Randomness in Interactive Proofs

Mihir Bellare* Oded Goldreich† Shafi Goldwasser‡

Abstract

This paper initiates a study of the quantitative aspects of randomness in interactive proof systems. Our result is a randomness-efficient error-reduction technique: given an Arthur-Merlin proof system (error probability $\leq \frac{1}{3}$) in which Arthur sends $l = l(n)$ random bits per round, we construct a new proof system which achieves error probability 2^{-k} at the cost of Arthur sending only $2l + O(k)$ random bits per round. The method maintains the number of rounds in the game.

Underlying the transformation is a novel sampling method for approximating the average value of an arbitrary function $f : \{0,1\}^l \rightarrow [0,1]$. The method evaluates the function on $O(\epsilon^{-2} \log \delta^{-1})$ sample points generated using only $2l + O(\log \delta^{-1})$ coin tosses to get an estimate which with probability $\geq 1 - \delta$ is within ϵ of the average value of the function.

1 Introduction

Interactive proofs [GMR],[Ba] are an extension of the classical NP notion of efficient provability in which non-determinism is enhanced by two new ingredients: *randomness* (the verifier flips coins) and *interaction* (the prover and verifier engage in a polynomial number of rounds of message exchange).

The power conferred by these new ingredients has been amply demonstrated: Fortnow, Karloff, Lund, and Nisan [FKLN] show that IP (the class of languages which possess interactive proofs of membership) contains the polynomial time hierarchy, and Shamir [Sh] shows that IP equals PSPACE. And furthermore, both ingredients are necessary: if the verifier did not flip coins then IP would collapse to NP (the prover could simulate all the moves of the verifier on his own) and if the parties did not interact then IP would clearly collapse to BPP.

This work is directed at understanding the power of interactive proofs as a function of the *amount of randomness* (number of coin tosses) used by the verifier. In contrast to the round complexity of interactive proofs which has received a great deal of attention (cf. [AGH],[Ba],[BM1],[BM2],[BHZ]), this "coin-flip" complexity has not previously been studied.

The particular question we investigate is the cost (in randomness) of reducing the error probability of Arthur-Merlin proof systems while maintaining the number of rounds.

1.1 Our Result: Randomness-Efficient Error-Reduction for Arthur-Merlin Games

An *Arthur-Merlin game* [Ba],[BM1] is a two-party protocol played by an all-powerful "prover", called *Merlin*, and a probabilistic polynomial-time "verifier", called *Arthur*. The game is played on a common input (and its purpose is to convince Arthur that the input belongs to some predetermined language). Arthur's role in the process is restricted to tossing coins and sending their outcome and then, at the end of the game, evaluating a polynomial-time predicate (applied to the common input and the full transcript of the interaction) in order to decide whether to accept or reject.

Arthur-Merlin games are thus a special form of interactive proof systems. However, their language recognition power has been shown by Goldwasser and Sipser [GS] to be equal to that of interactive proof systems.

We say the an Arthur-Merlin game is an Arthur-Merlin proof system for the language L if the error probability on any input w (the probability that Arthur accepts if $w \notin L$ or rejects if $w \in L$) is $\leq \frac{1}{3}$. This error probability can be decreased to 2^{-k} for any $k = k(n) \leq n^{O(1)}$ (while maintaining the number of rounds) by running the game $O(k)$ times in parallel and taking a majority vote on the outcomes

* MIT Laboratory for Computer Science, 545 Technology Square, Cambridge MA 02139. Partially supported by ARO grant No. DAAL 03-86-K-0171.

† Computer Science Dept., Technion, Haifa, Israel. Partially supported by grant No. 86-00301 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

‡ MIT Laboratory for Computer Science, 545 Technology Square, Cambridge MA 02139. Partially supported by NSF grant 25801, DARPA 71949, and grant No. 86-00301 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

CH2925-6/90/0000/0563$01.00 © 1990 IEEE

[Ba],[BM1]. Supposing that Arthur sent $l = l(n)$ random bits per round in the original game, this results in a game in each round of which Arthur sends $O(lk)$ random bits. We show the following

Theorem: *Given a $g = g(n)$ round Arthur-Merlin proof system for L in which Arthur sends $l = l(n)$ random bits per round, and given $k = k(n) \leq n^{O(1)}$, we can construct a g round Arthur Merlin proof system for L which achieves error probability $\leq 2^{-k}$ at the cost of Arthur sending $2l + O(k)$ random bits per round.*

This improves on a previous construction of ours which accomplished the same object at the cost of $O(l + gk)$ random bits per round, and on a later one of [BR] which used $O(l + k \log l)$ random bits per round.

The value $\frac{1}{3}$ in the bound on the error probability in the definition of L having an Arthur-Merlin proof system is not crucial: equivalent definitions are derived by letting the bound on the error probability be either $\frac{1}{2} - \frac{1}{n^{O(1)}}$ or $n^{-O(1)}$. However, until now it was not known whether this equivalence preserves the amount of randomness used (even up to a constant multiplicative factor). Results presented in this paper show that this is indeed the case.

1.2 Comparison with the Case of BPP

The problem of reducing the error probability in a randomness-efficient manner has received much attention in the context of the randomized complexity classes RP and BPP. However, the class IP seems more difficult to handle. To begin with, techniques based on assumptions of computational difficulty (which work for RP and BPP) are of no use against a prover who has the power to invert one-way functions. Thus the verifier cannot use pseudo-random sequences [BlMi] in place of random ones. Furthermore, the techniques of [CoWi],[IZ] (used for error-reduction in the classes RP and BPP) are not directly applicable here as we are not dealing with witness-sets which are fixed beforehand, but rather with an "adversary" (cheating "prover") that dynamically guides (by his responses to the verifiers coins) the search of the verifier for rejecting computations.

In general, not every result for RP and BPP translates easily (or at all) to a result on the class IP. Notable examples are results such as "BPP equals almost-P" [BG] and "BPP is contained in non-uniform P" [Ad]. The IP counterpart of the first was open for several years and finally proved by Nisan and Wigderson [NW], while the IP analogue of the second (i.e. "IP is contained in non-uniform NP") is not believed to be true.

1.3 Randomness-Efficient Approximation

Underlying our transformation is a new sampling method which combines ideas from [CG] and [AKS]. To discuss the merits of this method consider the problem of approximating the average value $\mathbf{E}[f] \stackrel{\text{def}}{=} 2^{-l} \sum_{x \in \{0,1\}^l} f(x)$ of an arbitrary function $f : \{0,1\}^l \to [0,1]$. An (ϵ, δ)-approximator (for this task) is a randomized process which picks several *sample points*, evaluates the function on them, and outputs an estimate which with probability $\geq 1 - \delta$ is within ϵ of $\mathbf{E}[f]$. The parameters to consider in designing such a process are the number of sample points and the number of coin tosses required to generate these sample points.

The straightforward method is to select $m = O(\epsilon^{-2} \log \delta^{-1})$ independent and uniformly distributed sample points and use as the estimate the average value of the function on these sample points. This requires $O(ml)$ coin tosses and m function evaluations. Our sampling method will require the same (up to a constant multiplicative factor) number of sample points (and function evaluations) but these sample points will be generated using only $2l + O(\log \delta^{-1})$ coin tosses. This result may be viewed as generalizing previous results of [AKS],[CoWi],[IZ],[CG] and [BR] (see §3 for a more complete discussion). Our (ϵ, δ)-approximator is optimal in the number of sample points generated, and, within the class of approximators using this number of sample points, also optimal in the number of coin tosses used (both to within constant multiplicative factors).

It is interesting to note that $(\frac{1}{6}, \delta)$-approximators for Boolean functions suffice for error-reduction in BPP (cf. [CoWi], [IZ]), whereas our amplification of IP relies on (ϵ, δ)-approximators of arbitrary functions ranging in $[0,1]$ with ϵ^{-1} being a polynomial.

2 Randomness-Efficient Error-Reduction for Arthur-Merlin Games

In this section we present our randomness-efficient error-reduction method. We begin with a review of Arthur-Merlin proof systems and the standard error-reduction method, and then proceed to describe and prove our protocol.

2.1 Arthur Merlin Games

An *Arthur-Merlin game* is a two-party protocol played by an all-powerful "prover", called *Merlin*, and a probabilistic polynomial-time "verifier", called *Arthur*. The game is played on a common input (and its purpose is to convince Arthur that the input belongs to some predetermined language). Arthur's role in the process is restricted to tossing coins and sending their outcome and then, at the end of the game, evaluating a polynomial-time predicate (applied to the common input and the full transcript of the interaction) in order to decide whether to accept or reject.

Let x denote the common input to the (Arthur-Merlin) game, $n = |x|$ its length, $l(n)$ the length of Arthur's messages, $q(n)$ the length of Merlin's, and $g(n)$ the number of rounds. We denote by $\rho(x, C) \in \{0,1\}$ Arthur's *decision* on input x and conversation C. The conversation C can be parsed uniquely into Arthur's and Merlin's messages: $C = r_1 y_1 \cdots r_g y_g$, where r_t is Arthur's t-th message and y_t is Merlin's response (we usually assume that Arthur plays first and Merlin second in each round). A strategy for Arthur, $A = (\rho, g, l, q)$, consists of the decision predicate ρ, as well as (polynomially bounded) functions specifying the number of rounds and the length of messages sent in each round by each party. For the sake of simplicity we assume that the length of the messages sent in each round is independent of the round.

Let M be a strategy for Merlin (i.e. M determines the next message of Merlin based on the common input and the messages received so far from Arthur). We denote by $\mathbf{P}[\,(A, M) \text{ accepts } x\,]$ the probability that $\rho(x, C) = 1$ when C is chosen at random (the probability space is that of all possible choices of $r_1, \ldots, r_{g(n)}$ taken with uniform distribution, and the y_t being set to $M(x, r_1 r_2 \ldots r_t)$).

Definition 2.1 We say that the Arthur strategy A defines an Arthur-Merlin proof system for L if the following conditions hold:

(1) *Completeness:* There exists a Merlin strategy M such that $\mathbf{P}[(A, M) \text{ accepts } x] \geq \frac{2}{3}$ for every $x \in L$.

(2) *Soundness:* $\mathbf{P}[\,(A, \widehat{M}) \text{ accepts } x\,] \leq \frac{1}{3}$ for every Merlin strategy $\widehat{M}$ and every $x \notin L$.

The strategy $\widehat{M}$ in the soundness conditions is sometimes called a *cheating* Merlin, while the strategy M in the completeness condition is called the *honest* Merlin. In fact, it suffices to consider (in both conditions) an "optimal Merlin", M_{opt_A}, that chooses all its messages in a way maximizing Arthur's accepting probability. Note that M_{opt_A} depends on A.

We define A's *accepting probability function* on partial conversations as follows (cf. [Ba],[BM1]):

- The value at a conversation is the value of A's deciding predicate:
$$acc(x, r_1 y_1 \ldots r_g y_g) = \rho(x, r_1 y_1 \ldots r_g y_g)\,.$$
- The value at a partial conversation ending with an Arthur message is the *maximum* value of all possible extensions by one move of Merlin:
$$acc(x, r_1 y_1 \ldots r_t y_t r_{t+1}) = \max_y acc(x, r_1 y_1 \ldots r_t y_t r_{t+1}.y)$$
for $t = g(n) - 1, \ldots, 0$.
- Finally the value of a partial conversation ending with a Merlin message is the *average* value of all its extensions by one move of Arthur:
$$acc(x, r_1 y_1 \ldots r_t y_t) = \mathbf{E}_r\, acc(x, r_1 y_1 \ldots r_t y_t.r)$$
for $t = g(n) - 1, \ldots, 0$.

The following

Proposition 2.2 *For any fixed history $r_1 y_1 \ldots r_t y_t$ one has*

$$acc(x, r_1 y_1 \ldots r_t y_t) \leq acc(x, r_1 y_1 \ldots r_{t-1} y_{t-1} r_t)$$

with equality holding when $y_j = M_{\mathsf{opt}_A}(r_1 \ldots r_{j-1})$ for all $j = 1, \ldots, t$.

follows directly from the definition.

A's accepting probability on input x is defined as

$$acc(x) \stackrel{\text{def}}{=} acc(x, \lambda)\,,$$

and the *error probability* of A on input x (with respect to a language L) is defined as

$$err_L(x) = \begin{cases} 1 - acc(x) & \text{if } x \in L \\ acc(x) & \text{otherwise.} \end{cases}$$

The error probability of A (with respect to L) is $e_L : \mathbf{N} \to [0,1]$ defined by $e_L(n) = \sup_{|x|=n} err_L(x)$. (Thus an Arthur strategy A defines a proof system for L if $e_L \leq \frac{1}{3}$).

2.2 Error-Reduction and its Standard Implementation

Error-reduction is the process of reducing the error probability of an Arthur-Merlin proof system from $\leq \frac{1}{3}$ to $\leq 2^{-k}$ for a given $k = k(n) \leq n^{O(1)}$. As an introduction to our error-reduction method, we review the standard one [Ba],[BM1].

Given $A = (\rho, g, l, q)$ defining an error $\leq \frac{1}{3}$ Arthur-Merlin proof system for L we are required to design A^* defining an error $\leq 2^{-k}$ Arthur-Merlin proof system for L. The solution is to play in parallel $m = O(k)$

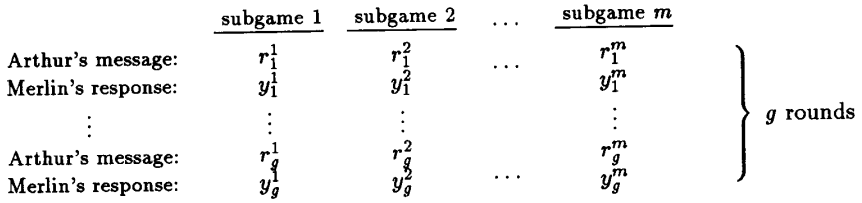

Figure 1: Framework of the Standard Error-Reduction Protocol

independent copies of the old game (the one defined by strategy A). The independence of Arthur's moves in the various "subgames" is used to prove that the error probability decreases exponentially with the number of subgames.

More concretely, A^* will, in round t, send ml random bits to Merlin. These bits are regarded as a sequence $r_t^1 \ldots r_t^m$ of m different round t messages of A. Merlin then responds with strings $y_t^1 \ldots y_t^m$, and y_t^i is regarded as the response of Merlin to r_t^i in the i-th subgame ($i = 1, \ldots, m$). This continues for g rounds (see Figure 1).

Finally, A^* will accept in the new game iff a majority of the subgames were accepting for the original A. That is, A^* accepts iff $|\{\, i : \rho(x, r_1^i y_1^i \ldots r_{g(n)}^i y_{g(n)}^i) = 1 \,\}| \geq \frac{m(n)}{2}$.

The bound on the error probability of the new game follows from the fact that the coin tosses used by Arthur in the different subgames are independent. However, the cost of this argument is in the large number of coin tosses used by A^*; namely $O(lk)$ coin tosses per round (to be contrasted with the l coin tosses used in each round of the original game).

2.3 Overview of Our Solution

We will play in parallel several *dependent* copies of the original game, and prove that although these copies depend on one another the error probability decreases exponentially with our "investment" in the randomness of each round.

More precisely, Arthur's message in round t will consist of a randomly chosen "seed" s_t. This (via an appropriate deterministic process) specifies a sequence of (statistically dependent) messages $r_t^1 \ldots r_t^m$ (m a function of our deterministic process) that will play the role of A's t-th round messages (recall, A is the original Arthur) for the different subgames. The Merlin of the new game computes the sequence of messages specified by the seed and replies with a sequence of m strings, $y_t^1 \ldots y_t^m$ that will be interpreted as his answers in the corresponding m subgames (see Figure 2). At the end, A^* will accept iff a majority of the subgames were accepting (i.e., iff $|\{\, i : \rho(x, r_1^i y_1^i \ldots r_{g(n)}^i y_{g(n)}^i) = 1 \,\}| \geq \frac{m(n)}{2}$).

We stress the Arthur's moves in the different rounds are still statistically independent (a new random seed is selected at each round), and that A^* actually sends in round t the (uniformly selected) seed s_t and both parties compute the sequence $r_t^1, \ldots, r_t^m$ specified by the seed s_t.

Within this template we rely on a combination of two different ideas.

The first idea is to guarantee that at each round the sequence of messages specified by the seed approximates (with very high probability) the average accepting probability of a sequence of independently chosen messages. That is, for each $t = 1, \ldots, g$, assuming $s_1, \ldots, s_{t-1}$ have been chosen, we guarantee that with high probability

$$\frac{1}{m}\sum_{i=1}^{m} acc(x, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i r_t^i) \approx \frac{1}{m}\sum_{i=1}^{m} \mathbf{E}_r\, acc(x, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r)$$

for the random choice of s_t (where $r_j^1 \ldots r_j^m$ is the sequence specified by s_j). Indeed if all seeds selected (for all rounds) provide good approximations in this sense then the rate of accepting subgames (in the new game) will approximate the accepting probability (in the original game).

The second idea is to use the above approach only to decrease the error probability to a non-negligible (i.e. $n^{-O(1)}$) amount and then play multiple copies of this new game, with the sequence of seeds at each round specified by a random walk on an expander graph.

We now proceed to describe these ideas in more detail. First, however, we make an assumption (without loss of generality) about Arthur's message lengths.

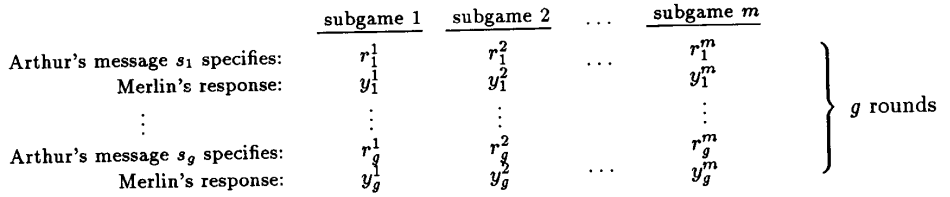

Figure 2: Framework of Our Error-Reduction Protocol

2.4 Restriction to a Special Case

For technical reasons it will be convenient to assume that Arthur messages are of length $\geq c \log n$ (for a specific constant c that will arise in our construction). This assumption does not reduce the generality of our results, since for every $c > 0$, any Arthur-Merlin game can transformed (without increasing the number of rounds or the total number of coin tosses) into one in which Arthur's messages are of length $\geq c \log n$. Namely, we have the following

Proposition 2.3 *Let $c > 0$ be a constant. Suppose $A = (\rho, g, l, q)$ defines an Arthur-Merlin proof system for L in which $l(n) \leq c \log n$. Then we can construct $A^* = (\rho^*, \frac{gl}{c \log n}, l^*, q^*)$ which defines an Arthur-Merlin proof system for L in which $l^*(n) = c \log n$.*

Proof: Group consecutive rounds of the given game into blocks in such a way that Arthur is sending $c \log n$ bits per block (we get $\frac{gl}{c \log n}$ blocks of $\frac{c \log n}{l}$ rounds each). The new game is formed by "collapsing" each of these blocks into a single round. In each round of the new game, Merlin (playing first) provides a response to each possible sequence of A's moves for that block (Merlin's message thus consists of n^c sequences, each consisting of $\frac{c \log n}{l}$ strings of length q). Arthur then uses $c \log n$ bits to select one of these sequences at random. ■

(Of course this reduction is truly valid only if Arthur used at least a super-logarithmic number of coins in total (i.e. if $\log n = o(lg)$). But this can be assumed without loss of generality since if $lg = O(\log n)$ then the language is in NP (and thus has an Arthur-Merlin proof system in which Arthur uses no coins)).

2.5 Reducing Error to any Non-Negligible Fraction

We show how to reduce the error probability to $\epsilon = n^{-O(1)}$ at the cost of doubling the number of coin tosses used in each round. The restriction on ϵ comes from the fact that the size of the game we construct here is polynomial in ϵ^{-1}.

It is well known that there exist (deterministic) procedures which take a $2l$ bit string s and specify a sequence of $m \leq 2^l$ strings of l bits each with the property that if s is chosen at random then the resulting sequence is pairwise independent (cf. [CaWe]). Let us fix one such procedure (which will be used also in §2.6 and §3). For example, regard s as the concatenation of two l bit strings a and b and then, identifying $\{0,1\}^l$ with $\mathrm{GF}(2^l)$, output $ax_1+b, \ldots, ax_m+b$ for some fixed sequence of m field elements $x_1, \ldots, x_m$.

We play m games in parallel where $m = poly(g, \epsilon^{-1})$ (we specify the exact value later). In each round t Arthur sends a $2l$ bit seed s_t and this is used to specify (via the above procedure) a sequence of m strings $r_t^1 \ldots r_t^m \in \{0,1\}^l$ (note that $m \leq 2^l$ since we may assume without loss of generality that $l \geq \log m$ (see §2.4)). These will play the role of A's round t messages. A^* accepts iff a majority of the subgames accept.

For the analysis, let $r_t^1 \ldots r_t^m$ be the sequence of messages specified by s_t and let $y_t^1 \ldots y_t^m$ be Merlin's response in round t ($t = 1, \ldots, g(n)$). Call a seed s_t *bad* for a history $\bar{r} = s_1 . y_1^1 \ldots y_1^m \cdots s_{t-1} . y_{t-1}^1 \cdots y_{t-1}^m$ if

$$\left| \tfrac{1}{m} \textstyle\sum_{i=1}^{m} \big(acc(x, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r_t^i) - acc(x, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i) \big) \right| \geq \tfrac{1}{6g} .$$

Then we have

Proposition 2.4 *At most a $(6g)^2/m$ fraction of the seeds are bad for any fixed history $\bar{r} = s_1 . y_1^1 \ldots y_1^m \cdots s_{t-1} . y_{t-1}^1 \cdots y_{t-1}^m$.*

This fact (which will also be used in §2.6) guarantees the correctness of our construction (as we shall see). First, let us prove it.

Proof: By Chebyshev's inequality we have

$$\mathbf{P}\left[\,\left|\tfrac{1}{m}\textstyle\sum_{i=1}^{m}\big(\,acc(x,r_1^i y_1^i\ldots r_{t-1}^i y_{t-1}^i.r_t^i)\right.\right.$$
$$\left.\left.-\,\mathbf{E}_r\,acc(x,r_1^i y_1^i\ldots r_{t-1}^i y_{t-1}^i.r)\big)\right|\geq\tfrac{1}{6g}\,\right]\leq\tfrac{(6g)^2}{m}$$

(the probability being over the random choice of s^t). But by the definition of the accepting probability function (see §2.1) we know that

$$\mathbf{E}_r\,acc(x,r_1^i y_1^i\ldots r_{t-1}^i y_{t-1}^i.r) = acc(x,r_1^i y_1^i\ldots r_{t-1}^i y_{t-1}^i)\,,$$

and Proposition 2.4 follows. ■

We now set $m \stackrel{\text{def}}{=} 36g^3\epsilon^{-1}$. The proof is completed by considering separately the cases of $x \in L$ and $x \notin L$.

Claim 2.5 *Suppose $x \in L$. Then there exists a Merlin strategy for which A^* accepts with probability $\geq 1-\epsilon$.*

Proof: We choose the strategy of setting $y_t^i = M_{\mathrm{opt}_A}(x, r_1^i\ldots r_{t-1}^i)$. By Proposition 2.2 it follows that

$$acc(x,r_1^i y_1^i\ldots r_t^i y_t^i) = acc(x,r_1^i y_1^i\ldots r_{t-1}^i y_{t-1}^i.r_t^i)\,.$$

By t applications of Proposition 2.4 it follows that at the end of t rounds we have

$$\tfrac{1}{m}\textstyle\sum_{i=1}^{m} acc(x,r_1^i y_1^i\ldots r_t^i y_t^i) > acc(x)-\tfrac{t}{6g}$$

with probability $\geq 1-\frac{t(6g)^2}{m}$ (recall that $acc(x,\lambda) = acc(x)$ (the accepting probability on input x)). Thus at the conclusion of the game ($t=g$) we are guaranteed that

$$\tfrac{1}{m}\textstyle\sum_{i=1}^{m} acc(x,r_i^1 y_i^1\ldots r_i^g y_i^g) > acc(x)-\tfrac{1}{6}$$

with probability $\geq 1-\epsilon$ (recall that $m = 36g^3\epsilon^{-1}$). But $\frac{1}{m}\sum_{i=1}^{m} acc(x,r_1^i y_1^i\ldots r_g^i y_g^i)$ is just the fraction of accepting subgames, and since $acc(x)\geq\frac{2}{3}$ we conclude that with probability $\geq 1-\epsilon$ a majority of the subgames accept. ■

Claim 2.6 *Suppose $x \notin L$. Then the probability that A^* accepts is $\leq \epsilon$.*

Proof: By Proposition 2.2 we know that

$$acc(x,r_1^i y_1^i\ldots r_t^i y_t^i) \leq acc(x,r_1^i y_1^i\ldots r_{t-1}^i y_{t-1}^i.r_t^i)\,.$$

By t applications of Proposition 2.4 it follows that at the end of t rounds we have

$$\tfrac{1}{m}\textstyle\sum_{i=1}^{m} acc(x,r_1^i y_1^i\ldots r_t^i y_t^i) < acc(x)+\tfrac{t}{6g}$$

with probability $\geq 1-\frac{t(6g)^2}{m}$. Thus at the conclusion of the game ($t=g$) we are guaranteed that

$$\tfrac{1}{m}\textstyle\sum_{i=1}^{m} acc(x,r_1^i y_1^i\ldots r_g^i y_g^i) < acc(x)+\tfrac{1}{6}$$

with probability $\geq 1-\epsilon$. The conclusion follows from the fact that $acc(x)\leq\frac{1}{3}$. ■

A similar argument can be applied even if the original game has error probability $\frac{1}{2}-\frac{1}{n^c}$ (use $m = g^3 n^{2c}\epsilon^{-1}$). Hence, we get

Theorem 2.7 *Let c be >0, and $A=(\rho,g,l,q)$ an Arthur strategy with error probability $\leq\frac{1}{2}-n^{-c}$ with respect to the language L. Then, for any $k=k(n)\leq n^{O(1)}$ we can construct another Arthur strategy $A^*=(\rho^*,g,2l,g^3n^{2c}k\cdot q)$ with error probability $\leq\frac{1}{k}$ with respect to L.*

That is, the error probability can be reduced $n^{-O(1)}$ with the number of rounds remaining invariant and the number of coin tosses used in each round only doubling.

2.6 Error Reduction at Logarithmic Cost

We now show how to reduce the error probability to 2^{-k} using $O(k)$ additional random bits per round, where $k=k(n)$ is $\leq n^{O(1)}$ with $\log n = o(k)$. The construction uses any explicit construction of expanders of fixed degree (see [GG],[LPS]). We will make use of the following

Lemma 2.8
(Expander Path Lemma) *Let $B_1, B_2, \ldots, B_k$ be a sequence of k subsets of the expander nodes such that each B_i has measure at most $\epsilon<\frac{1}{2}$. Consider a random walk of length k on the expander and denote the sequence of vertices visited at multiples of $L \stackrel{\text{def}}{=} \alpha\log\epsilon^{-1}$ by $v_1, v_2, \ldots, v_{k/L}$ (the constant $\alpha\geq 1$ depends only on the second eigenvalue of the expander). Then, for every b and every sequence of b indices, $1\leq i_1<i_2<\ldots<i_b\leq k/L$,*

$$\mathbf{P}\left[\,v_{i_1}\in B_{i_1},\ldots,v_{i_k}\in B_{i_k}\,\right]\leq(2\epsilon)^{b/2}\,.$$

(A random walk of length k means we begin at a random start vertex and then take $k-1$ steps, each of which consists of moving from the current vertex to a random neighbor. For a d-regular expander with vertex set $\{0,1\}^s$ the walk is thus specified by $s+(k-1)\log d$ random bits).

The proof of Lemma 2.8, following the ideas of [AKS], appears in Appendix A.

We fix an expander with vertex set $\{0,1\}^{2l}$. Let $\epsilon \stackrel{\text{def}}{=} \frac{1}{2}(24)^{-2}(kg)^{-4}$, $L \stackrel{\text{def}}{=} \alpha\log\epsilon^{-1}$, and $k^* \stackrel{\text{def}}{=} 24\alpha k/L = 24k/\log\epsilon^{-1}$ (where α is the constant of Lemma 2.8).

We will play in parallel k^* copies of the game presented in §2.5, setting m such that the error in each of these games is $\leq \epsilon$. Since each of our subgames consists itself of m subgames, we will be playing a total of k^*m subgames which are arranged in k^* blocks each consisting of m subgames. The sequence of messages to be regarded as A's moves in round t of our game will thus have the form $r_t^1 \ldots r_t^m, \cdots, r_t^{(k^*-1)m+1} \ldots r_t^{k^*m}$. The sequence of k^* seeds which specify the message blocks is itself specified by a random walk on the expander.

More precisely, Arthur's message in the t-th round consists of a "super-seed" S_t of length $2l + O(k)$ (the constant in the O depends on α and the degree of the expander). This super-seed is used to specify a random walk of length $24\alpha k$ on the expander. We denote the vertices visited at intervals of length L by $s_{t,1}, \ldots, s_{t,k^*} \in \{0,1\}^{2l}$. Each $s_{t,j}$ specifies a sequence of l bit strings (as in §2.5) which we denote by $r_t^{(j-1)m+1} \ldots r_t^{(j-1)m+m}$ where $m \stackrel{\text{def}}{=} (6g)^2\epsilon^{-1}$. The string $r_t^{(j-1)m+i}$ is regarded as Arthur's t-th round message in the $((j-1)m+i)$-th subgame. Merlin's (answer) message in the t-th round has the form $y_t^1 \ldots y_t^m \cdots y_t^{(k^*-1)m+1} \ldots y_t^{k^*m}$ where $y_t^{(j-1)m+i}$ is called Merlin's answer in the $((j-1)m+i)$-th subgame. After all g rounds are completed Arthur evaluates ρ in each of the subgames and accepts iff in a majority of blocks there is a majority of accepting conversations (i.e. iff $\frac{1}{m}\sum_{i=1}^{m} \rho(x, r_1^{(j-1)m+i} y_1^{(j-1)m+i} \ldots r_g^{(j-1)m+i} y_g^{(j-1)m+i})$ is $\geq \frac{1}{2}$ for a majority of the $j = 1, \ldots, k^*$).

To analyze the game, we remind the reader that in each block (of m subgames), for each round at most a $(6g)^2/m = \epsilon$ fraction of the seeds are bad for the current history (Proposition 2.4). By the Expander Path Lemma (Lemma 2.8), the probability that a particular sequence of b seeds is bad (in a sequence of k^* seeds generated by the random walk) is bounded above by $(2\epsilon)^{b/2}$. Thus, the probability that in the g rounds of the game at least $b = k^*/3$ of the k^*g seeds are bad is bounded above by

$$\begin{aligned}
\binom{k^*g}{b} \cdot (2\epsilon)^{b/2} &\leq (k^*g)^b (2\epsilon)^{b/2} \\
&= (k^*g\sqrt{2\epsilon})^{k^*/3} \\
&= \left(\frac{k^*g}{24(kg)^2}\right)^{k^*/3} \\
&= \left(\frac{24kg}{24(kg)^2 \log \epsilon^{-1}}\right)^{\frac{24k}{3\log \epsilon^{-1}}} \\
&= \left(\frac{1}{kg \log \epsilon^{-1}}\right)^{\frac{8k}{\log \epsilon^{-1}}} \\
&= \left(2^{-k}\right)^{\frac{8}{\log \epsilon^{-1}} \log(kg \log \epsilon^{-1})}.
\end{aligned}$$

But $1 \leq \log \epsilon^{-1} \leq O(1) + 4\log(kg)$ so

$$\frac{8}{\log \epsilon^{-1}} \log(kg \log \epsilon^{-1}) \geq 1$$

(for large enough n) and hence the above probability is $\leq 2^{-k}$. That is, the probability that at least $k^*/3$ of the blocks contain a bad seed in one of their rounds is bounded above by 2^{-k}. However, subgame-blocks with no bad seeds (in all rounds) necessarily have a majority of correct decisions. Hence a majority of the blocks may have a wrong majority with probability $\leq 2^{-k}$.

With changes in parameters the above argument can also be applied when the original protocol had error probability $\frac{1}{2} - n^{-O(1)}$ and thus we have

Theorem 2.9 *Let $c > 0$ and $A = (\rho, g, l, q)$ an Arthur strategy with error probability $\leq \frac{1}{2} - n^{-c}$ with respect to the language L. Then, for every $k = k(n) \leq n^{O(1)}$ with $\log n = o(k)$ we can construct another Arthur strategy $A^* = (\rho^*, g, 2l + O(k), O(g^7k^5n^{2c}/\log(kg)) \cdot q)$ with error probability $\leq 2^{-k}$ with respect to L.*

Remark: The *total* number of coins used by Arthur in our construction is $O(g(l+k))$ (as opposed to gl in the original game). In the special case that $l \leq O(\log n)$, however, the construction actually yields an even smaller number: namely, for any constant $c > 0$, we can construct Arthur achieving error 2^{-k} using a total of $O(g(l + \frac{lk}{c \log n}))$ coins. This is because we are actually applying our method to the modified proof system of Proposition 2.3.

3 Randomness-Efficient Approximation

Implicit in the previous section is a new sampling method. An application of particular interest is to the problem of approximating the average value $\mathbf{E}[f] \stackrel{\text{def}}{=} 2^{-l}\sum_{x \in \{0,1\}^l} f(x)$ of an arbitrary function $f : \{0,1\}^l \to [0,1]$

We define an (ϵ, δ)-approximator as a two stage process. In a first, randomized stage, it picks a collection of sample points. In the second, deterministic stage it evaluates the function f on these sample points and, based on these evaluations, outputs an estimate. We require that with probability $\geq 1 - \delta$ the value of this estimate is within ϵ of $\mathbf{E}[f]$.

The parameters to consider in designing an (ϵ, δ)-approximator are the number of sample points (which by the above definition is also the number of function evaluations) and the number of coin tosses used to generate these sample points.

The straightforward method is to select $m = O(\epsilon^{-2} \log \delta^{-1})$ independent and uniformly distributed sample points and use as the estimate the average value of the function on these sample points. This requires $O(ml)$ coin tosses (and m function evaluations).

Dramatic savings in the number of coin tosses is possible by selecting $O(\epsilon^{-2}\delta^{-1})$ pairwise independent sample points (cf. [CG]) which yields a (ϵ, δ)-approximator at the cost of $2l$ coin tosses. However, the number of sample points here grows inversely proportional to the desired error probability δ, and thus this method cannot be applied when δ is exponentially small. This restriction on the error probability is removed by [BR] whose techniques yield a (ϵ, δ)-approximator which outputs $poly(\epsilon^{-1}, \log \delta^{-1}, l)$ sample points using $O(l + \log \delta^{-1} \cdot \log l)$ coin tosses.

An alternative sampling method for the special case of boolean valued functions (i.e. f takes on only the values 0 and 1) is based on selecting a random walk on a 2^l node explicitly constructed expander graph (cf. [AKS],[CoWi],[IZ]). This method yields a $(\frac{1}{6}, \delta)$-approximator of boolean functions which outputs $O(\log \delta^{-1})$ sample points using $l + O(\log \delta^{-1})$ coin tosses†.

Combining expander graph methods with those of [CG] we construct a (ϵ, δ)-approximator (for arbitrary functions) which outputs $O(\epsilon^{-2} \log \delta^{-1})$ sample points using $2l + O(\log \delta^{-1})$ coin tosses. The method is implicit in the previous section, but for sake of clarity we present it explicitly below.

We fix an expander graph with vertex set 2^{2l}. Let $\mu \stackrel{\text{def}}{=} 2^{-9}$, $L \stackrel{\text{def}}{=} \alpha \log \mu^{-1}$, $k \stackrel{\text{def}}{=} L \log \delta^{-1}$, $k^* \stackrel{\text{def}}{=} k/L = \log \delta^{-1}$, and $m \stackrel{\text{def}}{=} \epsilon^{-2}\mu^{-1}$ (where α is the constant of Lemma 2.8). Using $2l + O(\log \delta^{-1})$ random bits we specify a walk of length k on the expander and let $s^1, \ldots, s^{k^*} \in \{0,1\}^{2l}$ be the vertices visited at intervals of length L. Each s^j is used to to specify a sequence of l bit strings (as in §2.5) which we denote $x_1^j, \ldots, x_m^j$. We compute the values

$$v^j = \frac{1}{m} \sum_{i=1}^{m} f(x_i^j) \quad (j = 1, \ldots, k^*)$$

and output as our estimate the median value of $v^1, \ldots, v^{k^*}$.

† One can obtain an (ϵ, δ)-approximator of arbitrary functions by using the ideas of [IZ], but this will require $\Omega(\epsilon^{-2} \log \delta^{-1})$ sample points generated using $l + \Omega(\epsilon^{-2} \log \delta^{-1})$ coin tosses.

For the analysis, call (a seed specifying) a sequence $x_1^j, \ldots, x_m^j$ *bad* if

$$\left| \frac{1}{m} \sum_{i=1}^{m} f(x_i^j) - \mathbf{E}[f] \right| \geq \epsilon \, .$$

By Chebyshev's inequality we know that for a pairwise independent sequence $x_1^j, \ldots, x_m^j$ one has

$$\mathbf{P}\left[\left| \frac{1}{m} \sum_{i=1}^{m} f(x_i^j) - \mathbf{E}[f] \right| \geq \epsilon \right] \leq \frac{1}{m\epsilon^2} = \mu$$

and thus at most a fraction μ of (the seeds specifying) our sequences are bad. By the Expander Path Lemma the probability that a majority of the seeds are bad is bounded above by

$$\begin{aligned} \binom{k^*}{k^*/2} \cdot (2\mu)^{k^*/4} &\leq (2(2\mu)^{1/4})^{k^*} \\ &= 2^{-\log \delta^{-1}} \\ &= \delta \, . \end{aligned}$$

Hence with probability $\geq 1 - \delta$ a majority of the values v^j are within ϵ of $\mathbf{E}[f]$ and thus with probability $\geq 1 - \delta$ the median of these values is within ϵ of $\mathbf{E}[f]$.

The new sampling method is optimal (up to a constant multiplicative factor) in terms of the number of sample points used. Namely, any (ϵ, δ)-approximator for all functions $f : \{0,1\}^l \to \{0,1\}$ must use $\Omega(\epsilon^{-2} \log \delta^{-1})$ sample points. Furthermore, among the (ϵ, δ)-approximators using $O(\epsilon^{-2} \log \delta^{-1})$ sample points, our scheme is also optimal (up to a constant multiplicative factor) in terms of the amount of randomness used.

4 Conclusion and Open Problems

Recall that $g = g(n)$ denotes the number of rounds in an Arthur-Merlin game, and $l = l(n)$ denotes the number of coin tosses used by Arthur in every round. Our main result (Theorem 2.9) is a transformation of Arthur Merlin games in which the error probability is reduced to 2^{-k} (where $k = k(n) \leq n^{O(1)}$) while maintaining the number of rounds and using $2l + O(k)$ coin tosses per round. Two challenging tasks are to

- Present a transformation in which the total number of coins flipped by Arthur is $O(gl + k)$. (Our result yields an Arthur Merlin game in which the total number of coins flipped by Arthur is $O(g(l + k))$ (but see the Remark at the end of section §2.6)). The problem is, of course when g is unbounded.
- Generalize this to (general) interactive proof systems. In fact, we don't know of any direct nontrivial amplification of general interactive proof

systems. By "direct" we mean without first transforming the interactive proof into an Arthur-Merlin game (a transformation which requires exponentially small error probability), and by "non-trivial" we mean more efficient in terms of coin tosses than playing independent copies of the same interactive proof.

A more modest challenge is to improve our result so that the number of coins is $l+O(k)$ per round (instead of $2l+O(k)$).

Acknowledgments

We thank John Rompel for many valuable ideas and discussions.

The first author thanks Silvio Micali.

References

[Ad] Adleman, L.M., "Two Theorems on Random Polynomial Time," FOCS 78.

[AGH] Aiello, W., S. Goldwasser and J. Håstad, "On the Power of Interaction," FOCS 86.

[AKS] Ajtai, M., J. Komlos and E. Szemeredi, "Deterministic Simulation in Logspace," STOC 87.

[Al] Alon, N., "Eigenvalues and Expanders," *Combinatorica* **6**(2), 83–96 (1986).

[Ba] Babai, L., "Trading Group Theory for Randomness," STOC 85.

[BM1] Babai, L. and S. Moran, "Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes," *J. Computer and System Sciences* **36**, 254–276 (1988).

[BM2] Babai, L. and S. Moran, "Proving Properties of Interactive Proofs by a Generalized Counting Technique," *Information and Computation* **82**, 185–197 (1989).

[BR] Bellare, M. and J. Rompel, "Randomness-Efficient Sampling of Arbitrary Functions," *MIT LCS Tech. Memo.* TM-433.

[BG] Bennet, C.H. and J. Gill, "Relative to a random Oracle A, $P^A \neq NP^A \neq coNP^A$, with probability 1," *SIAM J. on Computing* **10**, 96–113 (1981).

[BlMi] Blum, M., and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM Journal on Computing* **13** (4), 850-864 (1984).

[BHZ] Boppana, R., J. Hastad and S. Zachos, "Does co-NP have Short Interactive Proofs?," *Info. Processing Letters* **25**, 127–132 (1987).

[CaWe] Carter, L. and M. Wegman, "Universal Classes of Hash Functions," *J. Computer and System Sciences* **18**, 143–154 (1979).

[CG] Chor, B. and O. Goldreich, "On the Power of Two–Point Based Sampling," *J. of Complexity* **5**, 96–106 (1989).

[CoWi] Cohen, A. and A. Wigderson, "Dispersers, Deterministic Amplification, and Weak Random Sources," FOCS 89.

[FKLN] Fortnow, L., H. Karloff, C. Lund, and N. Nisan, "Algebraic Methods for Interactive Proof Systems," FOCS 90 (these proceedings).

[GG] Gabber, O. and Z. Galil, "Explicit Construction of Linear Sized Superconcentrators," *J. Computer and System Sciences* **22**, 407–420 (1981).

[GMR] Goldwasser, S., S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proofs," *SIAM J. Computing* **18**(1), 186–208 (1989).

[GS] Goldwasser, S. and M. Sipser, "Private Coins versus Public Coins in Interactive Proof Systems," STOC 86.

[IZ] Impagliazzo, R. and D. Zuckerman, "How to Recycle Random Bits," FOCS 89.

[LPS] Lubotzky, A., R. Phillips and P. Sarnak, "Explicit Expanders and the Ramanujan Conjectures," STOC 86.

[NW] Nisan, N. and A. Wigderson, "Hardness vs. Randomness," FOCS 88.

[Sh] Shamir, A., "IP=PSPACE," FOCS 90 (these proceedings).

A Appendix: Proof of the Expender Path Lemma

Modify the adjacency matrix (of the expander graph) by dividing the entries by d (the degree) and let A be the resulting N-by-N matrix. Let $\lambda_1 \leq \lambda_2 \cdots \leq \lambda_N$ and $e_1, e_2, \ldots, e_N$ be its eigenvalues and corresponding eigenvectors. By the Perron-Frobenius Theorem, non-negative A has $\lambda_1 = 1$. By the expander property, λ_2 is a constant < 1 (cf. [Al]). The space V_0 spanned by $e_2, \cdots, e_N$ is orthogonal to the space V_1 spanned by $e_1 = (1, 1, \cdots, 1)$, and A is invariant on these subspaces of $\mathbf{R}^N$. Any $X \in \mathbf{R}^N$ can be written as $X = X_0 + X_1$ where $X_0 \in V_0$ and $X_1 \in V_1$. Let $||X||$ denote the Euclidean norm and $||X||_1$ denote the L_1 norm (i.e. sum of absolute avlues). Let P_j be a projection matrix such that the i^{th} component of $P_j X$ equals the i^{th} component of X if $i \in S_j$ and zero otherwise. We have $||PY|| \leq ||Y||$ (for every Y). We show below that for every j and every $i > L \stackrel{\text{def}}{=} O(\log \epsilon^{-1})$

$$||P_j A^i X|| \leq \sqrt{2\epsilon} \cdot ||X||.$$

From this it follows that if

$$Y = P_{i_b} A^{(i_b - i_{b-1}) \cdot L} \cdots P_{i_2} A^{(i_2 - i_1) \cdot L} P_{i_1} A^{i_1 \cdot L} X$$

then $||Y|| \leq (2\epsilon)^{b/2} ||X||$. Thus, the probability that a random walk, starting at the uniform distribution, e_1/N, and terminating after k steps at distribution Y, passes through the set S_{i_j} in the $(i_j L)^{\text{th}}$ step for $i = 1, 2, ..., b$ is

$$||Y||_1 \leq \sqrt{N} \cdot ||Y|| \leq \sqrt{N} (2\epsilon)^{b/2} ||e_1/N|| = (2\epsilon)^{b/2}.$$

It is left to show that, for evey $i \geq L$, $||PA^i X|| \leq \sqrt{2\epsilon} ||X||$, where P is any of the above projection matrices. Let $X = X_0 + X_1$. Clearly $||X|| = (||X_0||^2 + ||X_1||^2)^{1/2}$. We get

$$\begin{aligned} ||PA^i X|| &\leq ||PA^i X_0|| + ||PA^i X_1|| \\ &\leq ||A^i X_0|| + ||PX_1|| \\ &\leq (2(||A^i X_0||^2 + ||PX_1||^2))^{1/2} \end{aligned}$$

(The first inequality is by the Triangle inequality and the second inequality uses $||PY|| \leq ||Y||$ and $AX_1 = X_1$). We now use $AX_0 = \sum_{i=2}^N c_i A e_i = \sum_{i=2}^N c_i \lambda_i e_i$ to bound $||AX_0||^2 \leq \lambda_2^2 ||X_0||^2$ and $||A^i X_0||^2 \leq \lambda_2^{2i} ||X_0||^2 \leq \lambda_2^{2L} ||X_0||^2$ follows. Hence $||A^i X_0||^2 \leq \lambda_2^{O(\log \epsilon^{-1})} ||X_0||^2$ and a suitable choice of the constant in the O-notation yields $||A^i X_0||^2 \leq \epsilon ||X_0||^2$. Next, bound $||PX_1||^2 \leq \epsilon ||X_1||^2$ (using the fact that X_1 has all components equal). Hence

$$||PA^i X|| \leq (2\epsilon ||X_0||^2 + 2\epsilon ||X_1||^2)^{1/2} = (2\epsilon)^{1/2} ||X|| \,.$$