Texts in Computational Complexity: IP, AM and round speed-up

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

November 12, 2005

Proving that $\mathcal{IP}(f) \subseteq \mathcal{AM}(O(f)) \subseteq \mathcal{AM}(f)$

We denote by $\mathcal{IP}(f)$ (resp., $\mathcal{AM}(f)$) the class of sets having interactive proof systems (resp., public-coin proof systems) in which a total of $f(|x|)$ messages are exchanged on common input x. We present proof of the following two results.

Theorem 1 (Round-efficient emulation of $\mathcal{IP}$ by $\mathcal{AM}$): *Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomially bounded function. Then $\mathcal{IP}(f) \subseteq \mathcal{AM}(f+3)$.*

We comment that, in light of the following linear speed-up in round-complexity for $\mathcal{AM}$, it suffices to establish $\mathcal{IP}(f) \subseteq \mathcal{AM}(O(f))$.

Theorem 2 (Linear speed-up for $\mathcal{AM}$): *Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomially bounded function. Then $\mathcal{AM}(2f) \subseteq \mathcal{AM}(f+1)$.*

Combining these two theorems, we obtain a linear speed-up for $\mathcal{IP}$; that is, for any polynomially bounded $f : \mathbb{N} \rightarrow (\mathbb{N} \setminus \{1\})$, it holds that $\mathcal{IP}(O(f)) \subseteq \mathcal{AM}(f) \subseteq \mathcal{IP}(f)$.

We mention that the proof of Theorem 1 relies on the fact that, for every f, error-reduction is possible for $\mathcal{IP}(f)$. Specifically, error-reduction can be obtained via *parallel* repetitions (see [3, Apdx. C.1]). We note that error-reduction (in the context of $\mathcal{AM}(f)$) is implicit also in the proof of Theorem 2 (and is explicit in the original proof of [1]).

1 Emulating general interactive proofs by AM-games

In this section we prove Theorem 1. Our proof differs from the original proof of Goldwasser and Sipser [5] only in the conceptualization and implementation of the iterative process.

1.1 The basic approach

Our aim is to transform a general interactive proof system (P, V) into a public-coin interactive proof system for the same problem. Suppose, without loss of generality, that P constitutes an optimal prover with respect to V (i.e., P maximizes the acceptance probability of V on any input). Then, for any yes-instance, the set of coin sequences that make V accept when interacting with this optimal prover contains all possible outcomes, whereas for a no-instance this set is very small.

The idea is having a public-coin system in which the prover prove to the verifier that the said set is big. Such a proof system can be constructed using ideas as in the case of approximate counting, while replacing the NP-oracle with a prover that is required to prove the correctness of its answers. Implementing this idea requires taking a closer look at the set of coin sequences that make V accept an input.

We demonstrate the implementation of the foregoing approach by considering an interactive proof system (such as the Graph Non-Isomorphism Protocol of [4]) in which the verifier V sends a single message to which the prover P responses. Further suppose that, when the common input is a yes-instance, each possible message of V is equally likely (which holds for a minor modification of the Graph Non-Isomorphism Protocol).[1] Specifically, suppose that on input x, the verifier V tosses $\ell = \ell(|x|)$ coins and sends one out of M possible messages (as determined by the input and the coin sequence). Then, in the *public-coin system*, the prover will claim that in the original proof there are M possible V-messages such that the original prover can respond to each of them in a way that is accepted by $2^\ell/M$ corresponding coin sequences of V. To prove this claim, the prover lets the verifier select at random one of the possible M messages (e.g., by selecting coins for V), denoted α, and the prover send back an adequate P-message, denoted β, and proves that β would have been accepted by $2^\ell/M$ possible coin sequences of V. The latter proof follows the idea of the reduction of approximate counting (of NP-witnesses) to $\mathcal{NP}$: The verifier applies a random sieve that lets only a $(2^\ell/M)^{-1}$ fraction of the elements pass, and the prover proves that some adequate sequence of V-coins has passed this sieve. The latter claim is proved by merely presenting such a sequence, denoted r, and the verifier can check whether indeed r passes the sieve as well as fits the initial message α and would have made V accept the prover message β (i.e., V would have accepted the input, on coins r, when receiving the prover message β). We stress that the foregoing interaction (and in particular the random sieve) can be implemented in the public-coin model.

A few technical problems arise. Firstly, recall that the random sieve only allows for an approximation of set sizes. However, since the gap between the acceptance probability of yes-instances and no-instances is big enough (or can be made big enough by parallel repetition), this suffices. Secondly, in general, it is not necessarily the case that each possible message of V is equally likely. However, the prover may cluster the V-messages into few (say ℓ) clusters such that the messages in each cluster are sent (by V) with roughly the same probability (say, up to a factor of two). Focusing on the cluster having the largest probability weight, the prover can proceed as in the simple case. This has a potential of cutting the probabilistic gap[2] between yes-instances and no-instances by a factor related to the number of clusters times the approximation level within clusters (e.g., a factor of $O(\ell)$), but this loss is negligible in comparison to the initial gap (which can be obtained via error-reduction). Lastly, there is the fact that we only dealt with a two-message system (i.e., $\mathcal{IP}(2)$).

It is tempting to say that the general case of $\mathcal{IP}(f)$ can be dealt by recursion (or rather iterations), and indeed this is almost the case. Recall that our treatment of the case of $\mathcal{IP}(2)$ boils down to having the verifier choose a random V-message, α, and having the prover send a P-response, β, and finally prove that β is acceptable by many V-coins. In other words, the prover should prove that in the conditional probability space defined by V-message α, the original verifier V accepts with high probability. In the general case (of $\mathcal{IP}(f)$), the latter claim refers to the

[1] In the original protocol, the verifier selects at random one of the two input graphs, and sends a random isomorphic copy of it. In the modification, the verifier creates a random isomorphic copy of each of the two input graphs, and sends them in a random order.

[2] The point is that in one case all clusters may have equal weight, and thus a corresponding factor is lost, while in the other case all probability mass may be concentrated in a single cluster.

probability of accepting in the residual interaction, which consists of $f-2$ messages, and thus the very same protocol can be applied iteratively (until we get to the last message, which is dealt as in the case of $\mathcal{IP}(2)$). The only problem is that in the residual interactions, it may not be easy for the verifier to select a random V-message (as done in the case of $\mathcal{IP}(2)$). Instead, the verifier will be assisted by the prover, while making sure that it is not being fooled by the prover. Indeed, this calls for an adequate "random selection" protocol, which need to be implemented in the public-coin model. For simplicity, we may consider the problem of selecting a uniform sequence of coins in the residual probability space, because such a sequence determines the desired random V-message.

1.2 Random selection

Various types of "random selection" protocols have appeared in the literature (see, e.g., [6, Sec. 6.4]). The common theme in these protocols is that they allow for a probabilistic polynomial-time player (called the *verifier*) to sample a set, denoted $S \subset \{0,1\}^\ell$, while being assisted by a second player (called the *prover*) that is powerful but not trustworthy. These nicknames fit the common conventions regarding interactive proofs and are further justified by the typical applications of such protocols as subroutines within an interactive proof system (where indeed the first party is played by the higher-level verifier while the second party is played by the higher-level prover). The various types of random selection protocols differ by what is known about the set S and what is required from the protocol.

Here we will assume that the verifier is given a parameter N, which is supposed to equal $|S|$, and the performance guarantee of the protocol will be meaningful only for sets of size at most N. We desire a constant-round (preferably two-round) public-coin protocol for this setting such that the following holds, with respect to a security parameter $\varepsilon = 1/\text{poly}(\ell)$.

1. If both players are honest and $N = |S|$ then the verifier's output is ε-close to the uniform distribution over S. Furthermore, the verifier always outputs an element of S.

2. For any set $S' \subseteq \{0,1\}^\ell$ if the verifier follows the protocol then, no matter how the prover behaves, the verifier's output resides in S' with probability at most $\text{poly}(\ell/\varepsilon) \cdot (|S'|/N)$.

Note that the second property is meaningful only for sets S' of size (significantly) smaller than N.

A three-round public-coin protocol can be obtained by using the ideas that underly uniform generation of NP-witnesses (as presented in [2]): Specifically, we use a high quality hashing function of $\{0,1\}^\ell$ to $\{0,1\}^m$, which in turn defines a partition of $\{0,1\}^\ell$ into 2^m cells. We set $m = \max(0, \log_2 N - O(\log \ell/\varepsilon))$ in order to guarantee that if $|S| = N$ then, with overwhelmingly high probability, each cell defined by the hashing function contains $(1 \pm \varepsilon) \cdot |S|/2^m$ elements of S. In the protocol, the prover selects a good hashing function (i.e., one defining such a good partition of S) and sends it to the verifier, which answers with a uniformly selected cell, to which the prover responds with a uniformly selected element of S that resides in this cell.[3]

Note that this protocol satisfies the aforementioned properties. In particular, the second property follows because for every possible hashing function, the fraction of cells containing an element

[3] A more natural version of this protocol consists of having the verifier select at random a hashing function as well as a cell, and asks the prover for a list of $(1-\varepsilon) \cdot N/2^m$ elements in this cell. The verifier then outputs an element that is uniformly selected in the list. This protocol provides a stronger guarantee with respect to cheating provers: the verifier's output resides in S' with probability at most $(|S'|/N) + \varepsilon$. However, even in case the prover is honest, this protocol does not guarantee that the verifier always outputs an element of S, because it may happen (rarely) that the hashing function selected by the verifier is not good. For this reason, we preferred the version presented in the main text.

of S' is at most $|S'|/2^m$, which is upper-bounded by $\text{poly}(\ell/\varepsilon) \cdot |S'|/N$. We stress that the protocol is indeed in the public-coin model, and comment that the fact that it uses three messages rather than two will have a minor effect on our application.

1.3 The iterated partition protocol

The random selection protocol discussed in §1.2 is meaningful only with respect to sets (i.e., S') that are smaller than the given parameter N. Here we explain why this suffices for our goals. We start with some notations.

Fixing any input x to (P, V), we denote by $t = t(|x|)$ the number of pairs of communication rounds (assuming that the verifier takes the first move in (P, V))[4] and by $\ell = \ell(|x|) > t$ the number of coins tossed by V. Recall that we assume that P is an optimal prover (with respect to V), and that (without loss of generality) P is deterministic. Let us denote by $\langle P, V(r)\rangle(x)$ the full transcript of the interaction of P and V on input x, when V uses coins r; that is, $\langle P, V(r)\rangle(x) = (\alpha_1, \beta_1, ..., \alpha_t, \beta_t, \sigma)$ if $\sigma = V(x, r, \beta_1, ..., \beta_t) \in \{0, 1\}$ is V's final verdict and for every $i = 1, ..., t$ it holds that $\alpha_i = V(x, r, \beta_1, ..., \beta_{i-1})$ and $\beta_i = P(x, \alpha_1, ..., \alpha_i)$. For any partial transcript ending with a P-message, $\gamma = (\alpha_1, \beta_1, ..., \alpha_{i-1}, \beta_{i-1})$, we denote by $\text{ACC}_x(\gamma)$ the set of coin sequences that are consistent with the partial transcript γ and lead V to accept x when interacting with P; that is, $r \in \text{ACC}_x(\gamma)$ if and only if for some γ' it holds that $\langle P, V(r)\rangle(x) = (\alpha_1, \beta_1, ..., \alpha_{i-1}, \beta_{i-1}, \gamma', 1)$. The same notation is also used for a partial transcript ending with a V-message; that is, $r \in \text{ACC}_x(\alpha_1, \beta_1, ..., \alpha_i)$ if and only if $\langle P, V(r)\rangle(x) = (\alpha_1, \beta_1, ..., \alpha_i, \gamma', 1)$ for some γ'.

Motivation. By suitable error reduction, we may assume that (P, V) has soundness error $\mu = \mu(|x|)$ that is smaller than $\text{poly}(\ell)^{-t}$. Thus, for any yes-instance x it holds that $|\text{ACC}_x(\lambda)| = 2^\ell$, whereas for any no-instance x it holds that $|\text{ACC}_x(\lambda)| \leq \mu \cdot 2^\ell$. Indeed, the gap between the set sizes is huge, and it will be preserved as long as we lose at most a factor of $\text{poly}(\ell)$ per each round. The key observations is that, for any partial transcript $\gamma = (\alpha_1, \beta_1, ..., \alpha_{i-1}, \beta_{i-1})$, it holds that

$$|\text{ACC}_x(\gamma)| = \sum_\alpha |\text{ACC}_x(\gamma, \alpha)|, \tag{1}$$

whereas $|\text{ACC}_x(\gamma, \alpha)| = \max_\beta\{|\text{ACC}_x(\gamma, \alpha, \beta)|\}$. Clearly, we can prove that $|\text{ACC}_x(\gamma, \alpha)|$ is big by providing an adequate β and proving that $|\text{ACC}_x(\gamma, \alpha, \beta)|$ is big. Likewise, proving that $|\text{ACC}_x(\gamma)|$ is big reduces to proving that the sum $\sum_\alpha |\text{ACC}_x(\gamma, \alpha)|$ is big. The problem is that this sum may contain exponentially many terms, and so we cannot even afford asking for the value of each of these terms.[5] As hinted in §1.1, we may cluster these terms into ℓ clusters, such that the j^{th} cluster contains sets of cardinality approximately 2^j (i.e., α's such that $2^j \leq |\text{ACC}_x(\gamma, \alpha)| < 2^{j+1}$). One of these clusters must account for a $1/2\ell$ fraction of the claimed size of $|\text{ACC}_x(\gamma)|$, and so we focus on this cluster; that is, the prover we construct will identify a suitable j and prove that there are at least $N = |\text{ACC}_x(\gamma)|/(2\ell \cdot 2^{j+1})$ sets (i.e., the $\text{ACC}_x(\gamma, \alpha)$ sets) each of size at least 2^j. Note that this establishes that $|\text{ACC}_x(\gamma)|$ is bigger than $N \cdot 2^j = |\text{ACC}_x(\gamma)|/O(\ell)$, which means that we lost a factor of $O(\ell)$ of the size of $\text{ACC}_x(\gamma)$. But as stated before, we may afford such a lost.

Before we turn to the actual protocol, let us discuss the method of proving that that there are at least N sets (i.e., $\text{ACC}_x(\gamma, \alpha)$'s) each of size at least 2^j. This claim is proved by employing the

[4] We note if the prover takes the first move in (P, V) then its first message can be emulated with no cost (in the number of rounds).

[5] Furthermore, we cannot afford verifying more than a single claim regarding the value of one of these terms, because examining at least two values per round will yield an exponential blow-up (i.e., time complexity that is exponential in the number of rounds).

random selection protocol (with size parameter set to N) with the goal of selecting such a set (or rather its index α). If indeed N such sets exists then the first property of the protocol guarantees that such a set is always chosen, and we will proceed to the next iteration with this set, which has size at least 2^j (and so we should be able to establish a corresponding lower-bound there). Thus,, entering the current iteration with a valid claim, we proceed to the next iteration with a new valid claim. On the other hand, suppose that $|\mathrm{ACC}_x(\gamma)| \ll N \cdot 2^j$. Then, the second property of the protocol guarantees that, with probability at least $1-(1/3t)$, the selected α is such that $|\mathrm{ACC}_x(\gamma,\alpha)| < \mathrm{poly}(\ell) \cdot |\mathrm{ACC}_x(\gamma)|/N \ll 2^j$, whereas at the next iteration we will need to prove that the selected set has size at least 2^j. Thus, entering the current iteration with a false claim that is wrong by a factor F, with probability at least $1-(1/3t)$, we proceed to the next iteration with a claim that is wrong by a factor of at least $F/\mathrm{poly}(\ell)$.

We note that, although the foregoing motivational discussion refers to proving lower-bounds on various set sizes, the actual implementation refers to randomly selecting elements in such sets. If the sets are smaller than claimed, the selected elements are likely to reside outside these sets, which will be eventually detected.

Construction 3 (the actual protocol). *On common input x, the $2t$-round interaction of P and V is* "quasi-emulated" *in t iterations, where $t = t(|x|)$. The i^{th} iteration starts with a partial transcript $\gamma_{i-1} = (\alpha_1,\beta_1,...,\alpha_{i-1},\beta_{i-1})$ and a claimed bound M_{i-1}, where in the first iteration γ_0 is the empty sequence and $M_0 = 2^\ell$. The i^{th} iteration proceeds as follows.*

1. *The prover determines an index j such that the cluster $C_j = \{\alpha : 2^j \leq |\mathrm{ACC}_x(\gamma_{i-1},\alpha)| < 2^{j+1}\}$ has size at least $N \stackrel{\text{def}}{=} M_{i-1}/(2^{j+2}\ell)$, and sends j to the verifier. Note that if $|\mathrm{ACC}_x(\gamma_{i-1})| \geq M_{i-1}$ then such a j exists.*

2. *The prover invokes the random selection protocol with size parameter N in order to select $\alpha \in C_j$, where for simplicity we assume that $C_j \subseteq \{0,1\}^\ell$. Recall that this public-coin protocol involves three messages with the first and last message being sent by the prover. Let use denote the outcome of this protocol by α_i.*

3. *The prover determines β_i such that $\mathrm{ACC}_x(\gamma_{i-1},\alpha_i,\beta_i) = \mathrm{ACC}_x(\gamma_{i-1},\alpha_i)$ and sends β_i to the verifier.*

   Towards the next iteration $M_i \leftarrow 2^j$ and $\gamma_i = (\alpha_1,\beta_1,...,\alpha_i,\beta_i) \equiv (\gamma_{i-1},\alpha_i,\beta_i)$.

After the last iteration,[6] *the prover invokes the random selection protocol with size parameter $N = M_t$ in order to select $r \in \mathrm{ACC}_x(\alpha_1,\beta_1,...,\alpha_t,\beta_t)$. Upon obtaining this r, the verifier accepts if and only if $V(x,r,\beta_1,...,\beta_t) = 1$ and for every $i = 1,...,t$ it holds that $\alpha_i = V(x,r,\beta_1,...,\beta_{i-1})$, where the α_i's and β_i's are as determined in the aforementioned iterations.*

Note that the three steps of each iteration involve a single message by the public-coin verifier, and thus the foregoing protocol can be implemented using $2t+3$ messages.

Clearly, if x is a yes-instance then the prover can make the verifier accept with probability one (because an adequately large cluster exists at each iteration, and the random selection protocol guarantees that the selected α_i will reside in this cluster). Thus, at the last invocation of the random selection protocol, the verifier always obtains $r \in \mathrm{ACC}_x(\gamma_t)$ and accepts. On the other

[6]Alternatively, we may modify (P,V) by adding a last V-message in which V sends its internal coin tosses (i.e., r). In this case, the additional invocation of the random selection protocol occurs as a special case of handling the added $t+1^{\text{st}}$ iteration.

hand, if x is a no-instance then by using the low soundness error of (P, V) we can establish the soundness of Construction 3. This is proved in the *following claim, which refers to a polynomial p that is sufficiently large.*

Claim 4 *Suppose that* $|\mathrm{ACC}_x(\lambda)| < \delta^{t+1} \cdot 2^\ell$, *where* $\delta = 1/p(\ell)$. *Then, the verifier of Construction 3 accepts x with probability smaller than* $1/2$.

Proof Sketch: We first prove that, for every $i = 1, ..., t$, if $|\mathrm{ACC}_x(\gamma_{i-1})| < \delta^{t+1-(i-1)} \cdot M_{i-1}$ then, with probability at least $1-(1/3t)$, it holds that $|\mathrm{ACC}_x(\gamma_i)| < \delta^{t+1-i} \cdot M_i$. Let j be the value selected by the prover in Step 1 (of iteration i), and define $S' = \{\alpha : |\mathrm{ACC}_x(\gamma_{i-1}, \alpha)| \geq \delta^{t+1-i} \cdot 2^j\}$. Then $|S'| \cdot \delta^{t+1-i} 2^j < \delta^{t+1-(i-1)} \cdot M_{i-1}$, and so $|S'| < \delta \cdot (M_{i-1}/2^j) = 4\ell\delta \cdot N$, where $N = M_{i-1}/(2^{j+2}\ell)$ is as used in Step 2 (of this iteration). By the second property of the random selection protocol it follows that $\mathsf{Pr}[\alpha_i \in S] \leq \mathrm{poly}(\ell) \cdot \delta = \mathrm{poly}(\ell)/p(\ell)$, which is smaller than $1/3t$ provided that the aforementioned polynomial p is sufficiently large. Thus, with probability at least $1 - (1/3t)$, it holds that $|\mathrm{ACC}_x(\gamma_{i-1}, \alpha_i)| < \delta^{t+1-i} \cdot 2^j$. The i^{th} claim follows by recalling that $M_i = 2^j$ (in Step 3) and that for every β it holds that $|\mathrm{ACC}_x(\gamma_{i-1}, \alpha_i, \beta)| \leq |\mathrm{ACC}_x(\gamma_{i-1}, \alpha_i)|$.

Recalling that $|\mathrm{ACC}_x(\gamma_0)| < \delta^{t+1} \cdot M_0$, with probability at least $2/3$, we have $|\mathrm{ACC}_x(\gamma_t)| < \delta \cdot M_t$. In this case, the random selection protocol produces an element of $\mathrm{ACC}_x(\gamma_t)$ with probability at most $1/6$, and the verifier rejects otherwise (because the conditions that the verifier checks regarding the output r of the random selection protocol are logically equivalent to $r \in \mathrm{ACC}_x(\gamma_t)$). The main claim follows. $\square$

2 Linear speed-up for $\mathcal{AM}$

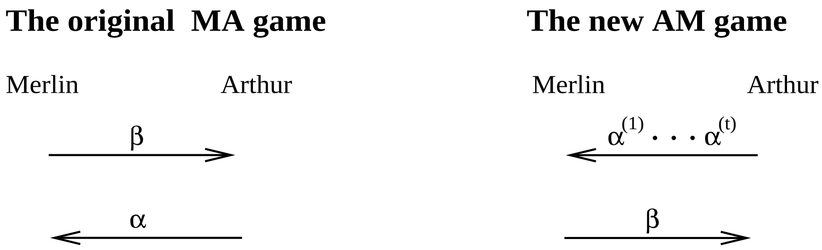
In this section we prove Theorem 2. Our proof differs from the original proof of Babai and Moran [1] in the way we analyze the basic switch (of MA to AM).

We assume that the reader is familiar with the terminology of public-coin (a.k.a Arthur-Merlin) interactive proofs, where the verifier is called Arthur and the prover is called Merlin. The *execution* of such a proof system, on any fixed common input x, can be viewed as a game (indexed by x) between an honest Arthur and powerful Merlin. These parties alternate in taking moves such that Arthur takes random moves and Merlin takes optimal moves with respect to a fixed (polynomial-time computable) predicate v_x that is *evaluated on the full transcript of the game's execution.* The value of the game is defined as the expected value of an execution of the game, where the expectation is taken over Arthur's moves (and Merlin's moves are assumed to be optimal).

Recall that $\mathcal{AM} = \mathcal{AM}(2)$ denotes a two-round system in which Arthur moves first and does not toss coins after receiving Merlin's answer, whereas $\mathcal{MA} = \mathcal{AM}(1)$ denotes a one-round system in which Merlin sends a single message and Arthur tosses additional coins after receiving this message. We may assume, without loss of generality, that all messages of Arthur are of the same length, denoted $\ell = \ell(|x|)$. Similarly, each of Merlin's messages is of length $m = m(|x|)$.

2.1 The basic switch (from MA to AM)

The basic idea is to transform an MA-game (i.e., a two-move game in which Merlin moves first and Arthur follows) into an AM-game (in which Arthur moves first and Merlin follows). Recall that, in the original game, first Merlin sends a message $\beta \in \{0,1\}^m$, then Arthur responds with a random $\alpha \in \{0,1\}^\ell$, and the value of this execution of the game is given by $v_x(\beta, \alpha) \in \{0,1\}$. In the new game (see Figure 1), the order of these moves will be switched, but to limit Merlin's potential gain from the switch we require it to provide a single answer that should “fit” several random messages of Arthur. That is, for a parameter t to be specified, first Arthur send a random sequence $(\alpha^{(1)}, ..., \alpha^{(t)}) \in \{0,1\}^{t\cdot\ell}$, then Merlin responds with a string $\beta \in \{0,1\}^m$, and the value of this transcript of the new game is defined as the conjunction of the values $v_x(\beta, \alpha^{(i)})$, for $i = 1, ..., t$. Intuitively, Merlin gets the advantage of choosing its move after seeing Arthur's move(s), but Merlin's choice must fit the t choices of Arthur's move, which intuitively leaves Merlin with little gain (if t is sufficiently large).



The value of the transcript (β, α) of the original MA-game is given by $v_x(\beta, \alpha)$, whereas the value of the transcript $((\alpha^{(1)}, ..., \alpha^{(t)}), \beta)$ of the new AM-game is given by $\wedge_{i=1}^t v_x(\beta, \alpha^{(i)})$.

Figure 1: The transformation of an MA-game into an AM-game.

Recall that the value, v'_x, of the transcript $(\overline{\alpha}, \beta)$ of the new game, where $\overline{\alpha} = (\alpha^{(1)}, ..., \alpha^{(t)})$, is defined as $\wedge_{i=1}^t v_x(\beta, \alpha^{(i)})$. Thus, the value of the new game is defined as

$$\mathsf{E}_{\overline{\alpha}}[\max_\beta\{\wedge_{i=1}^t v_x(\beta, \alpha^{(i)})\}], \tag{2}$$

which is upper-bounded by

$$\mathsf{E}_{\overline{\alpha}}\left[\max_\beta\left\{\frac{1}{t}\sum_{i=1}^t v_x(\beta, \alpha^{(i)})\right\}\right]. \tag{3}$$

Note that the upper-bound provided in Eq. (3) is tight in the case that the value of the original MA-game equals one (i.e., if x is a yes-instance), and that in this case the value of the new game is one (because in this case there exists a move β such that $v_x(\beta, \alpha) = 1$ holds for every α). However, the interesting case, where Merlin may gain something by the switch is when the value of the original MA-game is strictly smaller than one (i.e., when x is a no-instance). For this case, we upper-bound the probability that Merlin can obtain a significant gain by selecting β based on the sequence $\overline{\alpha}$ chosen by Arthur (in the new game, rather than obliviously of Arthur's move as in the original game). Specifically, we upper-bound the probability that Merlin's gain from the switch exceeds a parameter δ as follows.

$$\begin{aligned}
p_{x,\delta} &\stackrel{\text{def}}{=} \mathsf{Pr}_{(\alpha^{(1)},...,\alpha^{(t)})}\left[\max_\beta\left\{\frac{1}{t}\cdot\sum_{i=1}^t v_x(\beta, \alpha^{(i)})\right\} \geq \max_\beta\{\mathsf{E}_\alpha(v_x(\beta,\alpha))\} + \delta\right] \\
&\leq \mathsf{Pr}_{(\alpha^{(1)},...,\alpha^{(t)})}\left[\exists\beta\in\{0,1\}^m \text{ s.t. } \left|\frac{1}{t}\cdot\sum_{i=1}^t v_x(\beta, \alpha^{(i)}) - \mathsf{E}_\alpha(v_x(\beta,\alpha))\right| \geq \delta\right] \\
&\leq 2^m \cdot \exp(-\Omega(\delta^2 \cdot t)),
\end{aligned}$$

where the last inequality is due to combining the Union Bound with Chernoff Bound. Denoting by $V_x = \max_\beta\{\mathsf{E}_\alpha(v_x(\beta,\alpha))\}$ the value of the original game, we upper-bound Eq. (3) by $p_{x,\delta} + V_x + \delta$.

Using $t = O((m+k)/\delta^2)$ we have $p_{x,\delta} \leq 2^{-k}$, and thus

$$V'_x \stackrel{\rm def}{=} \mathsf{E}_{\overline{\alpha}}\left[\max_\beta\left\{\frac{1}{t}\sum_{i=1}^t v_x(\beta,\alpha^{(i)})\right\}\right] \;\leq\; \max_\beta\{\mathsf{E}_\alpha(v_x(\beta,\alpha))\} + \delta + 2^{-k}. \tag{4}$$

Needless to say, Eq. (3) is lower-bounded by V_x (since Merlin may just use the optimal move of the MA-game). In particular, using $\delta = 2^{-k} = 1/8$ and assuming that $V_x \leq 1/4$, we obtain $V'_x < 1/2$. Thus, starting from an MA proof system for some set, we obtain an AM proof system for the same set; that is, we just proved that $\mathcal{MA} \subseteq \mathcal{AM}$.

Extension. We note that the foregoing transformation as well as its analysis does not refer to the fact that $v_x(\beta,\alpha)$ is efficiently computable from (β,α). Furthermore, it does not refer to the fact that $v_x(\beta,\alpha)$ is in $\{0,1\}$. Thus, we may apply the foregoing transformation to the any two consecutive Merlin-Arthur moves in any public-coin interactive proof, provided that all subsequent moves are performed in t copies, where each copy corresponds to a different $\alpha^{(i)}$ used in the switch. That is, if the $j^{\rm th}$ move is by Merlin then we can switch the players in the j and $j+1$ moves, by letting Arthur take the $j^{\rm th}$ move, sending $(\alpha^{(1)},...,\alpha^{(t)})$, followed by Merlin's move, answering β. Subsequent moves will be played in t copies such that the $i^{\rm th}$ copy corresponds to the moves $\alpha^{(i)}$ and β. The value of the new game may increase by at most $2^{-k}+\delta < 1/4$, and so we obtain an "equivalent" game with the two steps switched. Schematically, acting on the middle MA (indicated in bold font), we can replace $[\mathrm{AM}]^{j_1}\mathrm{A}\mathbf{MA}[\mathrm{MA}]^{j_2}$ by $[\mathrm{AM}]^{j_1}\mathrm{A}\mathbf{AM}[\mathrm{MA}]^{j_2}$, which (for $j_2 \geq 1$) may be written as $[\mathrm{AM}]^{j_1}\mathrm{AAMMA}[\mathrm{MA}]^{j_2-1} = [\mathrm{AM}]^{j_1}\mathrm{AMA}[\mathrm{MA}]^{j_2-1}$ (and for $j_2 = 0$ yields $[\mathrm{AM}]^{j_1}\mathrm{AAM} = [\mathrm{AM}]^{j_1}\mathrm{AM}$). In particular, we get $\mathrm{A}[\mathrm{MA}]^{j+1} = \mathrm{A}[\mathrm{MA}]^j = \cdots = \mathrm{AMA} = \mathrm{AM}$. Thus, for any constant f, we get $\mathcal{AM}(f) = \mathcal{AM}(2)$.

We stress that the foregoing switching process can be obtained only a constant number of times, because each time we apply the switch the length of messages increases by a factor of $t = \Omega(m)$. Thus, a different approach is required to deal with a non-constant number of messages (i.e., unbounded function f).

2.2 The augmented switch (from $[MAMA]^j$ to $[AMA]^jA$)

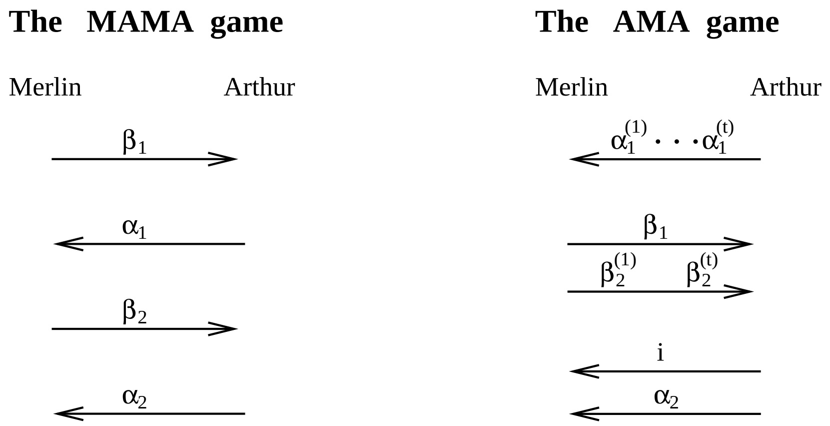
Sequential applications of the "MA-to-AM switch" allows for reducing the number of rounds by any additive constant. However, each time this switch is applied, all subsequent moves are performed t times (in parallel). That is, the "MA-to-AM switch" splits the rest of the game to t independent copies, and thus this switch cannot be performed more than a constant number of times. Fortunately, Eq. (3) suggests a way of shrinking the game back to a single copy: just have Arthur select $i \in [t]$ uniformly and have the parties continue with the $i^{\rm th}$ copy.[7] In order to avoid introducing an Arthur-Merlin alternation, the extra move of Arthur is postpone to after the following move of Merlin (see Figure 2). Schematically (indicating the action by bold font), we replace **MA**MA by **AM**MA**A**=**AMA** (rather than replacing **MA**MA by **AMA**MA and obtaining no reduction in the number of move-alternations).

The value of game obtained via the aforementioned augmented switch is given by Eq. (3), which can be written as

$$\mathsf{E}_{\alpha^{(1)},...,\alpha^{(t)}}[\max_\beta\{\mathsf{E}_{i\in[t]}(v_x(\beta,\alpha^{(i)}))\}],$$

which in turn is upper-bounded (in Eq. (4)) by $V_x + \delta + 2^{-k}$. As in §2.1, the argument applies to any two consecutive Merlin-Arthur moves in any public-coin interactive proof. Recall that in

[7]Indeed, the relaxed form of Eq. (3) plays a crucial role here (in contrast to Eq. (2)).



The value of the transcript $(\beta_1, \alpha_1, \beta_2, \alpha_2)$ *of the original MAMA-game is given by* $v_x(\beta_1, \alpha_1, \beta_2, \alpha_2)$, *whereas the value of the transcript* $((\alpha_1^{(1)}, ..., \alpha_1^{(t)}), (\beta_1, \beta_2^{(1)}, ..., \beta_2^{(t)}), (i, \alpha_2))$ *of the new AMA-game is given by* $v_x(\beta_1, \alpha_1^{(i)}, \beta_2^{(i)}, \alpha_2)$.

Figure 2: The transformation of MAMA into AMA.

order to avoid the introduction of an extra Arthur move, we actually postpone the last move of Arthur to after the next move of Merlin. Thus, we apply the augmented switch to any block of four consecutive moves that start with a Merlin move, transforming the schematic sequence MAMA into AMMAA=AMA (see Figure 2). The key point is that the moves that take place after the said block remain intact. Thus, we may apply the augmented "MA-to-AM switch" (which is actually an "MAMA-to-AMA switch") concurrently to disjoint segments of the game. Schematically, we can replace $[\mathrm{MAMA}]^j$ by $[\mathrm{AMA}]^j = A[\mathrm{MA}]^j$. Note that Merlin's gain from each such switch is upper-bounded by $\delta + 2^{-k}$, but selecting $t = \widetilde{O}(f(|x|)^2 \cdot m(|x|)) = \mathrm{poly}(|x|)$ allows to upper-bound the total gain by a constant (using, say, $\delta = 2^{-k} = 1/8f(|x|)$). We thus obtain $\mathcal{AM}(4f) \subseteq \mathcal{AM}(2f+1)$, and Theorem 2 follows.

References

[1] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes. *Journal of Computer and System Science*, Vol. 36, pp. 254–276, 1988.

[2] M. Bellare, O. Goldreich, and E. Petrank. Uniform Generation of NP-witnesses using an NP-oracle. *Information and Computation*, Vol. 163, pages 510–526, 2000.

[3] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness.* Algorithms and Combinatorics series (Vol. 17), Springer, 1999.

[4] 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*, Vol. 38, No. 3, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[5] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation,

S. Micali, ed.), pages 73–90, 1989. Extended abstract in *18th ACM Symposium on the Theory of Computing*, 1986.

[6] S. Vadhan. A Study of Statistical Zero-Knowledge Proofs. PhD Thesis, Department of Mathematics, MIT, 1999. Available from `http://www.eecs.harvard.edu/~salil/papers/phdthesis-abs.html`.