# On the Complexity of Interactive Proofs with Bounded Communication

Oded Goldreich*
Department of Computer Science
Weizmann Institute of Science
Rehovot, ISRAEL.
oded@wisdom.weizmann.ac.il

Johan Håstad
Department of Computer Science
Royal Institute of Technology
10044 Stockholm, SWEDEN.
johanh@nada.kth.se

November 2, 1997

**Abstract**

We investigate the computational complexity of languages which have interactive proof systems of bounded message complexity. In particular, denoting the length of the input by $n$, we show that

- If $L$ has an interactive proof in which the total communication is bounded by $c(n)$ bits then $L$ can be recognized by a probabilistic machine in time exponential in $O(c(n) + \log(n))$.
- If $L$ has a public-coin interactive proof in which the prover sends $c(n)$ bits then $L$ can be recognized by a probabilistic machine in time exponential in $O(c(n) \cdot \log(c(n)) + \log(n))$.
- If $L$ has an interactive proof in which the prover sends $c(n)$ bits then $L$ can be recognized by a probabilistic machine with an NP-oracle in time exponential in $O(c(n) \cdot \log(c(n)) + \log(n))$.

*Work done while being on a sabbatical leave at LCS, MIT.

# 1 Introduction

Proof systems are defined in terms of their verification procedures. The notion of a verification procedure assumes the notion of computation and furthermore the notion of efficient computation. This implicit assumption is made explicit in the definition of $\mathcal{NP}$, in which efficient computation is associated with (deterministic) polynomial-time algorithms. In light of the growing acceptability of randomized and distributed computations, it is only natural to associate the notion of efficient computation with probabilistic and interactive polynomial-time computations. This leads to the notion of an interactive proof system (cf., [9]) in which the verification procedure is interactive and randomized, rather than being non-interactive and deterministic. Intuitively, one may think of this interaction as consisting of "tricky" questions asked by the verifier, to which the prover has to reply "convincingly". The last sentence, as well as the definition, makes explicit reference to a prover, whereas a prover is only implicit in the traditional definitions of proof systems (e.g., NP-proofs).

The actual definition of *interactive proof systems* suggests probabilistic interpretations to the traditional notions of *completeness* and *soundness* associated with any proof system. Specifically, statistical soundness requires that there exists no strategy which makes the verifier accept false statements with probability greater than, say, 1/3. A further relaxation of this soundness condition is the notion of computational soundness: Here it is only required that there exists no *efficient* strategy which makes the verifier accept false statements with probability greater than 1/3. The difference between statistical soundness and computational soundness translates to a difference between interactive proof systems as defined by Goldwasser, Micali and Rackoff [9], and computationally-sound proof systems (aka argument systems) as defined by Brassard, Chaum and Crépeau [6].

A significant difference between interactive proof systems and computationally-sound proof systems has been observed in the domain of zero-knowledge. On one hand, it is widely believed that not all languages in $\mathcal{NP}$ have *perfect* zero-knowledge interactive proofs (cf., [7]).[1] In particular, the negation of this conjecture implies the collapse of the polynomial-time hierarchy (cf., [5]). On the other hand, assuming that factoring is hard, all languages in $\mathcal{NP}$ have *perfect* zero-knowledge computationally-sound proofs [6].

Our aim in this note is to point out another significant difference between interactive proof systems and computationally-sound proof systems. Specifically, we refer to the "expressive power" of the two types of proof systems when bounding their message complexity (i.e., the number of bits sent throughout the interaction). We will confront known positive results regarding the expressive power of computationally-sound proof systems of bounded message complexity with new negative results regarding the expressive power of interactive proof systems of the same message complexity.

**Computationally-sound proofs of bounded message complexity:** In 1992, Kilian demonstrated that *computationally-sound* proof systems may be able to recognize any language in $\mathcal{NP}$ while using only polylogarithmic message complexity [11]. Specifically, assuming the existence of hashing functions for which collisions cannot be found by subexponential-size circuits, Kilian showed that any language in $\mathcal{NP}$ has a *computationally-sound* proof system in which both the bi-directional message complexity and the randomness complexity are polylogarithmic. Furthermore, this proof system is in the public-coins (aka Arthur-Merlin) model of Babai [1].

[1] Perfect zero-knowledge is a strict variant of zero-knowledge. The above stated belief does *not* refer to the more relaxed notion of zero-knowledge (aka computational zero-knowledge). In fact, assuming the existence of commitment schemes, all languages in $\mathcal{NP}$ do have (computational) zero-knowledge interactive proofs [8].

**Interactive proofs of bounded message complexity:** Our first observation indicates that Kilian's result (as stated above) is unlikely for interactive proof (rather than computationally-sound) systems. It shows that if we bound the message and randomness complexity as in Kilian's result (i.e., to be polylogarithmic), then interactive proofs may exist only for languages in the class Quasi-Polynomial Time (i.e., $\mathrm{Dtime}(2^{\mathrm{poly}\log(\cdot)})$). We note that Quasi-Polynomial Time is widely believed not to contain $\mathcal{NP}$.

**Theorem 1** (interactive proofs with bounded message and randomness complexities): *Let $c(\cdot)$ be an integer function and $L \subseteq \{0,1\}^*$. Suppose that $L$ has an interactive proof system in which both the randomness and communication complexities are bounded by $c(\cdot)$. Then $L \in \mathrm{Dtime}(2^{O(c(\cdot))} \cdot \mathrm{poly}(\cdot))$.*

Theorem 1 is the starting point of our investigation. Its proof is facilitated by the fact that the hypothesis contains a bound on the randomness complexity of the verifier. However, what we consider fundamental in Kilian's result is the low message complexity. Thus, we wish to waive the extra hypothesis. In fact, waiving the bound on the randomness complexity, we obtain a very similar result

**Theorem 2** (interactive proofs with bounded message complexity): *Let $c(\cdot)$ be an integer function and $L \subseteq \{0,1\}^*$. Suppose that $L$ has an interactive proof system in which the communication complexity is bounded by $c(\cdot)$. Then $L \in \mathrm{BPtime}(2^{O(c(\cdot))} \cdot \mathrm{poly}(\cdot))$.*

Theorem 2 refers to interactive proof systems in which the bi-directional communication complexity is bounded. However, it seems that the more fundamental parameter is the uni-directional communication complexity in the prover-to-verifier direction. In fact, waiving also the bound on the verifier's message length, we obtain a similar result for the special case of public-coin (Arthur-Merlin) interactive proof systems.[2] Namely,

**Theorem 3** (public-coin interactive proofs with bounded prover-messages): *Let $c(\cdot)$ be an integer function and $L \subseteq \{0,1\}^*$. Suppose that $L$ has a public-coin interactive proof system in which the total number of bits sent by the prover is bounded by $c(\cdot)$. Then $L \in \mathrm{BPtime}(2^{O(c(\cdot)\log c(\cdot))} \cdot \mathrm{poly}(\cdot))$.*

Theorem 3 may not hold for general interactive proofs, and if it does this may be hard to establish. The reason being that supposedly hard languages such as Quadratic Non-Residuosity and Graph Non-Isomorphism have interactive proof systems in which the prover sends a single bit [9, 8]. Thus, we are currently content with a weaker result.

**Theorem 4** (interactive proofs with bounded prover-messages): *Let $c(\cdot)$ be an integer function and $L \subseteq \{0,1\}^*$. Suppose that $L$ has an interactive proof system in which the total number of bits sent by the prover is bounded by $c(\cdot)$. Then $L \in \mathrm{BPtime}(2^{O(c(\cdot)\log c(\cdot))} \cdot \mathrm{poly}(\cdot))^{NP}$.*

## 2 Formal Treatment

We assume that the reader is familiar with the basic definitions of interactive proofs as introduced by Goldwasser, Micali and Rackoff [9] and Babai [1]. Here we merely recall them, while focusing on some parameters. In particular, we use the (more liberal) two-sided error versions – this only makes our results stronger.

[2] Recall that Kilian's proof system is of the public-coin type.

### 2.1 Interactive Proof Systems and Parameters

**Definition 1** (interactive proof systems):

- *An* interactive proof system for a language $L$ *is a pair* $(P, V)$ *of interactive machines, so that* $V$ *is probabilistic polynomial-time, satisfying*
  - *Completeness: For every* $x \in L$, *the verifier* $V$ *accepts with probability* at least $\frac{2}{3}$, *after interacting with* $P$ *on common input* $x$.
  - *Soundness: For every* $x \notin L$ *and every potential prover* $P^*$, *the verifier* $V$ *accepts with probability* at most $\frac{1}{3}$, *after interacting with* $P^*$ *on common input* $x$.

  *An interactive proof system is said to be an* Arthur-Merlin game *if the verifier's message in each round consists of all coins it has tossed in this round.*

- *Let* $m$ *and* $r$ *be integer functions. The complexity class* $\mathcal{IP}(m(\cdot), r(\cdot))$ (resp., $\mathcal{AM}(m(\cdot), r(\cdot))$ *consists of languages having an interactive proof system* (resp., an Arthur-Merlin proof system) *in which, on common input* $x$, *the interaction consists of at most* $r(|x|)$ *communication rounds during which the* total *number of bits sent* from the prover to the verifier *is bounded by* $m(|x|)$.

### 2.2 Our Results

For an integer function $t$, we let BPtime$(t(\cdot))$ (resp., BPtime$(t(\cdot))^{NP}$) denote the class of languages recognizable by probabilistic $t(\cdot)$-time machines (resp., oracle machines with access to an oracle set in $\mathcal{NP}$) with error at most 1/3. Our main result is

**Proposition 5** (interactive proofs with bounded message and round complexity):

$$\begin{aligned} \mathcal{AM}(m(\cdot), r(\cdot)) &\subseteq \mathrm{BPtime}(2^{O(m(\cdot)+r(\cdot)\log r(\cdot))} \cdot \mathrm{poly}(\cdot)) \quad (1) \\ \mathcal{IP}(m(\cdot), r(\cdot)) &\subseteq \mathrm{BPtime}(2^{O(m(\cdot)+r(\cdot)\log r(\cdot))} \cdot \mathrm{poly}(\cdot))^{NP} \quad (2) \end{aligned}$$

Theorem 3 follows from Part (1) of Proposition 5, whereas Theorem 4 follows from Part (2). Theorems 1 and 2 will be proven directly before proving Proposition 5. The main ingredient in all our proofs are procedures for evaluating or approximating the value of the *game tree of a proof system.* This tree is defined next.

### 2.3 The Game Tree of a Proof System

Fixing a verifier $V$ we consider its interaction with a generic prover on any fixed common input, denoted $x$. The verifier's random choices can be thought of as corresponding to the contents of its random-tape, called the random-pad. We assume without loss of generality that $V$ sends the first message and that the prover sends the last one. In each round, $V$'s message is chosen depending on the history of the interaction so far and according to some probability distribution induced by $V$'s local random-tape. The history so far corresponds to a fixed subset of possible random-pads and the possible messages to be sent correspond to a partition of this subset. Thus, each possible message is sent with probability proportional to its part in this subset. The above description corresponds to general interactive proofs. In case of Arthur-Merlin games the situation is simpler: $V$ merely tosses a predetermined (by history) number of coins and sends the outcome to the prover.[3] As to the prover's messages, they are chosen arbitrarily (but are of length at most $\mathrm{poly}(|x|)$). The interaction goes on, for at most $\mathrm{poly}(|x|)$ rounds at which point the verifier stops outputting either *accept* or *reject*. The messages exchanged till that point are called a *transcript* of the interaction between the prover and $V$.

To simplify the exposition, we augment the transcript of the interaction by $V$'s random-pad. This way, $V$'s accept/reject decision is determined by the *augmented transcript* (and the input $x$). This convention is not needed for Arthur-Merlin games.

The interaction between the prover and $V$ on common input $x$ may be viewed as a game in which the prover's objective is to maximize the probability that $V$ accepts, and $V$'s strategy is fixed but mixed (i.e., probabilistic). It is useful to consider the corresponding *game tree*.

**Definition 2** (the game tree and its value): *Let $V$ and $x$ be fixed.*

- *The tree $T_x$: The nodes in the tree, denoted $T_x$, correspond to possible prefixes of the interaction of $V$ with an arbitrary prover. The root represents the empty interaction and is defined to be at level 0. For every $i = 0, 1...$ the edges going out from each $2i^{\text{th}}$ level node correspond to the messages $V$ may send given the history so far.* (We know that $V$ selects one of these edges/messages according to some predetermined by the node probability distribution.) *The edges going out from each $(2i+1)^{\text{st}}$ level node correspond to the messages a prover may send given the history so far.* (The prover may select an edge/message so to maximize the accepting probability of $V$.) *Nodes which correspond to an execution on which $V$ stops have as children one or more leaves, each corresponding to a possible $V$'s random-pad which is consistent with the interaction represented in the father. Thus, leaves correspond to augmented transcripts as defined above.*

- *The value of $T_x$: The value of the tree is defined bottom-up as follows. The* value of a leaf *is either 0 or 1 depending on whether $V$ accepts in the augmented transcript represented by it or not. The* value of an internal node at level $2i$ *is defined as the weighted average of the values of its children, where the weights correspond to the probabilities of the various verifier messages.* (This definition holds also for the fathers of leaves, when viewing $V$'s random-pad as an auxiliary, fictitious message sent by $V$.) *The* value of an internal node at level $2i-1$ *is defined as the maximum of the values of its children. This corresponds to the prover's strategy of trying to maximize $V$'s accepting probability. The* value of the tree *is defined as the value of its root.*

To decide if $x$ is in the language accepted by $V$, it suffices to approximate the value of the tree $T_x$ defined above. The reason being that the value of $T_x$ is a tight upper bound on the probability that $V$ accepts $x$ when interacting with any prover strategy. (The bound is achievable by an optimal prover which indeed selects each message as to maximize $V$'s acceptance probability.) Thus, the value of $T_x$ is at least $2/3$ if $x$ is in the language and at most $1/3$ otherwise. Thus, it suffices to approximate the value of $T_x$ within an additive term of $0.16 < \frac{1}{6}$. Below we present various procedures for obtaining such approximations. The more restrictions we have on the proof system, the simpler the procedure is.

[3] That is, we assume that, for every partial history of the interaction, the number of coins tossed by the verifier is predetermined (by the history of interaction so far). This assumption is more relaxed from what is typically assumed in the literature (i.e., typically it is assumed that the number of coin tosses may only depend on the round number or even is fixed for the entire interactive proof). Our results can be easily extended to the general case where the verifier may determine the number of coins tossed at each round depending on the outcome of previous coins tossed at this round.

**Comment:** It is easy to see that the optimal prover can be implemented in exp(poly($|x|$))-time, since within this time one may construct the tree $T_x$ as well as compute the value of all its nodes. In fact, it is a well-known folklore that the optimal prover can be implemented in polynomial-space.

## 2.4 Proof of Theorem 1

We start with the simplest case, where we have a bound $c \stackrel{\rm def}{=} c(|x|)$ on both the randomness and message complexity of the interactive proof on input $x$. In this case the number of nodes in $T_x$ is at most $2^{2c+1}$ (since the product of fan-out along any path from the root to a leaf is bounded by $2^c \cdot 2^c$, where the first factor is due to the actual transcript and the second to the number of possible random-pads augmenting any of these). Thus, we can construct $T_x$ in time $2^{2c+1} \cdot \text{poly}(|x|)$ and compute the value of each of its nodes (within the same time). The theorem follows. ■

## 2.5 Proof of Theorem 2

Here we only have a bound $c \stackrel{\rm def}{=} c(|x|)$ on the message complexity of the interactive proof on input $x$. In this case the number of internal nodes in $T_x$ is at most $2^{c+1}$ (since the product of fan-out along any path from the root to a father of a leaf is bounded by $2^c$.) However, $T_x$ itself may have exponentially many leaves (i.e., each last-level internal node may have exp poly($|x|$) many leaves corresponding to possible random-pads consistent with the transcript represented by this node). Our aim is to approximate the value of $T_x$ in time poly($2^c|x|$), so we cannot afford to construct $T_x$. Instead, we take a sample of $m \stackrel{\rm def}{=} \Theta(2^c c)$ random-pads, denoted $R$, and evaluate the residual tree $T_x^R$ which results from $T_x$ by omitting all nodes which are not consistent with some random-pad in $R$. (The weights in the tree $T_x^R$ are those induced by the various subsets of $R$ which are consistent with the transcript represented by each node.) We will show that, with very high probability, the value of $T_x^R$ approximates the value of $T_x$. We note that the value of $T_x^R$ can be computed in time proportional to its size (as done in previous subsection for $T_x$ itself), and that the size of $T_x^R$ is bounded by $2^c \cdot |R| = 2^{O(c)}$. Thus, the theorem follows from the following lemma.

**Lemma 6** *Let $V, x$, $m$, $T_x$ and $T_x^R$ be as above. Suppose that $r_1, ..., r_m$ are uniformly and independently chosen random-pads for $V(x)$ and let $R$ denote the multi-set $\{r_1, ..., r_m\}$. Then, with probability at least 0.99, the value of $T_x^R$ is within 0.1 of the value of $T_x$, where the probability is taken uniformly over all possible choices of $R$.*

**Proof:** It is useful to consider a "verifier"[4], denoted $V^R$, which selects its random-pad uniformly in $R$ and otherwise acts as $V$ does. Clearly, the value of $T_x^R$ represents a tight upper bound on the accepting probability of $V^R$ interacting with any prover strategy on common input $x$.

Fixing any prover strategy, denoted $P$, we consider the difference between the accepting probabilities of $V^R$ and $V$ when each interacts with $P$ on common input $x$. Denote this difference by $\Delta_P(R)$. Using Chernoff Bound (see Appendix A), with probability at most $2^{-\Omega(m)}$ over the choices of $R$, we have $|\Delta_P(R)| > 0.1$. Specifically, we consider random variables $\zeta_1, ... \zeta_m$, so that $\zeta_i = 1$ if the $i^{\rm th}$ random-pad in $R$ (i.e., $r_i$) makes $V^R$ accept $x$ when interacting with $P$. Since each $r_i$ is uniformly selected among all possible random-pads of $V$, the expected value of each $\zeta_i$ equals the probability that $V$ accepts $x$ when interacting with $P$. Since the $r_i$'s are chosen independently, the $\zeta_i$'s are independent random variables. Finally observe that the probability that $V^R$ accepts

[4] Such a "verifier" is not a standard interactive machine as defined in Definition 1, but rather one having access to an oracle $R$.

$x$ when interacting with $P$ is a random variable which equals the average of the random variables $\zeta_1, \ldots, \zeta_m$. Thus, applying Chernoff Bound indeed yields that with probability at most $2^{-2\cdot 0.1^2 \cdot m}$ over the choices of $R$, we have $|\Delta_P(R)| > 0.1$.

Noting that provers are functions from histories to next-messages, we conclude that there are at most $(2^c)^{2^c} = 2^{c2^c}$ possible provers (as both histories and next messages are of length at most $c-1$ bits). Thus, the probability that there exists a prover, $P$, such that $|\Delta_P(R)| > 0.1$ is at most

$$2^{c2^c} \cdot 2^{-2\cdot 0.1^2 \cdot m} \;<\; 2^{c2^c - (c2^c + 10)} \;<\; 0.01$$

where the probability is taken uniformly over all choices of $R$. (The first inequality uses $m = \Omega(c2^c)$.) The lemma follows. ■

## 2.6 Proof of Proposition 5

Here we only have a bound on the uni-directional communication from the prover to the verifier. Specifically, let $m \stackrel{\rm def}{=} m(|x|)$ be a bound on the total number of bits sent by the prover to $V$, on input $x$, and $r \stackrel{\rm def}{=} r(|x|)$ be a bound on the number of rounds in their interaction (on $x$). Our goal is to approximate the value of $T_x$ within complexity related to $m$ (and $r$). Thus, the approach of the previous subsection which used the assumption that $T_x$ has relatively few internal nodes will not do. Instead, we are going to construct a "representative subtree" of $T_x$ which (typically) has very few of the internal nodes of $T_x$. (We comment that the tree $T_x^R$, considered in the previous section, may have all internal nodes of $T_x$ yet few of its leaves.)

**Motivation:** The basic idea is that we do not need to consider all possible messages that $V$ may send at a particular point in the interaction. Considering a random sample of these messages should suffice, since with very high probability the average accepting-probability over this sample provides a good approximation to the (weighted) average over all possible messages. The latter assertion holds, provided we select the sample at random according to the weights assigned to the possible messages. Note that the argument holds with respect to $V$'s messages (as these are selected by $V$ at random), but cannot be applied to the prover's responses (which are selected to maximize $V$'s accepting probability).

**Back to the actual proof:** For each even-level node in $T_x$, we select a random sample of $\Theta(m^4)$ children (representing possible $V$ messages on the partial transcript associated with this node). The sample is selected according to the weights mentioned in Definition 2 (i.e., the probabilities of the various $V$'s messages). Each sample point is selected independently of the others, and so the sample may contain several occurrences of the same node. At this point we ignore the question of how one may select such a sample. This is indeed easy if the interactive proof is of an Arthur-Merlin type, but in general this may be a hard task (and an NP-oracle will be used to carry it out).

These samples (each per even-level node) defines an *approximation tree*, denoted $A_x$, in which each odd-level node has the same children as in $T_x$, whereas each even-level node has poly$(m)$ children. The value of the approximation tree is defined recursively as in Definition 2: Specifically, the leaves of $A_x$ have the same value as in $T_x$, the value of odd-level nodes is the maximum of the value of their children, and the value of even-level nodes is the (*unweighted*) average of the values of their children. We stress that although the averages taken in the even-level nodes of $T_x$ may be weighted, the averages taken in $A_x$ are not. However these weights have their effect in the randomized construction of $A_x$ (as described above). The following lemma shows that the value of $A_x$ is a good approximation of the value of $T_x$. The lemma does not refer to the complexity of constructing $A_x$ (considered below).

**Lemma 7** (the value of $A_x$): *With probability at least* 0.99*, the value of the approximation tree $A_x$ is within* 0.1 *away from the value of the corresponding game tree $T_x$, where the probability is taken uniformly over the random choices in the construction of $A_x$.*

**Proof:** Let $s = \Theta(m^4)$ be the size of the sample used for each even-level node. We consider $r+2$ *hybrid trees*, denoted $H_0, ..., H_{r+1}$, so that $H_i$ consists of the first $2i+1$ levels of $A_x$ and the rest of the levels taken from $T_x$. That is, each $2i^{\text{th}}$ level node of $H_i$ is the root of the $T_x$-subtree rooted at the corresponding node in $T_x$. Note that $H_0 \equiv T_x$ and $H_{r+1} \equiv A_x$. The value of $H_i$ is defined in the natural manner; that is, the values of nodes at level below $2i$ are defined as in $T_x$ (the corresponding edges going out of these even-level nodes have weights as in $T_x$), and the value of nodes in levels $2i-1$ and less are defined as in $A_x$. We will show that for every $i = 0, ..., r$, with probability at least $1 - \frac{0.01}{r+1}$, the values of $H_i$ and $H_{i+1}$ are within $\frac{0.1}{r+1}$ of one another.

Let us fix $i$ and consider any $2i^{\text{th}}$ level node in $H_i$, denoted $f$. Denote the children of this node (in $H_i$) by $c_1, ..., c_t$, and the weights associated with the edges leading to them by $w_1, ..., w_t$. Denote the value of $c_j$ in $H_i$ by $\mathtt{val}_i(j)$. Then, by definition of values in $H_i$, the value of $f$ in $H_i$ is $\sum_{j=1}^t w_j \cdot \mathtt{val}_i(j)$ (as in $T_x$). We may view $H_{i+1}$ as generated from $H_i$ by taking a sample of $s$ children of each $2i^{\text{th}}$ level node in $H_i$. The children of the node corresponding to $f$ in $H_{i+1}$ are selected among the nodes corresponding to the $c_j$'s according to the weights $w_j$'s. We represent these $s$ choices by the random variables $\gamma_1, ..., \gamma_s$ distributed in $\{1, ..., t\}$. Note that $\text{Prob}(\gamma_k = j) = w_j$, for every $j = 1, ..., t$ and $k = 1, ..., s$. As a function of each $\gamma_k$, we consider the random variable $\zeta_k \stackrel{\text{def}}{=} \mathtt{val}_i(\gamma_k)$. The expected value of each $\zeta_k$ equals $\sum_{j=1}^t w_j \cdot \mathtt{val}_i(j)$. Thus, the value of the node corresponding to $f$ in $H_{i+1}$ is a random variable which is the sum of $s = \Theta(m^4)$ independent random variables (i.e., the $\zeta_k$'s). Applying Chernoff bound, we observe that with probability at least $1 - 2\exp(-\frac{2s}{(100(r+1))^2}) > 1 - 2^{-m^2}$ the value of this node is within $\frac{0.1}{r+1}$ of its expected value (i.e., $\sum_{j=1}^t w_j \cdot \mathtt{val}_i(j)$). Since the number of $2i^{\text{th}}$ level node in $H_i$ is at most $2^m \cdot s^i = O(2^m \cdot m^{4r}) < \frac{0.01}{r+1} \cdot 2^{m^2}$, we conclude that with probability at least $1 - \frac{0.01}{r+1}$, the values of all corresponding $2i^{\text{th}}$ level nodes of $H_i$ and $H_{i+1}$ are within $\frac{0.1}{r+1}$ of one another. In such a case, the values of the (roots of the) trees $H_i$ and $H_{i+1}$ are within $\frac{0.1}{r+1}$ of one another. The lemma follows. ■

**The size of $A_x$:** The total size of the approximation tree is

$$2^m \cdot \text{poly}(m)^r = \text{poly}(2^m \cdot m^r) = \text{poly}(2^m \cdot r^r)$$

where the last equality is proven as follows: In case $2^m \geq m^r$, we have $2^m \cdot m^r \leq 2^{2m} = \text{poly}(2^m \cdot r^r)$. Otherwise, we have $2^m < m^r$ and so $m < r^2$ and $2^m \cdot m^r < 2^m \cdot (r^2)^r = \text{poly}(2^m \cdot r^r)$.

**Constructing $A_x$ in case of an Arthur-Merlin verifier:** In this case it is easy to select uniformly a sample of children of any even-level node in $T_x$ (as this amount to selecting a sample of the verifier next messages which are uniformly distributed in the set of strings of a predetermined length). Thus, we can construct $A_x$ (top-down) probabilistically in time related to its size (i.e., $\text{poly}(2^m \cdot r^r)$), and compute its value (bottom-up) within this time bound. Using Lemma 7, Part (1) of Proposition 5 follows.

**Constructing $A_x$ in the general case:** In this case we use a *Uniform Generation procedure* (see Appendix B). Loosely speaking, this procedure allows to uniformly select an NP-witness for a given input in an NP-language. The procedure runs in probabilistic polynomial-time using an NP-oracle. Here we use this procedure to uniformly select a random-pad (for $V$) consistent with a given partial transcript. (Note that the set of possible pairs $(x,t)$, where $t$ is a partial transcripts for $V$ on input $x$, is an NP-language with the random-pads acting as NP-witnesses.) Thus, given any even-level node in $T_x$ (partial transcript), we can uniformly select a consistent random-pad yielding a verifier next-message according to the right distribution. Thus, given access to an NP-oracle, we can construct $A_x$ (top-down) probabilistically in time related to its size (i.e., $\text{poly}(2^m \cdot r^r)$). Once we have constructed $A_x$, we compute its value (bottom-up) as before. Using Lemma 7, Part (2) of Proposition 5 follows.

# 3 Conclusions and Open Problems

Our conclusion is that computationally-sound proof systems of low message complexity seem to be much more powerful than interactive proof systems of the same message complexity bound. We wonder whether the results of Theorems 3 and 4 can be improved. In particular,

**Open Problem 1** (relatively minor): *Can the running-time bounds of the decision procedures provided by Theorems 3 and 4 be improved?*

In particular, time bounds exponential in $c(\cdot)$ (rather than in $c(\cdot)\log c(\cdot)$)) seem a natural goal. Note that there is little hope to go below $2^{c(\cdot)/O(1)}$ time – this would imply algorithms for any NP-complete problem operating in time which is subexponential in the length of the NP-witness (as each problem in NP has a trivial interactive proof in which the prover sends an NP-witness to the verifier). On the other hand, recall that for interactive proofs with $O(c(\cdot)/\log c(\cdot))$ rounds, we do have $\text{poly}(2^{c(\cdot)})$-time decision procedures (see Proposition 5).

**Open Problem 2** : *Can the probabilistic NP-oracle machine of the conclusion of Theorem 4 be replaced by a weaker process?*

There seems to be little hope to replace the probabilistic NP-oracle machine by an ordinary probabilistic (or non-deterministic) machine (of similar time-bounds), since languages for which the hypothesis of Theorem 4 holds with $c \equiv 1$ include Quadratic Non-Residuosity (widely believed not to be in $\mathcal{BPP}$) and Graph Non-Isomorphism (not known to be in $\mathcal{NP}$). But it seems plausible that, for $c(n) = O(\log n/\log\log n)$, the class $\mathcal{IP}(c(\cdot), c(\cdot))$ is contained in $\mathcal{IP}(\text{poly}(\cdot), O(1))$ (where notations are as in § 2.2). That is, we ask whether any language having an interactive proof system in which the prover sends a total of $O(\log n/\log\log n)$ bits (but may have as many rounds), has also a constant-round interactive proof system. More generally, we ask whether $\mathcal{IP}(c(\cdot), c(\cdot))$ is contained in a generalization of *constant-round* interactive proofs in which the verifier is allowed to run for $2^{O(c(\cdot)\log c(\cdot))} \cdot \text{poly}(\cdot)$ time. On a slightly different note, how about

**Open Problem 3** : *Can one provide evidence that $\mathcal{NP}$ is* NOT *contained in $\mathcal{IP}(c(\cdot), c(\cdot))$ for small $c$? How about constant $c$?*

Clearly, such indication will have to assume that $\mathcal{NP}$ is not in $\mathcal{BPP}$. But all we know under that assumption is that $\mathcal{NP}$ is not contained in public-coin classes such as $\mathcal{AM}(\log, O(1))$ (since $\mathcal{AM}(c(\cdot), O(1)) \subseteq \text{BPtime}(2^{O(c(\cdot))}\text{poly}(\cdot))$). In the same vain, how about

**Open Problem 4** : *Can one provide evidence that* co$\mathcal{NP}$ *is* NOT *contained in* $\mathcal{IP}(c(\cdot), c(\cdot))$ *for small, non-constant, function* $c$*? How about* $c(n) = \log\log n$*?*

It is widely believed that co$\mathcal{NP}$ is not contained in $\mathcal{IP}(\mathrm{poly}, O(1))$ (or else, for example, the polynomial-time hierarchy collapses [5]). Actually, stronger evidence for co$\mathcal{NP} \not\subseteq \mathcal{IP}(O(1), O(1))$ will be of interest too.

## Acknowledgments

We are grateful to the anonymous referees for their helpful comments.

## References

[1] L. Babai. Trading Group Theory for Randomness. In *17th STOC*, pages 421–429, 1985. (Publication [2] is considered the journal version.)

[2] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes. *JCSS*, Vol. 36, pages 254–276, 1988.

[3] M. Bellare, O. Goldreich and E. Petrank. Uniform Generation of NP-witnesses using an NP-oracle. In preparation, 1997.

[4] M. Bellare and E. Petrank. Making Zero-Knowledge Provers Efficient. In *24th STOC*, pages 711–722, 1992. See [3].

[5] R. Boppana, J. Håstad, and S. Zachos. Does Co-NP Have Short Interactive Proofs? *IPL*, Vol. 25, pages 127–132, May 1987.

[6] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS*, pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.

[7] L. Fortnow, The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 327–343, 1989.

[8] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38, No. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[9] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[10] M. Jerrum, L. Valiant and V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science*, Vol. 43, pages 169–188, 1986.

[11] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th STOC*, pages 723–732, 1992.

[12] M. Sipser. A Complexity Theoretic Approach to Randomness. In *15th STOC*, pages 330–335, 1983.

[13] L. Stockmeyer. The Complexity of Approximate Counting. In *15th STOC*, pages 118–126, 1983.

# Appendix A: Chernoff Bound

**Chernoff Bound:** Let $\zeta_1, ..., \zeta_m$ be independent random variables, each ranging in $[0,1]$ and having expected value $\mu$. Then,

$$\mathrm{Prob}\left(\left|\mu - \frac{1}{m}\sum_{i=1}^{m}\zeta_i\right| > \epsilon\right) \;\leq\; 2\exp\left(-2\epsilon^2 m\right)$$

# Appendix B: The Uniform Generation Procedure

The approximation algorithm presented in §2.6 uses a uniform generation procedure for selecting a NP-witness. Such a procedure originating in [12, 13, 10] has appeared in [4]. Here we follow the recent presentation of [3].

**Definition 3** (uniform generation of NP-witnesses): *Let $R$ be an NP-witness relation associated with the NP-language $L_R \stackrel{\mathrm{def}}{=} \{x : \exists y \text{ s.t } (x,y) \in R\}$. Let $R_x \stackrel{\mathrm{def}}{=} \{y : (x,y) \in R\}$ denote the set of witness for membership of $x$ in the language. A* uniform generation procedure for *$R$ is a probabilistic machine which given $x \in L_R$, with probability at least $1 - 2^{-|x|}$, outputs some witness for $x$* (i.e., a string $y$ in $R_x$)*. Furthermore, all possible strings in $R_x$ are output with the same probability. That is, for every $y_1, y_2 \in R_x$, the probability that the procedure* (on input $x$) *outputs $y_1$ equals the probability that it outputs $y_2$.*

Clearly we cannot expect such a procedure to be weaker than $\mathcal{NP}$ itself. On the other hand, without loss of generality, we may assume that when not outputting an NP-witness the procedure outputs a special symbol (e.g., $\perp$). Note that the definition is robust with respect to the choice of the lower bound on the probability that the procedure outputs a witness: Any procedure in which this lower bound is at least $1/\mathrm{poly}(|x|)$ can be converted to a procedure as above.

**Theorem 8** ([3]): *Let $R$ and $L_R$ be as above. Then there exists a probabilistic polynomial-time oracle machine which when given oracle to $\mathcal{NP}$* (i.e., to an NP-complete language)*, constitutes a uniform generation procedure for $R$.*

For sake of self-containment we present a sketch of the proof of this theorem. The proof is slightly different from what appears in any of the previous works. We start with a high level description of the execution of the procedure on input $x \in L_R$. We assume, without loss of generality that $R_x \subseteq \{0,1\}^n$, where $n = \mathrm{poly}(|x|)$, and that $n$ is polynomial-time computable from $x$. The uniform generation procedure for $R$ works as follows:

1. The procedure finds an $i$ such that $|R_x| < 2^{i+1}$ (and with probability at least 0.9 it holds $|R_x| \geq 2^i$). In addition, in case $i > \ell \stackrel{\mathrm{def}}{=} 2\lceil\log_2 n\rceil$, the procedure also obtains (see details below) a hash function $h : \{0,1\}^n \mapsto \{0,1\}^{i-\ell}$ so that for every $\alpha \in \{0,1\}^{i-\ell}$, we have $|R_{x,h,\alpha}| < 2n^2$, where $R_{x,h,\alpha} \stackrel{\mathrm{def}}{=} \{y \in R_x : h(y) = \alpha\}$. Furthermore, with probability at least 0.9, we have $|R_{x,h,\alpha}| \geq n^2/2$, for every $\alpha \in \{0,1\}^{i-\ell}$.

2. In case $i \leq \ell$, the procedure obtains $R_x$ and stops uniformly outputting a member of $R_x$.

3. Otherwise, using $i$ and $h$ found in Step 1, the procedure uniformly selects $\alpha \in \{0,1\}^{i-\ell}$, and obtains $R_{x,h,\alpha}$. The procedure halts outputting each $e \in R_{x,h,\alpha}$ with probability $1/2n^2$, and outputting $\perp$ otherwise (i.e., with probability $1 - \frac{|R_{x,h,\alpha}|}{2n^2}$). (In particular, in case $R_{x,h,\alpha} = \emptyset$, the procedure always outputs $\perp$.)

It can be easily verified that the above yields a uniform generation procedure for $R$. The question is how to implement all of the above steps. To simplify the exposition, we assume $n = |x|$ (rather than $n = \mathrm{poly}(|x|)$), and that $n^2 = 2^\ell$.

**Checking if $|R_x| < 2n^2$:** This is done using the NP-oracle by querying about membership of $x$ in the language

$$S_1 \stackrel{\mathrm{def}}{=} \{x' : \exists y_1 < y_2 < \cdots < y_{2|x'|^2} \text{ s.t. } (x', y_1), ..., (x', y_{2|x'|^2}) \in R\}$$

**Finding "good" $i$ and $h$, in case $|R_x| \geq 2n^2$:** Recall $n = |x|$. For each $i$, we use a family of $n$-wise hashing functions mapping $n$-bit strings into $(i-\ell)$-bit strings (e.g., use polynomials of degree $n-1$ over $\mathrm{GF}(2^n)$). For $i = \log_2 |R_x|$ and $h$ uniformly selected in this family, we have (using the $n^{\text{th}}$ moment method)

$$\mathrm{Prob}_h(\exists \alpha \text{ s.t. } |R_{x,h,\alpha}| < n^2/2 \text{ or } |R_{x,h,\alpha}| > 2n^2) < 0.1$$

We may verify that $|R_{x,h,\alpha}| \leq 2n^2$ for all $\alpha$'s, by checking with the NP-oracle that $(x,h)$ in not in the language

$$S_2 \stackrel{\mathrm{def}}{=} \{(x', h') : \exists \alpha\ \exists y_1 < y_2 < \cdots < y_{2|x'|^2+1} \text{ s.t. } (x', y_j) \in R \text{ and } h'(y_j) = \alpha, \forall j\}$$

Thus, trying $i = \ell, ..., n-1$, we select for each $i$ a random $h$ and test the above condition. In case we get to $i = n$, we set $h$ to return the $(n-\ell)$-bit prefix of the argument. Thus, we surely return a pair $(i,h)$ for which the condition holds and with probability at least 0.9 this pair will also satisfy $|R_{x,h,\alpha}| \geq n^2/2$ for all $\alpha$'s.

**Obtaining $R_x$ or $R_{x,h,\alpha}$ in case they are small:** For sake of simplicity, we consider here only the case $|R_x| \leq 2n^2$. Firstly, we use the NP-oracle to determine the size of $R_x$ by testing the membership of each $(x,1), ..., (x, 1^{2n^2})$ in the set

$$S_3 \stackrel{\mathrm{def}}{=} \{(x', 1^k) : \exists y_1 < y_2 < \cdots < y_k \text{ s.t. } (x', y_1), ..., (x', y_k) \in R\}$$

Once the cardinality of $R_x$, denoted $s$, is determined (and assuming $s \neq 0$), we find the $j^{\text{th}}$ bit of the $i^{\text{th}}$ element by testing the membership of $(x, 1^s, 1^i, 1^j)$ in the set

$$S_4 \stackrel{\mathrm{def}}{=} \{(x, 1^{s'}, 1^{i'}, 1^{j'}) : \exists y_1 < y_2 < \cdots < y_{s'} \text{ s.t. } (x, y_1), ..., (x, y_{s'}) \in R \text{ and the } (j')^{\text{th}} \text{ bit of } y_{i'} \text{ is zero}\}$$