Probabilistic Proof Systems: A Primer

Probabilistic Proof Systems: A Primer

Oded Goldreich

Department of Computer Science
and Applied Mathematics
Weizmann Institute of Science
Rehovot
Israel
oded.goldreich@weizmann.ac.il

now
the essence of knowledge

Boston – Delft

Foundations and Trends® in Theoretical Computer Science

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is O. Goldreich, Probabilistic Proof Systems: A Primer, Foundation and Trends® in Theoretical Computer Science, vol 3, no 1, pp 1–91, 2007

ISBN: 978-1-60198-152-3
© 2008 O. Goldreich

Foundations and Trends® in
Theoretical Computer Science
Volume 3 Issue 1, 2007

Editorial Scope

Foundations and Trends® in Theoretical Computer Science will publish survey and tutorial articles in the following topics:

- Algorithmic game theory
- Computational algebra
- Computational aspects of combinatorics and graph theory
- Computational aspects of communication
- Computational biology
- Computational complexity
- Computational geometry
- Computational learning
- Computational Models and Complexity
- Computational Number Theory
- Cryptography and information security
- Data structures
- Database theory
- Design and analysis of algorithms
- Distributed computing
- Information retrieval
- Operations Research
- Parallel algorithms
- Quantum Computation
- Randomness in Computation

Information for Librarians
Foundations and Trends® in Theoretical Computer Science, 2007, Volume 3, 4 issues. ISSN paper version 1551-305X. ISSN online version 1551-3068. Also available as a combined paper and online subscription.

Foundations and Trends® in
Theoretical Computer Science
Vol. 3, No. 1 (2007) 1–91
© 2008 O. Goldreich
DOI: 10.1561/0400000023

Probabilistic Proof Systems: A Primer

Oded Goldreich

Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, oded.goldreich@weizmann.ac.il

Abstract

Various types of probabilistic proof systems have played a central role in the development of computer science in the last couple of decades. These proof systems deviate from the traditional concept of a proof by introducing randomization and interaction into the verification process. Probabilistic proof systems carry an error probability (which is explicitly bounded and can be decreased by repetitions), but they offer various advantages over deterministic proof systems.

This primer concentrates on three types of probabilistic proof systems: interactive proofs, zero-knowledge proofs, and Probabilistically Checkable Proofs (PCP). Surveying the basic results regarding these proof systems, we stress the essential role of randomness in each of them.

Preface

A proof is whatever convinces me.
— Shimon Even (1935–2004)

The glory attached to the creativity involved in finding proofs makes us forget that it is the less glorified process of verification that gives proofs their value. Conceptually speaking, proofs are secondary to the verification process; whereas technically speaking, proof systems are defined in terms of their verification procedures.

The notion of a verification procedure presumes the notion of computation and furthermore the notion of efficient computation. This implicit stipulation is made explicit in the definition of $\mathcal{NP}$, where efficient computation is associated with deterministic polynomial-time algorithms. However, as argued next, we can gain a lot if we are willing to take a somewhat non-traditional step and allow *probabilistic* verification procedures.

In this primer, we shall survey three types of probabilistic proof systems, called *interactive proofs*, *zero-knowledge proofs*, and *probabilistic checkable proofs*. In each of these three cases, we shall present fascinating results that cannot be obtained when considering the analogous deterministic proof systems.

Indeed, the use of *probabilistic* verification procedures is common to the three aforementioned types of proof systems. We note that the association of efficient procedures with *deterministic* polynomial-time procedures is the basis for viewing NP-proof systems as the canonical formulation of proof systems (with efficient verification procedures). Now, since the notion of efficient computation has been extended to include *probabilistic* polynomial-time procedures, it is natural to allow the use of randomization also in the context of proof verification. Furthermore, it is natural to allow also a probability of error, which means that these probabilistic verification procedures may rule by (overwhelming) statistical evidence. Needless to say, this probability of error is explicitly bounded (and can be reduced by successive application of the proof system). Let us briefly review the three aforementioned types of probabilistic proof systems.

Interactive Proofs. Randomized and interactive verification procedures, giving rise to interactive proof systems, seem much more powerful than their deterministic counterparts. In particular, such interactive proof systems exist for any set in $\mathcal{PSPACE} \supseteq \text{co}\mathcal{NP}$ (e.g., for the set of unsatisfied propositional formulae), whereas it is widely believed that some sets in co$\mathcal{NP}$ do *not* have NP-proof systems (i.e., $\mathcal{NP} \neq \text{co}\mathcal{NP}$). We stress that a "proof" in this context is not a fixed and static object, but rather a randomized (and dynamic) process in which the verifier interacts with the prover. Intuitively, one may think of this interaction as consisting of questions asked by the verifier, to which the prover has to reply convincingly.

Zero-Knowledge. Such randomized and interactive verification procedures allow for the meaningful conceptualization of zero-knowledge proofs, which are of great theoretical and practical interest (especially in cryptography). Loosely speaking, zero-knowledge proofs are interactive proofs that yield nothing (to the verifier) beyond the fact that the assertion is indeed valid. For example, a zero-knowledge proof that a certain propositional formula is satisfiable does not reveal a satisfying assignment to the formula nor any partial information regarding such an assignment (e.g., whether the first variable can assume the value `true`). Thus, the successful verification of a zero-knowledge proof exhibit an extreme contrast between being convinced of the validity of a statement and learning nothing else (while receiving such a convincing proof). It turns out that, under reasonable complexity assumptions (i.e., assuming the existence of one-way functions), every set in $\mathcal{NP}$ has a zero-knowledge proof system.

Probabilistically Checkable Proofs. NP-proofs can be efficiently transformed into a (redundant) form that offers a trade-off between the number of locations (randomly) examined in the resulting proof and the confidence in its validity. In particular, it is known that any set in $\mathcal{NP}$ has an NP-proof system that supports probabilistic verification such that the error probability decreases exponentially with the number of bits read from the alleged proof. These redundant NP-proofs are called probabilistically checkable proofs (or PCPs). In addition to their conceptually fascinating nature, PCPs are closely related to the study of the complexity of numerous natural approximation problems.

Conventions and Organization

Most results surveyed in this text hold unconditionally. However, these results are only interesting if $\mathcal{NP} \neq \mathcal{P}$.

One Important Convention. When presenting a proof system, we state all complexity bounds in terms of the length of the assertion to be proved (which is viewed as an input to the verifier). Namely, when we say "polynomial-time" we mean time that is polynomial in the length of this assertion. Indeed, as will become evident, this is *the* natural choice in all the cases that we consider. Note that this convention is consistent with the definition of NP-proof systems.

Notational Conventions. We denote by `poly` the set of all integer functions that are upper-bounded by a polynomial, and by `log` the set of all integer functions bounded by a logarithmic function (i.e., $f \in \texttt{log}$ if and only if $f(n) = O(\log n)$). All complexity measures mentioned in this section are assumed to be constructible in polynomial-time.

Organization. In Section 1, we present the basic definitions and results regarding interactive proof systems. The definition of an interactive proof system is the starting point for a discussion of zero-knowledge proofs, which is provided in Section 2. Section 3, which presents the basic definitions and results regarding probabilistically checkable proofs (PCP), can be read independently of the other sections.

The study of probabilistic proof system is part of complexity theory (cf. e.g., [27]); in fact, the current text is an abbreviated (and somewhat revised) version of [27, Sec. 9].

Contents

1

Interactive Proof Systems

In light of the growing acceptability of randomized and interactive computations, it is only natural to associate the notion of efficient computation with probabilistic and interactive polynomial-time computations. This leads naturally to the notion of an interactive proof system in which the verification procedure is interactive and randomized, rather than being non-interactive and deterministic. Thus, a "proof" in this context is not a fixed and static object, but rather a randomized (dynamic) process in which the verifier interacts with the prover. Intuitively, one may think of this interaction as consisting of questions asked by the verifier, to which the prover has to reply convincingly.

The foregoing discussion, as well as the definition provided in Section 1.2, makes explicit reference to a prover, whereas a prover is only implicit in the traditional definitions of proof systems (e.g., NP-proof systems). Before turning to the actual definition, we highlight and further discuss this issue as well as some other conceptual issues.

1.1 Motivation and Perspective

We shall discuss the various interpretations given to the notion of a proof in different human contexts, and the attitudes that underly and/or accompany these interpretations. This discussion is aimed at emphasizing that the motivation for the definition of interactive proof systems is not replacing the notion of a mathematical proof, but rather capturing other forms of proofs that are of natural interest. Specifically, we shall contrast "written proofs" with "interactive proofs," highlight the roles of the "prover" and the "verifier" in any proof, and discuss the notions of completeness and soundness which underlie any proof. (Some readers may find it useful to return to this section after reading Section 1.2.)

1.1.1 A Static Object Vs. an Interactive Process

Traditionally in mathematics, a "proof" is a *fixed* sequence consisting of statements that are either self-evident or are derived from previous statements via self-evident rules. Actually, both conceptually and technically, it is more accurate to substitute the phrase "self-evident" by the phrase "commonly agreed upon" (because, at the last account, self-evidence is a matter of common agreement). In fact, in the formal study of proofs (i.e., logic), the commonly agreed statements are called *axioms*, whereas the commonly agreed rules are referred to as *derivation rules*. We highlight *a key property of mathematical proofs: these proofs are fixed* (*static*) *objects*.

In contrast, in other areas of human activity, the notion of a "proof" has a much wider interpretation. In particular, in many settings, a proof is not a fixed object but rather a process by which the validity of an assertion is established. For example, in the context of law, withstanding a cross-examination by an opponent, who may ask tough and/or tricky questions, is considered a proof of the facts claimed by the witness. Likewise, various debates that take place in daily life have an analogous potential of establishing claims and are then perceived as proofs. This perception is quite common in philosophical and political debates, and applies even in scientific debates. Needless to say, *a key property of such debates is their interactive* ("dynamic") *nature*. Interestingly, the appealing nature of such "interactive proofs" is reflected in the fact that they are mimicked (in a rigorous manner) in some mathematical *proofs* *by contradiction*, which emulate an imaginary debate with a potential (generic) skeptic.

Another difference between mathematical proofs and various forms of "daily proofs" is that, while the former aim at certainty, the latter are intended ("only") for establishing claims *beyond any reasonable doubt*. Arguably, an explicitly bounded error probability (as present in our definition of interactive proof systems) is an *extremely strong* form of establishing a claim beyond any reasonable doubt.

We also note that, in mathematics, proofs are often considered more important than their consequence (i.e., the theorem). In contrast, in many daily situations, proofs are considered secondary (in importance) to their consequence. These conflicting attitudes are well-coupled with the difference between written proofs and "interactive" proofs: If one values the proof itself then one may insist on having it archived, whereas if one only cares about the consequence then the way in which it is reached is immaterial.

Interestingly, the foregoing set of daily attitudes (rather than the mathematical ones) will be adequate in the current text, where *proofs are viewed merely as a vehicle for the verification of the validity of claims*. (This attitude gets to an extreme in the case of zero-knowledge proofs, where we actually require that the proofs themselves be useless beyond being convincing of the validity of the claimed assertion.)

In general, we will be interested in modeling various forms of proofs that may occur in the world, focusing on proofs that can be verified by automated procedures. These verification procedures are designed to check the validity of potential proofs, and are oblivious to additional features that may appeal to humans such as beauty, insightfulness, etc. In the current section, we will consider the most general form of proof systems that still allow efficient verification.

We note that the proof systems that we study refer to mundane theorems (e.g., asserting that a *specific* propositional formula is not satisfiable or that a party sent a message as instructed by a predetermined protocol). We stress that the (meta) theorems that we shall state regarding these proof systems will be proved in the traditional mathematical sense.

1.1.2 Prover and Verifier

The wide interpretation of the notion of a proof system, which includes interactive processes of verification, calls for the explicit introduction of two interactive players, called the *prover* and the *verifier*. The verifier is the party that employs the verification procedure, which underlies the definition of any proof system, while the prover is the party that tries to convince the verifier. In the context of static (or non-interactive) proofs, the prover is the transcendental entity providing the proof, and thus in this context the prover is often not mentioned at all (when discussing the verification of alleged proofs). Still, explicitly mentioning potential provers may be beneficial even when discussing such static (non-interactive) proofs.

We highlight the "distrustful attitude" toward the prover, which underlies any proof system. If the verifier trusts the prover then no proof is needed. Hence, whenever discussing a proof system, one should envision a setting in which the verifier is not trusting the prover, and furthermore is skeptical of anything that the prover says. In such a setting the prover's goal is to convince the verifier, while the verifier should make sure that it is not fooled by the prover. (See further discussion in Section 1.1.3.) Note that the verifier is "trusted" to protect his own interests by employing the predetermined verification procedure; indeed, the asymmetry with respect to whom we trust is an artifact of our focus on the verification process (or task). In general, each party is trusted to protect his own interests (i.e., the verifier is trusted to protect its own interests), but no party is trusted to protect the interests of the other party (i.e., the prover is not trusted to protect the verifier's interest of not being fooled by the prover).

Another asymmetry between the two parties is that our discussion focuses on the complexity of the verification task and ignores (as a first approximation) the complexity of the proving task (which is only discussed in Section 1.5.1). Note that this asymmetry is reflected in the definition of NP-proof systems; that is, verification is required to be efficient, whereas for sets $\mathcal{NP} \setminus \mathcal{P}$ finding adequate proofs is infeasible. Thus, as a first approximation, we consider the question of what can be efficiently verified when interacting with an arbitrary prover (which may be infinitely powerful). Once this question is resolved, we shall also consider the complexity of the proving task (indeed, see Section 1.5.1).

1.1.3 Completeness and Soundness

Two fundamental properties of a proof system (i.e., of a verification procedure) are its *soundness* (or *validity*) and *completeness*. The soundness property asserts that the verification procedure cannot be "tricked" into accepting false statements. In other words, *soundness* captures the verifier's ability to protect itself from being convinced of false statements (no matter what the prover does in order to fool it). On the other hand, *completeness* captures the ability of some prover to convince the verifier of true statements (belonging to some predetermined set of true statements). Note that both properties are essential to the very notion of a proof system.

We note that not every set of true statements has a "reasonable" proof system in which each of these statements can be proved (while no false statement can be "proved"). This fundamental phenomenon is given a precise meaning in results such as *Gödel's Incompleteness Theorem* and Turing's theorem regarding the *undecidability of the Halting Problem*. In contrast, recall that $\mathcal{NP}$ is defined as the class of sets having proof systems that support efficient deterministic verification (of "written proofs"). This section is devoted to the study of a more liberal notion of efficient verification procedures (allowing both randomization and interaction).

1.2 Definition

Loosely speaking, an interactive proof is a "game" between a computationally bounded verifier and a computationally unbounded prover whose goal is to convince the verifier of the validity of some assertion. Specifically, the verifier employs a probabilistic polynomial-time strategy (whereas no computational restrictions apply to the prover's strategy). It is required that if the assertion holds then the verifier always accepts (i.e., when interacting with an appropriate prover strategy). On the other hand, if the assertion is false then the verifier must reject with probability at least $\frac{1}{2}$, no matter what strategy is employed by the prover. (The error probability can be reduced by running such a proof system several times.)

We formalize the interaction between parties by referring to the *strategies* that the parties employ.[1] A strategy for a party is a *function mapping the party's view of the interaction so far to a description of this party's next move*; that is, such a strategy describes (or rather prescribes) the *party's next move* (i.e., its next message or its final decision) *as a function of the common input* (i.e., the aforementioned assertion), *the party's internal coin tosses, and all messages it has received so far*. Note that this formulation presumes (implicitly) that each party records the outcomes of its past coin tosses as well as all the messages it has received, and determines its moves based on these. Thus, an interaction between two parties, employing strategies A and B, respectively, is determined by the common input, denoted x, and the randomness of both parties, denoted r_A and r_B. Assuming that A takes the first move (and B takes the last "interactive move"), the corresponding (t-round) interaction transcript (on common input x and randomness r_A and r_B) is $\alpha_1,\beta_1,\ldots,\alpha_t,\beta_t$, where $\alpha_i = A(x,r_A,\beta_1,\ldots,\beta_{i-1})$ and $\beta_i = B(x,r_B,\alpha_1,\ldots,\alpha_i)$. The corresponding final decision of A is defined as $A(x,r_A,\beta_1,\ldots,\beta_t)$.

We say that a party employs a probabilistic polynomial-time strategy if its next move can be computed in a number of steps that is *polynomial in the length of the common input*. In particular, this means that, on common input x, the strategy may only consider a polynomial in $|x|$ many messages, which are each of poly($|x|$) length.[2] Intuitively, if the other party exceeds an *a priori* (polynomial in $|x|$) upper bound on the total length of the messages that it is allowed to send, then the execution is suspended.

[1] An alternative formulation refers to the interactive machines that capture the behavior of each of the parties (see, e.g., [25, Sec. 4.2.1.1]). Such an interactive machine invokes the corresponding strategy, while handling the communication with the other party and keeping a record of all messages received so far.

[2] Needless to say, the number of internal coin tosses fed to a polynomial-time strategy must also be bounded by a polynomial in the length of x.

Definition 1.1. (Interactive Proof Systems — IP)[3]: An *interactive proof system for a set S* is a two-party game, between a *verifier* executing a *probabilistic polynomial-time strategy*, denoted V, and a *prover* that executes a (computationally unbounded) strategy, denoted P, satisfying the following two conditions:

- *Completeness*: For every $x \in S$, the verifier V always accepts after interacting with the prover P on common input x.
- *Soundness*: For every $x \notin S$ and every strategy P^*, the verifier V rejects with probability at least $\frac{1}{2}$ after interacting with P^* on common input x.

We denote by $\mathcal{IP}$ the class of sets having interactive proof systems.

The error probability (in the soundness condition) can be reduced by successive applications of the proof system. In particular, repeating the proving process for k times, reduces the probability that the verifier is fooled (i.e., accepts a false assertion) to 2^{-k}, and we can afford doing so for any $k = \text{poly}(|x|)$. Variants on the basic definition are discussed in Section 1.4.

Note that NP-proof systems are obtained as a special case of interactive proof systems by eliminating interaction and randomness (i.e., restricting the communication to be uni-directional (from the prover to the verifier) and restricting the verifier to deterministic strategies). As we shall see next, interaction may be beneficial only if the verifier is probabilistic.

The Role of Randomness. Randomness is essential to the power of interactive proofs; that is, restricting the verifier to deterministic strategies yields a class of interactive proof systems that has no advantage over the class of NP-proof systems. The reason being that, in case the verifier is deterministic, the prover can predict the verifier's part

[3] We follow the convention of specifying strategies for both the verifier and the prover. An alternative presentation only specifies the verifier's strategy, while rephrasing the completeness condition as follows: *There exists a prover strategy P such that, for every $x \in S$, the verifier V always accepts after interacting with P on common input x.*

of the interaction. Thus, the prover can just supply its own sequence of answers to the verifier's sequence of (predictable) questions, and the verifier can just check that these answers are convincing. Actually, soundness error (and not merely randomized verification) is essential to the power of interactive proof systems (i.e., their ability to reach beyond NP-proofs).

Proposition 1.2. Suppose that S has an interactive proof system (P,V) with no soundness error; that is, for every $x \notin S$ and every potential strategy P^*, the verifier V rejects with probability one after interacting with P^* on common input x. Then $S \in \mathcal{NP}$.

Reflection. The uselessness of interacting with a deterministic verifier suggests a general moral by which there is no point to interact with a party whose moves are easily predictable, because such moves can be determined without any interaction. This moral represents the prover's point of view (regarding interaction with deterministic verifiers). In contrast, even an infinitely powerful party (e.g., a prover) may gain by interacting with an unpredictable party (e.g., a randomized verifier), because this interaction may provide useful information (e.g., information regarding the verifier's questions, which in turn allows the prover to increase its probability of answering convincingly). Furthermore, from the verifier's point of view it is beneficial to interact with the prover, because the latter is computationally stronger[4] (and thus its moves may not be *easily* predictable by the verifier even in the case that they are predictable in an information theoretic sense).

1.3 The Power of Interactive Proofs

We have seen that randomness is essential to the power of interactive proof systems in the sense that without randomness interactive proofs are not more powerful than NP-proofs. Indeed, the power of interactive proof arises from the combination of randomization and interaction. We first demonstrate this point by a simple proof system for a specific

[4] Or, just possesses secret information (regarding the common input).

coNP-set that is not known to have an NP-proof system, and next prove the celebrated result $\mathcal{IP} = \mathcal{PSPACE}$, which provides stronger evidence for the belief that interactive proofs are more powerful than NP-proofs.

1.3.1 A Simple Example

> One day on Olympus, bright-eyed Athena claimed that Nectar poured from the new silver-coated jars tastes less good than Nectar poured from the older gold-decorated jars. Mighty Zeus, who was forced to introduce the new jars by the practically minded Hera, was annoyed at the claim. He ordered that Athena be served one hundred glasses of Nectar, each poured at random either from an old jar or from a new one, and that she tell the source of the drink in each glass. To everybody's surprise, wise Athena correctly identified the source of each serving, to which the Father of the Gods responded "my child, you are either right or extremely lucky." Since all gods knew that being lucky was not one of the attributes of Pallas-Athena, they all concluded that the impeccable goddess was right in her claim.

The foregoing story illustrates the main idea underlying the interactive proof for Graph Non-Isomorphism, presented in Construction 1.3. Informally, this interactive proof system is designed for proving dissimilarity of two given objects (in the foregoing story these are the two brands of Nectar, whereas in Construction 1.3 these are two non-isomorphic graphs). We note that, typically, proving similarity between objects is easy, because one can present a mapping (of one object to the other) that demonstrates this similarity. In contrast, proving dissimilarity seems harder, because in general there seems to be no succinct proof of dissimilarity (e.g., clearly, showing that a particular mapping fails does not suffice, while enumerating all possible mappings (and showing that each fails) does not yield a succinct proof). More generally, it is typically easy to prove the existence of an easily verifiable structure in a given object by merely presenting this structure, but proving the non-existence of such a structure seems hard. Formally, membership in an NP-set is proved by presenting an NP-witness, but it is not clear how to prove the non-existence of such a witness. Indeed, recall that the common belief is that co$\mathcal{NP} \neq \mathcal{NP}$.

Two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, are called isomorphic if there exists a 1-1 and onto mapping, ϕ, from the vertex set V_1 to the vertex set V_2 such that $\{u,v\} \in E_1$ if and only if $\{\phi(v), \phi(u)\} \in E_2$. This ("edge preserving") mapping ϕ, in case it exists, is called an *isomorphism* between the graphs. The following protocol specifies a way of proving that two graphs are not isomorphic, while it is not known whether such a statement can be proved via a non-interactive process (i.e., via an NP-proof system).

Construction 1.3. (Interactive Proof for Graph Non-Isomorphism):

- *Common Input*: A pair of graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.
- *Verifier's First Step* (*V1*): The verifier selects at random one of the two input graphs, and sends to the prover a random isomorphic copy of this graph. Namely, the verifier selects uniformly $\sigma \in \{1,2\}$, and a random permutation π from the set of permutations over the vertex set V_σ. The verifier constructs a graph with vertex set V_σ and edge set

$$E \stackrel{\text{def}}{=} \{\{\pi(u), \pi(v)\} : \{u,v\} \in E_\sigma\}$$

  and sends (V_σ, E) to the prover.
- *Motivating Remark*: If the input graphs are non-isomorphic, as the prover claims, then the prover should be able to distinguish (not necessarily by an efficient algorithm) isomorphic copies of one graph from isomorphic copies of the other graph. However, if the input graphs are isomorphic, then a random isomorphic copy of one graph is distributed identically to a random isomorphic copy of the other graph.
- *Prover's Step*: Upon receiving a graph, $G' = (V', E')$, from the verifier, the prover finds a $\tau \in \{1,2\}$ such that the graph G' is isomorphic to the input graph G_τ. (If both $\tau = 1,2$

satisfy the condition then τ is selected arbitrarily. In case no $\tau \in \{1,2\}$ satisfies the condition, τ is set to 0.) The prover sends τ to the verifier.

- *Verifier's Second Step* (*V2*): If the message, τ, received from the prover equals σ (chosen in Step V1) then the verifier outputs 1 (i.e., accepts the common input). Otherwise the verifier outputs 0 (i.e., rejects the common input).

The verifier's strategy in Construction 1.3 is easily implemented in probabilistic polynomial-time. We do not know of a probabilistic polynomial-time implementation of the prover's strategy, but this is not required. The motivating remark justifies the claim that Construction 1.3 constitutes an interactive proof system for the set of pairs of non-isomorphic graphs. Recall that the latter set is not known to be in $\mathcal{NP}$.

1.3.2 The Full Power of Interactive Proofs

The interactive proof system of Construction 1.3 refers to a specific coNP-set that is not known to be in $\mathcal{NP}$. It turns out that interactive proof systems are powerful enough to prove membership in *any* coNP-set (e.g., prove that a graph is not 3-colorable). Thus, assuming that $\mathcal{NP} \neq \text{co}\mathcal{NP}$, this establishes that interactive proof systems are more powerful than NP-proof systems. Furthermore, the class of sets having interactive proof systems coincides with the class of sets that can be decided using a polynomial amount of work-space.

Theorem 1.4. (The IP Theorem): $\mathcal{IP} = \mathcal{PSPACE}$.

Recall that it is widely believed that $\mathcal{NP}$ is a *proper* subset of $\mathcal{PSPACE}$. Thus, under this conjecture, interactive proofs are more powerful than NP-proofs.

Sketch of the Proof of Theorem 1.4

We first show that $\text{co}\mathcal{NP} \subseteq \mathcal{IP}$, by presenting an interactive proof system for the co$\mathcal{NP}$-complete set of unsatisfiable CNF formulae. Next we extend this proof system to obtain one for the $\mathcal{PSPACE}$-complete set of unsatisfiable Quantified Boolean Formulae. Finally, we observe that $\mathcal{IP} \subseteq \mathcal{PSPACE}$.

We show that the set of unsatisfiable CNF formulae has an interactive proof system by using algebraic methods, which are *applied to an arithmetic generalization of the said Boolean problem* (rather than to the problem itself). That is, in order to demonstrate that this Boolean problem has an interactive proof system, we first introduce an arithmetic generalization of CNF formulae, and then construct an interactive proof system for the resulting arithmetic assertion (by capitalizing on the arithmetic formulation of the assertion). Intuitively, we present an iterative process, which involves interaction between the prover and the verifier, such that in each iteration the residual claim to be established becomes simpler (i.e., contains one variable less). This iterative process seems to be enabled by the fact that the various claims refer to the arithmetic problem rather than to the original Boolean problem. (Actually, one may say that the key point is that these claims refer to a generalized problem rather than to the original one.)

The Starting Point: We prove that co$\mathcal{NP} \subseteq \mathcal{IP}$ by presenting an interactive proof system for the set of unsatisfiable CNF formulae, which is co$\mathcal{NP}$-complete. Thus, our starting point is a given Boolean CNF formula, which is claimed to be unsatisfiable.

Arithmetization of Boolean (CNF) Formulae: Given a Boolean (CNF) formula, we replace the Boolean variables by integer variables, and replace the logical operations by corresponding arithmetic operations. In particular, the Boolean values `false` and `true` are replaced by the integer values 0 and 1 (respectively), OR-clauses are replaced by sums, and the top level conjunction is replaced by a product. This translation is depicted in Figure 1.1. Note that the Boolean formula is satisfied (resp., unsatisfied) by a specific truth assignment if and only if evaluating the resulting arithmetic expression at the corresponding 0–1 assignment yields a positive (integer) value (resp., yields the value zero). Thus, the claim that the original Boolean formula is unsatisfiable translates to the claim that the summation of the resulting arithmetic expression, over all 0–1 assignments to its variables, yields the value

	BOOLEAN	ARITHMETIC
variable values	`false`, `true`	0, 1
connectives	$\neg x$, $\vee$ and $\wedge$	$1-x$, $+$ and $\cdot$
final values	`false`, `true`	0, positive

Fig. 1.1 Arithmetization of CNF formulae.

zero. We highlight two additional observations regarding the resulting arithmetic expression:

1. The arithmetic expression is a low degree polynomial over the integers; specifically, its (total) degree equals the number of clauses in the original Boolean formula.
2. For any Boolean formula, the value of the corresponding arithmetic expression (for any choice of $x_1,\ldots,x_n \in \{0,1\}$) resides within the interval $[0,v^m]$, where v is the maximum number of variables in a clause, and m is the number of clauses. Thus, summing over all 2^n possible 0–1 assignments, where $n \le vm$ is the number of variables, yields an integer value in $[0,2^n v^m]$.

Moving to a Finite Field: In general, whenever we need to check equality between two integers in $[0,M]$, it suffices to check their equality mod q, where $q > M$. The benefit is that, if q is prime then the arithmetic is now in a finite field (mod q), and so certain things are "nicer" (e.g., uniformly selecting a value). Thus, proving that a CNF formula is not satisfiable reduces to proving an equality of the following form:

$$\sum_{x_1=0,1} \cdots \sum_{x_n=0,1} \phi(x_1,\ldots,x_n) \equiv 0 \pmod{q}, \tag{1.1}$$

where ϕ is a low-degree multi-variate polynomial (and q can be represented using $O(|\phi|)$ bits). In the rest of this exposition, all arithmetic operations refer to the finite field of q elements, denoted $\mathrm{GF}(q)$.

Overview of the Actual Protocol; *Stripping Summations in Iterations*: Given a formal expression as in Equation (1.1), we strip off summations in iterations, stripping a single summation at each iteration, and instantiate the corresponding free variable as follows. At the beginning of each iteration the prover is supposed to supply the univariate polynomial representing the residual expression as a function of the (single) currently stripped variable. (By Observation 1, this is a low degree polynomial and so it has a short description.)[5] The verifier checks that the polynomial (say, p) is of low degree, and that it corresponds to the current value (say, v) being claimed (i.e., it verifies that $p(0) + p(1) \equiv v$). Next, the verifier randomly instantiates the currently free variable (i.e., it selects uniformly $r \in \mathrm{GF}(q)$), yielding a new value to be claimed for the resulting expression (i.e., the verifier computes $v \leftarrow p(r)$, and expects a proof that the residual expression equals v). The verifier sends the uniformly chosen instantiation (i.e., r) to the prover, and the parties proceed to the next iteration (which refers to the residual expression and to the new value v). At the end of the last iteration, the verifier has a closed form expression (i.e., an expression without formal summations), which can be easily checked against the claimed value.

A Single Iteration (*detailed*): The ith iteration is aimed at proving a claim of the form:

$$\sum_{x_i=0,1} \cdots \sum_{x_n=0,1} \phi(r_1,\ldots,r_{i-1},x_i,x_{i+1},\ldots,x_n) \equiv v_{i-1} \pmod{q}, \tag{1.2}$$

where $v_0 = 0$, and $r_1,\ldots,r_{i-1}$ and v_{i-1} are as determined in previous iterations. The ith iteration consists of two steps (messages): a prover step followed by a verifier step. The prover is supposed to provide the verifier with the univariate polynomial p_i that satisfies

$$p_i(z) \stackrel{\text{def}}{=} \sum_{x_{i+1}=0,1} \cdots \sum_{x_n=0,1} \phi(r_1,\ldots,r_{i-1},z,x_{i+1},\ldots,x_n) \mod q\,. \tag{1.3}$$

Note that, modulo q, the value $p_i(0) + p_i(1)$ equals the l.h.s of Equation (1.2). Denote by p'_i the actual polynomial sent by the prover (i.e., the honest prover sets $p'_i = p_i$). Then, the verifier first checks if $p'_i(0) + p'_i(1) \equiv v_{i-1} \pmod{q}$, and next uniformly selects $r_i \in \mathrm{GF}(q)$

[5] We also use Observation 2, which implies that we may use a finite field with elements having a description length that is polynomial in the length of the original Boolean formula (i.e., $\log_2 q = O(vm)$).

and sends it to the prover. Needless to say, the verifier will reject if the first check is violated. The claim to be proved in the next iteration is

$$\sum_{x_{i+1}=0,1} \cdots \sum_{x_n=0,1} \phi(r_1,\ldots,r_{i-1},r_i,x_{i+1},\ldots,x_n) \equiv v_i \pmod{q}, \qquad (1.4)$$

where $v_i \overset{\text{def}}{=} p'_i(r_i) \bmod q$ is computed by each party.

Completeness of the Protocol: When the initial claim (i.e., Equation (1.1)) holds, the prover can supply the correct polynomials (as determined in Equation (1.3)), and this will lead the verifier to always accept.

Soundness of the Protocol: It suffices to upper-bound the probability that, for a particular iteration, the entry claim (i.e., Equation (1.2)) is false while the ending claim (i.e., Equation (1.4)) is valid. Indeed, let us focus on the ith iteration, and let v_{i-1} and p_i be as in Equations (1.2) and (1.3), respectively; that is, v_{i-1} is the (wrong) value claimed at the beginning of the ith iteration and p_i is the polynomial representing the expression obtained when stripping the current variable (as in Equation (1.3)). Let $p'_i(\cdot)$ be any potential answer by the prover. We may assume, without loss of generality, that $p'_i(0) + p'_i(1) \equiv v_{i-1} \pmod{q}$ and that p'_i is of low degree (since otherwise the verifier will definitely reject). Using our hypothesis (that the entry claim of Equation (1.2) is false), we know that $p_i(0) + p_i(1) \not\equiv v_{i-1} \pmod{q}$. Thus, p'_i and p_i are different low-degree polynomials, and so they may agree on very few points (if at all). Now, if the verifier's instantiation (i.e., its choice of a random r_i) does not happen to be one of these few points (i.e., $p_i(r_i) \not\equiv p'_i(r_i) \pmod{q}$), then the ending claim (i.e., Equation (1.4)) is false too (because the new value (i.e., v_i) is set to $p'_i(r_i) \bmod q$, while the residual expression evaluates to $p_i(r_i)$).

This establishes that the set of unsatisfiable CNF formulae has an interactive proof system. Actually, a similar proof system can be used to prove that a given formula has a given number of satisfying assignments, i.e., prove membership in the ("counting") set

$$\{(\phi,k) : |\{\tau : \phi(\tau) = 1\}| = k\}. \qquad (1.5)$$

Using adequate reductions, it follows that every problem in $\#\mathcal{P}$ has an interactive proof system (i.e., for every NP-relation R, the set

$\{(x,k) : |\{y : (x,y) \in R\}| = k\}$ is in $\mathcal{IP}$). Proving that $\mathcal{PSPACE} \subseteq \mathcal{IP}$ requires a little more work, as outlined next.

Obtaining Interactive Proofs for PSPACE (the basic idea): We present an interactive proof for the set of satisfied Quantified Boolean Formulae (QBF), which is complete for $\mathcal{PSPACE}$. Recall that the number of quantifiers in such formulae is unbounded (e.g., it may be polynomially related to the length of the input), that there are both existential and universal quantifiers, and furthermore these quantifiers may alternate. In the arithmetization of these formulae, we replace existential quantifiers by summations and universal quantifiers by products. Two difficulties arise when considering the application of the foregoing protocol to the resulting arithmetic expression. First, the (integral) value of the expression (which may involve a big number of nested formal products) is only upper-bounded by a double-exponential function (in the length of the input). Second, when stripping a summation (or a product), the expression may be a polynomial of high degree (due to nested formal products that may appear in the remaining expression). For example, both phenomena occur in the following expression

$$\sum_{x=0,1} \prod_{y_1=0,1} \cdots \prod_{y_n=0,1} (x + y_n),$$

which equals $\sum_{x=0,1} x^{2^{n-1}} \cdot (1+x)^{2^{n-1}}$. The first difficulty is easy to resolve by using the fact that if two integers in $[0,M]$ are different then they must be different modulo most of the primes in the interval $[3, \text{poly}(\log M)]$. Thus, we let the verifier select a random prime q of length that is linear in the length of the original formula, and the two parties consider the arithmetic expression reduced modulo this q. The second difficulty is resolved by noting that $\mathcal{PSPACE}$ is actually reducible to a special form of (non-canonical) QBF in which no variable appears both to the left and to the right of more than one universal quantifier. It follows that when arithmetizing and stripping summations (or products) from the resulting arithmetic expression, the corresponding univariate polynomial is of low degree (i.e., at most twice the length of the original formula, where the factor of two is due to the single universal quantifier that has this variable quantified on its left and appearing on its right).

IP is Contained in PSPACE: We shall show that, for every interactive proof system, there exists an *optimal prover strategy* that can be implemented in polynomial-space, where an optimal prover strategy is one that maximizes the probability that the prescribed verifier accepts the common input. It follows that $\mathcal{IP} \subseteq \mathcal{PSPACE}$, because (for every $S \in \mathcal{IP}$) we can emulate, in polynomial space, all possible interactions of the prescribed verifier with any fixed polynomial-space prover strategy (e.g., an optimal one), and accept if and only if the majority of these interactions accept.

Proposition 1.5. Let V be a probabilistic polynomial-time (verifier) strategy. Then, there exists a polynomial-space computable (prover) strategy f that, for every x, maximizes the probability that V accepts x. That is, for every P^* and every x it holds that the probability that V accepts x after interacting with P^* is upper-bounded by the probability that V accepts x after interacting with f.

Proof Idea. The strategy f can be defined recursively. Specifically, for each partial transcript of the interaction with V, the next message of f is determined such that the probability that V accepts the common input (when the subsequent prover messages are determined by f) is maximized. □

1.4 Variants and Finer Structure: An Overview

In this section we consider several variants on the basic definition of interactive proofs as well as finer complexity measures.

1.4.1 Arthur–Merlin Games a.k.a Public-Coin Proof Systems

The verifier's messages in a general interactive proof system are determined arbitrarily (but efficiently) based on the verifier's view of the interaction so far (which includes its internal coin tosses, which without loss of generality can take place at the onset of the interaction). Thus, the verifier's past coin tosses are not necessarily revealed by the messages that it sends. In contrast, in public-coin proof systems (a.k.a

Arthur–Merlin proof systems), the verifier's messages contain the outcome of any coin that it tosses *at the current round*. Thus, these messages reveal the randomness used toward generating them (i.e., this randomness becomes public). Actually, without loss of generality, the verifier's messages can be identical to the outcome of the coins tossed at the current round (because any other string that the verifier may compute based on these coin tosses is actually determined by them).

Note that the proof systems presented in the proof of Theorem 1.4 are of the public-coin type, whereas this is not the case for the Graph Non-Isomorphism proof system (of Construction 1.3). Thus, although not all natural proof systems are of the public-coin type, by Theorem 1.4 every set having an interactive proof system also has a public-coin interactive proof system. This means that, *in the context of interactive proof systems, asking random questions is as powerful as asking clever questions*. (A stronger statement appears at the end of Section 1.4.3.)

Indeed, public-coin proof systems are a syntactically restricted type of interactive proof systems. This restriction may make the design of such systems more difficult, but potentially facilitates their analysis (and especially when the analysis refers to a generic system). Another advantage of public-coin proof systems is that the verifier's actions (except for its final decision) are oblivious of the prover's messages. This property is used in the proof of Theorem 2.6.

1.4.2 Interactive Proof Systems With Two-sided Error

In Definition 1.1 error probability is allowed in the soundness condition but not in the completeness condition. In such a case, we say that the proof system has perfect completeness (or one-sided error probability). A more general definition allows an error probability (upper-bounded by, say, 1/3) in both the completeness and the soundness conditions. Note that sets having such generalized (two-sided error) interactive proofs are also in $\mathcal{PSPACE}$, and thus (by Theorem 1.4) allowing two-sided error does not increase the power of interactive proofs. See further discussion at the end of Section 1.4.3.

1.4.3 A Hierarchy of Interactive Proof Systems

Definition 1.1 only refers to the *total* computation time of the verifier, and thus allows an arbitrary (polynomial) number of messages to be exchanged. A finer definition refers to the number of messages being exchanged (also called the number of rounds).[6]

Definition 1.6. (The Round-Complexity of Interactive Proofs):

- For an integer function m, the complexity class $\mathcal{IP}(m)$ consists of sets having an interactive proof system in which, on common input x, at most $m(|x|)$ messages are exchanged between the parties.[7]
- For a set of integer functions, M, we let $\mathcal{IP}(M) \stackrel{\text{def}}{=} \bigcup_{m \in M} \mathcal{IP}(m)$. Thus, $\mathcal{IP} = \mathcal{IP}(\texttt{poly})$.

For example, interactive proof systems in which the verifier sends a single message that is answered by a single message of the prover corresponds to $\mathcal{IP}(2)$. Clearly, $\mathcal{NP} \subseteq \mathcal{IP}(1)$, yet the inclusion may be strict because in $\mathcal{IP}(1)$ the verifier may toss coins after receiving the prover's single message. (Also note that $\mathcal{IP}(0) = \text{co}\mathcal{RP}$.)

Definition 1.6 gives rise to a natural hierarchy of interactive proof systems, where different "levels" of this hierarchy correspond to different "growth rates" of the round-complexity of these systems. The following results are known regarding this hierarchy.

- *A linear speed-up* (see [6] and [33]): For every integer function, f, such that $f(n) \geq 2$ for all n, the class $\mathcal{IP}(O(f(\cdot)))$ collapses to the class $\mathcal{IP}(f(\cdot))$. In particular, $\mathcal{IP}(O(1))$ collapses to $\mathcal{IP}(2)$.

[6] An even finer structure emerges when considering also the total length of the messages sent by the prover (see [31]).

[7] We count the total number of messages exchanged, regardless of the direction of communication. Note that, without loss of generality, the last message is sent by the prover, the penultimate message is sent by the verifier, etc.

- The class $\mathcal{IP}(2)$ contains sets that are not known to be in $\mathcal{NP}$, e.g., Graph Non-Isomorphism (see Construction 1.3). However, under plausible intractability assumptions, $\mathcal{IP}(2) = \mathcal{NP}$ (see [42]).
- If co$\mathcal{NP} \subseteq \mathcal{IP}(2)$ then the Polynomial-Time Hierarchy collapses (see [15]).

It is conjectured that co$\mathcal{NP}$ is *not* contained in $\mathcal{IP}(2)$, and consequently that interactive proofs with an unbounded number of message exchanges are more powerful than interactive proofs in which only a bounded (i.e., constant) number of messages are exchanged.[8]

The class $\mathcal{IP}(1)$, also denoted $\mathcal{MA}$, seems to be *the* "real" randomized (and yet non-interactive) version of $\mathcal{NP}$: Here the prover supplies a candidate (polynomial-size) "proof," and the verifier assesses its validity probabilistically (rather than deterministically).

The IP-hierarchy (i.e., $\mathcal{IP}(\cdot)$) equals an analogous hierarchy, denoted $\mathcal{AM}(\cdot)$, that refers to public-coin (a.k.a Arthur–Merlin) interactive proofs. That is, for every integer function f, it holds that $\mathcal{AM}(f) = \mathcal{IP}(f)$. For $f \geq 1$, it is also the case that $\mathcal{AM}(2f) = \mathcal{AM}(O(f))$; actually, the aforementioned linear speed-up for $\mathcal{IP}(\cdot)$ is established by combining the following two results:

1. *Emulating $\mathcal{IP}(\cdot)$ by $\mathcal{AM}(\cdot)$*: $\mathcal{IP}(f) \subseteq \mathcal{AM}(f+3)$ [33].
2. *Linear speed-up for $\mathcal{AM}(\cdot)$*: $\mathcal{AM}(2f+1) \subseteq \mathcal{AM}(f+1)$ [6].

In particular, $\mathcal{IP}(O(1)) = \mathcal{AM}(2)$, even if $\mathcal{AM}(2)$ is restricted such that the verifier tosses no coins after receiving the prover's message. (Note that $\mathcal{IP}(1) = \mathcal{AM}(1)$ and $\mathcal{IP}(0) = \mathcal{AM}(0)$ are trivial.) We comment that it is common to shorthand $\mathcal{AM}(2)$ by $\mathcal{AM}$, which is indeed inconsistent with the convention of using $\mathcal{IP}$ as shorthand of $\mathcal{IP}(\texttt{poly})$.

The fact that $\mathcal{IP}(O(f)) = \mathcal{IP}(f)$ is proved by establishing an analogous result for $\mathcal{AM}(\cdot)$ demonstrates the advantage of the public-coin

[8] Note that the linear speed-up cannot be applied for an unbounded number of times, because each application may increase (e.g., square) the time-complexity of verification.

setting for the study of interactive proofs. A similar phenomenon occurs when establishing that the IP-hierarchy equals an analogous two-sided error hierarchy [23].

1.4.4 Something Completely Different

We stress that although we have relaxed the requirements from the verification procedure (by allowing it to interact with the prover, toss coins, and risk some (bounded) error probability), we did not restrict the soundness of its verdict by assumptions concerning the potential prover(s). This should be contrasted with other notions of proof systems, such as computationally sound ones (see Section 1.5.2), in which the soundness of the verifier's verdict depends on assumptions concerning the potential prover(s).

1.5 On Computationally Bounded Provers: An Overview

Recall that our definition of interactive proofs (i.e., Definition 1.1) makes no reference to the computational abilities of the potential prover. This fact has two opposite consequences:

1. The completeness condition does not provide any upper bound on the complexity of the corresponding proving strategy (which convinces the verifier to accept valid assertions).
2. The soundness condition guarantees that, regardless of the computational effort spend by a cheating prover, the verifier cannot be fooled to accept invalid assertions (with probability exceeding the soundness error).

Note that providing an upper-bound on the complexity of the (prescribed) prover strategy P of a specific interactive proof system (P,V) only strengthens the claim that (P,V) is an interactive proof system for the corresponding set (of valid assertions). We stress that the prescribed prover strategy is referred to only in the completeness condition (and is irrelevant to the soundness condition). On the other hand, relaxing the definition of interactive proofs such that soundness holds only for a specific class of cheating prover strategies (rather than for all cheating prover strategies) weakens the corresponding claim. In this advanced section we consider both possibilities.

1.5.1 How Powerful Should The Prover Be?

Suppose that a set S is in $\mathcal{IP}$. This means that there exists a verifier V that can be convinced to accept any input in S but cannot be fooled to accept any input not in S (except with small probability). One may ask how powerful should a prover be such that it can convince the verifier V to accept any input in S. Note that Proposition 1.5 asserts that an optimal prover strategy (for convincing any fixed verifier V) can be implemented in polynomial-space, and we cannot expect any better for a generic set in $\mathcal{PSPACE} = \mathcal{IP}$. Still, we may seek better upper-bounds on the complexity of some prover strategy that convinces a *specific* verifier, which in turn corresponds to a specific set S. More interestingly, considering all possible verifiers that give rise to interactive proof systems for S, we wish to upper-bound the computational power that suffices for convincing any of these verifiers (to accept any input in S).

We stress that, unlike the case of computationally sound proof systems (see Section 1.5.2), we do not restrict the power of the prover in the soundness condition, but rather consider the minimum complexity of provers meeting the completeness condition. Specifically, we are interested in *relatively efficient* provers that meet the completeness condition. The term "relatively efficient prover" has been given three different interpretations, which are briefly surveyed next.

1. A prover is considered *relatively efficient* if, when given an auxiliary input (in addition to the common input in S), it works in (probabilistic) polynomial-time. Specifically, in case $S \in \mathcal{NP}$, the auxiliary input maybe an NP-proof that the common input is in the set. Still, even in this case the interactive proof need not consist of the prover sending the auxiliary input to the verifier, for example, an alternative procedure may allow the prover to be zero-knowledge (see Construction 2.4).

This interpretation is adequate and in fact crucial for applications in which such an auxiliary input is available to the otherwise polynomial-time parties. Typically, such auxiliary input is available in cryptographic applications in which parties wish to prove in (zero-knowledge) that they have correctly conducted some computation. In these cases, the NP-proof is just the transcript of the computation by which the claimed result has been generated, and thus the auxiliary input is available to the party that plays the role of the prover.

2. A prover is considered *relatively efficient* if it can be implemented by a probabilistic polynomial-time oracle machine with oracle access to the set S itself. Note that the prover in Construction 1.3 has this property.

   This interpretation generalizes the notion of self-reducibility of NP-proof systems. Recall that by self-reducibility of an NP-set (or rather of the corresponding NP-proof system) we mean that the search problem of finding an NP-witness is polynomial-time reducible to deciding membership in the set. Here we require that implementing the prover strategy (in the relevant interactive proof) be polynomial-time reducible to deciding membership in the set.

3. A prover is considered *relatively efficient* if it can be implemented by a probabilistic machine that runs in time that is polynomial in the deterministic complexity of the set. This interpretation relates the time-complexity of convincing a “lazy person” (i.e., a verifier) to the time-complexity of determining the truth (i.e., deciding membership in the set).

   Hence, in contrast to the first interpretation, which is adequate in settings where assertions are generated along with their NP-proofs, the current interpretation is adequate in settings in which the prover is given only the assertion and has to test its validity by itself (before trying to convince a lazy verifier of this claim).

1.5.2 Computational Soundness

Relaxing the soundness condition such that it only refers to relatively efficient ways of trying to fool the verifier (rather than to all possible ways) yields a fundamentally different notion of a proof system. The verifier's verdict in such a system is not absolutely sound, but is rather sound *provided that the potential cheating prover does not exceed the presumed complexity limits*. As in Section 1.5.1, the notion of "relative efficiency" can be given different interpretations, the most popular one being that the cheating prover strategy can be implemented by a (non-uniform) family of polynomial-size circuits. The latter interpretation coincides with the first interpretation used in Section 1.5.1 (i.e., a probabilistic polynomial-time strategy that is given an auxiliary input (of polynomial length)). Specifically, in this case, the soundness condition is replaced by the following computational soundness condition that asserts that it is infeasible to fool the verifier into accepting false statements. Formally:

> For every prover strategy that is implementable by a family of polynomial-size circuits $\{C_n\}$, and every sufficiently long $x \in \{0,1\}^* \setminus S$, the probability that V accepts x when interacting with $C_{|x|}$ is less than $1/2$.

As in case of standard soundness, the computational-soundness error can be reduced by repetitions. We warn, however, that unlike in the case of standard soundness (where both sequential and parallel repetitions will do), the computational-soundness error cannot *always* be reduced by parallel repetitions (see [9, 45]).

It is common and natural to consider proof systems in which the prover strategies considered both in the completeness and soundness conditions satisfy the same notion of relative efficiency. Protocols that satisfy these conditions with respect to the foregoing interpretation are called arguments. We mention that argument systems may be more efficient (e.g., in terms of their communication complexity) than interactive proof systems (see [39] vs. [31]).

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, "Proof verification and intractability of approximation problems," *Journal of the ACM*, vol. 45, pp. 501–555, 1998, (preliminary version in *33rd FOCS*, 1992).

[2] S. Arora and S. Safra, "Probabilistic checkable proofs: A new characterization of NP," *Journal of the ACM*, vol. 45, pp. 70–122, 1998, (preliminary version in *33rd FOCS*, 1992).

[3] L. Babai, "Trading group theory for randomness," in *17th ACM Symposium on the Theory of Computing*, pp. 421–429, 1985.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy, "Checking computations in polylogarithmic time," in *23rd ACM Symposium on the Theory of Computing*, pp. 21–31, 1991.

[5] L. Babai, L. Fortnow, and C. Lund, "Non-deterministic exponential time has two-prover interactive protocols," *Computational Complexity*, vol. 1, no. 1, pp. 3–40, 1991, (preliminary version in *31st STOC*, 1990).

[6] 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.

[7] B. Barak, *Non-Black-Box Techniques in Crypptography*. PhD thesis, Weizmann Institute of Science, 2004.

[8] M. Bellare, O. Goldreich, and M. Sudan, "Free bits, PCPs and non-approximability — towards tight results," *SIAM Journal on Computing*, vol. 27, no. 3, pp. 804–915, 1998, (extended abstract in *36th FOCS*, 1995).

[9] M. Bellare, R. Impagliazzo, and M. Naor, "Does parallel repetition lower the error in computationally sound protocols?," in *38th IEEE Symposium on Foundations of Computer Science*, pp. 374–383, 1997.

[10] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway, "Everything provable is probable in zero-knowledge," in *Crypto88*, vol. 403, pp. 37–56, Springer-Verlag Notes in Computer Science, 1990.

[11] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, "Multi-prover interactive proofs: How to remove intractability," in *20th ACM Symposium on the Theory of Computing*, pp. 113–131, 1988.

[12] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan, "Robust PCPs of proximity, shorter PCPs, and applications to coding," *SIAM Journal on Computing*, vol. 36, no. 4, pp. 889–974, 2006, (extended abstract in *36th STOC*, 2004).

[13] E. Ben-Sasson and M. Sudan, "Simple PCPs with poly-log rate and query complexity," in *37th ACM Symposium on the Theory of Computing*, pp. 266–275, 2005.

[14] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," *Journal of Computer and System Science*, vol. 47, no. 3, pp. 549–595, 1993.

[15] R. Boppana, J. Håstad, and S. Zachos, "Does Co-NP have short interactive proofs?," *Information Processing Letters*, vol. 25, pp. 127–132, May 1987.

[16] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *Journal of Computer and System Science*, vol. 37, no. 2, pp. 156–189, 1988, (preliminary version by Brassard and Crépeau in *27th FOCS*, 1986).

[17] I. Dinur, "The PCP theorem by gap amplification," in *38th ACM Symposium on the Theory of Computing*, pp. 241–250, 2006.

[18] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra, "Characterizations of NP: Towards a polynomially-small error-probability," in *31st ACM Symposium on the Theory of Computing*, pp. 29–40, 1999.

[19] I. Dinur and O. Reingold, "Assignment-testers: Towards a combinatorial proof of the PCP-Theorem," *SIAM Journal on Computing*, vol. 36, no. 4, pp. 975–1024, 2006, (extended abstract in *45th STOC*, 2004).

[20] U. Feige, S. Goldwasser, L. Lovász, and S. Safra, "On the complexity of approximating the maximum size of a clique," Unpublished manuscript, 1990.

[21] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, "Approximating clique is almost NP-complete," *Journal of the ACM*, vol. 43, pp. 268–292, 1996, (preliminary version in *32nd STOC*, 1991).

[22] L. Fortnow, J. Rompel, and M. Sipser, "On the power of multi-prover interactive protocols," in *3rd IEEE Symposium on Structure in Complexity Theory*, pp. 156–161, 1988. (See errata in *5th IEEE Symposium on Structure in Complexity Theory* pages 318–319, 1990).

[23] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos, "On completeness and soundness in interactive proof systems," in *Advances in Computing Research: A Research Annual*, (S. Micali, ed.), pp. 429–442, 1989.

[24] O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudorandomness,* Algorithms and Combinatorics Series*,* vol. 17. Springer, 1999.

[25] O. Goldreich, *Foundation of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[26] O. Goldreich, *Foundation of Cryptography: Basic Applications.* Cambridge University Press, 2004.

[27] O. Goldreich, *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[28] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game — A completeness theorem for protocols with honest majority," in *19th ACM Symposium on the Theory of Computing*, pp. 218–229, 1987.

[29] 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, pp. 691–729, 1991, (preliminary version in *27th STOC*, 1986).

[30] O. Goldreich and E. Petrank, "Quantifying knowledge complexity," *Computational Complexity*, vol. 8, pp. 50–98, 1999.

[31] O. Goldreich, S. Vadhan, and A. Wigderson, "On interactive proofs with a laconic provers," *Computational Complexity*, vol. 11, pp. 1–53, 2002.

[32] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, pp. 186–208, 1989, (preliminary version in *17th STOC*, 1985. Earlier versions date to 1982).

[33] S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," in *Advances in Computing Research: A Research Annual*, (S. Micali, ed.), pp. 73–90, 1989, (extended abstract in *18th STOC*, 1986. Extended abstract in *18th STOC*, pp. 59–68, 1986).

[34] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan, "A tight characterization of NP with 3 query PCPs," in *39th IEEE Symposium on Foundations of Computer Science*, pp. 8–17, 1998.

[35] J. Håstad, "Clique is hard to approximate within $n^{1-\epsilon}$," *Acta Mathematica*, vol. 182, pp. 105–142, 1999, (preliminary versions in *28th STOC* (1996) and *37th STOC* (1996)).

[36] J. Håstad, "Getting optimal in-approximability results," *Journal of the ACM*, vol. 48, pp. 798–859, 2001, (extended abstract in *29th STOC*, 1997).

[37] J. Håstad and S. Khot, "Query efficient PCPs with pefect completeness," in *42nd IEEE Symposium on Foundations of Computer Science*, pp. 610–619, 2001.

[38] R. Impagliazzo and M. Yung, "Direct Zero-Knowledge Computations," in *Crypto87*, vol. 293, pp. 40–51, Springer-Verlag Lecture Notes in Computer Science, 1987.

[39] J. Kilian, "A note on efficient zero-knowledge proofs and arguments," in *24th ACM Symposium on the Theory of Computing*, pp. 723–732, 1992.

[40] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM*, vol. 39, no. 4, pp. 859–868, 1992, (preliminary version in *31st STOC*, 1990).

[41] S. Micali, "Computationally sound proofs," *SIAM Journal on Computing*, vol. 30, no. 4, pp. 1253–1298, 2000, (preliminary version in *35th STOC*, 1994).

[42] P. B. Miltersen and N. V. Vinodchandran, "Derandomizing Arthur–Merlin games using hitting sets," *Computational Complexity*, vol. 14, no. 3, pp. 256–279, 2005, (preliminary version in *40th STOC*, 1999).

[43] J. Naor and M. Naor, "Small-bias probability spaces: Efficient constructions and applications," *SIAM Journal on Computing*, vol. 22, pp. 838–856, 1993, (preliminary version in *22nd STOC*, 1990).

[44] M. Nguyen, S. J. Ong, and S. Vadhan, "Statistical zero-knowledge arguments for NP from any one-way function," in *47th IEEE Symposium on Foundations of Computer Science*, pp. 3–14, 2006.
[45] K. Pietrzak and D. Wikström, "Parallel repetition of computationally sound protocols, revisited," in *4th TCC,* vol. 4392, pp. 86–102, Springer, Lecture Notes in Computer Science, 2007.
[46] R. Raz, "A parallel repetition theorem," *SIAM Journal on Computing*, vol. 27, no. 3, pp. 763–803, 1998, (extended abstract in *27th STOC*, 1995).
[47] R. Rubinfeld and M. Sudan, "Robust characterization of polynomials with applications to program testing," *SIAM Journal on Computing*, vol. 25, no. 2, pp. 252–271, 1996.
[48] A. Shamir, "IP = PSPACE," *Journal of the ACM*, vol. 39, no. 4, pp. 869–877, 1992, (preliminary version in *31st STOC*, 1990).
[49] S. Vadhan, "A study of statistical zero-knowledge proofs," PhD Thesis, Department of Mathematics, MIT, available from http://www.eecs.harvard.edu/∼salil/papers/phdthesis-abs.html, 1999.
[50] S. Vadhan, "An unconditional study of computational zero knowledge," *SIAM Journal on Computing*, vol. 36, no. 4, pp. 1160–1214, 2006, (extended abstract in *45th STOC*, 2004).
[51] U. Zwick, "Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint," in *9th SODA*, pp. 201–210, 1998.