A Taxonomy of Proof Systems*

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

September 1996

Abstract

Several alternative formulations of the concept of an efficient proof system are nowadays coexisting in our field. These systems include the classical formulation of $\mathcal{NP}$, *interactive proof systems* (giving rise to the class $\mathcal{IP}$), *computationally-sound proof systems*, and *probabilistically checkable proofs* ($\mathcal{PCP}$), which are closely related to *multi-prover interactive proofs* ($\mathcal{MIP}$). Although these notions are sometimes introduced using the same generic phrases, they are actually very different in motivation, applications and expressive power. The main objective of this essay is to try to clarify these differences.

*This is a revised version of a survey which has appeared in *Complexity Theory Retrospective II*, L.A. Hemaspaandra and A. Selman (eds.), 1996.

1 Introduction

In recent years, alternative formulations of the concept of an efficient proof system have received much attention. Not only have talks and papers concerning these systems flooded the field of theoretical computer science, but also some of these developments have reached the non-theory community and a few were even reported in non-scientific forums such as the *New York Times*. Thus, I am quite sure that the reader has heard of phrases such as "interactive proofs" and results such as $\mathcal{IP} = \mathcal{PSPACE}$.

By no means am I suggesting that the interest in the various formulations of efficient proof systems has gone out of proportion. Certainly, the notion of an efficient proof system is central to the field of computer science and I find it hard to conceive of circumstances in which one might say that it was receiving too much attention. Furthermore, the research area established by these notions has been one of the most successful and rewarding enterprises in which the theoretical computer science community has been involved. For example, zero-knowledge proofs have revolutionized the design of cryptographic protocols, and the characterization of $\mathcal{NP}$ in terms of probabilistically checkable proofs has contributed to (and, in fact, revived) the attempts to classify the complexity of approximation problems.

Except for $\mathcal{NP}$, all proof systems reviewed below are probabilistic and furthermore have a non-zero error probability. However, the error probability is explicitly bounded and can be reduced by successive applications of the proof system. In all cases, non-zero error probability is essential to the interesting properties and consequences that these probabilistic proof systems have.

Referencing convention

I have decided to proceed in a somewhat unconventional way and have decoupled the technical exposition from the story behind its evolution. These parts appear in separate sections, which can be read independently of each other. When reading the technical part, the reader should bear in mind that the references in this part are minimal (and definitely sub-standard), with the *sole objective* of referring the reader to the best source for more details. (I wish to stress that the "best source for more details" is *not* necessarily the source that deserves the most credit!) The situation is reversed in the "story part," which contains only credits, or more accurately *my evaluation* of the contribution of the various works to the development of the field.

Addendum

The current version is augmented by an open problems section.

2 A Technical Exposition

The notion of a proof is one of the more fundamental notions of our culture. In particular, it is central to science and specifically to mathematics and computer science. Yet, although people always talk of *proofs*, the fundamental issue is the *verification* procedure. Proof systems are defined by their verification procedure, and it is the verification procedure that gives them their value.

The notion of a verification procedure assumes the notion of computation and furthermore efficient computation. This implicit assumption is made explicit in the definition of $\mathcal{NP}$. It is the association of efficient computation with (deterministic) polynomial-time algorithms that yields the association of efficient proof systems with the class $\mathcal{NP}$. Namely, to prove the validity of some statement ϕ, one supplies a (relatively short) proof π, and the verification procedure consists of running a polynomial-time algorithm on input (ϕ, π).

Technical Remarks: All complexity measures mentioned in the subsequent exposition are assumed to be constructible in polynomial time. We denote by `poly` the set of all polynomials and by `log` the set of all logarithmic functions (i.e., integer functions bounded by $O(\log n)$). We adopt the standard notations $\mathcal{EXP} \stackrel{\rm def}{=} \mathrm{DTIME}(2^{\rm poly})$ and $\mathcal{NEXP} \stackrel{\rm def}{=} \mathrm{NTIME}(2^{\rm poly})$.

2.1 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 (as in $\mathcal{NP}$). A sketch[1] of the formal definition is given in Item (1) below. (We stress that no computational restrictions are placed on the prover.) Items (2) and (3) introduce additional complexity measures that can be ignored in a first reading.

Definition 1 (Interactive Proofs – IP)

1. *An* interactive proof system (ips) for a language *L is a pair (P, V) of interactive machines, so that V is a probabilistic polynomial-time machine, satisfying*

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

2. *Let m and r be integer functions. The complexity class $\mathcal{IP}(m(\cdot), r(\cdot))$ consists of languages having an interactive proof system in which, on common input x, the verifier uses at most $r(|x|)$ coin tosses and the* total number *of messages exchanged between the parties is bounded by $m(|x|)$.*

3. *Let M and R be sets of integer functions. Then,*

$$\mathcal{IP}(M, R) \stackrel{\rm def}{=} \cup_{m \in M, r \in R} \mathcal{IP}(m(\cdot), r(\cdot)).$$

Finally, $\mathcal{IP}(m(\cdot)) \stackrel{\rm def}{=} \mathcal{IP}(m(\cdot), \texttt{poly})$ and $\mathcal{IP} \stackrel{\rm def}{=} \mathcal{IP}(\texttt{poly})$.

[1] We avoid the definition of "interacting machines." This definition can be found in [48, 43].

In Item (1), we have followed the common convention of specifying both the verifier and the prover. An alternative presentation only specifies the verifier while rephrasing the completeness condition as follows:

> There exists a machine P (a prover) so that, for every $x \in L$, the verifier V always accepts after interacting with P on common input x.

The soundness condition allows for errors; that is, executions in which the verifier accepts $x \notin L$. Yet, the error is explicitly bounded by $\frac{1}{2}$. In general, one may consider the error probability (in the soundness condition) as another parameter. It is not hard to see that the error probability in interactive proofs can be reduced by independent sequential and/or parallel repetitions. Actually, this holds even for somewhat dependent parallel repetitions; see [14], which is instructive also for the simpler case of independent parallel repetitions. On the other hand, requiring zero soundness error, in interactive proof systems, restricts their existence to languages in $\mathcal{NP}$ [38].

We stress that although we have relaxed the requirements from the verification procedure, by allowing it to interact, toss coins and err with bounded probability, we did not restrict the validity of the assertions by assumptions concerning the potential prover. (This should be contrasted with later notions of proof systems, such as computationally-sound ones and multi-prover ones, in which the validity of the soundness condition depends on assumptions concerning the external proving entity.)

2.1.1 Known results

Clearly, $\mathcal{IP}(0, \texttt{poly})$ equals co$\mathcal{RP}$, whereas $\mathcal{IP}(1,0)$ equals $\mathcal{NP}$. Furthermore, $\mathcal{IP}(1, \texttt{poly})$ contains $\mathcal{BPP}$ (see [57] or [38]). Hence, $\mathcal{IP} \supseteq \mathcal{IP}(1, \texttt{poly})$ contains $\mathcal{BPP} \cup \mathcal{NP}$, whereas we currently do not know whether $\mathcal{NP}$ contains $\mathcal{BPP}$. It is also easy to see that $\mathcal{IP}(0, \texttt{log})$ collapses to $\mathcal{IP}(0,0) = \mathcal{P}$, whereas $\mathcal{IP}(\texttt{poly}, \texttt{log})$ collapses to $\mathcal{IP}(1,0) = \mathcal{NP}$. The main result concerning interactive proof systems is that they exist for any language recognizable in polynomial space. Namely,

Theorem 1 [60, 71] $\mathcal{IP} = \mathcal{PSPACE}$.[2]

Theorem 1, was established using algebraic methods. In particular, the following approach – unprecedented in complexity theory – was employed: In order to demonstrate that a particular language is in a particular class, an arithmetic generalization of the Boolean problem is presented, and (elementary) algebraic methods are applied to show that the arithmetic problem is solvable within the class. Interestingly, this technique "does not relativize." and, furthermore, yields results (e.g., $\mathcal{IP} = \mathcal{PSPACE}$) that are false relative to most oracles, providing a dramatic refutation of the "Random Oracle Hypothesis"; see [29].

Concerning the finer structure of the IP-hierarchy, the following is known:

- For every integer function, f, so that $f(n) \geq 2$ for all n, the class $\mathcal{IP}(O(f(\cdot)))$ collapses to the class $\mathcal{IP}(f(\cdot))$, and in particular $\mathcal{IP}(O(1))$ collapses to $\mathcal{IP}(2)$ [10].
- The class $\mathcal{IP}(2)$ contains languages not known to be in $\mathcal{NP}$, e.g., Graph Non-Isomorphism [43].

[2] See [60] for the general technique and [71] for its application yielding the quoted result.

- The class $\mathcal{IP}(2)$ is contained in $\mathcal{NP}$/poly (i.e., nonuniform-NP), analogously to $\mathcal{BPP} \subseteq \mathcal{P}$/poly.
- If co$\mathcal{NP} \subseteq \mathcal{IP}(2)$ then the polynomial-time hierarchy collapses [26].

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.

The IP-hierarchy (i.e., $\mathcal{IP}(\cdot)$) equals an analogous hierarchy in which the verifier is restricted to send the outcome of any coin it tosses [49]. The latter restricted proof systems are called *Arthur-Merlin games* or *public-coin interactive proofs*. In addition, aside from the zero-level and 1-level, the IP-hierarchy equals an analogous two-sided error hierarchy [38]. In the latter proof systems the completeness condition is relaxed so that the verifier is required to accept each $x \in L$ with probability at least $\frac{2}{3}$.

2.1.2 Zero-Knowledge and Knowledge Complexity

Zero-knowledge is a central notion in cryptography. Here, we only discuss its conceptual significance to the theory of proof systems. Zero-knowledge proofs are interesting as they exhibit a somewhat extreme contrast between being convinced of the validity of a statement and learning something in addition while receiving such a convincing proof. Namely, zero-knowledge proofs have the remarkable property of being convincing while yielding nothing to the verifier beyond the fact that the statement is valid.

In the formulation of the statement "(P, V) is a *zero-knowledge* proof system for the language L" one considers two probability distributions, for each input x in L:

1. The output distribution of the verifier[3] after interacting with the specified prover P on common input x.
2. The output distribution of some probabilistic polynomial-time machine (not interacting with anyone), on input x.

The basic paradigm of zero-knowledge asserts that for every distribution of type (1) there exist a "similar" distribution of type (2). The specific variants differ by the interpretation given to "similarity." The most strict interpretation, leading to *perfect zero-knowledge*, is that similarity means equality. A somewhat relaxed interpretation, leading to *almost-perfect zero-knowledge*, is that similarity means statistical closeness (i.e., negligible difference between the distributions). The most liberal interpretation, leading to the standard usage of the term zero-knowledge (and sometimes referred to as *computational zero-knowledge*), is that similarity means computational indistinguishability (i.e., failure of any efficient procedure to tell the two distributions apart).

The most important result concerning zero-knowledge is that, assuming the existence of one-way functions, each language in $\mathcal{NP}$ (and actually even in $\mathcal{IP}$) has a zero-knowledge interactive proof system; see [43, 64, 50] (and [19], respectively). This result should be contrasted with the results regarding the complexity of *almost-perfect* zero-knowledge proof systems, namely, that such proof systems exist only for languages in $\mathcal{IP}(2) \cap \text{co}\mathcal{IP}(2)$ [2, 35].[4] Also, a recent result indicates that one-way functions are essential for the existence of zero-knowledge proofs for hard languages (i.e., languages that cannot be decided in average polynomial time) [67]. Non-zero error probability is

[3] The verifier is not necessarily the one specified (i.e., V) – yet, for sake of simplicity this issue is ignored here.

[4] See also an appendix in [45] indicating an error in [35].

essential also to zero-knowledge proofs (which otherwise exist only for co$\mathcal{RP}$ [44]). Hence, besides the apparent strengthening of expressive power, interactive proof systems do offer some properties (specifically, zero-knowledge) that cannot be offered by an NP-proof system.

An extensive treatment of zero-knowledge can be found in [41]. Zero-knowledge is the lowest level of several knowledge-complexity hierarchies that quantify the "amount of knowledge revealed" by a proof system. Definitions and results concerning these hierarchies can be found in [46], [45] and [1].

2.1.3 How powerful should the ("completeness") prover be?

Assume that a language L is in $\mathcal{IP}$. This means that there is a verifier V that can be convinced to accept any input in L but cannot be convinced to accept an input not in L. One can ask how powerful should a prover be so that it can convince the verifier V to accept an input in L. More interestingly, considering all possible verifiers that give rise to an interactive proof system for L, what is the minimum power required from a prover that satisfies the completeness requirement with respect to one of these verifiers? We stress that, unlike in the case of computationally-sound proof systems (discussed below), 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 (meeting the completeness condition). The term "relatively efficient prover" has been given three different interpretations.

1st interpretation: A prover is considered *relatively efficient* if, when given an auxiliary input (in addition to the common input in L), it works in (probabilistic) polynomial time. Specifically, in the case $L \in \mathcal{NP}$, the auxiliary input may be an NP-witness that the common input is in the language. 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 [43]). 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, the auxiliary input is available in cryptographic applications in which both the input and an NP-witness for it are generated by some party who later wishes to prove (in zero-knowledge) that the input is in the language.[5] See [43].

2nd interpretation: A prover is considered *relatively efficient* if it can be implemented by a probabilistic polynomial-time oracle machine with oracle access to the language L itself. This interpretation generalizes the notion of self-reducibility of NP languages. (By self-reducibility of an NP language we mean that the search problem of finding an NP-witness is Cook-reducible to deciding membership in the language. Thus, every NP-complete language has a relatively efficient proof system.) See [16].

3rd interpretation: 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 language. This interpretation relates the difficulty of convincing a "lazy" verifier to the complexity of finding the truth alone. Hence, in contrast to the first interpretation, which is adequate in settings where NP-assertions are generated along with their NP-proofs, the current interpretation is adequate in settings in which the prover is given only an assertion and has to convince itself of the validity of the assertion (before trying to convince a lazy verifier of its validity). See [63].

[5] For example, suppose a party generates a composite number by multiplying together two primes, and that it later wishes to prove in (in zero-knowledge) that the number it has chosen indeed has this form. Then this is an NP-statement and the primes are the NP-witness for it.

2.2 MIP and PCP

In contrast to the setting of interactive proofs, where no restrictions have been placed on the prover, the two settings discussed in this section do impose restrictions on the prover. Interestingly, the ("expressive") power of the proof system is increased by these restrictions, unless $\mathcal{PSPACE} = \mathcal{NEXP}$ (in which case the systems are equally powerful). We wish to stress that there is nothing wrong in the fact that a proof system becomes more powerful once the prover is restricted. Indeed, this restricts the prover's abilities in case the input is in the language; but it also restricts the prover's ability in case the input is not in the language. Thus, in general, when restricting the power of the prover, the expressive power of the proof system can change in an arbitrary way (and in particular – stay the same). Yet, the effect of these restrictions in typically more drastic on the soundness condition than on the completeness condition (since the former employs an existential quantifier whereas the latter employs a universal quantifier).

But what is the justification for restricting the prover? One answer is that in particular applications this restriction can be imposed (on the potential provers). A second answer is that this restriction yields an alternative characterization for a fundamental complexity class (i.e., $\mathcal{NP}$) and that this alternative characterization enables one to get important results.

2.2.1 Multi-Prover Interactive Proof Systems (MIP)

In the multi-prover interactive proof setting, the prover is split into two (or more) entities and the restriction (or assumption) is that these entities cannot interact with each other. Actually, the formulation allows them to coordinate their strategies prior to interacting with the verifier[6] but it is crucial that they do not exchange messages among themselves while interacting with the verifier. It is customary to call each of these proving-entities a "prover" and hence the term "multi-prover proof systems." It turns out that two-prover systems are as powerful as multi-prover ones (even such in which the number of provers is a parameter that is polynomially related to the input length). For sake of concreteness, a definition of two-prover proof systems is given below.

Definition 2 (Two-Prover Interactive Proofs) *A* two-prover interactive proof system for a language *L is a triplet (P_1, P_2, V) of interactive machines, so that V is a probabilistic polynomial-time machine, satisfying*

- *Completeness: For every $x \in L$ the verifier V always accepts after interacting separately and concurrently with the provers P_1 and P_2, on common input x.*
- *Soundness: For every $x \notin L$ and every potential pair of provers, P_1^* and P_2^*, the verifier V rejects with probability at least $\frac{1}{2}$, after interacting separately and concurrently with the provers P_1^* and P_2^*, on common input x.*

The set of languages having two-prover proof systems is denoted by $\mathcal{MIP}$.

The two-prover model is reminiscent of the common police procedure of isolating collaborating suspects and interrogating each of them separately. A typical application in which the two-prover model may be assumed is an ATM that verifies the validity of a pair of smart-cards inserted in two isolated slots of the ATM. The advantage in using such a split system is that it enables the presentation of (perfect) zero-knowledge proof systems for any language in $\mathcal{NP}$, using no intractability assumptions [20].

[6]This is implicit in the universal quantifier used in the soundness condition.

2.2.2 Probabilistically Checkable Proofs (PCP)

When viewed in terms of an interactive proof system, the probabilistically checkable proof (PCP) setting consists of a prover that is memoryless. Namely, one can think of the prover as being an oracle and of the messages sent to it as being queries. A more appealing interpretation is to view the PCP setting as an alternative way of generalizing $\mathcal{NP}$. Instead of receiving the entire proof and conducting a deterministic polynomial-time computation (as in the case of $\mathcal{NP}$), the verifier may toss coins and query the proof only at locations of its choice. Potentially, this allows the verifier to utilize very long proofs (i.e., of super-polynomial length) or alternatively inspect very few bits of a (polynomially long) proof. The basic definition of the PCP setting is given in Item (1) below. Yet, the complexity measures introduced in Items (2) and (3) are of key importance for the subsequent discussions, and should not be ignored.

Definition 3 (Probabilistic Checkable Proofs – PCP)

1. *A* probabilistic checkable proof system (pcp) for a language *L is a probabilistic polynomial-time oracle machine (called* verifier*), denoted V, satisfying*
   - *Completeness: For every $x \in L$ there exists an oracle set π_x so that V, on input x and access to oracle π_x, always accepts x.*
   - *Soundness: For every $x \notin L$ and every oracle set π, machine V, on input x and access to oracle π, rejects x with probability at least $\frac{1}{2}$.*
2. *Let r and q be integer functions. The complexity class $\mathcal{PCP}(r(\cdot), q(\cdot))$ consists of languages having a probabilistic checkable proof system in which the verifier, on any input of length n, uses at most $r(n)$ random coins and makes at most $q(n)$ queries.*
3. *Let R and Q be sets of integer functions. Then,*
$$\mathcal{PCP}(R, Q) \stackrel{\text{def}}{=} \cup_{r\in R, q\in Q}\mathcal{PCP}(r(\cdot), q(\cdot)).$$

Note that the oracle π_x in a pcp system constitutes a proof in the standard mathematical sense. (Jumping ahead, the oracles in pcp systems characterizing $\mathcal{NP}$ have the property of being NP proofs themselves.) Yet, this oracle has the extra property of enabling a lazy verifier to toss coins, take its chances and verify the proof without reading all of it (but rather by reading a tiny portion of it).
Typical applications for probabilistically checkable proofs arise from settings in which the prover is "committed" to a single "proof" and cannot modify it depending on previous queries of the verifier. See, for example, [8, 55].

2.2.3 The expressive power of PCP

Clearly, $\mathcal{PCP}(\texttt{poly}, 0)$ equals co$\mathcal{RP}$, whereas $\mathcal{PCP}(0, \texttt{poly})$ equals $\mathcal{NP}$. It is easy to prove an upper bound on the nondeterministic time complexity of languages in the PCP hierarchy. Namely,

Proposition 1 *For every integer function $r(\cdot)$, the class $\mathcal{PCP}(r(\cdot), \texttt{poly})$ is contained in* $\mathrm{NTIME}(2^{O(r(\cdot)+\log(\cdot))})$. *Hence, $\mathcal{PCP}(\texttt{log}, \texttt{poly}) \subseteq \mathcal{NP}$.*

Proof Sketch: Observe that guessing the best oracle amounts to guessing only $2^{r(n)} \cdot \mathrm{poly}(n)$ many oracle values. □

These upper bounds turn out to be tight, but proving this is much more difficult (to say the least).

Theorem 2 [8, 32, 6, 5] *$\mathcal{NP}$ is contained in $\mathcal{PCP}(\texttt{log}, O(1))$.*[7]

Corollary 1 (The PCP characterization of NP) $\mathcal{NP} = \mathcal{PCP}(\texttt{log}, O(1))$.

I consider the proof of Theorem 2 to be one of the most complicated proofs in computer science and believe that it is very important to find a simpler proof that may be taught in an advanced complexity theory class. By adapting the proof of Theorem 2, one gets also:

Theorem 3 (Theorem 2 – Generalized): *Let $t(\cdot)$ be an integer function so that $n < t(n) < 2^{\mathrm{poly}(n)}$, for all n's. Then, the class* NTIME$(t(\cdot))$ *is contained in the class* $\mathcal{PCP}(O(\log t(\cdot)), O(1))$.

Corollary 2 $\mathcal{NEXP} = \mathcal{PCP}(\texttt{poly}, O(1))$.

Interestingly, the two complexity measures in the PCP-characterization of $\mathcal{NP}$ can be traded off, so that at the extremes we get $\mathcal{NP} = \mathcal{PCP}(\texttt{log}, O(1))$ and $\mathcal{NP} = \mathcal{PCP}(0, \texttt{poly})$, respectively.

Proposition 2 *There exist constants $\alpha, \beta > 0$ such that for every integer function $l(\cdot)$, so that $0 \leq l(n) \leq \alpha \log_2 n$,*

$$\mathcal{NP} = \mathcal{PCP}(r(\cdot), q(\cdot)),$$

where $r(n) = \alpha \log_2 n - l(n)$ and $q(n) = \beta 2^{l(n)}$.

Proof Sketch: Starting with Theorem 2, one can scan all possibilities for the $l(n)$-long prefix of the random tape of the verifier. □

Sequential repetitions can be used to reduce the error probability of pcp/mip systems. Furthermore, error reduction can be obtained at very moderate cost in the randomness complexity (cf. [15]). Parallel repetition is a much more complex matter (than in the context of interactive proof systems); see [69] (and do not get misled by an error in an early version of [36]). On the other hand, non-zero error probability is essential to the above results as otherwise one can eliminate randomness altogether and use $\mathcal{PCP}(0, q(\cdot)) \subseteq \mathrm{DTIME}(2^q \cdot \mathrm{poly})$.

Finally, we mention that one can convert a multi-prover interactive proof system into a probabilistically checkable proof, and vice versa. The translation in the first direction is easy (i.e., just prefix each verifier-message by the identity of the prover), but the translation in the other direction is more complex; see [36, 73].

2.2.4 PCP and Approximation

Interestingly, Theorem 2 can be rephrased without mentioning the class $\mathcal{PCP}$ at all.[8] Instead, a new type of polynomial-time reduction, which we call amplifying, emerges.

[7] The result $\mathcal{NP} \subseteq \mathcal{PCP}(\mathrm{poly}\log, \mathrm{poly}\log)$ can be found in [8], although this paper uses a different model and terminology. Furthermore, $\mathcal{NP} \subseteq \mathcal{PCP}(\log, \mathrm{poly}\log)$ is easily obtained from [8], by using standard de-randomization techniques. For a proof of $\mathcal{NP} \subseteq \mathcal{PCP}(f(\cdot), f(\cdot))$ with $f(n) = O(\log n \cdot \log\log n)$, see [32]. The proof of the quoted result is more complex and can be found in [6, 5] (see also [3, 72]).

[8] Below we prove that Theorem 2 implies the rephrased form. The converse is proven by starting with an amplifying reduction of 3SAT to itself and constructing a pcp system for 3SAT as follows. The oracle in this system is viewed as a function from variables (of the reduced formula) to Boolean values. The verifier uniformly selects a clause (in the reduced formula) and inspects the value of the variables that appear in it.

Theorem 4 (Theorem 2 – Rephrased): *There exist a constant $\rho < 1$, and a polynomial-time reduction of 3SAT to itself, so that the reduction maps non-satisfiable 3CNF formulae to 3CNF formulae for which every truth assignment satisfies at most a ρ fraction of the clauses.*

Proof Sketch: Start by considering the pcp system for 3SAT (guaranteed by Theorem 2). Use the fact that the pcp system used in the proof of Theorem 2 is non-adaptive[9] (i.e., the queries are determined as a function of the input and the random-tape – and do *not* depend on answers to previous queries). Next, associate the bits of the oracle with Boolean variables and introduce a Boolean formula for each possible value of the random tape, describing whether the verifier would have accepted given this value of the random tape. Finally, using auxiliary variables, convert each formula into 3CNF and obtain (as the output of the reduction) the conjunction of all these polynomially many formulae. □

As an immediate corollary one gets results concerning the intractability of approximation. For example,

Corollary 3 (Hardness of Approximating Max3SAT) *There exists a constant $\rho < 1$ so that the following approximation problem* (known as Max3SAT) *is NP-hard:*

> *Given a satisfiable 3CNF formula, find a truth assignment that satisfies at least a ρ fraction of its clauses.*

Thus, given a satisfiable 3CNF formula, it is as hard to find a truth assignment that satisfies a ρ fraction of its clauses as it is to find a truth assignment that satisfies all clauses.

Consequently, for any approximation problem in the class MAX-SNP-complete (cf. [68] and [54]), there exists a constant so that approximating the problem up to this constant is NP-hard. It follows that, unless $\mathcal{P} = \mathcal{NP}$, there exist no polynomial-time approximation schemes (i.e., a sequence of polynomial-time approximation algorithms, one for each constant approximation ratio) for any problem in the class MAX-SNP-complete. Results for approximation problems not in the class MAX-SNP can be derived as well – see [4]. An alternative perspective, aimed at obtaining tight non-approximability results, is presented in [15].

I believe that amplifying reductions are interesting for their own sake and may find other applications in complexity theory.

2.3 Computationally Sound Proof Systems

In the two settings just discussed (i.e., MIP and PCP) the restrictions imposed on the prover were of a "physical" nature. In the current section we discuss models derived from the model of interactive proofs by imposing computational restrictions (specifically, time bounds) on the prover. Although these restrictions apply to potential provers in both the completeness condition and the soundness condition, the effect of the restriction is more dramatic on the soundness condition.[10] Hence, proof systems with computational restrictions on the potential provers are commonly referred to as "computationally sound." The computational restriction in the soundness condition seems to

[9] Actually, this is not essential as one can convert an adaptive system into a non-adaptive one, while incurring an exponential blowup in the query complexity (which in our case is a constant).

[10] In fact, the prover in the completeness condition typically has bounded complexity although the definition of an interactive proof system does not impose such a restriction. In particular, every language in $\mathcal{IP}$ has a proof system in which the prover works in polynomial space. For further discussion see subsection 2.1.3.

upset the puristic intuition about proofs (even more than the restrictions discussed in the previous section). Yet, from the point of view of computer science, the computational restriction is quite natural and furthermore it is justified in many applications.

There are two types of computationally sound proof systems. In the first type, called *arguments*,[11] the computational restriction is that the potential provers, given access to an auxiliary input, run in polynomial time (or more generally, in time that is polynomially related to the nondeterministic complexity of the language). In the second type, called *CS-proofs*,[12] the computational restriction is that the potential provers run in time that is polynomially related to the deterministic complexity of the language. These computational restrictions are analogous to two of the interpretations discussed in subsection 2.1.3. Specifically, in argument system the "1st interpretation" is imposed on the provers of both soundness and completeness condition, whereas in CS-proofs the "3rd interpretation" is imposed.

2.3.1 Argument Systems

The definition of an argument system is derived from the definition of an interactive proof system by modifying the completeness and soundness conditions as follows.

- *Completeness*: The prover P runs in time polynomial in the length of the common input. For every $x \in L$, there exists an auxiliary input (for the prover), w_x, so that the verifier V always accepts after interacting with $P(w_x)$ on common input x.
- *Soundness*: For every probabilistic polynomial-time[13] machine P^*, for all sufficiently long $x \notin L$, and for all $w \in \{0,1\}^*$, the verifier V rejects with probability at least $\frac{1}{2}$, after interacting with $P^*(w)$ on common input x.

Both conditions can be rephrased by using (non-uniform) families of circuits of polynomial size. As discussed above, argument systems are adequate for modeling the behavior of parties in a real-life setting. Under strong intractability assumptions, argument systems exhibit advantages over interactive proof systems.[14] Let us start by stating these assumptions.

Definition 4 (Collision-Free Hashing) *Consider a family of hash functions, indexed by strings,* $F \stackrel{\text{def}}{=} \{f_\alpha : \{0,1\}^{2|\alpha|} \mapsto \{0,1\}^{|\alpha|}\}_\alpha$, *so that there exists a polynomial-time algorithm for evaluating* F *(i.e., on input* α *and* x *returns* $f_\alpha(x)$*). The family* F *is called* collision-free w.r.t. complexity $c(\cdot)$ *if for every non-uniform family of circuits* $\{C_n\}$ *with size bounded by* $c(\cdot)$*, and all sufficiently large* n*'s, the probability that* C_n*, given a uniformly chosen* $\alpha \in \{0,1\}^n$*, outputs a pair* (x,y) *so that* $f_\alpha(x) = f_\alpha(y)$*, is bounded above by* $1/c(n)$*. The family* F *is called* collision-free *if it is collision-free w.r.t. all polynomials, and is called* strongly collision-free *if, for some* $\epsilon > 0$*, it is collision-free w.r.t. the function* $f(n) \stackrel{\text{def}}{=} 2^{n^\epsilon}$.

[11] In some early works, and in particular in [27], argument systems are negligently referred to as "interactive proofs." This is quite confusing since arguments differ from interactive proofs not only by definition but also in expressive power (unless $\mathcal{PSPACE} \subseteq \mathcal{IP}(1)$) and in zero-knowledge properties [27, 35] (unless the polynomial-time hierarchy collapses [26]).

[12] Actually, there are three variants of this model – see [62, 63]. In the current subsection we concentrate on the "interactive" variant of [62]. A brief discussion of the "non-interactive" variants of [63] is postponed to subsection 2.4.

[13] Again, this means a running time polynomial in the length of the common input.

[14] Below, we consider the expressing power of both models. An additional advantage of argument systems is that, under strong intractability assumptions, there exist *perfect* zero-knowledge arguments (rather than *computational* zero-knowledge interactive proofs) for any language in $\mathcal{NP}$ [27].

Collision-free functions exist assuming the intractability of factoring integers (i.e., in polynomial time). Strong collision-free functions exist if integers cannot be factored in time 2^{n^ϵ}, for some $\epsilon > 0$.

Theorem 5 [55]: *Let $L \in \mathcal{NP}$ and assume the existence of collision-free functions* (resp., strong collision-free functions)*. Then, for every $\epsilon > 0$, there exists an argument system for L in which the randomness and communication complexities of the verifier, on inputs of length n, are both bounded by n^ϵ* (resp., $\text{poly}(\log(n))$)*. Furthermore, the computational complexity of the verifier is quadratic in the length of the input.*

We stress that Theorem 5 is meaningful also in case $L \in \mathcal{P}$; in particular, it offers quadratic verification time, independently of the (possibly higher) deterministic complexity of the language. Interestingly, the results of Theorem 5 are unlikely for interactive proof systems, due to the following[15]:

Proposition 3 *Suppose that L has an interactive proof system in which both the randomness and communication complexities of the verifier are bounded by an integer function $c(\cdot)$. Then $L \in \text{DTIME}(2^{O(c(\cdot)+\log(\cdot))})$.*

Proof Sketch: Consider the tree of all possible executions. $\square$

2.3.2 CS-Proof Systems

The definition of a CS-proof system is derived from the definition of an interactive proof system analogously to the way the definition of an argument system is derived. The only difference is that here the potential provers are *uniform* probabilistic machines, with no auxiliary inputs, running in time polynomial *in the deterministic complexity of the language*. A result analogous to Theorem 5 is obtainable also in the current setting. Specifically,

Theorem 6 [62]: *Let $L \in \mathcal{EXP}$. Then, assuming the existence of strong collision-free functions, there exists a CS-proof system for L. Furthermore, the computational complexity of the verifier is quadratic in the length of the input and polylogarithmic in the deterministic complexity of L.*

2.4 Other Types of Proof Systems

Of the other types of proof systems, I'm going to discuss only two, which are slightly problematic, and also this will be done quite abruptly.

2.4.1 Non-Interactive Proof Systems

The phrase "non-interactive" is often misleading. Indeed, in all models that are called "non-interactive," the interaction between the prover and the verifier is minimal; it consists of the prover sending a single message to the verifier (as in the case of an NP-proof). Yet, in most "non-interactive" models, both the prover and the verifier interact with a (trusted) random string (cf., [22, 21]) or even query a random oracle (cf., the so-called "Guaranteed CS-proofs" of [63]).

However, in addition to NP-proofs, there is another model that is truly non-interactive, namely, the so-called Cryptographic CS-proof [63]. Cryptographic CS-proofs are short "certificates" that can be efficiently verified (like NP-proofs), are "relatively easy" to find for inputs in the language, but are very hard to find (rather than do not exist) for inputs *not* in the language. Namely, for

[15] See [42] for further investigations.

	IP	arguments	CS-proof	PCP	MIP
restrictions on prover	none	poly-time + aux. input	polynomial in Dtime	memoryless (i.e., oracle)	split entity
motivation (as I see it)	generalize NP	restrict IP (see Remark 1)		augment NP	see Remark 2
expressive power	PSPACE	$\mathcal{IP}(1) \subseteq \mathcal{PH}$	EXP[16]	scalable: NTIME($2^{l(n)}$), for rnd+query $= O(l(n))$	

Figure 1: Comparison of various proof systems

valid assertions, Cryptographic CS-proofs can be found in time polynomial in the (deterministic) complexity of (deciding) the language; whereas, for invalid assertions, false certificates cannot be found within such time bounds. Unfortunately, the existence of (non-trivial) Cryptographic CS-proofs is not known to be reducible to standard complexity assumptions; yet, plausibility arguments towards the existence of the former can be found in [63].

2.4.2 Proofs of Knowledge

The concept of a proof of knowledge is very appealing, yet its precise formulation is much more complex than one may expect; see [13]. A key notion in the definition is that of a *knowledge extractor*. Loosely speaking, a knowledge-verifier for a relation R guarantees the existence of a knowledge extractor that on input x and access to any interactive machine P^* outputs a y so that $(x, y) \in R$, within complexity that is inversely proportional to the probability that the verifier accepts x when interacting with P^*.

2.5 Comparison

In Figure 1, I've tried to summarize the differences between the various notions of efficient proof systems. The class $\mathcal{NP}$ has been omitted for obvious reasons. I view $\mathcal{IP}$ as the natural *generalization* of $\mathcal{NP}$, obtained by relaxing the notion of efficient computation so that probabilism and interaction are allowed. Except for the negligible probability of error, which can be controlled by the verifier, the original flavor of a proof is maintained. Also, I view $\mathcal{PCP}(\texttt{log}, O(1))$ as an *augmentation* of $\mathcal{NP}$ with the extra property of allowing a hasty verifier to take its chances and verify the proof in a super-fast manner. In contrast, the two notions of computationally sound proof systems (i.e., arguments and CS-proofs) deviate significantly from the conservative approach of absolute proofs. Yet, computational soundness seems adequate in most practical settings. The only word of warning is that typical results in these latter settings depend on intractability assumptions, and when evaluating these results one should not ignore the relative severeness of these assumptions.

Remark 1: Arguments and CS-proof systems are derived by imposing computational restrictions on the potential provers in both the completeness and soundness conditions. In both cases the motivation for these restrictions is to obtain properties that interactive proofs do not (seem to) have. In the case of argument systems the advantageous properties are very low communication complexity and *perfect* zero-knowledge (for $\mathcal{NP}$). Interestingly, the expressive power of the system does not increase in this case (but rather decreases). In the case of CS-proof systems the advantageous property is the linking of the complexity of proving to the complexity of deciding. Interestingly, the expressive power of the system seems to increase as well (unless $\mathcal{PSPACE} = \mathcal{EXP}$).

[16] Depending on (strong) intractability assumptions.

Remark 2: The MIP model indeed generalizes the IP model. However, in my opinion, this generalization is less natural than the generalization of NP to IP. As far as I am concerned, the MIP model is justified by cryptographic applications (see subsection on MIP). (The transformations between MIP systems and PCP systems does not mean that the motivation of one model can be moved to the other.)

3 The Story

In this section I will try to provide a historical account of the ideas that led to the exciting developments concerning the various types of efficient proof systems. My account is certainly a subjective one and it concentrates on my evaluation of the conceptual contributions, their declared intentions, and their effect on subsequent developments.

The story of these research developments, as any real story, has a complex structure that needs to be simplified so that the reader does not get lost in its web. Thus I have broken the story into several independent linearly structured stories, which do cross each other. I will begin with *the story of the evolution of proof systems*, and then pass to the story of their applications to *deriving hardness results for approximation problems*, *program checking* and *zero-knowledge proofs*.

3.1 The Evolution of Proof Systems

The story of $\mathcal{NP}$ is well-known (except maybe for the fact that NP-completeness was discovered by Levin [58] independently of Cook [30] and Karp [52]; see Trakhtenbrot's survey of Russian research on NP [74]). Our story thus starts with the definition of interactive proof systems.

3.1.1 Interactive Proof Systems

Motivated by the desire to formulate the most general type of "proofs" that may be used within cryptographic protocols, Goldwasser, Micali and Rackoff introduced the notion of an *interactive proof system* [48]. Although the main thrust of their paper is the introduction of a special type of interactive proofs (i.e., ones that are *zero-knowledge*), the computational complexity potential of the class of languages possessing interactive proof systems, denoted $\mathcal{IP}$, has been pointed out.

The first evidence of the surprising power of interactive proofs was given by Goldreich, Micali and Wigderson, who presented an interactive proof system for Graph Non-Isomorphism [43], a language not known to be in $\mathcal{NP}$. Interactive proof systems gained much attention also due to the interest in the construction of zero-knowledge proofs for languages in $\mathcal{NP}$, presented in [43] (assuming the existence of one-way functions).

Babai [7], independently[17] of [48], suggested a different formulation of interactive proofs, termed *Arthur-Merlin Games* (and giving rise to the class $\mathcal{AM}$). Syntactically, Arthur-Merlin Games are a restricted form of interactive proof systems, yet Goldwasser and Sipser have subsequently shown that these restricted systems are as powerful as the general ones; namely, $\mathcal{AM} = \mathcal{IP}$ [49]. Babai's motivation was to place a group-theoretic problem, previously placed in $\mathcal{NP}$ under some group-theoretic assumptions, "as close to $\mathcal{NP}$ as possible" without using any assumptions.[18] Interestingly, Babai underestimated the power of the new class, conjecturing that the class $\mathcal{AM}$ (even with an unbounded number of rounds) is "very close" to $\mathcal{NP}$.

3.1.2 Multi-Prover Interactive Proof Systems

A generalization of interactive proofs to *multi-prover interactive proofs* has been suggested by Ben-Or, Goldwasser, Kilian and Wigderson [20]. Again, the main motivation came from zero-knowledge aspects; specifically, introducing multi-prover zero-knowledge proofs for $\mathcal{NP}$ without

[17] Here "independence" does not mean concurrently. Indeed, both the works of Babai and of Goldwasser, Micali and Rackoff first appeared in the same conference (i.e., *17th STOC*, 1985), yet early versions of the paper of Goldwasser, Micali and Rackoff existed as early as 1982 and were rejected three times from major conferences (i.e., *FOCS83*, *STOC84*, and *FOCS84*).

[18] Indeed, Babai placed the Matrix Representation Problem in $\mathcal{AM}$ using only two rounds.

relying on intractability assumptions. Yet, the complexity theoretic prospects of the new class, denoted $\mathcal{MIP}$, have not been ignored. A more appealing, to my taste, formulation of the class $\mathcal{MIP}$ has been presented by Fortnow, Rompel and Sipser [36]. The latter formulation exactly coincides with the formulation now known as *probabilistically checkable proofs* (i.e., $\mathcal{PCP}$).

3.1.3 Algebraic Methods Demonstrate the Power of Interactive Proofs

The amazing power of interactive proof systems has been demonstrated by using algebraic methods. The basic technique has been introduced by Lund, Fortnow, Karloff and Nisan, who applied it to show that the polynomial-time hierarchy (and actually $\mathcal{P}^{\sharp\mathcal{P}}$) is in $\mathcal{IP}$ [60]. Subsequently, Shamir used the technique to show that $\mathcal{IP} = \mathcal{PSPACE}$ [71], and Babai, Fortnow and Lund used it to show that $\mathcal{MIP} = \mathcal{NEXP}$ [9].

The technique of Lund, Fortnow, Karloff and Nisan has been inspired by ideas coming from works on "program checking" (cf., [23]). In particular, their interactive proof system for the permanent combines Lipton's "self-testing" procedure for the permanent (which represents the permanent as a multi-linear polynomial) [59], and the "downwards self-reducibility" procedure of Blum, Luby and Rubinfeld [25]. Another idea that is implicit in [60] and made explicit in the subsequent works of [71, 9] is the representation, introduced by Beaver and Feigenbaum [11], of Boolean formulae as multi-linear polynomials.

It may be of interest to note that the technique of Lund et. al. has been first applied in the context of multi-prover interactive proofs, yielding $\mathcal{P}^{\sharp\mathcal{P}} \subseteq \mathcal{MIP}$, and that the result quoted above (concerning $\mathcal{IP}$) followed later. Hence, $\mathcal{MIP}$ has played a role in the historical development leading to the characterization of $\mathcal{IP}$.

3.1.4 Scaling Down the BFL Proof System Yields a New Class

The abovementioned multi-prover proof system of Babai, Fortnow and Lund [9] (hereafter referred to as the BFL proof system) has been the starting point for fundamental developments regarding $\mathcal{NP}$. The first development was the discovery that the BFL proof system can be "scaled-down"[19] from $\mathcal{NEXP}$ to $\mathcal{NP}$. This important discovery was made independently by two sets of authors: Babai, Fortnow, Levin and Szegedy [8] and Feige, Goldwasser, Lovasz and Safra [31].[20] However, the manner in which the BFL proof is scaled-down is different in the two papers, and so are the consequences of the scaling-down.

Babai, Fortnow, Levin and Szegedy start by considering only inputs encoded using a special error-correcting code. The encoding of strings, relative to this error-correcting code, can be computed in polynomial time. They presented a polynomial-time algorithm that transforms NP-witnesses (to inputs in a language $L \in \mathcal{NP}$) into *transparent proofs* that can be verified as vouching for the correctness of the encoded assertion in (probabilistic) polylogarithmic time (by a Random Access Machine). (The fact that the verification procedure never reads the entire "proof" should not come as a surprise, as the procedures of [60, 71, 9] also have this property.) Thus, once "statements" and "proofs" are in the right (error-correcting) form, verification is "super-fast." Babai et. al. [8] stress the practical aspects of transparent proofs – specifically, for rapidly checking transcripts of long computations.

[19] The term "scaled-down" is used here as a (standard) technical term. Doing so, I do *not* mean to underestimate the technical difficulty of obtaining these results.

[20] At a later stage, Szegedy improved the randomness and query complexities of the system in [31] and joined the latter paper, which has appeared as [32].

In the proof system of Babai et. al. [8], the total running time of the verifier is reduced (i.e., "scaled-down") to polylogarithmic. In contrast, in the proof system of Feige, Goldwasser, Lovasz and Safra [31], the verifier stays polynomial-time and only two more refined complexity measures, specifically the randomness and query complexities, are reduced to polylogarithmic. This eliminates the need to assume that the input is in a special error-correcting form, and yields a more appealing (i.e., less cumbersome) complexity class. This complexity class is a refinement of the class introduced in [36]. The refinement is obtained by specifying the randomness and query complexities. Namely, $\mathcal{PCP}(r(\cdot), q(\cdot))$ denotes the class of languages having probabilistically checkable proofs in which, on input x, the verifier tosses at most $r(|x|)$ coins and makes at most $q(|x|)$ (Boolean) queries to the proof. Hence, whereas the result of Babai, Fortnow and Lund [9] can be restated as

$$\mathcal{NEXP} = \mathcal{PCP}(\text{poly}, \text{poly}), \tag{1}$$

the result of Feige, Goldwasser, Lovasz, Safra and Szegedy [32] is restated as

$$\mathcal{NP} \subseteq \mathcal{PCP}(f(\cdot), f(\cdot)), \qquad \text{where } f(n) = O(\log n \cdot \log\log n). \tag{2}$$

It should be stressed that the result of Babai, Fortnow, Levin and Szegedy [8] also implies a containment of the above form (with $f(n) = \text{poly}\log n$). Interest in the new complexity class became immense since Feige et. al. [31, 32] demonstrated its relevance to proving the intractability of approximating some combinatorial problems (specifically, Max-Clique). I will elaborate on this aspect of their work in subsection 3.2. Here, I only wish to point out that, when using the method of Feige et. al., the randomness and query complexities of the verifier (in a pcp system for an NP-complete language) relate to the strength of the negative results obtained for approximation problems. This fact provided a very strong motivation for trying to reduce these complexities and obtain a tight characterization of $\mathcal{NP}$ in terms of $\mathcal{PCP}(\cdot,\cdot)$.

3.1.5 Tightening the Relation between NP and PCP

Once the work of Feige et. al. [32] had been presented, the challenge was clear: showing that $\mathcal{NP}$ equals $\mathcal{PCP}(\texttt{log}, \texttt{log})$. This challenge was met by Arora and Safra [6]. The proof system constructed by Arora and Safra is very complex, involving recursive use of proof systems and concatenation tests that are much more efficient than the length of strings being tested. (Interestingly, the idea of encoding inputs in an error-correcting form, as suggested in [8], is essential to make this recursion work.) Actually, Arora and Safra showed that

$$\mathcal{NP} = \mathcal{PCP}(\texttt{log}, f(\cdot)), \qquad \text{where } f(n) = o(\log n). \tag{3}$$

Hence, a new challenge arose, namely, further reducing the query complexity – in particular to a constant – while maintaining the logarithmic randomness complexity. Again, additional motivation for this challenge came from the relevance of such a result to the study of approximation problems (see subsection 3.2). The new challenge was met by Arora, Lund, Motwani, Sudan and Szegedy [5] and is captured by the equation

$$\mathcal{NP} = \mathcal{PCP}(\texttt{log}, O(1)). \tag{4}$$

In addition to building on the ideas of Arora and Safra [6], the above result of [5] utilizes ideas and techniques from the works on self-testing/self-correcting [25], degree-tests for multi-variant polynomials [40, 70], and parallelization of multi-prover proof systems [56].

3.1.6 Computationally Sound Proof Systems

Argument systems were defined in 1986 by Brassard, Chaum and Crépeau [27], but their complexity-theoretic significance became apparent only in 1992. This happened when Kilian, using early results on $\mathcal{PCP}$ (due to [32]), showed that, under some intractability assumptions, every language in $\mathcal{NP}$ has a computationally-sound proof in which the randomness and communication complexities are polylogarithmic [55].

Consequently, Micali suggested three new types of computationally-sound proof systems that he calls CS-proofs [62, 63]. Micali showed that Kilian's construction can be applied also for languages beyond $\mathcal{NP}$, and more importantly that using interaction with a trusted random oracle allows one to get rid of the interaction between the parties as well as of the intractability assumptions [63]. The latter idea can be traced back to a paper of Fiat and Shamir [34]

3.1.7 Other Types of Proof Systems

The setting of non-interactive proofs was first introduced by Blum, Feldman and Micali [22]. The concept of proofs of knowledge was introduced in the paper of Goldwasser, Micali and Rackoff [48], but it is only recently that it has been given a satisfactory formal treatment [13].

3.2 PCP and Approximation

As stated above, probabilistic checkable proofs became the focus of so much attention due to their relation to the difficulty of approximation.

The relation between PCP and approximation was first pointed out by Feige, Goldwasser, Lovasz and Safra [31] (see [32]). Specifically, they showed how to construct a graph representing the possible computations of a pcp verifier on a specific input so that the size of the maximum clique in this graph equals the accepting probability (scaled down by an easy to determine factor). The time needed to construct the graph is (linearly) exponential in the randomness and query complexity of the pcp system. It follows, for example, that any polynomial-time algorithm approximating Max-Clique, up to a constant factor, yields an algorithm for deciding a language in $\mathcal{PCP}(r(\cdot), q(\cdot))$ such that the algorithm runs in time $2^{r(n)+q(n)} \cdot \text{poly}(n)$. This has provided a strong motivation towards showing that $\mathcal{NP} \subseteq \mathcal{PCP}(\log, \log)$.

Although Feige et. al. only related PCP to the approximation of Max-Clique, it was understood that the relation between PCP and approximation is not specific to the Max-Clique Problem. The question was how well do the results for Max-Clique translate to other approximation problems, or alternatively whether better results can be obtained by "directly" relating PCP to the approximation of different problems.

It turned out that stronger non-approximability results are possible when treating the two complexity measures of $\mathcal{PCP}$ (i.e., randomness and query complexities) separately. In particular, languages in the class $\mathcal{PCP}(r(\cdot), O(1))$ are naturally represented by 3CNF formulae of size $O(2^{r(n)})$, so that if the input is in the language then the formula is satisfiable and otherwise no truth assignment can satisfy more than a constant (< 1) fraction of its clauses (see this chapter's Technical Part). Furthermore, the time required to compute this formula is (linearly) exponential in $r(\cdot)$. Hence, any polynomial-time approximation scheme for Max3SAT yields a polynomial-time decision procedure for any language in $\mathcal{PCP}(\texttt{log}, O(1))$. This implication has provided a strong motivation towards showing that $\mathcal{NP} \subseteq \mathcal{PCP}(\log, O(1))$. Additional motivation comes from the fact that the latter inclusion also implies that Max-Clique is NP-hard to approximate even to within a factor of N^ϵ, for some $\epsilon > 0$, where N denotes the number of vertices in the graph. These facts were observed by Arora, Lund, Motwani, Sudan and Szegedy [5]. They also observed that, since Max3SAT is in the class MAX-SNP [68], it followed that each complete problem in that class (e.g., Max2SAT, MaxCUT, MinVC) is hard to approximate to within some constant factor.[21]

The results of [32, 6, 5] have renewed interest in the complexity of approximation problems, after many years of frustration. Subsequent work established the hardness of approximation problems via the following two approaches.

1st approach: generic reductions from PCP/MIP. That is, reducing the evaluation of the acceptance probability of a PCP/MIP system *directly* to the approximation problem being considered (cf., Bellare [12], Lund and Yannakakis [61]). Furthermore, sometimes such a reduction is coupled with a new PCP/MIP system that is more efficient in some *parameters* yielding stronger negative results than those obtained by reducing previously known PCP/MIP systems (cf., Bellare, Goldwasser, Lund and Russell [17], Bellare and Sudan [18] and Bellare, Goldreich and Sudan [15]). Lastly, it is important to select the "right" complexity parameters to be optimized (cf., [17, 33, 18]). For an elaborate discussion of the latter point see [15].

2nd approach: reductions among approximation problems. The newly acquired collection of hard approximation problems motivates and facilitates attempts to prove the hardness of new problems by using approximation-preserving reductions. Typical examples are given in the works of Lund and Yannakakis [61], Khanna, Linial and Safra [53], and Fürer [37].

3.3 Interactive Proofs and Program Checking

When introducing the concept of *program checking*, Blum (cf. [23]) was indeed aware of its relation to interactive proof systems. In particular, his program checker for Graph Isomorphism mimics the interactive proof of [43]. Subsequently, the development paths of the two concepts (i.e., program checking and efficient proof systems) have continued to intersect. The contribution of program checking techniques to the results concerning $\mathcal{IP}$ and $\mathcal{PCP}$ has been discussed above. Here I wish to further discuss the opposite direction in which various proof systems have yielded applications to program checking.

In [8], Babai, Fortnow, Levin and Szegedy stress the application of their result to program checking. At the expense of having a powerful computer do some extra work, its computation can be checked by a much weaker computer. Specifically, the powerful computer outputs a certificate for the correctness of the computation so that the certificate has length comparable to the "length" of the computation and yet its correctness can be verified in time that is much shorter than the time of the original computation. This holds in a Random Access Machine computation model.

Recently, Micali showed that using "Cryptographic CS-proofs" (see this chapter's Technical Part) one can transform programs to ones that in addition to the result of the computation supply a very short certificate of the validity of the computation [63]. This certificate can then be checked in time that is negligible (also in a Turing Machine model) compared to the original computation time.

Both the results of [8] and [63] refer to program checking in a sense different from Blum's (cf. [23]), namely, the computation of a program π on a specific input is checked against the program π itself (rather than against a functional description of what π is supposed to do). The prover, which is an extension of π, needs to run more time than π (but not drastically more), yet the advantage is that a verifier can check this long computation very fast.

[21] On the other hand, each problem in this class is easy to approximate to within a (different) constant.

3.4 Zero-Knowledge Proofs

As mentioned a few times above, the search for zero-knowledge proofs has been the driving force behind many of the definitions of proof systems. In particular, zero-knowledge has motivated the first leap beyond $\mathcal{NP}$ taken when interactive proofs were conceived by Goldwasser, Micali and Rackoff [48]. The reason being that a more liberal notion of a proof seemed (and in fact is [44]) necessary in order to enable simulation of the transcript without ability to perform the interaction ("in real time").

The notion of computational indistinguishability, introduced by Goldwasser and Micali (in the context of defining security of encryption schemes [47]) and Yao (in full generality [75]), is central to the definition of zero-knowledge. However, most of the complexity-theoretic impact of the former notion was on the theory of pseudorandomness (cf., for example, [24, 75] and [66, 65]) which has evolved almost concurrently to the evolution of interactive proof systems.

The wide applicability of zero-knowledge proofs has been demonstrated by Goldreich, Micali and Wigderson [43]. Most importantly, they showed how to construct zero-knowledge proof systems for any language in $\mathcal{NP}$.[22] Their construction uses a cryptographic primitive called a commitment scheme that may be implemented using any one-way function [50, 64]. Actually, under the same assumption, zero-knowledge proofs exists for any language in $\mathcal{IP}$ [51, 19], but the latter elegant result has almost no applications.

Subsequently, zero-knowledge proofs have become the focus of much attention in cryptographic circles and, as one may expect, many interesting results followed. But this is a story for a separate essay.

[22]Some sources credit Brassard and Crépeau [28] for independently discovering the same result. To say the very least, this is technically inaccurate since Brassard and Crépeau use a much stronger intractability assumption – specifically, the intractability of the Quadratic Residue Problem.

4 Addendum: Some Open Problems

Open Problem 1: Polynomials play a fundamental role in the construction of interactive proof systems for co$\mathcal{NP}$, and this trend continues in the construction of PCP systems for $\mathcal{NP}$. Furthermore, it does not seem possible to abstract that role. I consider it important to obtain an alternative proof of co$\mathcal{NP} \subseteq \mathcal{IP}$; a proof in which all the underlying ideas can be presented at an abstract level.

Open Problem 2: Suppose that $L \in \mathcal{IP}(r)$. What can be said about $\overline{L}$? Currently, we only know that $\overline{L} \in \mathcal{IP}(\mathrm{poly})$. This does not utilize the extra information regarding the $\mathcal{IP}(\cdot)$ level in which L resides. On the other hand, we don't expect $\overline{L}$ to be in $\mathcal{IP}(g(r))$, for any function g, since this will put co$\mathcal{NP} \subseteq$ co$\mathcal{IP}(1)$ in $\mathcal{IP}(2)$. Thus, another parameter may be relevant here; for example, the lengths of the messages exchanged in the interaction. Indeed, if L has an interactive proof in which the total message length is m then $\overline{L}$ has an interactive proof in which the total message length is $O(m^3)$. I consider it important to obtain a better result. In general, it would be interesting to get a better understanding of the $\mathcal{IP}(\cdot)$ Hierarchy.

Open Problem 3: Regarding subsection 2.1.3, it will be interesting to understand the limitations of "relatively efficient provers" according to the 2nd and 3rd interpretations. A better understanding of the self-reducibility on NP-languages is also long due. A specific challenge (cf., [16]): provide an NP-proof system for Quadratic Non-Residucity (QNR), using a probabilistic polynomial-time prover with access to QNR language.

Open Problem 4: Try to provide firm grounds for the heuristics of making proof systems non-interactive by use of "random public functions" (cf., [34, 63]): I advise not to try to define the latter notion (in a general form), but rather devise some ad-hoc method, using some specific but widely believed complexity assumptions (e.g., hardness of deciding Quadratic Residucity modulo a composite number), for this specific application.

Open Problem 5: It would be interesting to figure out and utilize the minimal possible assumption required for constructing "zero-knowledge protocols for NP" in various models like constant-round interactive proofs, the "non-interactive" model, and perfect zero-knowledge arguments.

Open Problem 6: As a first step towards the simplification of the proof of the PCP Characterization, one may want to provide an alternative (randomness-efficient) "parallelization" procedure which does not rely on polynomials or any other algebraic creatures. A first step towards this partial goal was taken by Safra and myself (cf., TR96-047 of ECCC): We have constructed an efficient low-degree test which utilizes a simple/inefficient low-degree test which is parallelized using a new "combinatorial consistency lemma".

Acknowledgments

I am grateful to Shafi Goldwasser, Leonid Levin, Dana Ron, Madhu Sudan and two anonymous reviewers for helpful remarks.

References

[1] W. Aiello, M. Bellare and R. Venkatesan. Knowledge on the Average – Perfect, Statistical and Logarithmic. In *27th ACM Symposium on the Theory of Computing*, pages 469–478, 1995.

[2] W. Aiello and J. Håstad. Statistical Zero-Knowledge Languages can be Recognized in Two Rounds. *Journal of Computer and System Science*, Vol. 42, pages 327–345, 1991.

[3] S. Arora. *Probabilistic Checking of Proofs and the Hardness of Approximation Problems.* PhD thesis, U.C. Berkeley, 1994. Available from *ECCC*.

[4] S. Arora and C. Lund. Hardness of Approximations. In *Approximation Algorithms for NP-hard Problems*, D. Hochbaum ed., PWS, 1996.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. In *33rd IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[6] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. In *33rd IEEE Symposium on Foundations of Computer Science*, pages 1–13, 1992.

[7] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–420, 1985.

[8] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd ACM Symposium on the Theory of Computing*, pages 21–31, 1991.

[9] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, Vol. 1, No. 1, pages 3–40, 1991. Preliminary version in *31st FOCS*, 1990.

[10] 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, pages 254–276, 1988.

[11] D. Beaver and J. Feigenbaum. Hiding Instances in Multioracle Queries. In *7th Symposium on Theoretical Aspects of Computer Science*, Springer Verlag, LNCS Vol. 415, pages 37–48, 1990.

[12] M. Bellare. Interactive Proofs and Approximation: Reductions from Two Provers in One Round. In *Proc. 2nd Israel Symp. on Theory of Computing and Systems* (*ISTCS93*), IEEE Computer Society Press, pages 266–274, 1993.

[13] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto92*, Springer Verlag, LNCS Vol. 740, pages 390–420, 1992.

[14] M. Bellare, O. Goldreich and S. Goldwasser. Randomness in interactive proofs. *Computational Complexity*, Vol. 4, No. 1, pages 319–354, 1993.

[15] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. Extended abstract in *36th IEEE Symposium on Foundations of Computer Science*, pages 422–431, 1995. Full version appeared as TR95-024 of *ECCC*.

[16] M. Bellare and S. Goldwasser. The Complexity of Decision versus Search. *SIAM J. on Computing,* Vol. 23, No. 1, pages 97–119, 1994.

[17] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient Probabilistically Checkable Proofs and Applications to Approximation. In *25th ACM Symposium on the Theory of Computing*, pages 294–304, 1993. A revised version is available from the authors.

[18] M. Bellare and M. Sudan. Improved Non-Approximability Results. In *26th ACM Symposium on the Theory of Computing*, pages 184–193, 1994.

[19] 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*, Springer Verlag, LNCS Vol. 403, pages 37–56, 1990

[20] 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*, pages 113–131, 1988.

[21] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge Proof Systems. *SIAM J. on Computing,* Vol. 20, No. 6, pages 1084–1118, 1991.

[22] M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and its Applications. In *20th ACM Symposium on the Theory of Computing*, pages 103–112, 1988.

[23] M. Blum and S. Kannan. Designing Programs that Check their Work. In *21st ACM Symposium on the Theory of Computing*, pages 86–97, 1989.

[24] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, Vol. 13, pages 850–864, 1984. In *23rd FOCS*, pages 80–91, 1982.

[25] 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, pages 549–595, 1993. Preliminary version in *22th STOC*, 1990.

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

[27] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Science*, Vol. 37, No. 2, pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.

[28] G. Brassard and C. Crépeau. Zero-Knowledge Simulation of Boolean Circuits. In *Crypto86*, Springer Verlag, LNCS Vol. 263, pages 223–233, 1987.

[29] R. Chang, B. Chor, O. Goldreich, J. Hartmanis, J. Håstad, D. Ranjan, and P. Rohatgi. The Random Oracle Hypothesis is False. *Journal of Computer and System Science*, Vol. 49, No. 1, pages 24–39, 1994.

[30] S. A. Cook. The Complexity of Theorem-Proving Procedures. In *3rd ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[31] U. Feige, S. Goldwasser, L. Lovász and S. Safra. On the Complexity of Approximating the Maximum Size of a Clique. Unpublished manuscript, 1990.

[32] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. In *32nd IEEE Symposium on Foundations of Computer Science*, pages 2–12, 1991.

[33] U. Feige and J. Kilian. Two Prover Protocols – Low Error at Affordable Rates. In *26th ACM Symposium on the Theory of Computing*, pages 172–183, 1994.

[34] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solution to Identification and Signature Problems. In *Crypto86*, Springer Verlag, LNCS Vol. 263, pages 186–189, 1987.

[35] 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.

[36] L. Fortnow, J. Rompel and M. Sipser. On the Power of Multi-Prover Interactive Protocols. *Theoretical Computer Science*, Vol. 134, pages 545–557, 1994. Preliminary version in *Proc. 3rd IEEE Symp. on Structure in Complexity Theory*, 1988.

[37] M. Fürer. Improved Hardness Results for Approximating the Chromatic Number. In *36th IEEE Symposium on Foundations of Computer Science*, pages 414–421, 1995.

[38] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.

[39] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[40] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In *23th ACM Symposium on the Theory of Computing*, pages 32–42, 1991.

[41] O. Goldreich. *Foundation of Cryptography – Fragments of a Book*. February 1995. Available from *ECCC*.

[42] O. Goldreich and J. Håstad. On the Message Complexity of Interactive Proof Systems. TR96-018 of *ECCC*, March 1996.

[43] 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. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[44] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, Vol. 7, No. 1, pages 1–32, 1994.

[45] O. Goldreich, R. Ostrovsky and E. Petrank, Knowledge Complexity and Computational Complexity. In *26th ACM Symposium on the Theory of Computing*, pages 534–543, 1994.

[46] O. Goldreich and E. Petrank. Quantifying Knowledge Complexity. In *32nd IEEE Symposium on Foundations of Computer Science*, pages 59–68, 1991.

[47] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.

[48] 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.

[49] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 73–90, 1989.

[50] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudorandom Generator from any One-Way Function. To appear in *SIAM J. on Computing.* Preliminary versions by Impagliazzo et. al. in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).

[51] R. Impagliazzo and M. Yung. Direct Zero-Knowledge Computations. In *Crypto87*, Springer Verlag, LNCS Vol. 293, pages 40–51, 1987.

[52] R.M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, (Raymond E. Miller and James W. Thatcher, eds.), Plenum Press, pages 85–103, 1972.

[53] S. Khanna, N. Linial and S. Safra. On the Hardness of Approximating the Chromatic Number. In *Proc. 2nd Israel Symp. on Theory of Computing and Systems* (*ISTCS93*), IEEE Computer Society Press, pages 250–260, 1993.

[54] S. Khanna, R. Motwani, M. Sudan and U. Vazirani. On Syntatic versus Computational Views of Approximability. In *35th IEEE Symposium on Foundations of Computer Science*, pages 819–830, 1994.

[55] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th ACM Symposium on the Theory of Computing*, pages 723–732, 1992.

[56] D. Lapidot and A. Shamir. Fully Parallelized Multi Prover Protocols for NEXP-time. In *32nd IEEE Symposium on Foundations of Computer Science*, pages 13–18, 1991.

[57] C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, Vol. 17 (4), pages 215–217, 1983.

[58] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal Search Problems: in Russian). *Problemy Peredachi Informatsii*, 9 (3), pages 265–266, 1973.

[59] R.J. Lipton. New Directions in Testing, Unpublished manuscript, 1989.

[60] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.

[61] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems, *Journal of the ACM*, Vol. 41, pages 960–981, 1994. Preliminary version in *25th STOC*, 1993.

[62] S. Micali. CS Proofs. Unpublished manuscript, 1992.

[63] S. Micali. CS Proofs. In *35th IEEE Symposium on Foundations of Computer Science*, pages 436–453, 1994.

[64] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.

[65] N. Nisan. Pseudorandom Generators for Space-Bounded Computation. *Combinatorica*, Vol. 12, No. 4, pages 449–461, 1992. Preliminary version in *22nd STOC*, 1990.

[66] N. Nisan and A. Wigderson. Hardness vs Randomness. *Journal of Computer and System Science*, Vol. 49, No. 2, pages 149–167, 1994. Preliminary version in *29th FOCS*, 1988.

[67] R. Ostrovsky and A. Wigderson. One-Way Functions are essential for Non-Trivial Zero-Knowledge, In *Proc. 2nd Israel Symp. on Theory of Computing and Systems* (*ISTCS93*), IEEE Computer Society Press, pages 3–17, 1993.

[68] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Science*, Vol. 43, pages 425–440, 1991. Preliminary version in *20th STOC*, 1988.

[69] R. Raz. A Parallel Repetition Theorem. In *27th ACM Symposium on the Theory of Computing*, pages 447–456, 1995.

[70] R. Rubinfeld and M. Sudan. Robust Characterizations of Polynomials with Applications to Program Checking. *SIAM J. of Computing*, Vol. 25, No. 2, pages 252–271, 1996. Preliminary version in *3rd SODA*, 1992.

[71] A. Shamir. IP=PSPACE. *Journal of the ACM*, Vol. 39, No. 4, pages 869–877, 1992. Preliminary version in *31st FOCS*, 1990.

[72] M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. ACM Distinguished Theses, Springer Verlag, LNCS Vol. 1001, 1995.

[73] A. Ta-Shma. A Note on PCP vs. MIP. TR 94–12, Institute of Computer Science, Hebrew University, Israel, 1994. To appear in *Information Processing Letters*.

[74] B.A. Trakhtenbrot. A Survey of Russian Approaches to *Perebor* (Brute-Force Search) Algorithms. *Annals of the History of Computing*, Vol. 6, No. 4, pages 384–400, 1984.

[75] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.