On Promise Problems

in memory of Shimon Even (1935–2004)*

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

January 30, 2005

Abstract

The notion of promise problems was introduced and initially studied by Even, Selman and Yacobi (*Inform. and Control*, Vol. 61, pages 159–173, 1984). In this article we survey some of the applications that this notion has found in the twenty years that elapsed. These include the notion of "unique solutions", the formulation of "gap problems" as capturing various approximation tasks, the identification of complete problems (especially for the class $\mathcal{SZK}$), the indication of separations between certain computational resources, and the enabling of presentations that better distill the essence of various proofs.

Keywords: Complexity Theory, NP, reductions, Unique Solutions, Approximate Counting, Complexity of Approximation, Property Testing, BPP, Statistical Zero-Knowledge, Complete Problems, circuit complexity, derandomization, PCP, machines that take advice, infinitely often classes.

*This survey was written towards publication in a forthcoming book in memory of Shimon Even.

Contents

1 Introduction

The Theory of Computation excels in identifying fundamental questions and formulating them at the right level of abstraction. Unfortunately, the field's preoccupation with innovation, comes sometimes at the expense of paying relatively modest attention to the proper presentation of these fundamental questions and the corresponding notions and results. One striking example is the way the basics are being taught.

For example, in typical *Theory of Computation* classes, the focus is on "language recognition" devices, and fundamental questions like "P versus NP" are presented in these terms (i.e., is the class of languages recognized by deterministic polynomial-time machines equal the class of languages recognized by non-deterministic polynomial-time machines). In my opinion, such a formulation diminishes the importance of the problem in the eyes of non-bright students, and hides the fundamental nature of the question (which is evident when formulated in terms of "solving problems versus checking the correctness of solutions"). Similarly, one typically takes the students throughout the proof of Cook's Theorem before communicating to them the striking message (i.e., that "universal" problems exist at all, let alone that many natural problems like SAT are universal). Furthermore, in some cases, this message is not communicated explicitly at all.

This article focuses on a less dramatic case of a bad perspective, but still one that deserves considerable attention: I refer to the notion of *promise problems*, and to its presentation in theory of computation classes. Let me start by posing the following rhetoric question:

> How many of the readers have learned about *promise problems* in an undergraduate "theory of computation" course or even in a graduate course on complexity theory?

On the other hand, I would argue that almost all of the readers refer to this notion when thinking about computational problems, although they may be often unaware of this fact.

1.1 What are promise problems

My view is that any decision problem is a promise problem, although in some cases the promise is trivial or tractable (and is thus possible to overlook or ignore). Formally, a promise problem is a partition of the set of all strings into three subsets:

1. The set of strings representing YES-instances.
2. The set of strings representing NO-instances.
3. The set of disallowed strings (which represent neither YES-instances nor NO-instances).

The algorithm (or process) that is supposed to solve the promise problem is required to distinguish YES-instances from NO-instances, and is allowed arbitrary behavior on inputs that are neither YES-instances nor NO-instances. Intuitively, this algorithm (or rather its designer) is "promised" that the input is either a YES-instance or a NO-instance, and is only required to distinguish these two cases. Thus, the union of the first two sets (i.e., the set of all YES-instances and NO-instances) is called the promise.

The common perception by which promise problems are useful for treatment of some *abnormal* situations is in my opinion wrong. On the contrary, in my opinion promise problems *are the norm*: Indeed, the standard and natural presentation of natural decision problems is actually in terms of promise problems, although the the presentation rarely refers explicitly to the terminology of promise problems. Consider a standard entry in [16] (or any similar compendium). It reads something like "given a planar graph, determine whether or not ..." A more formal statement will refer to strings that represent planar graphs. Either way, one may wonder what should the decision procedure do when the input is *not* a (string representing a) planar graph. One common formalistic answer is that all strings are interpreted as representation of planar graphs (typically, by using a decoding convention by which every "non-canonical" representation is interpreted as a representation of some fixed planar graph). Another (even more) formalistic "solution" is to discuss the problem of distinguishing YES-instances from anything else (i.e., effectively viewing strings that violate the promise as NO-instances). Both conventions miss the true nature of the original computational problem, which is concerned with distinguishing planar graphs of one type from planar graphs of another type (i.e., the complementary type). That is, the conceptually correct perspective is that the aforementioned problem is a promise problem in which the promise itself is an easily recognizable set.

But, as observed by Even, Selman and Yacobi [12], the promise need not be an easily recognizable set, and in such a case the issue cannot be pushed under the carpet. Indeed, consider a computational problem that, analogously to the one above, reads "given a Hamiltonian graph, determine whether or not ..." In this case, the two formalistic conventions mentioned above fail: The first one cannot be implemented, whereas the second one may drastically effect the complexity of the problem.

Jumping ahead, we mention that the formulation of promise problems is avoided not without reason. Firstly, it is slightly more cumbersome than the formulation of ordinary decision problems (having a trivial promise that consists of the set of all strings). More importantly, as observed by Even, Selman and Yacobi [12], in some cases "well-known" *structural* relations (which refer to standard decision problems) need not hold for promise problems (in which the promise itself is hard to test for membership). For example, the existence of a promise problem in $\mathcal{NP} \cap \text{co}\mathcal{NP}$ that is $\mathcal{NP}$-hard (under Cook-reduction) does not seem to imply that $\mathcal{NP} = \text{co}\mathcal{NP}$. Still, the benefits of formulating computational problems in terms of promise problems is often more than worthwhile the aforementioned costs.

1.2 Some definitions

In accordance with the above discussion, promise problems are defined as follows.

Definition 1.1 (promise problems): *A promise problem* Π *is a pair of non-intersecting sets, denoted* $(\Pi_{\rm YES}, \Pi_{\rm NO})$*; that is,* $\Pi_{\rm YES}, \Pi_{\rm NO} \subseteq \{0,1\}^*$ *and* $\Pi_{\rm YES} \cap \Pi_{\rm NO} = \emptyset$*. The set* $\Pi_{\rm YES} \cup \Pi_{\rm NO}$ *is called the* promise*.*

An alternative formulation, used in the original paper [12], is that a promise problem is a pair (P, Q), where P is the promise and Q is a super-set of the YES-instances. Indeed, in some cases, it is more natural to use the original formulation (e.g., let P be the set of Hamiltonian graphs and Q be the set of 3-colorable graphs), but Definition 1.1 refers more explicitly to the actual computational problem at hand (i.e., distinguishes inputs in $\Pi_{\rm YES} = P \cap Q$ from inputs in $\Pi_{\rm NO} = P \setminus Q$).

Standard "language recognition" problems are cast as the special case in which the promise is the set of all strings (i.e., $\Pi_{\rm YES} \cup \Pi_{\rm NO} = \{0,1\}^*$). In this case we say that the promise is trivial. The standard definitions of complexity classes (i.e., classes of languages) extend naturally to promise problems. In formulating such an extension, rather than thinking on the standard definition as referring to the set of YES-instances and its complement, one better think of it as referring to two (non-intersecting) sets: the set of YES-instances and the set of NO-instances. We thus have definitions of the following form.

Definition 1.2 (three classes of promise problems):

$\mathcal{P}$ *is the class of promise problems that are solvable in* (deterministic) *polynomial-time. That is, the promise problem* $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ *is in* $\mathcal{P}$ *if there exists a polynomial-time algorithm* A *such that*

- *For every* $x \in \Pi_{\rm YES}$ *it holds that* $A(x) = 1$.
- *For every* $x \in \Pi_{\rm NO}$ *it holds that* $A(x) = 0$.

$\mathcal{NP}$ *is the class of promise problems that have polynomially long proofs of membership that are verifiable in* (deterministic) *polynomial-time. That is, the promise problem* $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ *is in* $\mathcal{NP}$ *if there exists a polynomially bounded binary relation* R *that is recognized in by a polynomial-time algorithm such that*

- *For every* $x \in \Pi_{\rm YES}$ *there exists* y *such that* $(x, y) \in R$.
- *For every* $x \in \Pi_{\rm NO}$ *and every* y *it holds that* $(x, y) \notin R$.

We say that $R \subseteq \{0,1\}^* \times \{0,1\}^*$ *is* polynomially bounded *if there exists a polynomial* p *such that* $|y| \leq p(|x|)$ *for every* $(x, y) \in R$. *We say that* R *is recognized in by the polynomial-time algorithm* A *if* $A(x, y) = 1$ *if and only if* $(x, y) \in R$ *for every* (x, y).

$\mathcal{BPP}$ *is the class of promise problems that are solvable in probabilistic polynomial-time. That is, the promise problem* $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ *is in* $\mathcal{BPP}$ *if there exists a* probabilistic *polynomial-time algorithm* A *such that*

- *For every* $x \in \Pi_{\rm YES}$ *it holds that* $\Pr[A(x) = 1] \geq 2/3$.
- *For every* $x \in \Pi_{\rm NO}$ *it holds that* $\Pr[A(x) = 0] \geq 2/3$.

That is, in each case, the conditions used in the standard definition (of language recognition) are applied to the partition of the promise (i.e., $\Pi_{\rm YES} \cup \Pi_{\rm NO}$), and nothing is required with respect to inputs that violate the promise.

The notion of a reduction among computational problems also extends naturally to promise problems. The next definition extends the most basic type of reductions (i.e., Karp and Cook reductions).

Definition 1.3 (reductions among promise problems): *The promise problem* $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ *is* Karp-reducible *to the promise problem* $\Pi' = (\Pi'_{\rm YES}, \Pi'_{\rm NO})$ *if there exists a polynomial-time computable function* f *such that*

- *For every* $x \in \Pi_{\rm YES}$ *it holds that* $f(x) \in \Pi'_{\rm YES}$.
- *For every* $x \in \Pi_{\rm NO}$ *it holds that* $f(x) \in \Pi'_{\rm NO}$.

The promise problem $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ *is* Cook-reducible *to the promise problem* $\Pi' = (\Pi'_{\rm YES}, \Pi'_{\rm NO})$ *if there exists a polynomial-time oracle machine* M *such that*

- *For every* $x \in \Pi_{\rm YES}$ *it holds that* $M^{\Pi'}(x) = 1$.
- *For every* $x \in \Pi_{\rm NO}$ *it holds that* $M^{\Pi'}(x) = 0$.

where query q to Π' is answered by 1 if $q \in \Pi'_{\text{YES}}$, by 0 if $q \in \Pi'_{\text{NO}}$, and arbitrarily otherwise. Alternatively, we may consider the computation of M when given access to any total function $\sigma : \{0,1\}^* \to \{0,1,\perp\}$ that satisfies $\sigma(x) = 1$ if $x \in \Pi'_{\text{YES}}$ and $\sigma(x) = 0$ if $x \in \Pi'_{\text{NO}}$, where for $x \notin \Pi'_{\text{YES}} \cup \Pi'_{\text{NO}}$ the value of $\sigma(x)$ may be anything (in $\{0,1,\perp\}$). *Such a function σ is said to* conform with Π'. *We then require that there exists a polynomial-time oracle machine M such that for every total function $\sigma : \{0,1\}^* \to \{0,1,\perp\}$ that conforms with Π' the following holds:*

- *For every $x \in \Pi_{\text{YES}}$ it holds that $M^\sigma(x) = 1$.*
- *For every $x \in \Pi_{\text{NO}}$ it holds that $M^\sigma(x) = 0$.*

Randomized reductions are defined analogously.

We stress that the convention by which queries that do not satisfy the promise may be answered arbitrarily is consistent with the notion of solving a promise problem. Recall that solving the latter means providing correct answers to instances that satisfy the promise, whereas nothing is required of the "solver" in case it is given an instance that violates the promise. In particular, such a potential "solver" (represented by σ in the alternative formulation) may either provide wrong answers to instances that violate the promise or provide no answer at all (as captured by the case $\sigma(x) = \perp$). On the other hand, reductions are suppose to capture what can be done when given access to a device (represented by σ) that solves the problem at the target of the reduction. Thus, a reduction to a promise problem should yield the correct answer regardless of how one answers queries that violate the promise. We stress that the standard meaning of a reduction is preserved: if Π is Cook-reducible to a promise problem in $\mathcal{P}$ (or in $\mathcal{BPP}$) then Π is in $\mathcal{P}$ (resp., in $\mathcal{BPP}$).

The above natural convention (regarding oracle calls to a promise problem) is the source of technical problems. In particular, unlike in the case of languages, a Cook-reduction to a promise problem in $\mathcal{NP} \cap \text{co}\mathcal{NP}$ does not guarantee that the reduced problem is in $\mathcal{NP}$. (For further discussion, see Section 5.1. We stress, again, that a Cook-reduction to a promise problem in $\mathcal{P}$ does guarantee that the reduced problem is in $\mathcal{P}$.)

1.3 Some indispensable uses of promise problems

As argued in Section 1.1, promise problems are actually more natural than language recognition problems, and the latter are preferred mainly for sake of technical convenience (i.e., using less cumbersome formulations). However, in many cases, promise problems are indispensable for capturing important computational relations. For example, the notion of one computational problem being a special case (or a restriction) of another problem is best captured this way: The promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is a special case of $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$ if both $\Pi_{\text{YES}} \subseteq \Pi'_{\text{YES}}$ and $\Pi_{\text{NO}} \subseteq \Pi'_{\text{NO}}$.

The above paragraph refers to the importance of promise problems in providing the nicest presentation of simple ideas, where by a nice presentation we mean one in which conceptual issues are explicitly represented (rather than hidden by technical conventions). We note that when simple ideas are concerned one may survive ugly presentations, but this becomes more difficult when the issues at hand are less simple. Furthermore, in some cases the notion of a promise problem is essential to the main results themselves. Most of this article will be devoted to surveying some of these cases, and a brief overview of some of them follows.

1. The study of the complexity of problems with unique solution must be formally cast in terms of promise problems. For example, unique-SAT is the promise problem having as YES-instances Boolean formulas that have a unique satisfying assignment and having as NO-instances unsatisfiable Boolean formulas. (See Section 2 for further discussion.)

2. The study of the hardness of approximation problems may be formally cast in terms of promise problems. This is especially appealing when one wants to establish the hardness of obtaining an approximation of the optimal value. Specifically, one often refers to "gap problems" that are promise problem having as YES-instances objects that have a relatively high (resp., low) optimum value and NO-instances that are objects with relatively low (resp., high) optimum value. (See Section 3 for further discussion.)

3. Promise problem allow to introduce complete problems for classes that are not known to have complete languages. A notable example is the class $\mathcal{BPP}$, and another important one is $\mathcal{SZK}$ (i.e., the class of problems having statistical zero-knowledge proof systems). Indeed, promise problems played a key role in the study of the latter class. (See Section 4 for further discussion.)

4. Promise problem were used to indicate separations between certain computational devices with certain resource bounds. Examples appeared in the study of circuit complexity, derandomization, PCP and zero-knowledge. (See Section 5.2 for further discussion.)

Finally, we wish to call the reader attention also to the expositionary benefits of promise problems, further discussed in Sections 6 and 7. In particular, in Section 6.1 we discuss the use of promise problems in the content of proving various complexity lower-bounds. In Section 7 we present a suggestion for casting various "modified" complexity classes (i.e., "computations that take advice" and "infinitely often" classes) in terms of the classes themselves where the latter are understrood as classes of promise problems.

1.4 Relation to Shimon Even (a personal comment)

As hinted above, promise problems were explicitly introduced by Even, Selman and Yacobi [12], and their study was initiated in [12]. Moreover, in my opinion, the powerful combination of the natural notion that promise problems capture, their simple definition and its wide applicability is one of Shimon Even's trade-marks. I vividly recall him telling me in one of our first meetings

> *The very simple facts and the basic approaches are the ones that have most impact: they are the ones that get disseminated across the discipline and even influence other disciplines. A work's most influential contribution may be introducing a good notation.*

Needless to say, science progresses by coping with difficult problems. Most scientific works are too complicated to have a far-reaching impact by themselves, but at times they lead to paradigm shifts that do have far-reaching impact (indeed, cf. Kuhn [39]). These paradigm shifts, which are the most important contributions of science, are typically simple from a technical point of view. Thus, both Even and Kuhn viewed simplicity (at the frontier of science) as positively correlated with impact and importance.

In view of the above, I believe that in surveying the notion of promise problems and its wide applicability, I am surveying a central theme in Shimon's research, a theme which is prominently present also in other works of his.

1.5 Organization

We survey various research directions (in complexity theory) in which the notion of promise problems plays a major role. Specifically, Section 2 refers to finding "unique solutions" and to approximately counting the number of solutions, Section 3 refers to the role of "gap problems" in the study of approximation, and Section 4 refers to identifying complete problems (especially for the class $\mathcal{SZK}$). In Section 5 we discuss the use of promise problems as indicators of complexity, and in Section 6.2 we use promise problems in distilling the essence of a known result (i.e., $\mathcal{BPP} \subseteq \mathcal{PH}$). Other expositionary advantages of promise problems are discussed in Sections 6.1 and 7. Concluding comments, which refer to the implications on langauge recogniztion problems and to the applicability to search problems, appear in Section 8.

In all cases, we start with the main conceptual message, which refers both to the topic being treated and the use of promise problems in that treatment. In some cases, we "beef up" the exposition by providing proof sketches of results that we mention; these proof sketches can be ignored with no loss to the conceptual message, and are placed in separate subsections in order to facilitate skipping them.

2 Unique Solutions and Approximate Counting of Solutions

In this section, we review the use of promise problems in stating central results regarding the complexity of finding unique solutions and the complexity of approximating the number of solutions (to NP-problems). We call the reader's attention to the indispensable role of promise problems in the definition of "problems with unique solutions" and their role in formulating a decision version of the problem of "approximate counting". The latter theme will reappear in Section 3.

2.1 The complexity of finding unique solutions

The widely believed intractability of SAT cannot be due to instances that have a "noticeable fraction" of satisfying assignments. For example, given an n-variable formula that has at least $2^n/n$ satisfying assignments, it is easy to find a satisfying assignment (by trying $O(n)$ assignments at random). Going to the other extreme, one may ask whether SAT instances having very few satisfying assignments (e.g., a unique satisfying assignment) can be hard. As shown by Valiant and Vazirani [56], the answer is positive. Specifically, they showed that the ability to solve such instances yields the ability to solve arbitrary instances. Actually, they showed that distinguishing *uniquely* satisfiable formulae from unsatisfied ones is not easier that distinguishing satisfiable formulae from unsatisfied ones.

In order to formulate the above discussion, we refer to the notion of promise problems. Specifically, we refer to the promise problem of distinguishing instances with a unique solution from instances with no solution. For example, unique-SAT (or uSAT) is the promise problem with YES-instances being formulae having a unique satisfying assignment and NO-instances being formulae having no satisfying assignment.

Theorem 2.1 [56]: SAT *is randomly reducible to* uSAT. *That is, there exists a randomized Cook-reduction of* SAT *to* uSAT.

A proof sketch is presented in Section 2.3. The same result holds for any known NP-complete problem; in some cases this can be proven directly and in other cases by using suitable parsimonious reductions.[1]

[1] A parsimonious reduction (between NP-sets) is a Karp-reduction that preserves the number of solutions (i.e., NP-witnesses). That is, for NP-sets $L_R = \{x : \exists y\, (x,y) \in R\}$ and $L_{R'} = \{x : \exists y\, (x,y) \in R'\}$, the mapping f is a parsimonious reduction from L_R to $L_{R'}$ if for every x it holds that $|R'(f(x))| = |R(x)|$, where $R(x) \stackrel{\rm def}{=} \{y : (x,y) \in R\}$ and $R'(x) \stackrel{\rm def}{=} \{y : (x,y) \in R'\}$.

2.2 The complexity of approximately counting the number of solutions

A natural computational problem associated with an NP-relation R is to determine the number of solutions for a given instance; that is, given x, determine the cardinality of $R(x) \stackrel{\rm def}{=} \{y : (x,y) \in R\}$. This problem is the counting problem associated with R. Certainly, the counting problem associated with R is not easier than the problem of deciding membership in $L_R = \{x : \exists y \text{ s.t. } (x,y) \in R\}$, which can be cast as determining, for a given x, whether $|R(x)|$ is positive or zero.

We focus on the problem of approximating $|R(x)|$, when given x, up to a factor of $f(|x|)$, for some function $f : \mathsf{N} \to \{r \in \mathsf{R} : r \geq 1\}$. Formulating this problem in terms of decision problems has several advantages (see analogous discussion at the end of Section 3.1), and such a formulation can be obtained in terms of promise problems. Specifically, the problem of approximating $|R(x)|$ can be easily cast as (or rather reduced to) a promise problem with YES-instances being pairs (x,N) such that $|R(x)| \geq N$ and NO-instances being pairs (x,N) such that $|R(x)| < N/f(|x|)$, where we assume that $N \geq 1$.[2] We denote this promise problem by $\#R^f$. Clearly, for every $f : \mathsf{N} \to \{r \in \mathsf{R} : r \geq 1\}$, deciding $\#R^f$ is at least as hard as deciding L_R. Interestingly, for any f that is bounded away from 1 and for any known NP-relation R, deciding $\#R^f$ is not harder than deciding L_R. We state this fact for the witness relation of SAT, denoted $R_{\tt SAT}$.

Theorem 2.2 [51]: *For every $f : \mathsf{N} \to \mathsf{R}$ such that $f(n) > 1 + (1/\text{poly}(n))$, the counting problem $\#R^f_{\tt SAT}$ is randomly Karp-reducible to* SAT.

A proof sketch is presented in Section 2.3. The same result holds for any known NP-complete problem; in some cases this can be proven directly and in others by using suitable parsimonious reductions.

2.3 Proof sketches for both results

We prove Theorem 2.2 first, and establish Theorem 2.1 next using similar techniques. We start by observing that solving the counting problem $\#R_{\tt SAT}$ for very narrow margins of error is reducible to solving it for very large margins of error. That is, for $f(n) = 1 + (1/\text{poly}(n))$ and $g(n) < \exp(n^c)$ for some $c \in (0,1)$, it holds that $\#R^f_{\tt SAT}$ is Karp-reducible to $\#R^g_{\tt SAT}$. The reduction is based on the observation that, for formulae $\phi_1, ..., \phi_t$ over disjoint sets of variables, it holds that $R_{\tt SAT}(\wedge_{i=1}^t \phi_i) = \{\langle \tau_1, ..., \tau_t \rangle : \forall i\, \tau_i \in R_{\tt SAT}(\phi_i)\}$. Thus, $|R_{\tt SAT}(\phi^t)| = |R_{\tt SAT}(\phi)|^t$, where ϕ^t is the formula obtained by concatenating t copies of the formula ϕ (while using different variables in each copy). It follows that, for any polynomially bounded t, the problem $\#R^f_{\tt SAT}$ is Karp-reducible to $\#R^g_{\tt SAT}$, where $g(t(n) \cdot n) = f(n)^{t(n)}$, by mapping (ϕ, N) to $(\phi^{t(|\phi|)}, N^{t(|\phi|)})$.

Reducing $\#R^g_{\tt SAT}$ to SAT, for $g(n) = n^2$. In view of the above, we may focus on randomly reducing $\#R^g_{\tt SAT}$ to SAT, for $g(n) = n^2$. Given an instance (ϕ, N), with $1 \leq N < g(|\phi|)$, we reduce ϕ to itself, and notice that YES-instances are certainly satisfiable (because $N \geq 1$), whereas NO-instances are not satisfiable (because they have less than $N/g(|\phi|) < 1$ satisfying assignments). Thus, in this case the reduction is valid, but the interesting case (treated next) is when $N \geq g(|\phi|) > |\phi|$.

[2]For every $f : \mathsf{N} \to \mathsf{R}$ such that $f(n) > 1 + (1/\text{poly}(n))$, approximating $|R(x)|$ up to a factor of $f(|x|)$ is Cook-reducible to $\#R^f$ (as defined in the main text). On input x, the Cook-reduction issues the queries $(x, f(|x|)^i)$, for $i = 0, 1, ..., \ell$, where $\ell = \text{poly}(|x|)/\log_2(f(|x|))$. The oracle machine returns 0 if the first query was answered by 0, and $f(|x|)^i$ if i is the largest integer such that $(x, f(|x|)^i)$ was answered by 1.

Given an instance (ϕ, N), with $N > |\phi|$, our goal is to create a random formula ϕ' such that the *expected* cardinality of $R_{\mathtt{SAT}}(\phi')$ equals $|R_{\mathtt{SAT}}(\phi)|/2^k$, where $k \stackrel{\text{def}}{=} \log_2 N - \log_2 |\phi| \geq 0$. Furthermore, with very high probability, if $|R_{\mathtt{SAT}}(\phi)| \geq N$ then $|R_{\mathtt{SAT}}(\phi')| > N/2^{k+1} > 1$ and if $|R_{\mathtt{SAT}}(\phi)| < N/g(|\phi|)$ then $R_{\mathtt{SAT}}(\phi') = \emptyset$ (because $N/2^k = |\phi|/g(|\phi|) \ll 1$).

We create the formula ϕ' as the conjunction of ϕ and ϕ_h, where $h : \{0,1\}^n \to \{0,1\}^{n-k}$ is a randomly chosen hashing function and $\phi_h(x_1, ..., x_n) = 1$ if and only if $h(x_1, ..., x_n) = 0^{n-k}$. We stress that ϕ and ϕ_h use the same variables $x_1, ..., x_n$, and that ϕ_h can be obtained by a parsimonious reduction of the computation of h (i.e., verifying that $h(x_1, ..., x_n) = 0^{n-k}$) to SAT. That is, we consider the randomized mapping

$$(\phi, N) \to \phi' \qquad \text{where } \phi'(x) \stackrel{\text{def}}{=} \phi(x) \wedge (h(x) = 0^{|x| - \log_2(N/|\phi|)}) \qquad (1)$$
$$\text{and } h : \{0,1\}^{|x|} \to \{0,1\}^{|x| - \log_2(N/|\phi|)} \text{ is a random hash function.}$$

Using the "Leftover Hashing Lemma" [50, 8, 35] it follows that, with very high probability, if $|R_{\mathtt{SAT}}(\phi)| \geq N$ then $|R_{\mathtt{SAT}}(\phi')| > N/2^{k+1} > 1$ and if $|R_{\mathtt{SAT}}(\phi)| < N/g(|\phi|)$ then $R_{\mathtt{SAT}}(\phi') = \emptyset$. Thus, we randomly reduced the instance (ϕ, N) of $\#R^g_{\mathtt{SAT}}$ to deciding whether or not ϕ' is satisfiable. That is, *the above randomized mapping* $(\phi, N) \mapsto \phi'$ *is a randomized Karp-reduction of* $\#R^g_{\mathtt{SAT}}$ *to* SAT.

Proof of Theorem 2.1: To prove Theorem 2.1 we combine the above ideas with two additional observations. The first observation is that if an n-variable formula ϕ is unsatisfiable then, for every $i \in \{0, 1, ..., n\}$, the pair $(\phi, 2^i)$ is a NO-instance of $\#R^g_{\mathtt{SAT}}$, whereas in case ϕ is satisfiable then, for $i = \lfloor \log_2 |R_{\mathtt{SAT}}(\phi)| \rfloor$, the pair $(\phi, 2^i)$ is a YES-instance of $\#R^g_{\mathtt{SAT}}$. Furthermore, in the latter case, $2^i \leq |R_{\mathtt{SAT}}(\phi)| < 2^{i+1}$. For sake of simplicity, we assume below that $i \geq \log |\phi|$. The second observation is that in case $2^i \leq |R_{\mathtt{SAT}}(\phi)| < 2^{i+1}$, with very high probability, the formula ϕ'_i (randomly constructed as above when using $N = 2^i$), has at least m satisfying assignments and at most $8m$ satisfying assignments, where $m \stackrel{\text{def}}{=} 2^i/2^{k+1} = |\phi|/2$. Our goal is to reduce the problem of counting the number of satisfying assignments of ϕ'_i to uSAT. We consider a Cook-reduction that for every possible value $j \in \{m, ..., 8m\}$, constructs a formula $\phi''_{i,j}$ that is satisfiable if and only if ϕ'_i has at least j satisfying assignments; for example, we may use

$$\begin{aligned} &\phi''_{i,j}(x_1^{(1)}, ..., x_n^{(1)}, ..., x_1^{(j)}, ..., x_n^{(j)}) \\ &= \bigwedge_{\ell=1}^{j} \phi'_i(x_1^{(\ell)}, ..., x_n^{(\ell)}) \bigwedge \bigwedge_{\ell=1}^{j-1} \left((x_1^{(\ell)}, ..., x_n^{(\ell)}) < (x_1^{(\ell+1)}, ..., x_n^{(\ell+1)})\right) \qquad (2) \end{aligned}$$

where $(x_1^{(\ell)}, ..., x_n^{(\ell)}) < (x_1^{(\ell+1)}, ..., x_n^{(\ell+1)})$ if and only if $x_q^{(\ell)} < x_q^{(\ell+1)}$ for some q and $x_q^{(\ell)} \leq x_q^{(\ell+1)}$ for every q. Furthermore, note that if ϕ'_i has exactly j satisfying assignments then $\phi''_{i,j}$ has a unique satisfying assignment. This suggests the following randomized Cook-reduction from SAT to uSAT:

1. On input an n-variable formula ϕ, the oracle machine constructs the formulae $\phi''_{i,j}$, for every $i \in \{\log |\phi|, ..., |\phi|\}$ and $j \in \{1, ..., 8m\}$, where ϕ'_i is obtained by applying Eq. (1) to the pair $(\phi, 2^i)$, and $\phi''_{i,j}$ is obtained by applying Eq. (2) to ϕ'_i.

   (The case that ϕ has less than $|\phi|$ satisfying assignments is covered by $i = \log |\phi|$, where effectively no hashing takes place, and thus $\phi'_i = \phi$. For this reason, we have let j range in $\{1, ..., 8m\}$ rather than in $\{m, ..., 8m\}$, a change that causes no harm to larger values of i.)

2. The oracle machine queries the oracle on each of the formulae $\phi''_{i,j}$ and accepts if and only if at least one answer is positive.

Note that if ϕ is satisfiable then, with very high probability, at least one of the formulae $\phi''_{i,j}$ has a unique satisfying assignment (and thus the corresponding query will be answered positively). On the other hand, if ϕ is unsatisfiable then all the formulae $\phi''_{i,j}$ are unsatisfiable (and thus all queries will be answered negatively).

3 Gap Problems – Representing Notions of Approximation

Gap problems are special type of promise problems in which instances are partitioned according to some metric leaving a "gap" between YES-instances and NO-instances. We consider two such metrics: in the first metric instances are positioned according to the value of the best corresponding "solution" (with respect to some predetermined objective function), whereas in the second metric instances are positioned according to their distance from the set of objects that satisfy some predetermined property.

3.1 Approximating the value of an optimal solution

When presenting efficient approximation algorithms, one typically discuss algorithms that given an input find an almost-optimal solution with respect to some desired objective function. After all, in many settings, one seeks a solution rather than merely its value, and typically finding such solutions enables to determine their value (thus making the positive result stronger). However, when presenting negative results (i.e., hardness of approximation results), it is natural to consider the possibly easier task of approximating the value of an optimal solution (rather than finding the solution itself). This makes the negative result stronger, and typically makes the argument more clear.

Promise problems are the natural vehicle for casting computational problems that refer to approximating the value of an optimal solution. Specifically, one often refers to "gap problems" that are promise problem having as YES-instances objects that have a relatively high (resp., low) optimum value and NO-instances that are objects with relatively low (resp., high) optimum value. Indeed, this has been the standard practice since [7].

Let us demonstrate this approach by considering the known results regarding several famous approximation problems. For example, the complexity of MaxClique is captured by the gap problem $\texttt{gapClique}_{b,s}$, where b and s are functions of the number of vertices in the instance graph. The gap problem $\texttt{gapClique}_{b,s}$ is a promise problem consisting of YES-instances that are N-vertex graphs containing a clique of size $b(N)$ and NO-instances that are N-vertex graphs containing no clique of size $s(N)$. Hastad's famous result asserts that, for every $\epsilon > 0$, the promise problem $\texttt{gapClique}_{b_\epsilon,s_\epsilon}$ is NP-hard (under probabilistic Karp-reductions) [33], where $b_\epsilon(N) = N^{1-\epsilon}$ and $s_\epsilon(N) = N^\epsilon$.

Another famous approximation problem is Max3SAT. Consider the gap problem $\texttt{gap3SAT}_s$, where s is a constant, that is a promise problem consisting of YES-instances that are satisfiable 3CNF formulae and NO-instances that are 3CNF formulae in which every truth assignment satisfies less than an s fraction of the clauses. (By 3CNF formula we mean a conjunction of clauses, each consisting of exactly three different literals.) Note that the gap problem $\texttt{gap3SAT}_{7/8}$ is trivial, because every 3CNF formula has a truth assignment that satisfies at least a $7/8$ fraction of its clauses. On the other hand, Hastad showed that, for every $\epsilon > 0$, the promise problem $\texttt{gap3SAT}_{(7/8)+\epsilon}$ is NP-hard (under Karp-reductions) [34].

On the benefits of the framework of gap problems. The reader may wonder how essential is the use of gap (promise) problems in stating results of the aforementioned type. Indeed, one often states the MaxClique result by saying that, for every $\epsilon > 0$, it is NP-hard to approximate the size of the maximum-clique in an N-vertex graph to within a factor of $N^{1-\epsilon}$. Firstly, we comment that the latter is merely a corollary of Hastad's result [33], which is actually a (randomized) Karp-reduction of $\mathcal{NP}$ to $\texttt{gapClique}_{N^{1-\epsilon},N^{\epsilon}}$. The same holds with respect to all hardness of approximation results that are obtained through PCPs: They are obtained by Karp-reductions of PCPs with certain parameters to gap problems, where the former PCPs are shown to exist for $\mathcal{NP}$. In our opinion, it is nicer to present these results as hardness of certain gap problems (which reflects what is actually proved), and their meaning is at least as clear when stated in this way. More importantly, in some cases information is lost when using the approximation-factor formulation. Consider for example the assertion that, for every $\epsilon > 0$, it is NP-hard to approximate Max3SAT to within a factor of $(7/8) + \epsilon$. The latter assertion does not rule out the possibility that, given a satisfiable 3CNF formula, one can find an assignment that satisfies 90% of the clauses. This possibility is ruled out by the fact that $\texttt{gap3SAT}_{(7/8)+\epsilon}$ is NP-hard, and we comment that proving the latter result seems to require more work than proving the former [34].[3] Lastly, the formulation of promise problems seems essential to "reversing the PCP to approximation" connection [7, Sec. 8] (i.e., showing that certain NP-hardness results regarding approximation yield PCP systems with certain parameters).

3.2 Property Testing – the distance between YES and NO-instances

In some sense, all research regarding property testing (cf. [48, 21]) can be cast in terms of promise problems, although this is typically not done – for reasons discussed below.

Property testing is a relaxation of decision problems, where the (typically sub-linear time) algorithm is required to accept (with high probability) any instance having the property (i.e., any instance in some predetermined set) and reject (with high probability) any instance that is "far from having the property" (i.e., being at large distance from any instance in the set). The algorithm, called a tester, may run in sub-linear time because it is given oracle access to the tester object, and thus need not read it entirely. Needless to say, in all interesting cases, this algorithm needs to be probabilistic.

Typically, the distance parameter is given as input to the tester (rather than being fixed as in Section 3.1)[4], which makes the positive results stronger and more appealing (especially in light of a separation recently shown in [5]). In contrast, negative results typically refer to a fixed value of the distance parameter. Thus, fixing a *distance function* (e.g., Hamming distance between bit strings) and a property P, two natural types of promise problems emerge:

1. Instances are pairs (x, δ), where x is a description of an object and δ is a distance parameter. The YES-instances are pairs (x, δ) such that x has property P, whereas (x, δ) is a NO-instance if x is δ-far from any x' that has property P.

2. Fixing the distance parameter δ, the instances are merely descriptions of objects, and the partition to YES and NO instances is as above.

[3] Specifically, proving that $\texttt{gap3SAT}_{(7/8)+\epsilon}$ is NP-hard seems to require using a PCP with "perfect completeness" (as constructed in [34, Thm. 3.4]), whereas Hastad's initial construction [34, Thm. 2.3] does not have perfect completeness (and establishes the NP-hardness of distinguishing 3CNF formulae having a truth assignment that satisfies at least $1 - \epsilon$ fraction of the clauses from 3CNF formulae in which every truth assignment satisfies less than a $(7/8) + \epsilon$ fraction of the clauses [34, Thm. 3.1]).

[4] In fact, an analogous treatment applies to approximation problems as briefly surveyed in Section 3.1. Such a formulation of approximation problems in which the approximation factor is part of the input corresponds to the notion of an approximation scheme.

For example, for some fixed integer d, consider the following promise problem, denoted $\mathtt{BPG}_d$, regarding bipartiteness of bounded-degree graphs. The YES-instance are pairs (G, δ) such that G is a bipartite graph of maximum degree d, whereas (G, δ) is a NO-instance if G is an N-vertex graph of maximum degree d such that more than $\delta \cdot dN/2$ edges must be omitted from G in order to obtain a bipartite graph. Similarly, for fixed integer d and $\delta > 0$, the promise problem $\mathtt{BPG}_{d,\delta}$ has YES-instances that are bipartite graphs of maximum degree d and NO-instances that are N-vertex graphs of maximum degree d such that more than $\delta \cdot dN/2$ edges must be omitted from the graph in order to obtain a bipartite graph. In [24] it was shown that any tester for $\mathtt{BPG}_{3,0.01}$ must make $\Omega(\sqrt{N})$ queries (to the description of the graph, given as an oracle). In contrast, for every d and δ, the tester presented in [25] decides $\mathtt{BPG}_{d,\delta}$ in time $\widetilde{O}(\sqrt{N}/\text{poly}(\delta))$. In fact, this algorithm decides $\mathtt{BPG}_d$ in time $\widetilde{O}(\sqrt{N}/\text{poly}(\delta))$, where N and δ are explicitly given parameters.

Indeed, all research on property testing refers to the two aforementioned types of promise problems, where positive results typically refer to the first type and negative results to the second type. However, a strictly formal statement of the result is typically not provided, because it is rather cumbersome and believed to be unnecessary (especially in light of the greater focus on positive results). Indeed, property testing is positioned between algorithmic research and complexity theory, and seems to be more influenced by the algorithmic terminology. (We comment that the positioning of a discipline is determined both by its contents and by sociology-of-science factors.)

Let us consider what is required for a formal statement of property testing results. The starting point is a specification of a property and a distance function, the combination of which yields a promise problem (of the first type), although the latter fact is never stated. The first step is to specify that the potential "solvers" (i.e., property testers) are probabilistic oracle machines that are given oracle access to the "primary" input (i.e., the object in the aforementioned problem types). Indeed, this step need to be taken and is taken no matter how one states property testing results. Secondly, for a formal asymptotic complexity statement, one needs to specify the explicit inputs, which consist of various problem-dependent parameters (e.g., N in the above examples) and the distance parameter δ (in case of $\mathtt{BPG}_d$ and any other problem of the first aforementioned type). This step is rarely done explicitly. Finally, one states the complexity of the tester in terms of these explicit inputs.

4 Promise problems provide complete problems

The bulk of this section is devote to the key role that promise problems have played in the study of Statistical Zero-Knowledge proof systems. But we start with reviewing the situation in the seemingly lower complexity class $\mathcal{BPP}$.

4.1 A complete problem for BPP

In terms of language recognition, finding a complete problem for $\mathcal{BPP}$ is a long-standing challenge. The same hold for establishing hierarchy theorems for BPtime (cf. [6, 15]). However, in terms of promise problems, both challenges are rather easy (as is the case for analogous questions regarding $\mathcal{P}$). Indeed, the following promise problem is complete (under deterministic Karp-reductions) for the (promise problem) class $\mathcal{BPP}$: The YES-instances are Boolean circuits that evaluate to 1 on at least a 2/3 fraction of their inputs, whereas the NO-instances are Boolean circuits that evaluate to 0 on at least a 2/3 fraction of their inputs. (Thus, the promise "rules out" circuits that evaluate to 1 on a p fraction of their input, where $p \in (1/3, 2/3)$.) A reduction from $\Pi \in \mathcal{BPP}$ to the aforementioned promise problem merely maps x to C_x, where C_x is a circuit that on input r emulates the computation of M on input x and random-tape r, where M is a probabilistic polynomial-time machine deciding Π.

Needless to say, the above also holds with respect to other complexity classes that are aimed to capture efficient randomized computation (e.g., $\mathcal{RP}$ and $\mathcal{ZPP}$).

4.2 Complete problems for Statistical Zero-Knowledge

Statistical zero-knowledge (SZK) is a subclass of standard zero-knowledge (ZK, aka computational zero-knowledge), where the simulation requirement is more strict (i.e., requiring simulation that is statistically close to the true interaction rather than only computationally indistinguishable from it). For background see either [17] or [18]. Typically (as is the case in all results reviewed below), the study of SZK is carried out *without* referring to any intractability assumptions (in contrast to the study of standard ZK that is usually based on one-way functions; cf. [23] but see [54] for a recent exception).

Promise problem have played a key role in the comprehensive study of statistical zero-knowledge. (This study was carried out in the late 1990's and is nicely summarized in Vadhan's thesis [53].) This study of statistical zero-knowledge (SZK) was conducted by presenting and extensively studying two complete (promise) problems for the (promise problem) class SZK. Specifically, these promise problems facilitate the establishment of various important properties of the class SZK, because the definition of these promise problems is very simple in comparison to the actual definition of the class SZK. Furthermore, the fact that the class has natural complete problems is of independent interest.

The two aforementioned complete problems are gapSD and gapENT, introduced and shown complete for SZK in [49] and [29], respectively. Both problems refer to pairs of distributions, where each distribution is represented by a "sampling circuit" (i.e., a circuit C represents the distribution seen at its output wires when feeding the input wires with uniformly distributed values). The YES-instances of gapSD are distributions that are at (statistical) distance at most 1/3 apart, and the NO-instances are distributions that are at distance at least 2/3 apart. The YES-instances of gapENT are pairs of distributions in which the first distribution has entropy greater by one unit than the entropy of the second distribution, and in the NO-instances the first distribution has entropy that is smaller by one unit from the entropy of the second distribution.

To demonstrate the power of the complete problem approach to the study of SZK, note that the fact that gapENT is complete (under Karp-reductions) for SZK immediately implies that SZK is closed under complementation, which is a highly non-trivial result.

4.3 A more detailed presentation

In the rest of this section, we provide a more detailed presentation of the material presented above (in Section 4.2) and describe some of the ideas underlying the proof of central results. We start by recalling a few underlying notions.

The statistical difference (or variation distance) between the distributions (or the random variables) X and Y is defined as

$$\Delta(X,Y) \stackrel{\rm def}{=} \frac{1}{2} \cdot \sum_e |\Pr[X\!=\!e] - \Pr[Y\!=\!e]| \;=\; \max_S\{\Pr[X \in S] - \Pr[Y \in S]\} \qquad (3)$$

We say that X and Y are δ-close if $\Delta(X,Y) \le \delta$ and that they are δ-far if $\Delta(X,Y) \ge \delta$. Note that X and Y are identical if and only if they are 0-close, and are disjoint (or have disjoint support) if and only if they are 1-far. The entropy of a distribution (or random variables) X is defined as

$$\mathrm{H}(X) \stackrel{\mathrm{def}}{=} \sum_e \Pr[X=e] \cdot \log_2(1/\Pr[X=e])\,. \tag{4}$$

The entropy of a distribution is always non-negative and is zero if and only if the distribution is concentrated on a single element. In general, a distribution that has support size N has entropy at most $\log_2 N$.

The distribution represented (or generated) by a circuit $C : \{0,1\}^n \to \{0,1\}^m$ assigns each string $\alpha \in \{0,1\}^m$ probability $|\{s : C(s) = \alpha\}|/2^n$. The corresponding random variable is $C(U_n)$, where U_n denotes a random variable uniformly distributed over $\{0,1\}^n$. A function $\mu : \mathsf{N} \to [0,1]$ is called negligible if it decreases faster than any polynomial fraction; that is, for every positive polynomial p and all sufficiently large n it holds that $\mu(n) < 1/p(n)$. A function $\nu : \mathsf{N} \to [0,1]$ is called noticeable if $\nu(n) > 1/p(n)$ for some positive polynomial p and all sufficiently large n.

The class $\mathcal{SZK}$ consists of promise problems that have an interactive proof system that is "statistically zero-knowledge" (with respect to the honest verifier). Recall that proof systems are two-party protocols in which a computationally unbounded prover may convince a probabilistic polynomial-time verifier to accept YES-instances, whereas no prover may fool the verifier into accepting NO-instances. Both assertions hold with high probability, which can be amplified by repetitions.

Definition 4.1 ([14, 22], following [31]) *The two-party protocol (P,V) is called an* interactive proof system for the promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *if V is a probabilistic polynomial-time interactive machine and the following two conditions hold*

1. Completeness: *For any $x \in \Pi_{\mathrm{YES}}$, with probability at least $2/3$, the verifier V accepts after interacting with the prover P on common input x.*
2. Soundness: *For any $x \in \Pi_{\mathrm{NO}}$, with probability at least $2/3$, the verifier V rejects after interacting with* any strategy *on common input x.*

We denote by $\langle P,V\rangle(x)$ the local view of V when interacting with P on common input x, where the local view consists of x, the internal coin tosses of V, and the sequence of messages it has received from P. The proof system (P,V) is said to be statistical zero-knowledge *if there exists a probabilistic polynomial-time machine S, called a* simulator, *such that for $x \in \Pi_{\mathrm{YES}}$ the statistical difference between $\langle P,V\rangle(x)$ and $S(x)$ is negligible as a function of $|x|$.*

We stress that the completeness and zero-knowledge conditions refer only to YES-instances, whereas the soundness condition refers only to NO-instances. We mention that Definition 4.1 refers only to honest-verifiers, but it is known that any problem that has an interactive proof satisfying Definition 4.1 also has one that is statistical zero-knowledge in general (i.e., with respect to arbitrary verifiers); see [26, 29].

Definition 4.2 *The class $\mathcal{SZK}$ consists of all promise problems that have a statistical zero-knowledge interactive proof system.*

Recall that $\mathcal{SZK}$ contains some promise problems that are widely believed not to be in $\mathcal{BPP}$ (e.g., one equivalent to the Discrete Logarithm Problem, cf. [22]). On the other hand, $\mathcal{SZK} \subseteq \mathcal{AM} \cap \mathrm{co}\mathcal{AM}$ (cf. [14, 3]), which in turn lies quite low in the Polynomial-Time Hierarchy.

Approximating the distance between distributions. We consider promise problems that take as input a pair of circuits and refer to the statistical difference between the two corresponding distributions (generated by the two circuits). For (threshold) functions $c, f : \mathsf{N} \to [0,1]$, where $c \leq f$, the promise problem $\texttt{GapSD}^{c,f} = (\texttt{Close}^c, \texttt{Far}^f)$ is defined such that $(C_1, C_2) \in \texttt{Close}^c$ if $\Delta(C_1, C_2) \leq c(|C_1|+|C_2|)$ and $(C_1, C_2) \in \texttt{Far}^f$ if $\Delta(C_1, C_2) > f(|C_1|+|C_2|)$. In particular, we focus on promise problem $\texttt{GapSD} \stackrel{\text{def}}{=} \texttt{GapSD}^{\frac{1}{3},\frac{2}{3}}$. Interestingly, the complexity of the latter gap problem, which captures quite a good approximation requirement, is computationally equivalent to a very crude approximation requirement. That is, the former problem is Karp-reducible to the latter:

Theorem 4.3 [49]: *For some $\alpha > 0$, there exists a Karp-reduction of $\texttt{GapSD}^{\frac{1}{3},\frac{2}{3}}$ to $\texttt{GapSD}^{\epsilon,1-\epsilon}$, where $\epsilon(n) = 2^{-n^\alpha}$. More generally, for every polynomial-time computable $c, f : \mathsf{N} \to [0,1]$ such that $c(n) < f(n)^2 - (1/\text{poly}(n))$ it holds that $\texttt{GapSD}^{c,f}$ is Karp-reducible to $\texttt{GapSD}^{\epsilon,1-\epsilon}$.*

Using a trivial reduction in the other direction, we conclude that *for every $c, f : \mathsf{N} \to [0,1]$ such that $c(n) \geq 2^{-n^\alpha}$, $c(n) < f(n)^2 - (1/\text{poly}(n))$ and $f(n) \geq 1 - 2^{-n^\alpha}$, the problems $\texttt{GapSD}^{c,f}$ and $\texttt{GapSD} = \texttt{GapSD}^{\frac{1}{3},\frac{2}{3}}$ are computationally equivalent* (under Karp reductions). This equivalence is useful in determining the complexity of GapSD (as well as all these $\texttt{GapSD}^{c,f}$'s). Specifically, in order to show that $\mathcal{SZK}$ is Karp-reducible to GapSD, it is shown that $\mathcal{SZK}$ is Karp-reducible to $\texttt{GapSD}^{\frac{1}{2p^2},\frac{1}{p}}$, for some polynomial p. On the other hand, in order to show that GapSD is in $\mathcal{SZK}$, it is shown that for $\epsilon(n) = 2^{-n^\alpha}$ the problem $\texttt{GapSD}^{\epsilon,1-\epsilon}$ is in $\mathcal{SZK}$. Thus, one gets

Theorem 4.4 [49]: *The promise problem* GapSD *is $\mathcal{SZK}$-complete* (under Karp-reductions).

We stress that the promise problem nature of GapSD seems essential for showing that $\texttt{GapSD} \in \mathcal{SZK}$. On the other hand, the class $\mathcal{SZK}$ reduces naturally to a promise problem with a non-trivial promise. For details, see the outline of the proof ideas provided below.

Approximating the entropy of a distribution. We consider two computational problems related to approximating the entropy of a distribution. The first problem is captured by promise problems that take as input a circuit and a value and refers to the relation between the entropy of (the distribution generated by) the circuit and the given value. The second type of promise problems take as input a pair of circuits and refer to the relation between the entropies of the corresponding distributions (generated by the two circuits). Note that the two types of problems are computationally equivalent (i.e., each is Cook-reducible to the other). We focus on the second type of problems, because (unlike the first type) they are known to be complete for $\mathcal{SZK}$ under Karp-reductions. Specifically, for a positive (slackness) function $s : \mathsf{N} \to \mathsf{R}^+$, the promise problem $\texttt{GapENT}^s = (\texttt{Smaller}^s, \texttt{Larger}^s)$ is defined such that $(C_1, C_2) \in \texttt{Smaller}^s$ if $\text{H}(C_1) \leq \text{H}(C_2) - s(|C_1| + |C_2|)$ and $(C_1, C_2) \in \texttt{Larger}^f$ if $\text{H}(C_1) \geq \text{H}(C_2) + s(|C_1| + |C_2|)$. We focus on promise problem $\texttt{GapENT} \stackrel{\text{def}}{=} \texttt{GapENT}^1$, and mention the following two simple facts:

Fact 1: For every positive polynomial p and $\ell_\epsilon(n) = n^{1-\epsilon}$ for any $\epsilon > 0$, it holds that the problems $\texttt{GapENT}^{1/p}$, GapENT and $\texttt{GapENT}^{\ell_\epsilon}$ are computationally equivalent (under Karp reductions).

Fact 2: The problem $\texttt{GapENT} = (\texttt{Smaller}, \texttt{Larger})$ is Karp-reducible to its complement (i.e., the problem $(\texttt{Larger}, \texttt{Smaller})$): e.g., by the reduction that maps (C_1, C_2) to (C_2, C_1).

It turns out that the computational problems (regarding entropy) are computationally equivalent to the computational problems regarding statistical distance:

Theorem 4.5 [29]: *The promise problems* GapENT *and* GapSD *are computationally equivalent under Karp reductions.*

Combining Theorems 4.4 and 4.5, it follows that GapENT is $\mathcal{SZK}$-complete (under Karp-reductions).

Comments regarding the proofs of Theorems 4.3–4.5

The proofs of Theorems 4.3 and 4.5 rely on sophisticated manipulations of distributions (or rather the corresponding sampling circuits). Although these proofs are quite interesting, we focus on the proof of Theorem 4.4, which provides the bridge between the specific problems and the class $\mathcal{SZK}$. Indeed, the proof of Theorem 4.4 highlights the role of promise problems (with non-trivial promises) in the study of $\mathcal{SZK}$, whereas the proofs of Theorems 4.3 and 4.5 merely translate one promise problem (with a non-trivial promise) to another.

Theorem 4.4 was proven by Sahai and Vadhan [49], and here we sketch the ideas underlying their proof. Their proof consists of two parts: (1) showing that GapSD has a statistical zero-knowledge proof system, and (2) showing that any problem in $\mathcal{SZK}$ is Karp-reducible to GapSD.

The problem GapSD has a statistical zero-knowledge proof system: Using Theorem 4.3, it suffices to show such a proof system for $\texttt{GapSD}^{\epsilon,1-\epsilon}$, where $\epsilon : \mathsf{N} \to [0,1]$ is a negligible function (e.g., $\epsilon(n) = 2^{-n^{\alpha}}$ for some $\alpha > 0$). Actually, we present such a proof system for the complement problem (i.e., $(\texttt{Far}^{1-\epsilon}, \texttt{Close}^{\epsilon})$), and rely on the (highly non-trivial) fact that GapSD is reducible to its complement.[5] Following an idea that originates in [31, 23], the protocol proceeds as follows, with the aim of establishing that the two input distributions are far apart. The verifier selects one of the input distributions at random and presents the prover with a random sample generated according to this distribution. The verifier accepts if and only if the prover correctly identifies the distribution from which the sample was taken. Observe that if the input distributions are far apart then the prover can answer correctly with very high probability. On the other hand, if the input distributions are very close then the prover cannot guess the correct answer with probability significantly larger than 1/2. This establishes that the protocol is an interactive proof (and thus that GapSD is in co$\mathcal{AM}$). It can be shown that this protocol is actually statistical zero-knowledge, intuitively because the verifier learns nothing from the prover's correct answer which is a priori known to to the verifier (in case the two distributions are far apart).

Any problem in $\mathcal{SZK}$ is Karp-reducible to GapSD: We rely on Okamoto's Theorem by which any problem in $\mathcal{SZK}$ has a *public-coin*[6] statistical zero-knowledge proof system [43]. (We comment that an alternative proof of that theorem has subsequently appeared in [29], who showed that $\mathcal{SZK}$ is Karp-reducible to GapENT while the latter problem has a public-coin statistical zero-knowledge proof system.) We consider an arbitrary (*public-coin*) statistical zero-knowledge proof system. Following Fortnow [14], we observe a discrepancy between the behavior of the simulator on YES-instances versus NO-instances:

[5] This fact follows by combining Theorem 4.5 an Fact 2. An alternative proof of the fact that GapSD is reducible to its complement was given in [49], before Theorem 4.5 was stated (let alone proved). Another alternative, is to rely on an even earlier result of Okamoto by which $\mathcal{SZK}$ is closed under complementation [43].

[6] An interactive proof is said to be of the public-coin type if the verifier is required to send the outcome of any coin it tosses as soon as it sees it. In other words, the verifier's messages are uniformly distributed strings (of predetermined length), and the verifier's decision depends only on the messages exchanged (rather than on some secret random choices made by the verifier).

- In case the input is a YES-instance, the simulator outputs transcripts that are very similar to those in the real interaction. In particular, these transcripts are accepting and the verifier's behavior in them is as in a real interaction. In particular, resorting to the public-coin condition, this means that the verifier's messages in the simulation are (almost) uniformly distributed independently of prior messages.
- In case the input is a NO-instance, the simulator must output either rejecting transcripts or transcripts in which the verifier's behavior is significantly different from the verifier's behavior in a real interaction. In particular, the only way the simulator can produce accepting transcripts is by producing transcripts in which the verifier's messages are not "random enough" (i.e., they depend, in a noticeable way, on previous messages).

Thus assuming, without loss of generality, that the simulator only produces accepting transcripts, we consider two types of distributions. The first type of the distributions is obtained by truncating a random simulator-produced transcript at a random "location" (after some verifier message), whereas the second type is obtained by doing the same while replacing the last verifier message by a random one. Note that both distributions can be implemented by polynomial-size circuits that depend on the input to the proof system being analyzed (and that these two circuits can be constructed in polynomial-time given the said input). The key observation is that if the input is a YES-instance then the two corresponding distributions will be very close, whereas if the input is a NO-instance then there will be a noticeable distance between the two corresponding distributions. Thus, we reduced any problem having a (public-coin) statistical zero-knowledge proof system to $\texttt{GapSD}^{\mu,\nu}$, where μ is a negligible function and $\nu(n)$ is a noticeable function. The proof is completed by using Theorem 4.3 (while noting that $\mu(n) < \nu(n)^2 - (1/\text{poly}(n))$).

Alternative proofs of Theorems 4.4 and 4.5: In sketching the proof of Theorem 4.4, we relied on two theorems of Okamoto [43]: The closure of $\mathcal{SZK}$ under complementation and the existence of public-coin statistical zero-knowledge proof systems for any problem in $\mathcal{SZK}$. Since Okamoto's arguments are hard to follow, it is worthwhile noting that an alternative route does exist. In [29] it is proved that gapENT is $\mathcal{SZK}$-complete (under Karp-reductions), without relying on Okamoto's results (but while using some of his ideas). Furthermore, the statistical zero-knowledge proof system presented for gapENT is of the public-coin type. Thus, the two aforementioned theorems of Okamoto follow (using the fact that gapENT is easily reducible to its complement). Consequently, the proof of Theorem 4.4 need not refer to Okamoto's paper [43]. (Theorem 4.5 follows immediately from the fact that both gapENT and gapSD are $\mathcal{SZK}$-complete, but a direct proof is possible by employing the ideas underlying [29, 49].)

5 Promise problems as indicators of complexity: Pros and Cons

Given the common desire to appeal to traditional notions, one typically tries to avoid promise problems and formulate the assertions in terms of language recognition problems. As we have seen in previous sections, in some cases this desire can not be satisfied due to inherent (or seemingly inherent) reasons. In other cases, one turns to promise problems after failing to prove an analogous result for language recognition problems, although there seems to be no inherent reason to justify the failure (see examples in Section 5.2). The question, however, is whether we lose something important when working with promise problems (rather than with language recognition problems). Since we have already seen some of the benefit of promise problems, we start by considering the dark side (i.e., the latter question).

5.1 Con: the failure of some structural consequences

The problem with results regarding promise problems is that sometimes they do not have the same *structural* consequences as analogous results regarding language recognition. The most notorious example is that the existence of an $\mathcal{NP}$-hard (under Cook reductions) promise problem in $\mathcal{NP} \cap \mathrm{co}\mathcal{NP}$ does not seem to have any structural consequences, whereas an analogous result for a language recognition problem implies that $\mathcal{NP} = \mathrm{co}\mathcal{NP}$ (see Theorem 5.2 below). This fact was observed by Even, Selman and Yacobi [12], who presented the following $\mathcal{NP}$-complete problem, denoted $\mathtt{xSAT}$: The YES-instances are pairs (ϕ_1, ϕ_2) such that $\phi_1 \in \mathtt{SAT}$ and $\phi_2 \notin \mathtt{SAT}$, whereas the NO-instances are pairs (ϕ_1, ϕ_2) such that $\phi_1 \notin \mathtt{SAT}$ and $\phi_2 \in \mathtt{SAT}$.

Theorem 5.1 [12, Thm. 4]: *$\mathcal{NP}$ is Cook-reducible to $\mathtt{xSAT}$, which in turn is in $\mathcal{NP} \cap \mathrm{co}\mathcal{NP}$.*

Proof sketch: To see that $\mathtt{xSAT}$ is in $\mathcal{NP}$, consider the witness relation $R_1 = \{((\phi_1, \phi_2), \tau) : (\phi_1, \tau) \in R_{\mathtt{SAT}}\}$, whereas $\mathtt{xSAT}$ is in co$\mathcal{NP}$ by virtue of the witness relation $R_2 = \{((\phi_1, \phi_2), \tau) : (\phi_2, \tau) \in R_{\mathtt{SAT}}\}$. A Cook-reduction of $\mathtt{SAT}$ to $\mathtt{xSAT}$ may consist of the following oracle machine that, on input a formula ϕ, tries to find a satisfying assignment to ϕ, and accepts if and only if it succeeds. On input ϕ and oracle access to $\mathtt{xSAT}$, the machine proceeds as follows, starting with $\phi_\lambda \stackrel{\rm def}{=} \phi$ and $\tau = \lambda$ (the empty prefix of a potential satisfying assignment), and continuing as long as ϕ_τ has free variables.

1. Let $\phi_{\tau\sigma}$ be the formula obtained from ϕ_τ by setting the $|\tau| + 1^{\rm st}$ variable to σ.
2. Invoke the oracle on query $(\phi_{\tau 1}, \phi_{\tau 0})$. If the answer is 1 then let $\tau \leftarrow \tau 1$, otherwise $\tau \leftarrow \tau 0$.

Note that if ϕ_τ is satisfiable and the query $(\phi_{\tau 1}, \phi_{\tau 0})$ is answered with σ then $\phi_{\tau\sigma}$ is satisfiable, because the claim holds trivially if both $\phi_{\tau 1}$ and $\phi_{\tau 0}$ are satisfiable, and the oracle answer is definitely correct if only one of these formulae is satisfiable (since the promise is satisfied in this case). Thus, the above process finds a satisfying assignment to ϕ if and only if one exists. ■

What happened? We stress that a Cook-reduction to a promise problem does maintain the standard meaning of the concept; that is, if the target (promise) problem is tractable (i.e., is in $\mathcal{P}$ or $\mathcal{BPP}$) then so is the reduced problem. The issue is that if the target problem is in $\mathcal{NP} \cap \mathrm{co}\mathcal{NP}$ then (unlike in the case of trivial promises (i.e., language recognition problems)) it does not necessarily follow that the reduced problem is in $\mathcal{NP} \cap \mathrm{co}\mathcal{NP}$. This fact will be clarified by looking at the proof Theorem 5.2, which refers to "smart reductions" to promise problems.

Smart reduction. Note that the reduction used in the proof of Theorem 5.1 may make queries that violate the promise (something that cannot possibly happen in the case the promise is trivial). Still, we have shown that the reduction remains valid regardless of the answers given to these queries (i.e., to queries that violate the promise). One may eliminate the problems arising from such queries by requiring that the reduction does not make them (i.e., does not make queries that violate the promise). Such a reduction is called smart [32] (probably because it is smart to avoid making queries that violate the promise, although one may argue that it is even more clever to be able to use answers to such queries). Note that any Karp-reduction is smart. Smart reductions maintain the situation established in the case of language recognition problems.

Theorem 5.2 [32, Thm. 2]: *Suppose that the promise problem Π' is reducible to the promise problem $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ via a smart reduction, and that $\Pi \in \mathcal{NP} \cap \mathrm{co}\mathcal{NP}$. Then $\Pi' \in \mathcal{NP} \cap \mathrm{co}\mathcal{NP}$.*

Proof sketch: We prove that $\Pi' \in \mathcal{NP}$ and the proof that $\Pi' \in \text{co}\mathcal{NP}$ is similar. Let M be the (polynomial-time) oracle machine guaranteed by the hypothesis. The transcript of the execution of $M^{\Pi}(x)$ contains the sequence of queries and answers to the oracle as well as the final decision of M, but the transcript itself (as a string) does not guarantee the correctness of the answers and thus the authenticity of the execution. The key observation is that the said answers can be augmented by corresponding NP-witnesses that guarantee the correctness of the answers, and thus the authenticity of the execution.

Specifically, on any input x (which satisfies the promise of Π'), machine M makes queries that are either in Π_{YES} or in Π_{NO}, and in each of these cases there is an NP-witness guaranteeing the correctness of the answer (because $\Pi \in \mathcal{NP}$ and $\Pi \in \text{co}\mathcal{NP}$). Thus, an NP-witness for x may consist of the sequence of (answers and) corresponding NP-witnesses, each proving either that the query is in Π_{YES} or that the query is in Π_{NO}, thus certifying the correctness of the answers. Indeed, these NP-witnesses are all correct, because it is guaranteed that each query satisfies the promise (since the reduction is smart). Note that this sequence of NP-witnesses uniquely determines the execution of M, on input x and oracle access to Π, and thus vouches for the correctness of the outcome of this computation. ■

In contrast to the proof of Theorem 5.2, note that a query that violates the promise does not necessarily have an NP-witness (e.g., asserting that it violates the promise, or anything else). Thus, we cannot insist on having NP-witnesses for all queries, and once we allow "uncertified answers" (i.e., answers not backed by NP-witnesses) all bets are off.

Another look. Indeed, smart reduction salvage the structural consequences of reductions to language recognition problems, but this comes at the cost of restricting the consequences to smart reductions. That is, for $\Pi \in \mathcal{NP} \cap \text{co}\mathcal{NP}$, rather than saying "if Π is NP-hard then $\mathcal{NP} = \text{co}\mathcal{NP}$" one may only say "if Π is NP-hard under smart reductions then $\mathcal{NP} = \text{co}\mathcal{NP}$". However, there is another way out, provided we know more about the promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$. For example, suppose that in addition to knowing that $\Pi \in \mathcal{NP} \cap \text{co}\mathcal{NP}$, we know that the set Π_{NO} is in $\text{co}\mathcal{NP}$ (i.e., $(\{0,1\}^* \setminus \Pi_{\text{NO}}, \Pi_{\text{NO}}) \in \mathcal{NP}$). Then, we can ask for NP-witnesses asserting either membership in $\{0,1\}^* \setminus \Pi_{\text{NO}}$ or membership in Π_{NO}.

Theorem 5.3 (implicit in [11], see [19]):[7] *Suppose that the promise problem Π' is reducible to the promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \text{co}\mathcal{NP}$ and that $(\{0,1\}^* \setminus \Pi_{\text{NO}}, \Pi_{\text{NO}}) \in \mathcal{NP}$. Then $\Pi' \in \mathcal{NP} \cap \text{co}\mathcal{NP}$.*

Note that $(\{0,1\}^* \setminus \Pi_{\text{NO}}, \Pi_{\text{NO}}) \in \mathcal{NP}$ implies that $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \mathcal{NP}$, and thus the latter was not stated as a hypothesis in Theorem 5.3. To demonstrate the applicability of Theorem 5.3, we mention that it was recently shown (cf. [2] improving upon [20]) that certain promise problems (i.e., gap problems) regarding lattices are in $\mathcal{NP} \cap \text{co}\mathcal{NP}$. It is actually obvious that the set of the corresponding NO-instances is in $\text{co}\mathcal{NP}$. Applying Theorem 5.3, it follows that these (gap) problems are unlikely to be $\mathcal{NP}$-hard (rather than restricting the claim to smart reductions).

[7] This theorem is implicit in [11], which observes an oversight of [20]. In [20] certain gap problems regarding lattices were shown to be in $\mathcal{NP} \cap \text{co}\mathcal{AM}$, and it was inferred that these (gap) problems are unlikely to be $\mathcal{NP}$-hard under smart reductions (because such a reduction will imply that $\mathcal{AM} = \text{co}\mathcal{AM}$, which in turn will cause collapse of the Polynomial-time Hierarchy). In [11] it was observed that these problems are unlikely to be $\mathcal{NP}$-hard (under any Cook-reduction). Specifically, they showed that, for these gap problems, the argument of Theorem 5.2 can be extended using NP-witnesses that exist for the corresponding set $\{0,1\}^* \setminus \Pi_{\text{NO}}$. This argument was abstracted in [19], where a theorem analogous to Theorem 5.3 is presented (referring to $\mathcal{AM}$ rather than to $\mathcal{NP}$).

Proof sketch: Following the proof of Theorem 5.2, an NP-witness for x may consist of the sequence of (answers and) corresponding NP-witnesses, each "proving" either that the query is in $\{0,1\}^* \setminus \Pi_{\rm NO}$ or that the query is in $\Pi_{\rm NO}$. Note that these witnesses exist for every query, but indeed, in case the query violates the promise, witnesses may exist to both claims. Still, these witnesses do guarantee the correctness of all answers to queries that satisfy the promise (although they do not indicate which queries satisfy the promise). However, guaranteeing the correctness of all queries that satisfy the promise suffices for guaranteeing the correctness of the outcome of the computation. Thus, although the sequence of witnesses does not determine (uniquely) the execution of M on input x and oracle access to Π, it does vouch for the correctness of the outcome of the computation. ■

Generalization of Theorem 5.3. The following elegant generalization of Theorem 5.3 was suggested to us by Salil Vadhan. It considers two sets, $S_{\rm Y}$ and $S_{\rm N}$, such that $S_{\rm Y}$ (resp., $S_{\rm N}$) contains all YES-instances (resp., NO-instances) of Π but none of the NO-instances (resp., YES-instances).

Theorem 5.4 (Vadhan [priv. comm.]): *Let $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ be a promise problem, and $S_{\rm Y}$ and $S_{\rm N}$ be sets such that $S_{\rm Y} \cup S_{\rm N} = \{0,1\}^*$, $\Pi_{\rm YES} \subseteq S_{\rm Y} \subseteq \{0,1\}^* \setminus \Pi_{\rm NO}$ and $\Pi_{\rm NO} \subseteq S_{\rm N} \subseteq \{0,1\}^* \setminus \Pi_{\rm YES}$. Suppose that the promise problems $(S_{\rm Y}, \Pi_{\rm NO})$ and $(S_{\rm N}, \Pi_{\rm YES})$ are both in $\mathcal{NP}$. Then, every promise problem that is Cook-reducible to Π, is in $\mathcal{NP} \cap {\rm co}\mathcal{NP}$.*

We stress that $S_{\rm Y}$ and $S_{\rm N}$ cover the set of all strings but are not necessarily a partition of it (i.e., $S_{\rm Y} \cup S_{\rm N} = \{0,1\}^*$ but $S_{\rm Y} \cap S_{\rm N}$ may be non-empty). Theorem 5.3 is obtained as a special case by considering $S_{\rm Y} = \{0,1\}^* \setminus \Pi_{\rm NO}$ and $S_{\rm N} = \Pi_{\rm NO}$. The proof of Theorem 5.4 generalizes the proof of Theorem 5.3: the answer to each query is augmented by a corresponding NP-witness (asserting either membership in $S_{\rm Y}$ or membership in $S_{\rm N}$). Again, "witnesses" exist for each query, and they are guaranteed to be correct in case the query satisfies the promise.

5.2 Pro: shedding light on questions concerning complexity classes

Recall that working with promise problems (rather than with language recognition problems) may result in the loss of some structural consequence. We stress, however, that the most fundamental feature of general reductions is maintained: if a problem is reducible to a tractable problem, then the former is also tractable. We note that the results in this subsection address the issue of tractability. (An analogue comment refers to all previous sections, with the exception of Section 4. Regarding the latter, we note that the structural results obtained for the promise problem class $\mathcal{SZK}$ yield similar results for the corresponding language recognition class. For further discussion, see Section 8.)

As mentioned in Section 3, promise problems play an important role in stating results regarding approximation problems. We refer specifically to the notion of a gap problem discussed in Section 3. We have also discussed the fundamental role of promise problems in providing the framework for the study of the complexity of "unique solutions" (see Section 2.1) and in providing complete problems for the classes $\mathcal{BPP}$ and $\mathcal{SZK}$ (see Section 4, which emphasizes the important consequences of these complete problems for the study of $\mathcal{SZK}$). Here we discuss promise problems that were not covered by each of the above cases: These promise problems are neither gap problems (at least not in a natural sense), nor problems of "unique solutions" or problems that are complete for any natural class. Still, they are important for the study of some natural complexity classes. Specifically, they indicate (or provide evidence) to separations between complexity classes that represent the computing power of certain computational devices with certain resource bounds.

Separating monotone and non-monotone circuit complexities. Several researchers have observed that Razborov's celebrated super-polynomial lower-bound on the monotone circuit complexity of MaxClique [45, Thm. 2] actually establishes a lower-bound on a promise problem that is in $\mathcal{P}$.[8] Thus, this result actually establishes a super-polynomial separation between the monotone and non-monotone circuit complexities (of a monotone problem). Actually, a less-known result in the same paper [45, Thm. 3] asserted a similar lower-bound for Perfect Matching (cf. [46]), and so the said separation could have been established by a language recognition problem (but not by the more famous result of [45]). Interestingly, the clique lower-bound was improved to exponential in [4], but a similar result was not known for Perfect Matching. Thus, at that time, an *exponential separation of the monotone and non-monotone circuit complexities required referring to a promise problem* (i.e., the one mentioned in Footnote 8). Subsequently, an exponential separation was shown by providing an exponential lower-bound on the monotone complexity of some other polynomial-time computable (monotone) function [52].

The derandomization of BPP versus the derandomization of MA. One obvious fact, rarely noted, is that results about derandomization of $\mathcal{BPP}$ imply results on the derandomization of $\mathcal{MA}$, where $\mathcal{MA}$ is the class of problems having a "randomized verification procedure" (i.e., the analogue of $\mathcal{NP}$ in which the validity of witnesses is determined by a probabilistic polynomial-time algorithm rather than by a deterministic polynomial-time algorithm). This observation holds provided that the former derandomization results relate to $\mathcal{BPP}$ as a class of promise problems (as in Definition 1.2) rather than as to a class of language recognition problems. We note that all known derandomization results have this property. In any case, in terms of promise problem classes, we have that $\mathcal{BPP} \subseteq \mathrm{DTIME}(t)$ implies $\mathcal{MA} \subseteq \mathrm{NTIME}(\mathrm{poly}(t))$, provided that the function t is "nice". Specifically, $\mathcal{BPP} = \mathcal{P}$ implies $\mathcal{MA} = \mathcal{NP}$. For details see [30, Sec. 5.4] (or the Appendix).

Relations among PCP classes. Some of the appealing transformations among PCP classes are only known when these classes are defined in terms of promise problems (see, e.g., [7, Sec. 11] and [28, Sec. 4]). For example, the intuitive meaning of [7, Prop. 11.2] is that the randomness in a PCP can be reduced to be logarithmic in the length of the proof oracle, but the actual result is a randomized Karp reduction of any problem having a PCP to a *promise problem* having a PCP with the same query (and/or free-bit) complexity and proof-length but with logarithmic randomness. Similarly, the main PCP result of [28, Sec. 4] is a almost-linear length PCP not for SAT but rather for a promise problem to which SAT can be randomly Karp-reduced (by an almost length preserving reduction). We mention that the random reduction was eliminated by the subsequent work of [9].

Supporting the conjectured non-triviality of statistical zero-knowledge. Seeking to provide further evidence to the conjectured non-triviality of statistical zero-knowledge (i.e., the conjecture that $\mathcal{SZK}$ extends beyond $\mathcal{BPP}$), researchers tried to show statistical zero-knowledge proof systems for "hard" (language recognition) problems. At the time (i.e., late 1980's), it was known that Quadratic Residuocity and Graph Isomorphism are in $\mathcal{SZK}$ (cf., [31] and [23], respectively), but the belief that these problems are hard seems weaker than the belief that factoring integers or the Discrete Logarithm Problem are hard. So the goal was to present a statistical zero-knowledge proof system for a language recognition problem that is computationally equivalent to any of these search problems. This was almost done in [22], who showed an *analogous result for a promise*

[8] For $k \approx N^{2/3}$, this promise problem has YES-instances that are N-vertex graphs having a clique of size k, and NO-instances that are complete $(k-1)$-partite N-vertex graphs.

problem. Specifically, they presented a statistical zero-knowledge proof for a promise problem that is computationally equivalent to the Discrete Logarithm Problem.[9] We mention that the "gap" between YES and NO instances in this promise problem, plays a key role in showing that this problem is in $\mathcal{SZK}$. Thus, *based on this promise problem, the non-triviality of $\mathcal{SZK}$ is supported by the conjectured intractability of the Discrete Logarithm Problem.*

Following a great tradition. The last example follows a central tradition in the closely related field of Cryptography, where one often considers promise problems. These problems are often search problems that refer to inputs of a special form (although computationally equivalent decision (promise) problems are sometimes stated too). Typical examples include "cryptanalyzing" a sequence of ciphertexts that are "promised" to have been produced using the same encryption-key, and factoring an integer that in the product of two primes of approximately the same size. Indeed, these examples were among the concrete motivations to the definition of promise problems introduced by Even, Selman and Yacobi [12], following prior work of Even and Yacobi [13]. The latter paper (combined with [41]) has also demonstrated that NP-hardness (i.e., worst-case hardness) of the "cryptanalysis" task is a poor evidence for cryptographic security. Indeed, subsequent works in cryptography typically relate to the average-case complexity of "cryptanalysis", and the "promise problem nature" of the task is incorporated (implicitly) in the formulation by assigning zero (probability) weight to instances that violate the promise.

And something completely different. Finally, we mention the role of promise problems in the study of Quantum Computation and Communication. I am referring to two elegant mathematical models of controversial relevance to the theory of computation, and admit that I do not understand the real meaning of these models. Still, I am told that the very definition of "Quantum NP-completeness" refers to promise problems, and that the known complete problems are all promise problems (see, e.g., [37, 38] where the names QBNP and QMA are used). As for Quantum Communication, the only separations known between the power of classical and quantum communication complexity is for a promise problem (see, e.g., Raz's paper [44]).

6 Promise problems as facilitators of nicer presentation

In previous sections, we have discussed the role of promise problems in providing a framework for several natural studies and in enabling several appealing results (e.g., complete problems for $\mathcal{SZK}$). In the current section we focus on their role as facilitators of nicer presentation of various results. We believe than an explicit use of promise problems in such cases clarifies the argument as well as reveals its real essence. We start with a rather generic discussion, and later turn to one concrete example (i.e., the well-known result $\mathcal{BPP} \subseteq \mathcal{PH}$).

[9] More specifically, they considered the promise problem $\mathtt{DLP} = (\mathtt{High}, \mathtt{Low})$, where the promise (i.e., $\mathtt{High} \cup \mathtt{Low}$) consisted of triples (p, g, y) such that p is a prime, g is a primitive element of Z_p^*, and $y = g^x \bmod p$ for $x \in H_p \cup L_p$, where $H_p = [(p/2)+1, ..., (p/2) + (p/(100 \log_2 p))]$ and $L_p = [1, ..., (p/(100 \log_2 p))]$. The problem is to distinguish the case that $x \in H_p$ from the case that $x \in L_p$ (i.e., the decision refers to the "half" bit of the Discrete Logarithm). That is, $(p, g, y) \in \mathtt{High}$ corresponds to $x \in H_p$, whereas $(p, g, y) \in \mathtt{Low}$ corresponds to $x \in L_p$. It is easy to show that the promise problem DLP is computationally equivalent to the Discrete Logarithm Problem (in which, given (p, g, y), where p and q are as above, the task is to find x such that $y = g^x \bmod p$). The main result of [22] is that DLP has a statistical zero-knowledge proof system, and the "good distance" between the "high" and "low" intervals (guaranteed by the promise) plays a key role in their proof system.

6.1 Presenting lower-bound arguments

Numerous lower-bound arguments proceed by focusing on special cases of the original decision problem. As stated in Section 1.3, these special cases are promise problems. To be concrete, we refer to an example mentioned in Section 5.2: Razborov's lower-bound on the monotone circuit complexity of MaxClique [45, Thm. 2] is commonly presented as a lower-bound on a promise problem that, for $k \approx N^{2/3}$, has YES-instances that are N-vertex graphs having a clique of size k, and NO-instances that are complete $(k-1)$-partite N-vertex graphs.

A similar strategy is adopted in numerous works (which are too numerous to be cited here). The benefit of this strategy is that it introduces additional structure that facilitates the argument. In some cases the act of restricting attention to special cases is even repeated several times. Needless to say, a proper formulation of this process involves the introduction of promise problems (which correspond to these special cases). It also relies on the trivial fact that any "solver" of a problem also solves its special cases (i.e., if some device solves the promise problem $(\Pi_{\rm YES}, \Pi_{\rm NO})$ then it also solves any $(\Pi'_{\rm YES}, \Pi'_{\rm NO})$ that satisfies both $\Pi'_{\rm YES} \subseteq \Pi_{\rm YES}$ and $\Pi'_{\rm NO} \subseteq \Pi_{\rm NO}$).

The use of promise problems becomes almost essential when one proves a lower-bound by a reduction from a known lower-bound for a promise problem, and the reduction uses the promise in an essential way. Consider, for example, the seperation between rank and communication complexity proven by Nisan and Wigderson [42]. Their communication complexity lower bound is by a reduction of "unique disjointness" to their communication problem, while noting that the linear lower-bound on disjointness established by Razborov [47] holds also for the promise problem "unique disjointness" (where the sets are either disjoint or have an single emelent in their intersection). We stress that their reduction of unique disjointness uses the promise in an essential way (and may fail for instances that violate the promise).

6.2 BPP is in the Polynomial-time Hierarchy, revisited

It is well-known that $\mathcal{BPP}$ is in the Polynomial-time Hierarchy (see proofs by Lautemann [40] and Sipser [50]). However, the known proofs actually establish stronger results. In my opinion, the nicest formulation of the result (as well as the one that best distills the essence of the proofs) is in terms of promise problem classes. Specifically, we consider the extension of the classes $\mathcal{RP}$ and $\mathcal{BPP}$ to promise problems, and show that $\mathcal{BPP} = \mathcal{RP}^{\mathcal{RP}}$.

Recall that $\mathcal{BPP}$ is the class of promise problems that are solvable by a *two-sided* error probabilistic polynomial-time algorithm, whereas $\mathcal{RP}$ and co$\mathcal{RP}$ refer to problems that are solvable by a *one-sided* error probabilistic polynomial-time algorithm. Specifically, $\Pi \in \mathcal{RP}$ (resp., $\Pi \in {\rm co}\mathcal{RP}$) if there exists a *probabilistic* polynomial-time algorithm A such that

- For every $x \in \Pi_{\rm YES}$ it holds that $\Pr[A(x)=1] \geq 1/2$ (resp., $\Pr[A(x)=1]=1$).
- For every $x \in \Pi_{\rm NO}$ it holds that $\Pr[A(x)=0]=1$ (resp., $\Pr[A(x)=0] \geq 1/2$).

It is evident that $\mathcal{RP}^{\mathcal{RP}} \subseteq \mathcal{BPP}^{\mathcal{BPP}} = \mathcal{BPP}$ (where the last equality utilizes standard "error reduction"). Thus, we focus on the other direction (i.e., $\mathcal{BPP} \subseteq \mathcal{RP}^{\mathcal{RP}}$), following the proof ideas of Lautemann [40] (or rather presenting his proof using a different formalism).

Theorem 6.1 ([10]): *Any problem in $\mathcal{BPP}$ is reducible by a one-sided error randomized Karp-reduction to co$\mathcal{RP}$.*

Proof: Consider any BPP-problem with a characteristic function χ (which, in case of a promise problem, is a partial function, defined only over the promise). That is, for some probabilistic polynomial-time algorithm A and for every x on which χ is defined it holds that $\Pr[A(x) \neq \chi(x)] \leq 1/3$. Thus, for some polynomial p and some polynomial-time recognizable relation $R \subseteq \cup_{n\in\mathsf{N}}(\{0,1\}^n \times \{0,1\}^{p(n)})$ and for every x on which χ is defined it holds that

$$\Pr_{r\in\{0,1\}^{p(|x|)}}[R(x,r)\neq\chi(x)] \leq \frac{1}{3} \tag{5}$$

where $R(x,y)=1$ if $(x,y)\in R$ and $R(x,y)=0$ otherwise. By straightforward "error reduction" we have, for some other NP-relation R and polynomial p,

$$|\{r \in \{0,1\}^{p(|x|)} : R(x,r)\neq\chi(x)\}| < \frac{2^{p(|x|)}}{2p(|x|)} \tag{6}$$

We show a randomized one-sided error (Karp) reduction of χ to (the promise problem extension of) co$\mathcal{RP}$. We start by stating the simple reduction, and next define the target promise problem.

The reduction: On input $x \in \{0,1\}^n$, the randomized polynomial-time mapping uniformly selects $s_1,...,s_m \in \{0,1\}^m$, and outputs the pair $(x,\overline{s})$, where $m=p(|x|)$ and $\overline{s}=(s_1,...,s_m)$.

The promise problem: We define the following co$\mathcal{RP}$ promise problem, denoted $\Pi=(\Pi_{\mathrm{YES}},\Pi_{\mathrm{NO}})$.

- The yes-instances are pairs $(x,\overline{s})$ such that for every $r\in\{0,1\}^m$ there exists an i satisfying $R(x,r\oplus s_i)=1$, where $\overline{s}=(s_1,...,s_m)$ and $m=p(|x|)$.
- The no-instances are pairs $(x,\overline{s})$ such that for at least half of the possible $r\in\{0,1\}^m$, it holds that $R(x,r\oplus s_i)=0$ for every i, where again $\overline{s}=(s_1,...,s_m)$ and $m=p(|x|)$.

To see that Π is indeed a co$\mathcal{RP}$ promise problem, we consider the following randomized algorithm. On input $(x,(s_1,...,s_m))$, where $m=p(|x|)=|s_1|=\cdots=|s_m|$, the algorithm uniformly selects $r\in\{0,1\}^m$, and accepts if and only if $R(x,r\oplus s_i)=1$ for some $i\in\{1,...,m\}$. Indeed, yes-instances of Π are accepted with probability 1, whereas no-instances are rejected with probability at least 1/2.

Analyzing the reduction: We claim that the above randomized mapping, denoted M, reduces χ to Π. Specifically, we will prove:

Claim 1: If x is a yes-instance (i.e., $\chi(x)=1$) then $\Pr[M(x)\in\Pi_{\mathrm{YES}}] > 1/2$.

Claim 2: If x is a no-instance (i.e., $\chi(x)=0$) then $\Pr[M(x)\in\Pi_{\mathrm{NO}}]=1$.

We start with Claim 2, which refers to $\chi(x)=0$ (and is easier to establish). Recall that $M(x)=(x,(s_1,...,s_m))$, where $s_1,...,s_m$ are uniformly and independently distributed in $\{0,1\}^m$. Observe that (by Eq. (6)), for every possible choice of $s_1,...,s_m\in\{0,1\}^m$ and every $i\in\{1,...,m\}$, the fraction of r's that satisfy $R(x,r\oplus s_i)=1$ is at most $\frac{1}{2m}$. Thus, for every possible choice of $s_1,...,s_m\in\{0,1\}^m$, the fraction of r's for which there exists an i such that $R(x,r\oplus s_i)=1$ holds is at most $m\cdot\frac{1}{2m}=\frac{1}{2}$. Thus, the reduction always maps such an x to a no-instance of Π (i.e., an element of Π_{NO}).

Turning to Claim 1 (which refers to $\chi(x)=1$), we will show shortly that in this case, with very high probability, the reduction maps x to a yes-instance of Π. We upper-bound the probability that the reduction fails (in case $\chi(x)=1$):

$$\begin{aligned}\Pr[M(x)\notin\Pi_{\mathrm{YES}}] &= \Pr_{s_1,\ldots,s_m}[\exists r\in\{0,1\}^m \text{ s.t. } (\forall i)\ R(x,r\oplus s_i)=0] \\ &\leq \sum_{r\in\{0,1\}^m}\Pr_{s_1,\ldots,s_m}[(\forall i)\ R(x,r\oplus s_i)=0] \\ &\leq 2^m\cdot\left(\frac{1}{2m}\right)^m \ll \frac{1}{2}\end{aligned}$$

Thus, the randomized mapping M reduces χ to Π, with one-sided error on YES-instances. Recalling that $\Pi \in \text{co}\mathcal{RP}$, the theorem follows. ■

Comment: The traditional presentation uses the above reduction to show that $\mathcal{BPP}$ is in the Polynomial-Time Hierarchy. One defines the polynomial-time computable predicate $\varphi(x,\overline{s},r) \stackrel{\text{def}}{=} \bigvee_{i=1}^{m}(R(x,s_i \oplus r)=1)$, and observes that

$$\chi(x)=1 \quad \Rightarrow \quad \exists \overline{s}\, \forall r \;\; \varphi(x,\overline{s},r) \tag{7}$$

$$\chi(x)=0 \quad \Rightarrow \quad \forall \overline{s}\, \exists r \;\; \neg\varphi(x,\overline{s},r) \tag{8}$$

Note that Claim 1 establishes that *most* sequences $\overline{s}$ satisfy $\forall r\, \varphi(x,\overline{s},r)$, whereas Eq. (7) only requires the existence of *at least one* such $\overline{s}$. Similarly, Claim 2 establishes that for every $\overline{s}$ *most* choices of r violate $\varphi(x,\overline{s},r)$, whereas Eq. (8) only requires that for every $\overline{s}$ there exists *at least one* such r.

7 Using promise problems to define modified complexity classes

In continuation to Section 6, we survey a recent suggestion of Vadhan for defining "modified" complexity classes in terms of promise problems [55]. We refer to classes such as $\mathcal{BPP}/\log$ and io-$\mathcal{BPP}$ (i.e., "computations that take advice" and "infinitely often" classes). We comment that such classes are typically defined by modification to the operation of the computing device (or the conditions applied to its computations). Vadhan's suggestion is to define such classes by modification to the class itself, provided that the (resulting) class is understood as a class of promise problems. Indeed, this approach is sometimes taken with respect to language classes like $\mathcal{P}/\text{poly}$ but the extension to $\mathcal{BPP}$ and other probabilistic classes seems to require the use of promise problems. (Indeed, in view of the fact that $\mathcal{BPP}/\text{poly} = \mathcal{P}/\text{poly}$ (cf. [1]), we demonstrate the approach with respect to $\mathcal{BPP}/\log$ (which is the focus of some recent studies [6, 15]).)

7.1 Probabilistic machines that take advice

Recall that $\mathcal{P}/\log$ may be defined as the class of decision problems that can be solved by a deterministic polynomial-time machine M provided that M obtains an adequate advice (of logarithmic length). Specifically, focusing on language classes, one says that $L \in \mathcal{P}/\log$ if there exists a deterministic polynomial-time machine M and sequence $a_1, a_2, ...$ such that $|a_n| = \log n$ and $M(x, a_{|x|}) = \chi_L(x)$ for every x, where $\chi_L(x) = 1$ if and only if $x \in L$. Equivalently, one may say that $L \in \mathcal{P}/\log$ if and only if there exists a language $L' \in \mathcal{P}$ and a sequence $a_1, a_2, ...$ such that $|a_n| = \log n$ and $x \in L$ if and only if $(x, a_{|x|}) \in L'$. Indeed, L' can be defined as the language accepted by the aforementioned machine M.

The latter approach seems impossible in case of a probabilistic class like $\mathcal{BPP}$; the problem being that the first formulation does not imply the second one. Indeed, consider a probabilistic polynomial-time machine M and sequence $a_1, a_2, ...$ such that $|a_n| = \log n$ and $\Pr[M(x, a_{|x|}) = \chi_L(x)] \geq 2/3$ for every x. Then, it is unclear which pair language (extending $L \in \mathcal{BPP}/\log$) is in $\mathcal{BPP}$ (i.e., $\{(x,a) : \Pr[M(x,a)=1] \geq 2/3$ is not necessarily a BPP-set). However, we can define an adequate promise problem that is in the (promise problem) class $\mathcal{BPP}$. Specifically, consider $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$ such that $\Pi'_{\text{YES}} = \{(x, a_{|x|}) : x \in L\}$ and $\Pi'_{\text{NO}} = \{(x, a_{|x|}) : x \notin L\}$. Similarly, for a promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \mathcal{BPP}/\log$ we consider $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$ such that $\Pi'_{\text{YES}} = \{(x, a_{|x|}) : x \in \Pi_{\text{YES}}\}$ and $\Pi'_{\text{NO}} = \{(x, a_{|x|}) : x \in \mathcal{P}_{\text{NO}}\}$. Thus, we may say that a promise problem $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ is in the (promise problem) class $\mathcal{BPP}/\log$ if there exists a promise problem $\Pi' = (\Pi'_{\rm YES}, \Pi'_{\rm NO})$ in $\mathcal{BPP}$ and a sequence $a_1, a_2, ...$ such that $|a_n| = \log n$ and $x \in \Pi_{\rm YES}$ implies $(x, a_{|x|}) \in \Pi'_{\rm YES}$ while $x \in \Pi_{\rm NO}$ implies $(x, a_{|x|}) \in \Pi'_{\rm NO}$.

Needless to say, the same approach can be applied to other probabilistic classes (e.g., $\mathcal{RP}$ and $\mathcal{AM}$) and to any bound on the advice length. We note that the need for promise problems arises here only in case of probabilistic classes (and not in case of deterministic or non-deterministic classes). The issue at hand is related to the difficulties regarding complete problems and hierarcy theorems (cf. Section 4.1); that is, not every advice (or machine) induces a *bounded-error* probabilistic computation, and focusing on those advices (or machines) that do induce such a computation is done by introducing a promise.

7.2 Infinitely often probabilistic classes

Similar issues arise in the case of "infinitely often" classes. The standard definition of io-$\mathcal{BPP}$ is that a problem is in this class if there exists a probabilistic polynomial-time algorithm that solves it correctly for infinitely many input lengths. The alternative formulation would say that a promise problem $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ is in the class io-$\mathcal{BPP}$ if there exists a promise problem $\Pi' = (\Pi'_{\rm YES}, \Pi'_{\rm NO})$ in $\mathcal{BPP}$ such that for infinitely many values of n it holds that $\Pi'_{\rm YES} \cap \{0,1\}^n = \Pi_{\rm YES} \cap \{0,1\}^n$ and $\Pi'_{\rm NO} \cap \{0,1\}^n = \Pi_{\rm NO} \cap \{0,1\}^n$.

8 Concluding Comments

We conclude with a couple of comments of "opposite direction" (i.e., restricting attention versus extending it). On one hand, we point out the relevance of promise problems to the study of "traditional" complexity classes that refer to language recognition problems. On the other hand, we mention that the concept of promise problems is applicable also to the study of search (rather than decision) problems.

8.1 Implications on the study of classes of language

We have argued that promise problems are at least as natural as traditional language recognition (decision) problems. Moreover, promise problems are a generalization of language recognition problems, and classes of promise problems refer to the computational powers of certain resources in the same way as classes of languages. In many cases, the generalization (i.e., promise problems) allows to represent natural concepts (e.g., special cases, unique solutions, and approximation) that cannot be represented in terms of language recognition problems. In other cases, the generalization allows to derive appealing results that are not known to hold for the corresponding language classes. The latter results refer to the same computational resources, and thus do not seem less fundamental than the analogous results unknown for language classes.

Still tradition and simplicity (offered by language classes) have their appeal. We thus mention that in some cases, *the study of promise problems yields results about language classes.* The best example is the study of Statistical Zero-Knowledge ($\mathcal{SZK}$), which is surveyed in Section 4: Using promise problems it is possible to present clear proofs of certain properties of the promise problem class $\mathcal{SZK}$ (e.g., that $\mathcal{SZK}$ is closed under complementation [49] and that any problem in $\mathcal{SZK}$ has a public-coin statistical zero-knowledge proof system [29]). But, then, it follows that the same properties hold for the class of languages having Statistical Zero-Knowledge proofs.

8.2 Applicability to search problems

As surveyed above, promise problems are a generalization of language recognition problems, and thus constitute a general form of decision problems. However, one may apply the concept of promise problems also in the context of search problems, and indeed such an application is at least as natural. Specifically, it is most natural to state search problems in terms of "promise problems" (rather than requiring their solver also to handle instances that have no solution (and hence also solve the corresponding decision problem)). That is, for a polynomially bounded relation R, the search (promise) problem is given x that has a solution (i.e., there exists y such that $(x,y) \in R$) to find such a solution (i.e., find a y such that $(x,y) \in R$). Hence, the promise is that x has a solution (i.e., y such that $(x,y) \in R$), and nothing is required in case x has no solution. (Note that the promise is important in case R is not an NP-relation.)

As in case of decision problems, search (promise) problems offer a formalism for the intuitive notion of special cases. In addition to the natural appeal of the promise problem formulation of search problems, such promise search problems offer a few fundamental advantages over traditional search problems. The best example is the connection between (bounded fan-in) circuit depth and communication complexity established by Karchmer and Wigderson [36]. Specifically, the (bounded fan-in) circuit depth of a function $f : \{0,1\}^n \to \{0,1\}$ is shown to equal the communication complexity of the search promise-problem in which one party is given $x = x_1 \cdots x_n \in f^{-1}(1)$, the other party is given $y = y_1 \cdots y_n \in f^{-1}(0)$, and the task is to determine an $i \in [n]$ such that $x_i \neq y_i$. Furthermore, the monotone depth of f equals the communication complexity of a search problem with the same promise, where the task is to find an $i \in [n]$ such that $x_i = 1$ and $y_i = 0$. We stress that removing the promise yields a trivial communication complexity lower bound of n (e.g., by reduction from the communication complexity of the `identity` function), which of course has no relevance to the circuit depth of the function.

Acknowledgments

I am grateful to Salil Vadhan for various insightful comments and suggestions. I also wish to thank Ran Raz and Avi Wigderson for answering my questions and making additional suggestions.

References

[1] L. Adleman. Two theorems on random polynomial-time. In *19th FOCS*, pages 75–83, 1978.

[2] D. Aharonov and O. Regev. Lattice problems in $\mathcal{NP} \cap \text{co}\mathcal{NP}$. In *45th FOCS*, pages 362–371, 2004.

[3] W. Aiello and J. Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *JCSS*, Vol. 42 (3), pages 327–345, 1991. Preliminary version in *28th FOCS*, 1987.

[4] N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, Vol. 7 (1), pages 1–22, 1987.

[5] N. Alon and A. Shapira. Extermal Graphs, Recursive Functions and a Separation Theorem in Property Testing. Unpublished manuscript, 2004.

[6] B. Barak. A Probabilistic-Time Hierarchy Theorem for "Slightly Non-uniform" Algorithms. In *Random'02*, LNCS 2483, pages 194–208, 2002.

[7] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. *SICOMP*, Vol. 27, No. 3, pages 804–915, 1998. Extended abstract in *36th FOCS*, pages 422–431, 1995.

[8] C.H. Bennett, G. Brassard and J.M. Robert. Privacy Amplification by Public Discussion. *SICOMP*, Vol. 17, pages 210–229, 1988. Preliminary version in *CRYPTO'85*, Springer-Verlag LNCS Vol. 218, pages 468–476 (titled "How to Reduce your Enemy's Information").

[9] E. Ben-Sasson, M. Sudan, S. Vadhan and A. Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *35th STOC*, pages 612–621, 2003.

[10] H. Buhrman and L. Fortnow. One-sided versus two-sided randomness. In Proceedings of the *16th Symposium on Theoretical Aspects of Computer Science*. LNCS, Springer, Berlin, 1999.

[11] J. Cai and A. Nerurkar. A note on the non-NP-hardness of approximate lattice problems under general Cook reductions. *IPL*, Vol. 76, pages 61-66, 2000. See comment in [19].

[12] S. Even, A.L. Selman, and Y. Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Inform. and Control*, Vol. 61, pages 159–173, 1984.

[13] S. Even and Y. Yacobi. Cryptography and NP-Completeness. In proceedings of *7th ICALP*, Springer-Verlag, LNCS Vol. 85, pages 195–207, 1980. See [12].

[14] L. Fortnow, The complexity of perfect zero-knowledge. In *Advances in Computing Research*, Vol. 5 (S. Micali, ed.), JAC Press Inc., pages 327–343, 1989. Preliminary version in *19th STOC*, 1987.

[15] L. Fortnow and R. Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th FOCS*, pages 316–324, 2004.

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

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

[18] O. Goldreich. Zero-Knowledge twenty years after its invention. To appear in *Quaderni di Matematica*, volume on complexity theory (2005). Available from `http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html`.

[19] O. Goldreich. Web-page `http://www.wisdom.weizmann.ac.il/~oded/p_lp.html`.

[20] O. Goldreich and S. Goldwasser, On the Limits of Non-Approximability of Lattice Problems. *JCSS*, Vol. 60, pages 540–563, 2000. Extended abstract in *30th STOC*, 1998.

[21] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.

[22] O. Goldreich and E. Kushilevitz. A Perfect Zero-Knowledge Proof for a Decision Problem Equivalent to Discrete Logarithm. *Jour. of Crypto.*, Vol. 6 (2), pages 97–116, 1993. Extended abstract in proceedings of *Crypto'88*.

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

[24] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002. Extended abstract in *29th STOC*, 1997.

[25] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999. Extended abstract in *30th STOC*, 1998.

[26] O. Goldreich, A. Sahai, and S. Vadhan. Honest-Verifier Statistical Zero-Knowledge equals general Statistical Zero-Knowledge. In *30th STOC*, pages 399–408, 1998.

[27] O. Goldreich, A. Sahai, and S. Vadhan. Can Statistical Zero-Knowledge be Made Non-Interactive? or On the Relationship of SZK and NISZK. In *Proceedings of Crypto99*, Springer LNCS Vol. 1666, pages 467–484.

[28] O. Goldreich and M. Sudan. Locally Testable Codes and PCPs of Almost-Linear Length. Preliminary version in *43rd FOCS*, 2002. ECCC Report TR02-050, 2002.

[29] O. Goldreich and S. Vadhan. Comparing Entropies in Statistical Zero-Knowledge with Applications to the Structure of SZK. In *14th IEEE Conference on Computational Complexity*, pages 54–73, 1999.

[30] O. Goldreich and D. Zuckerman. Another proof that BPP subseteq PH (and more). ECCC Report TR97-045, 1997.

[31] 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.

[32] J. Grollmann and A.L. Selman. Complexity Measures for Public-Key Cryptosystems. *SIAM Jour. on Comput.*, Vol. 17 (2), pages 309–335, 1988.

[33] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, Vol. 182, pages 105–142, 1999. Preliminary versions in *28th STOC* (1996) and *37th FOCS* (1996).

[34] J. Hastad. Getting optimal in-approximability results. In *29th STOC*, pages 1–10, 1997.

[35] R. Impagliazzo, L.A. Levin and M. Luby. Pseudorandom Generation from One-Way Functions. In *21st STOC*, pages 12–24, 1989.

[36] M. Karchmer and A. Wigderson. Monotone Circuits for Connectivity Require Super-logarithmic Depth. *SIAM J. Discrete Math.*, Vol. 3 (2), pages 255–265, 1990. Preliminary version in *20th STOC*, 1988.

[37] E. Knill. Quantum randomness and nondeterminism. `quant-ph/9610012`, 1996.

[38] A.Yu. Kitaev, A.H. Shen and M.N. Vyalyi. Classical and Quantum Computation. Vol. 47 of *Graduate Studies in Mathematics*, AMS, 2002.

[39] T. Kuhn. *The Structure of Scientific Revolution*. University of Chicago, 1962.

[40] C. Lautemann. BPP and the Polynomial Hierarchy. *IPL*, 17, pages 215–217, 1983.

[41] A. Lempel. Cryptography in Transition. *Computing Surveys*, Dec. 1979.

[42] N. Nisan and A. Wigderson. A note on Rank versus Communication Complexity. *Combinatorica*, Vol. 15 (4), pages 557–566, 1995.

[43] T. Okamoto. On relationships between statistical zero-knowledge proofs. *JCSS*, Vol. 60 (1), pages 47–108, 2000. Preliminary version in *28th STOC*, 1996.

[44] R. Raz. Exponential Separation of Quantum and Classical Communication Complexity. In *31st STOC*, pages 358–367, 1999.

[45] A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. In *Doklady Akademii Nauk SSSR*, Vol. 281, No. 4, 1985, pages 798-801. English translation in *Soviet Math. Doklady*, 31:354-357, 1985.

[46] A. Razborov. Lower bounds of monotone complexity of the logical permanent function. *Matematicheskie Zametki*, Vol. 37, No. 6, 1985, pages 887-900. English translation in *Mathematical Notes of the Academy of Sci. of the USSR*, 37:485-493, 1985.

[47] A. Razborov. On the Distributional Complexity of Disjointness. *Theoretical Computer Science*, Vol. 106, pages 385–390, 1992.

[48] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2), pages 252–271, 1996.

[49] A. Sahai and S. Vadhan. A complete problem for Statistical Zero-Knowledge. *Journal of the ACM*, 50(2):196–249, March 2003. Preliminary version in *38th FOCS*, 1997.

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

[51] L. Stockmeyer. On Approximation Algorithms for #P. *SIAM Journal on Computing*, Vol. 14 (4), pages 849–861, 1985. Preliminary version in *15th STOC*, pages 118–126, 1983.

[52] E. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, Vol. 8 (1), pages 141–142, 1988.

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

[54] S. Vadhan. An Unconditional Study of Computational Zero Knowledge. In *45th FOCS*, pages 176–185, 2004.

[55] S. Vadhan. Using promise problems to define modified complexity classes. Email to the author (Jan. 24, 2005).

[56] L.G. Valiant and V.V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theoretical Computer Science*, Vol. 47 (1), pages 85–93, 1986. Preliminary version in *17th STOC*, pages 458–463, 1985.

Appendix: On the derandomization of BPP versus MA

The following presentation is adapted from [30, Sec. 5.4]. We denote by $\mathcal{MA}$ the class of promise problems of the form $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$, where there exists a polynomial p and a polynomial-time (verifier) V such that

$$\begin{aligned} x \in \Pi_{\rm YES} \;\; &\Longrightarrow \;\; \exists w \in \{0,1\}^{p(|x|)} \;\; {\rm Pr}_{r \in \{0,1\}^{p(|x|)}}[V(x,w,r)=1] \;=\; 1 \\ x \in \Pi_{\rm NO} \;\; &\Longrightarrow \;\; \forall w \in \{0,1\}^{p(|x|)} \;\; {\rm Pr}_{r \in \{0,1\}^{p(|x|)}}[V(x,w,r)=1] \;\leq\; \frac{1}{2} \end{aligned}$$

(All other complexity classes used below also refer to promise problems.)

Proposition (folklore): *Suppose that $\mathcal{RP} \subseteq {\rm DTIME}(t)$, for some monotonically non-decreasing and time-constructible function $t : \mathsf{N} \to \mathsf{N}$. Then, $\mathcal{MA} \subseteq \cup_{i \in \mathsf{N}} {\rm NTIME}(t_i)$, where $t_i(n) = t(n^i)$.*

In particular, $\mathcal{RP} = \mathcal{P}$ implies $\mathcal{MA} = \mathcal{NP}$.

Proof: Each promise problem $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$ in $\mathcal{MA}$ gives rise to a promise problem $\Pi' = (\Pi'_{\rm YES}, \Pi'_{\rm NO})$, where

$$\begin{aligned} \Pi'_{\rm YES} \;&\stackrel{\rm def}{=}\; \{(x,w) : \forall r \in \{0,1\}^{p(|x|)} \;\; V(x,w,r)=1\} \\ \Pi'_{\rm NO} \;&\stackrel{\rm def}{=}\; \{(x,w) : x \in \Pi_{\rm NO}\}\,. \end{aligned}$$

where p and V are as above. Note that, for every $(x,w) \in \Pi'_{\rm YES}$ it holds that ${\rm Pr}_{r \in \{0,1\}^{p(|x|)}}[V(x,w,r)=1] = 1$, whereas for every $(x,w) \in \Pi_{\rm NO}$ it holds that ${\rm Pr}_{r \in \{0,1\}^{p(|x|)}}[V(x,w,r)=1] \leq 1/2$. Thus, $\Pi' \in {\rm co}\mathcal{RP}$ (i.e., $(\Pi'_{\rm NO}, \Pi'_{\rm YES}) \in \mathcal{RP}$). Using the hypothesis (and the closure of DTIME under complementation), we have $\Pi' \in {\rm DTIME}(t)$. On the other hand, note that for every $x \in \Pi_{\rm YES}$ there exists $w \in \{0,1\}^{p(|x|)}$ such that $(x,w) \in \Pi'_{\rm YES}$, whereas for every $x \in \Pi_{\rm NO}$ and every $w \in \{0,1\}^{p(|x|)}$ it holds that $(x,w) \in \Pi'_{\rm NO}$. Thus, Π is "non-deterministically reducible" to Π' (i.e., by a "reduction" that maps x to (x,w), where $w \in \{0,1\}^{p(|x|)}$, such that $x \in \Pi_{\rm YES}$ is mapped to $(x,w) \in \Pi'_{\rm YES}$), and $\Pi \in {\rm NTIME}(t')$ follows, where $t'(n) = t(n+p(n)) < t(n^i)$ for some $i \in \mathsf{N}$. The proposition follows. ■