On Doubly-Efficient Interactive Proof Systems

Oded Goldreich
Weizmann Institute of Science
oded.goldreich@weizmann.ac.il

March 29, 2018

Abstract

An interactive proof system is called doubly-efficient if the prescribed prover strategy can be implemented in polynomial-time and the verifier's strategy can be implemented in almost-linear-time. Such proof systems, introduced by Goldwasser, Kalai, and Rothblum (*JACM*, 2015), make the benefits of interactive proof system available to real-life agents who are restricted to polynomial-time computation.

We survey some of the known results regarding doubly-efficient interactive proof system. We start by presenting two simple constructions for t-no-`CLIQUE` (due to Goldreich and Rothblum (*ECCC*, TR17-018 and TR18-046)), where the first construction offers the benefit of being generalized to any "locally characterizable" set, and the second construction offers the benefit of preserving the combinatorial flavor of the problem. We then turn to two more general constructions of doubly-efficient interactive proof system: the proof system for sets having (uniform) bounded-depth circuits (due to Goldwasser, Kalai and Rothblum (*JACM*, 2015)), and the proof system for sets that are recognized in polynomial-time and small space (due to Reingold, Rothblum, and Rothblum (*STOC*, 2016)). Our presentation of the GKR construction is quite complete (and is somewhat different from the original presentation), but for the RRR construction we only provide an overview.

Contents

Chapter 1

Introduction

The notion of interactive proof systems, put forward by Goldwasser, Micali, and Rackoff [28], and the demonstration of their power by Lund, Fortnow, Karloff, and Nisan [35] and Shamir [44] are among the most celebrated achievements of complexity theory. Recall that an interactive proof system for a set S is associated with an interactive verification procedure, V, that can be made to accept any input in S but no input outside of S. That is, there exists an interactive strategy for the prover that makes V always accept any input in S, but no strategy can make V accept an input outside of S, except with negligible probability. (See Appendix A.1 for a formal definition of interactive proofs, and [19, Chap. 9] for a wider perspective.)

The original definition does not restrict the complexity of the strategy of the prescribed prover and the constructions of [35, 44] use prover strategies of high complexity. This fact limits the applicability of these proof systems in practice. (Nevertheless, such proof systems may be actually applied when the prover knows something that the verifier does not know, such as an NP-witness to an NP-claim; this is beneficial when the proof system offers an advantage (over NP-proof systems) such as being zero-knowledge [28, 22].)

1.1 The notion of doubly-efficient interactive proof systems

Seeking to make interactive proof systems available for a wider range of applications, Goldwasser, Kalai and Rothblum put forward a notion of *doubly-efficient* interactive proof systems (also called *interactive proofs for muggles* [27] and *interactive proofs for delegating computation* [42]). In these proof systems the prescribed prover strategy can be implemented in polynomial-time and the verifier's strategy can be implemented in almost-linear-time. That is, doubly-efficient interactive proof systems are restricted by two additional efficiency requirements:

> *Prover's efficiency requirement*: The prescribed prover strategy (referred to in the completeness condition) should be implemented in polynomial-time.
>
> *Verifier's efficiency requirement*: The verifier strategy should be implemented in almost-linear time.

(We stress that unlike in *argument systems*, the soundness condition holds for all possible cheating strategies (not only for feasible ones).)[1]

[1] See further discussion in Section 1.5.

Restricting the prescribed prover to run in polynomial-time implies that such systems may exist only for sets in $\mathcal{BPP}$, whereas a polynomial-time verifier can check membership in such sets by itself. However, restricting the verifier to run in almost-linear-time implies that something can be gained by interacting with a more powerful prover, even though the latter is restricted to polynomial-time.

The potential applicability of doubly-efficient interactive proof systems was demonstrated by Goldwasser, Kalai and Rothblum [27], who constructed such proof systems for any set that has log-space uniform circuits of small depth (e.g., log-space uniform $\mathcal{NC}$). A recent work of Reingold, Rothblum, and Rothblum [42] provided doubly-efficient (constant-round) proof systems for any set that can be decided in polynomial-time and small amount of space (e.g., for all sets in $\mathcal{SC}$). These two results are actually incomparable. The two constructions will be reviewed in Chapters 3 and 4, respectively, but before doing so we shall consider simpler constructions (see Section 1.2 and Chapter 2).

Terminology: We keep the term *almost linear* vague on purpose, but whenever appropriate we shall spell-out a specific interpretation of it. The most strict interpretation is that a function $f : \mathbb{N} \to \mathbb{N}$ is almost linear if $f(n) = \widetilde{O}(n) = \text{poly}(\log n) \cdot n$. This interpretation is suitable for Chapter 3 and most of Chapter 2 (i.e., for the results of [27] and [24, 25], resp). In contrast, Chapter 4 (following [42]) uses a much more liberal interpretation by which a sequence of functions of the form $f_\epsilon : \mathbb{N} \to \mathbb{N}$ (representing the complexities of a sequence of constructions) is almost linear if $f_\epsilon(n) = O(n^{1+\epsilon})$ for every $\epsilon > 0$. We mention that these interpretations have been used in the past also in other settings (see discussion in [20, Sec. 13.3.3]).

1.2 Doubly-efficient NP-proof systems

For starters, we mention that doubly-efficient interactive proof systems exist for some sets in $\mathcal{P}$ that are believed not to have almost-linear decision procedures (or at least are not known to have such procedures). Examples include `perfect matching` and t-`CLIQUE`, for any constant $t \geq 3$. In these cases, the proof systems are actually of the NP-type. The point is that in each of these cases, an easily verified NP-witness (i.e., one that can be verified in almost-linear time) can be found in polynomial-time.

The foregoing assertion is quite evident for `perfect matching`, by virtue of the known matching algorithms and the fact that checking whether a set of edges is a perfect matching in a given graph can be done in linear time. The same hold for the maximum flow problem, by virtue of the min-cut dual. More generally, morally, the same holds for linear programming (again by duality; i.e., by checking the feasibility of the dual program).[2] Turning to fixed-parameter problems, such as t-`CLIQUE`, let us now consider the (popular) sets

t-`CLIQUE`**:** The set of n-vertex graphs that contain a clique of size t.

t-`SUM`**:** The set of n-long sequences of integers, say, in $[-n^{t+3}, n^{t+3}]$, that contain t elements that sum-up to zero.

[2]Formally, issues may arise with respect to the length of the description of the solutions to the primal and dual programs, but these issues can be resolved by padding the input to that length (which seems quite natural in this context).

t-OV: The set of n-long sequences of vectors over $\mathrm{GF}(2)^{\log^2 n}$ that contain t vectors such that their coordinate-wise multiplication yields the all-zero vector.

In all cases, the NP-witness is a set of t elements, which can be found in time $\binom{n}{t}$ and verified in $\mathrm{poly}(t \cdot \log n)$-time. Recall that t-CLIQUE is conjectured to require time $n^{c' \cdot t}$, where c' is any constant smaller than one third of the Boolean matrix multiplication exponent (see, e.g., [1]), and that 3-SUM is conjectured to require almost quadratic time (see, e.g., [40], which promoted it), whereas t-OV generalizes the Orthogonal Vectors problem (see [6]).[3] Furthermore, t-CLIQUE is $\mathcal{W}[1]$-complete [15], solving it in time $n^{o(t)}$ refutes the ETH [13], and lower bounds for t-SUM are known to follow from lower bounds for t-CLIQUE (see [2]).

In general, the class of sets having doubly-efficient NP-proof systems is a subclass of $\mathcal{P} \cap \mathrm{Ntime}(\tilde{L})$, where $\tilde{L}$ denotes the set of almost-linear functions. (Indeed, the class of sets having doubly-efficient NP-proof systems consists of all sets $S \in \mathrm{Ntime}(\tilde{L})$ associated with a witness relation R such that given $x \in S$ we can find $y \in R(x) \stackrel{\text{def}}{=} \{w : (x,w) \in R\}$ in polynomial-time.)[4] As in the case of $\mathcal{IP}$-versus-$\mathcal{NP}$, the question is what can be gained by allowing the verifier to be interactive, toss coins, and rule by statistical evidence. That is, moving beyond doubly-efficient NP-proof systems, we focus on the power of doubly-efficient *interactive* proof systems.

1.3 The power of doubly-efficient interactive proof systems

The bulk of this survey is devoted to demonstrating the power of doubly-efficient interactive proof systems. We shall start by presenting two different (doubly-efficient interactive) proof systems for t-no-CLIQUE (i.e., the complement of t-CLIQUE). These proof systems (presented in Sections 2.1 and 2.3) are considerably simpler than the proof systems that can be derived from the general results captured by Theorems 1.1 and 1.2 (and presented in Chapters 3 and 4).[5]

Before turning to these more general results, we briefly discuss the two doubly-efficient interactive proof systems for t-no-CLIQUE . One of these systems (i.e., the one presented in Section 2.3) proceeds in t rounds such that, *in the i^{th} round, a claim regarding the number of $(t-i+1)$-cliques in a graph is reduced to a claim regarding the number of $(t-i)$-cliques in a related graph.* Hence, in each iteration, a natural computational problem is reduced to a closely related computational problem, while preserving the combinatorial flavor of the original problem. This proof system can also handle varying t, yielding an alternative interactive proof system for $\#\mathcal{P}$.

The idea that underlies the other proof system for t-no-CLIQUE (presented in Section 2.1) can be applied to a natural class of "locally characterizable" sets (defined in Section 2.2, following [24,

[3] Indeed, the Orthogonal Vectors problem corresponds to the case of $t = 2$. Our formulation of t-OV is different but equivalent to the one in [6], where the sequence of vectors is partitioned into t equal parts and a YES-instance has to take a single vector from each part.

[4] Formally, S has a doubly-efficient NP-proof system if and only if there exists a relation R such that

1. if $(x,y) \in R$, then $|y|$ is almost linear in $|x|$;
2. $S = \{x : \exists w \text{ s.t. } (x,w) \in R\}$;
3. there exists a polynomial-time algorithm that, on input $x \in S$, outputs an element of $\{w : (x,w) \in R\}$;
4. there exists an almost-linear time algorithm that, on input (x,y), decides whether or not $(x,y) \in R$.

[5] The two simple proof systems for t-no-CLIQUE are due to [24, Sec. 3] and [25, Sec. 2], resp., whereas Theorems 1.1 and 1.2 are due to [27] and [42], resp.

Sec. 5]). This class, which contains t-no-CLIQUE, is a subclass of $\mathcal{NC} \cap \mathcal{SC}$. This means that doubly-efficient interactive proof systems for locally characterizable sets can be obtained from either Theorem 1.1 or Theorem 1.2, but the point of presenting the direct proof systems for locally characterizable sets (in Section 2.2) is that they are considerably simpler than those obtained by either Theorem 1.1 or Theorem 1.2. Still, the latter theorems yield the most general results known regarding doubly-efficient interactive proof systems.

Theorem 1.1 (doubly-efficient interactive proof systems for log-space uniform $\mathcal{NC}$ [27]): *Every set that is decidable by a family of log-space uniform circuits of depth d such that $d(n) = \widetilde{O}(n)$, has a doubly-efficient interactive proof system. Furthermore, the proof system uses $O(\log n) \cdot d(n)$ rounds, and the verifier runs in $\widetilde{O}(n + d(n))$-time.*

Although circuit size and depth is related to time and space [10], this relation is not tight enough to relate $\mathcal{NC}$ and $\mathcal{SC}$.[6] Hence, Theorem 1.1 is incomparable to the following

Theorem 1.2 (doubly-efficient interactive proof systems for $\mathcal{SC}$ [42]): *Every set that is decidable by an algorithm that runs in polynomial time and has space complexity s such that $s(n) < \sqrt{n}$, has a doubly-efficient interactive proof system. Furthermore, for any constant $\delta > 0$, the proof system uses $\exp(\widetilde{O}(1/\delta))$ rounds, and verifier runs in $(\widetilde{O}(n) + s(n)^2 \cdot n^{\delta})$-time.*

As noted in [42], Theorem 1.2 can be extended to randomized algorithms by first reducing their randomness complexity to linear (using adequate pseudorandom generators), and then letting the verifier toss coins for the derived algorithm and send the outcomes to the prover (asking it to prove membership in the corresponding residual set).[7] A begging open problem is whether the upper bound of $s(n)^2$ can be replaced by $s(n)$; that is,

Problem 1.3 (a possible quantitative improvement of Theorem 1.2): *Does every set that is decidable by an algorithm that runs in polynomial time and linear space have a doubly-efficient interactive proof system?*

Another intriguing question is whether the round complexity in Theorem 1.2 can be reduced to $\mathrm{poly}(1/\delta)$.

We mention that all the aforementioned proof systems, which will be surveyed in the subsequent chapters, are of the public-coin type.[8] Note that the known transformation of general interactive proof systems to public-coin ones does not apply to doubly-efficient interactive proof systems, since the resulting prover strategy is not efficient and this seems inherent [48]. This begs the question of whether general systems (i.e., ones that are not of the public-coin type) can offer some advantages in the context of doubly-efficient interactive proof systems.

[6] The point is that the translations between depth and space do not preserve polynomial bounds on the size and time, respectively. Specifically, $\mathcal{SC}$ can be emulated by uniform circuits of polylogarithmic depth (and quasi-polynomial size), whereas log-space uniform $\mathcal{NC}$ can be emulated by algorithms of polylogarithmic space complexity (that run in quasi-polynomial time).

[7] When applied to a randomized algorithm with two-sided error this yields an interactive proof system with two-sided error (a.k.a imperfect completeness (see Appendix A.1)). Recall, however, that we our focus is on interactive proof systems with perfect completeness (as in Definition A.1); such proof systems are obtained here when applying the foregoing reduction to a randomized algorithm that always accepts YES-instances. A similar comment holds with respect to all subsequent statements about $\mathcal{BPP}$ (see, e.g., Theorem 1.4 and Section 1.5); that is, when considering proof systems of perfect completeness, one may replace $\mathcal{BPP}$ by co$\mathcal{RP}$.

[8] In the public coin model, at each round, the verifier tosses a predetermined number of coins and sends the outcome to the prover (see discussion at the end of Appendix A.1).

1.4 An upper bound on doubly-efficient interactive proof systems

As stated upfront, doubly-efficient interactive proof systems exists only for sets in $\mathcal{BPP}$. This is the case, since a decision procedure can just emulate the interaction between the prescribed polynomial-time prover (and the polynomial-time verifier). Using the hypothesis that the verifier runs in almost linear time, it follows that such sets are decidable in almost linear space. This is the case since the hypothesis implies that the communication and the verifier's randomness are (at most) almost-linear (in the length of the input), and the same holds for the space complexity of the verifier. Hence, a machine of almost linear space complexity can decide membership in the set by emulating all possible interactions. For future reference, let us state the conclusion of the foregoing discussion.

Theorem 1.4 (upper bound): *Every set that has a doubly-efficient interactive proof system can be decided in $\mathcal{BPP} \cap \mathrm{Dspace}(\tilde{\ell})$, where $\tilde{\ell}$ is an almost linear function.*

Note that even if Theorem 1.2 is improved as suggested in Problem 1.3, a gap will remain between it and Theorem 1.4, since $\mathrm{TiSp}(T,s)$ is not necessarily equal $\mathrm{Time}(T) \cap \mathrm{Space}(s)$. Hence, another begging open problem is whether the power of doubly-efficient interactive proof systems is captured by $\mathrm{TiSp}(\mathrm{poly},s)$ or by $\mathcal{BPP} \cap \mathrm{Space}(s)$ (or by neither).

Problem 1.5 (on the gap between Theorems 1.2 and 1.4): *Prove or provide evidence against at least one of the following conjectures:*

1. *For $s(n) = \widetilde{O}(n)$, every set in $\mathcal{P} \cap \mathrm{Dspace}(s)$ has a doubly-efficient interactive proof system. Even establishing the claim for some $s(n) = \omega(\log n)$ would be interesting.*[9]
2. *Every set that has a doubly-efficient interactive proof system can be decided by a probabilistic algorithm that runs in polynomial time and almost linear space.*

We mention that the second conjecture contradicts the conjecture that *log-space uniform circuits of linear depth and polynomial size cannot be emulated by polynomial-time algorithms of almost-linear space complexity*, since Theorem 1.1 provides doubly-efficient interactive proof systems for the former. Can stronger evidence be brought against the second conjecture? On the other hand, we propose a milder form of the second conjecture asserting that every set that has a *constant-round* doubly-efficient interactive proof system can be decided by a probabilistic algorithm that runs in polynomial time and almost linear space.

1.5 On doubly-efficient argument systems

Recall that argument systems are defined as interactive proof systems with the exception that the soundness condition is replaced by a computational soundness condition. That is, while the (standard) soundness condition requires that no cheating strategy can make the prover accept false assertions, except with negligible probability, the computational soundness condition only requires

[9] Note that, by Theorem 1.1, the claim holds for any $s(n) = O(\log n)$, since $\mathcal{L}$ is contained in log-space uniform $\mathcal{NC}$.

the infeasibility of cheating (i.e., that cheating strategies that can be implemented by polynomial size circuits may only fool the verifier with negligible probability).[10]

Doubly-efficient argument systems for any set in $\mathcal{BPP}$ are implicit in Kilian's argument system for sets in $\mathcal{NP}$, which relies on collision resistance hashing [32].[11] This system uses a constant number of rounds. Furthermore, assuming the existence of computational PIR schemes [33], every set in $\mathcal{BPP}$ has a two-message doubly-efficient argument system [30, 31, 11].[12] On the other hand, any set having a doubly-efficient argument system is in $\mathcal{BPP}$ (since we can decide membership in such a set by emulating the interaction between the prescribed prover and verifier strategies).[13]

The fact that argument systems are always asserted by relying on an intractability assumption is no coincidence, since these asserted systems do not satisfy the information theoretic soundness requirement. In fact, the existence of an argument system that is not an interactive proof system (i.e., does not satisfy standard soundness) implies a complexity separation (which is not known unconditionally). Specifically:

Theorem 1.6 (arguments that are not proofs imply separations): *Let V be a verifier strategy for an argument system for a set S, and suppose that V does not satisfy the* (information theoretic) *soundness requirement. Then, $\mathcal{PSPACE}$ is not contained in $\mathcal{P}/\mathrm{poly}$. In particular, $\mathcal{BPP} \neq \mathcal{PSPACE}$.*[14]

In other words, Theorem 1.6 asserts that a gap between information theoretic soundness and computational soundness means a gap between computationally unbounded (prover) strategies and computationally bounded (prover) strategies. Recalling that, in the current setting (of fooling a probabilistic polynomial-time verifier), the former can be implemented in $\mathcal{PSPACE}$, whereas the computational restriction refers to $\mathcal{P}/\mathrm{poly}$, the main claim follows (and $\mathcal{BPP} \neq \mathcal{PSPACE}$ follows, since $\mathcal{BPP} \subset \mathcal{P}/\mathrm{poly}$).

Proof: Assume, for simplicity and without loss of generality, that in the said argument system each message of the prover consists of a single bit, and let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ denote an optimal prover strategy with respect to this system (i.e., $f(x,\gamma)$ is the message sent by the prover on common input x after receiving the sequence of verifier messages described in the communication transcript γ). Then, $f \in \mathcal{PSPACE}$, since an optimal prover strategy (w.r.t a probabilistic polynomial-time verifier) can be implemented in polynomial space. On the other hand, $f \notin \mathcal{P}/\mathrm{poly}$, since otherwise a polynomial size circuit could implement the optimal strategy (for convincing V), and so there will be no gap between the information theoretic soundness and the computational soundness of this proof system. ■

[10] Polynomial size circuits are preferred over probabilistic polynomial-time algorithms in order to account for auxiliary information that may be available to the prover (esp., when used as a subroutine inside a higher-level application).

[11] The claim is evident for sets in $\mathcal{P}$, whereas proving membership in $\mathcal{BPP}$-sets can be reduced to proving membership in $\mathcal{P}$-set as follows. First note that the hypothesis (i.e., the existence of collision resistance hashing) implies the existence of one-way functions, and hence of pseudorandom generators [29]. Using such a generator, we can reduce the randomness complexity of the decision procedures for $\mathcal{BPP}$ to linear, and let the verifier send a random-tape for such a procedure (as part of its first message). Hence, it suffices to verify a claim that refers to the residual set, which is in $\mathcal{P}$.

[12] The result of [31], which builds on [30], uses a computational PIR of quasi-polynomial security. The assumption was weakened to standard (polynomial security) by [11]. The original results that are stated for $\mathcal{P}$ can be extended to $\mathcal{BPP}$ (cf. Footnote 11).

[13] The validity of this decision procedure refers only to the probability that the prescribed prover convinces the (prescribed) verifier.

[14] Note that $\mathcal{BPP} \neq \mathcal{PSPACE}$ implies $\mathcal{BPP} \subset \mathcal{PSPACE}$, since $\mathcal{BPP} \subseteq \mathcal{PSPACE}$.

1.6 Preliminaries: The Sum-Check protocol

The Sum-Check protocol, designed by Lund, Fortnow, Karloff, and Nisan [35], is a key ingredient in some of the constructions that we present. In particular, it will used in Sections 2.1 and 2.2 as well as in Chapter 3.

Fixing a finite field $\mathcal{F}$ and a set $H \subset \mathcal{F}$ (e.g., H may be a two-element set), we consider an m-variate polynomial $P : \mathcal{F}^m \to \mathcal{F}$ of individual degree d. Given a value v, the Sum-Check protocol is used to prove that

$$\sum_{\sigma_1,...,\sigma_m \in H} P(\sigma_1, ..., \sigma_m) = v, \tag{1.1}$$

assuming that the verifier can evaluate P by itself. The Sum-Check protocol proceeds in m iterations, such that in the i^{th} iteration the number of summations (over H) decreases from $m-i+1$ to $m-i$. Specifically, the i^{th} iteration starts with a claim of the form $\sum_{\sigma_i,...,\sigma_m \in H} P(r_1, ..., r_{i-1}, \sigma_i, ..., \sigma_m) = v_{i-1}$, where $r_1, ..., r_{i-1}$ and v_{i-1} are as determined in prior iterations (with $v_0 = v$), and ends with a claim of the form $\sum_{\sigma_{i+1},...,\sigma_m \in H} P(r_1, ..., r_i, \sigma_{i+1}, ..., \sigma_m) = v_i$, where r_i and v_i are determined in the i^{th} iteration. Initializing the process with $v_0 = v$, in the i^{th} iteration the parties act as follows.

Prover's move: The prover computes a univariate polynomial of degree d over $\mathcal{F}$

$$P_i(z) \stackrel{\text{def}}{=} \sum_{\sigma_{i+1},...,\sigma_m \in H} P(r_1, ..., r_{i-1}, z, \sigma_{i+1}, ..., \sigma_m), \tag{1.2}$$

where $r_1, ..., r_{i-1}$ are as determined in prior iterations, and sends P_i to the verifier (claiming that $\sum_{\sigma \in H} P_i(\sigma) = v_{i-1}$).

Verifier's move: Upon receiving a degree d polynomial, denoted $\widetilde{P}$, the verifier checks that $\sum_{\sigma \in H} \widetilde{P}(\sigma) = v_{i-1}$ and rejects if inequality holds. Otherwise, it selects r_i uniformly in $\mathcal{F}$, and sends it to the prover, while setting $v_i \leftarrow \widetilde{P}(r_i)$.

If all m iterations are completed successfully (i.e., without the verifier rejecting in any of them), the verifier conducts a final check. It computes the value of $P(r_1, ..., r_m)$ and accepts if and only if this value equals v_m.

Clearly, if Eq. (1.1) holds (and the prover acts according to the protocol), then the verifier accepts with probability 1. Otherwise, no matter what the prover does, the verifier accepts with probability at most $m \cdot d/|\mathcal{F}|$, because in each iteration if the prover provides the correct polynomial, then the verifier rejects (since $\sum_{\sigma \in H} P_i(\sigma) = P_{i-1}(r_{i-1}) \neq v_{i-1}$), and otherwise the (degree d) polynomial sent agrees with P_i on at most d points.[15]

The complexity of verification is dominated by the complexity of evaluating P (on a single point). As for the prescribed prover, it may compute the relevant P_i's by interpolation, which is based on computing the value of P at $(d+1) \cdot |H|^{m-i}$ points, for each $i \in [m]$. (That is, the polynomial P_i is computed by obtaining its values at $d+1$ points, where the value of P_i at each point is obtained by summing the values of P at $|H|^{m-i}$ points.)[16]

[15] If P_i does not satisfy the current claim (i.e., $\sum_{\sigma \in H} P_i(\sigma) \neq v_{i-1}$), then the prover can avoid upfront rejection only if it sends $\widetilde{P} \neq P_i$. But in such a case, $\widetilde{P}$ and P_i (both being degree d polynomials) may agree on at most d points. Hence, if the chosen $r_i \in \mathcal{F}$ is not one of these points, it holds that $v_i = \widetilde{P}(r_i) \neq P_i(r_i)$, which means that the next iteration will also start with a false claim. Hence, starting with a false claim (i.e., $\sum_{\sigma \in H} P_1(\sigma) \neq v_0$ since Eq. (1.1) does not hold), with probability at least $1 - m \cdot d/|\mathcal{F}|$, after m iterations we reach a false claim regarding the value of P at a single point.

[16] Specifically, the value of P_i at p is obtained from the values of P at the points $(r_1, ..., r_{i-1}, p, \sigma)$, where $\sigma \in H^{m-i}$.

1.7 Organization

In Chapter 2 we present simple constructions of doubly-efficient interactive proof systems for t-no-CLIQUE, as well as for a natural class that contains it and is contained in uniform $\mathcal{NC}$ (and also in $\mathcal{SC}$). These proof systems are due to Goldreich and Rothblum [24, 25]. In Chapter 3 we present the proof systems of Goldwasser, Kalai and Rothblum [27], which are applicable to sets that are recognized by small depth circuits (e.g., uniform $\mathcal{NC}$). In Chapter 4 we provide an outline of the proof systems of Reingold, Rothblum, and Rothblum [42], which are applicable to sets recognized in polynomial-time and small space (e.g., sets in $\mathcal{SC}$).

As hinted in the foregoing paragraph, Chapter 2 is much easier to read than Chapters 3 and 4. Furthermore, while Chapters 2 and 3 provide full expositions of the claimed proof systems, Chapter 4 provides only an overview of the claimed proof system (while referring the interested reader to the 70-page description in the original work [42]).

Each of the following three chapters starts with several overview paragraphs that outline the contents of the chapter. Furthermore, Chapters 3 and 4 proceed with overview sections (see Sections 3.1 and 4.1, respectively).

We conclude (in Chapter 5) with an attempt to provide a high-level digest of the four fundamentally different proof systems reviewed above and with some speculations regarding the study of interactive proof systems at large.

Conventions: We assume that the verifier (resp., prover) has *direct access* to the common input; that is, each bit in the input can be read in unit cost. Unless explicitly stated differently, all logarithms are to base 2.

Chapter 2

Simple doubly-efficient interactive proof systems

In this chapter, we present simple constructions of doubly-efficient interactive proof systems for sets in $\mathcal{P}$ that are believed to have relatively high complexity. Specifically, two such constructions are presented for t-no-CLIQUE (i.e., the set of n-vertex graphs that do not contain a clique of size t). Generalizing one of these two constructions, we also present a generic construction of doubly-efficient interactive proof systems for sets in a natural class, which on the one hand contains t-no-CLIQUE and on the other hand is contained in $\mathcal{NC} \cap \mathcal{SC}$. Hence, doubly-efficient interactive proof systems for sets in the former class can be obtained from either Theorem 1.1 or Theorem 1.2, but the constructions presented in the current chapter are considerably simpler than the constructions reviewed in Chapters 3 and 4 (towards proving Theorems 1.1 and 1.2, resp.).

As stated above, we shall present two different doubly-efficient interactive proof systems for t-no-CLIQUE. The first proof system, which is due to Goldreich and Rothblum [24, Sec. 3] and is presented in Section 2.1, uses the Sum-Check protocol (reviewed in Section 1.6). This proof system is based on the observation that membership of an n-vertex graph in t-no-CLIQUE can be captured by the sum of the evaluations of an explicit low degree polynomial at n^t points, and that this polynomial can be evaluated in almost-linear time (in the size of the graph). Hence, applying the Sum-Check protocol yields the desired proof system. Following [24, Sec. 5], the same strategy can be applied to a natural class of "locally characterizable" sets, defined in Section 2.2, yielding a generic proof system for any locally-characterizable set.

The second construction of doubly-efficient interactive proof systems for t-no-CLIQUE, which originates in [25, Sec. 2] and is presented in Section 2.3, does not use the Sum-Check protocol. The interactive proof proceeds in t iterations such that in the i^{th} iteration verifying the number of $(t-i+1)$-cliques in a graph is reduced to verifying the number of $(t-i)$-cliques in a related graph. Hence, in each iteration, a natural computational problem is reduced to a closely related computational problem, while preserving the combinatorial flavor of the original problem. The resulting t-round interactive proof systems extends also to varying $t = t(n)$, yielding alternative interactive proof systems for sets in $\#\mathcal{P}$. Furthermore, on input an n-vertex graph, the prescribed prover strategy can be implemented in $\widetilde{O}(n^2)$-time when given oracle access to counting $(t(n)-1)$-cliques in $\text{poly}(t(n)) \cdot \widetilde{O}(n)$-vertex graphs.

Organization. Section 2.1 presents the first proof system for t-no-CLIQUE. The generic construction for any *locally-characterizable set* is presented in Section 2.2: The corresponding class is defined in Section 2.2.1, the basic construction appear in Section 2.2.2, and ramifications appear in Sections 2.2.3 and 2.2.4.[1] Section 2.3 presents the second proof system for t-no-CLIQUE.

2.1 The first construction for t-no-CLIQUE

For a parameter $t \in \mathbb{N}$, given a graph $G = ([n], E)$, the problem we consider is determining whether there exist t vertices such that the subgraph induced by them is a clique; that is, does there exist distinct $v_1, ..., v_t \in [n]$ such that $\bigwedge_{j,k \in [t]: j<k} \chi_{v_j,v_k}$, where $\chi_{u,v} = 1$ if and only if $\{u, v\} \in E$. (Our focus is on simple graphs, and so we assume that $\chi_{v,v} = 0$ for every $v \in [n]$.)

We actually consider the set of YES-instances (i.e., graphs having a t-clique), denoted t-CLIQUE, and the set of NO-instances, denoted t-no-CLIQUE. As outlined in Section 1.2, the set t-CLIQUE has an NP-proof system that uses proofs of length $t \log n$. We shall present a doubly-efficient interactive proof for the set t-no-CLIQUE.

Fixing a generic graph $G = ([n], E)$ (and recalling that $\chi_{u,v} = 1$ if and only if $\{u, v\} \in E$), we consider the arithmetic expression

$$\sum_{v^{(1)},...,v^{(t)} \in [n]} \prod_{j,k \in [t]: j<k} \chi_{v^{(j)},v^{(k)}}. \tag{2.1}$$

Observe that Eq. (2.1) is zero if and only if G is in t-no-CLIQUE. Hence, proving that G is in t-no-CLIQUE reduces to proving that the sum of n^t terms evaluates to zero, and the Sum-Check protocol seems adequate for this task, except that the expression is over the integers (rather than over a finite field) and a generic term (in Eq. (2.1)) is not expressed as a polynomial in $v^{(1)}, ..., v^{(t)}$.

The first problem is easy to handle: we can just use a finite field $\mathcal{F}$ of prime size greater than n^t (and consider Eq. (2.1) as an expression in $\mathcal{F}$). The second problem is more acute: we wish to represent a choice from the sequence of values $(\chi_{u,w})_{u,w \in [n]}$, which may be viewed as a function $\chi : [n] \times [n] \to \{0, 1\}$, as a (low degree) polynomial (in the description of this choice). The way to do it is to identify $[n]$ with $\{0, 1\}^\ell$, where $\ell = \log_2 n$, and to consider a low-degree extension of the function $\chi : \{0, 1\}^\ell \times \{0, 1\}^\ell \to \{0, 1\}$. Specifically, we consider a multi-linear polynomial $\widehat{\chi} : \mathcal{F}^\ell \times \mathcal{F}^\ell \to \mathcal{F}$ such that $\widehat{\chi}(y, z) = \chi(y, z)$ for every $y, z \in \{0, 1\}^\ell$. The latter polynomial can be written as the sum, over all $\alpha, \beta \in \{0, 1\}^\ell$, of $\mathtt{EQ}(\alpha\beta, yz) \cdot \chi(y, z)$, where $\mathtt{EQ} : \mathcal{F}^{2\ell} \times \mathcal{F}^{2\ell} \to \mathcal{F}$ is a multi-linear polynomial that satisfies $\mathtt{EQ}(\alpha\beta, yz) = 1$ if $yz = \alpha\beta$ and $\mathtt{EQ}(\alpha\beta, yz) = 0$ if $yz \in \{0, 1\}^{2\ell} \setminus \{\alpha\beta\}$ (for every $\alpha, \beta \in \{0, 1\}^\ell$).[2] Indeed, $\mathtt{EQ}(\gamma, x) = \prod_i (\gamma_i x_i + (1 - \gamma_i)(1 - x_i))$, which means that the polynomial $\widehat{\chi} : \mathcal{F}^\ell \times \mathcal{F}^\ell \to \mathcal{F}$ can be written as

$$\widehat{\chi}(y, z) = \sum_{\alpha,\beta \in \{0,1\}^\ell} \prod_{i \in [\ell]} (y_i\alpha_i + (1 - y_i)(1 - \alpha_i)) \cdot \prod_{i \in [\ell]} (z_i\beta_i + (1 - z_i)(1 - \beta_i)) \cdot \chi_{\alpha,\beta}.$$

[1] Although the generic construction for the class of locally-characterizable sets, which is presented in Section 2.2.2, almost meets the parameters of construction tailored for t-no-CLIQUE, we present the latter construction first (in Section 2.1). This is done since the construction tailored for t-no-CLIQUE is simpler (and actually more efficient).

[2] Hence, the value of the sum $\sum_{\alpha,\beta \in \{0,1\}^\ell} \mathtt{EQ}(\alpha\beta, yz) \cdot \chi(y, z)$ at $yz \in \{0, 1\}^{2\ell}$ is determined by the only term associated with the pair (α, β) that satisfies $\mathtt{EQ}(\alpha\beta, yz) = 1$ (i.e., $\alpha\beta = yz$), whereas all other terms vanish. Note that $\mathtt{EQ}(\gamma, x) = \prod_i (\gamma_i x_i + (1 - \gamma_i)(1 - x_i))$ equals 1 if $\gamma = x \in \{0, 1\}^{2\ell}$, since in this case each factor evaluates to 1, and equals 0 if $\gamma \neq x$ are both in $\{0, 1\}^{2\ell}$, since in this case at least one of the factors evaluates to 0.

(There is some abuse of notation here: In the first two occurrences, α and β are viewed as elements of $\mathcal{F}^\ell$, which happen to reside in $\{0,1\}^\ell \subset \mathcal{F}^\ell$, whereas in the last occurrence α and β are viewed as elements of $[n] \equiv \{0,1\}^\ell$.)

Hence, fixing a generic graph $G = ([n], E)$ and letting $\ell = \log n$, we consider a finite field $\mathcal{F}$ of prime size greater than n^t, and identify $\{0,1\}$ with the set H containing the zero and one elements of $\mathcal{F}$. Using this identification (and recalling that $\chi_{u,v} = 1$ if and only if $\{u,v\} \in E$), we define a polynomial $P : (\mathcal{F}^\ell)^t \to \mathcal{F}$ such that

$$P(z^{(1)}, ..., z^{(t)}) = \prod_{j,k \in [t]: j<k} \sum_{\alpha,\beta \in H^\ell} \mathtt{EQ}(\alpha\beta, z^{(j)} z^{(k)}) \cdot \chi_{\alpha,\beta} \tag{2.2}$$

$$\text{where } \mathtt{EQ}(\overline{\gamma}, \overline{z}) = \textstyle\prod_{i \in [2\ell]} (z_i \gamma_i + (1 - z_i)(1 - \gamma_i)). \tag{2.3}$$

Note that for each $\overline{z} \in (H^\ell)^t$ it holds that $P(\overline{z}) \in \{0,1\}$, and that $P(\overline{z}) = 1$ if and only if $\overline{z}$ describes a sequence of t distinct vertices that induces a clique in $G = ([n], E)$. (This holds since, for distinct $v^{(1)}, ..., v^{(t)} \in H^\ell$, it holds that $P(v^{(1)}, ..., v^{(t)}) = 1$ if and only if $\{v^{(j)}, v^{(k)}\} \in E$ for every $j < k$.)[3]

Note that P has individual degree $O(t)$, and that it is straightforward to evaluate P in time $O(t^2 \cdot 2^{2\ell} \cdot \ell) = \widetilde{O}(t^2 \cdot n^2)$. Also, for $\overline{\gamma}, \overline{z} \in H^{2\ell}$, it holds that $\mathtt{EQ}(\overline{\gamma}, \overline{z}) = 1$ if $\overline{\gamma} = \overline{z}$ and $\mathtt{EQ}(\overline{\gamma}, \overline{z}) = 0$ otherwise. Hence, for $\overline{z} \in H^{t\ell}$, it holds that $P(\overline{z}) = \prod_{j,k \in [t]: j<k} \chi_{z^{(j)}, z^{(k)}}$.

The key observation is that the graph G is a NO-instance if and only if for all $\overline{z} \in (H^\ell)^t$ it holds that $P(\overline{z}) = 0$. Hence, *we obtain an interactive proof system* (for the set of NO-instances) *by applying the Sum-Check protocol to the claim* $\sum_{\overline{z} \in (H^\ell)^t} P(\overline{z}) = 0$.

The complexity of the verifier's strategy is dominated by the evaluation of P (as defined in Eq. (2.2)), which reduces to $\binom{t}{2}$ computations of sums having the form

$$\sum_{\alpha,\beta \in H^\ell} \mathtt{EQ}(\alpha\beta, r^{(j)} r^{(k)}) \cdot \chi_{\alpha,\beta} \tag{2.4}$$

where $j < k \in [t]$ and $r^{(1)} \cdots r^{(t)} \in (\mathcal{F}^\ell)^t$ are determined in the execution of the Sum-Check protocol. Writing Eq. (2.4) as $\sum_{(\alpha,\beta) \in H^{2\ell} \cap E} \mathtt{EQ}(\alpha\beta, r^{(j)} r^{(k)})$, we can evaluate this sum in time $O(|E| \cdot \ell) = \widetilde{O}(|E| + n)$.

Turning our attention to the complexity of proving, recall that the prover strategy in the Sum-Check protocol is dominated by the complexity of evaluating the polynomial P at $O(t^2 \cdot \ell) \cdot |H|^{t\ell} = \widetilde{O}(n^t)$ points, where the first factor represents the degree of P (as defined in Eq. (2.2)) and the second factor represents the number of terms in the sum $\sum_{\overline{z} \in (H^\ell)^t} P(\overline{z})$. Hence, the prover's complexity is definitely bounded by $\widetilde{O}(n^t \cdot |E|)$. A closer inspection reveals that we can do better (cf. [24, Sec. 3]); that is, the prover strategy can be implemented in $\widetilde{O}(n^t)$-time. Hence, we get

Proposition 2.1 (doubly-efficient interactive proof systems for t-no-CLIQUE): *For every constant t, the set t-no-CLIQUE has a doubly-efficient interactive proof system. Specifically, on input $G = ([n], E)$, the verifier runs in $\widetilde{O}(|G|)$-time and the prescribed prover runs in $\widetilde{O}(n^t)$-time. Furthermore, the proof system uses logarithmically many rounds.*

Indeed, the point of Proposition 2.1 is that is the simplicity of its proof; that is, the fact that it is proved by merely applying the Sum-Check protocol to the claim $\sum_{\overline{z} \in (H^\ell)^t} P(\overline{z}) = 0$, where P is defined in Eq. (2.2).

[3] Also, if $v^{(j)} = v^{(k)}$ for some $j < k$, then $P(v^{(1)}, ..., v^{(t)}) = 0$, since $\chi_{v^{(j)}, v^{(k)}} = 0$.

Related works. As stated upfront, the foregoing construction is reproduced from the work of Goldreich and Rothblum [24]. This proof system is comparable to system presented by Thaler [46]. In addition, several recent works [12, 51, 9] constructed *non-interactive* proof systems for t-no-CLIQUE and related problems. For a detailed discussion, the interested reader is referred to [24, Sec. 1.1]. We also mention that, as discussed in [24, Sec. 1.1], the foregoing proof system is not only simpler than those of [27, 42] (which will be reviewed in Chapters 3 and 4, respectively), but also yields better round complexity.

2.2 A generic construction for locally-characterizable sets

In this section we present simple constructions of proof systems for a natural sub-class of $\mathcal{NC} \cap \mathcal{SC}$, which is believed not to be in Dtime(p) for any fixed polynomial p. Loosely speaking, this class consists of all sets that can be locally-characterized by the conjunction of polynomially many local conditions that can be expressed by Boolean formulae of polylogarithmic size (see definition in Section 2.2.1). The following result will become precise once the definition of locally-characterizable sets (i.e., Definition 2.3) is in place.

Theorem 2.2 (doubly-efficient interactive proof systems for locally-characterizable sets): *Every locally-characterizable set has a simple doubly-efficient interactive proof system. Specifically, on input of length n, the verifier runs in $\widetilde{O}(n)$-time and the strategy of the prescribed prover can be implemented in $\widetilde{O}(n^{c+1})$-time, where n^c denotes the number of local conditions in the characterization. Furthermore, the interactive proof system uses public coins, has logarithmic round-complexity, and uses polylogarithmic communication.*

We mention that that using different settings of parameters yields a general trade-off between the computational complexity and the number of communication rounds (see Theorem 2.6). The class of "locally-characterizable" sets contains two t-parametrized problems that are widely believed not to be in, say, Dtime($n^{t/2}$):

t-no-CLIQUE**:** The set of n-vertex graphs that do not contain a clique of size t.

t-no-SUM**:** The set of n-long sequences of integers that contain no t elements that sum-up to zero.[4]

(Indeed, the aforementioned sets are the complements of the sets called t-CLIQUE and t-SUM, which are popular candidates for "hardness in $\mathcal{P}$" [50].) Recall that a direct construction of a proof system for t-no-CLIQUE was presented in Section 2.1, with a corresponding prover strategy that can be implemented in time $\widetilde{O}(n^t)$. An analogous (but more cumbersome) construction for t-no-SUM appears in [24, Sec. 4].

2.2.1 A natural class: locally-characterizable sets

Loosely speaking, a set is said to be locally-characterizable if membership in it can be expressed by the conjunction of polynomially many local conditions such that the conditions as well as the relevant locations can be computed by Boolean formulae of polylogarithmic size (i.e., the functions ϕ_n and $\pi_{n,j}$'s in Definition 2.3).

[4] Alternatively, we may consider sequences of positive integers and ask if they contain a t-subset that sums-up to target integer, which is also given as part of the input.

This definition is related but different from the definition of "local characterization" that is often used in the property testing literature (see, Sudan's survey [45]). Most importantly, on the one hand, the latter definition does not specify the complexity of the "local characterization" (i.e., the functions ϕ_n and $\pi_{n,j}$'s in Definition 2.3).[5] On the other hand, typically, in the context of property testing, the number of locations in each "local condition" is postulated to be a constant, whereas in Definition 2.3 this number is polylogarithmic (see the polynomial p).

Definition 2.3 (locally-characterizable sets): *A set S is* locally-characterizable *if there exist a constant c, a polynomial p and a polynomial-time algorithm that on input n outputs* $\mathrm{poly}(\log n)$*-sized formulae $\phi_n : [n]^{p(\log n)} \times \{0,1\}^{p(\log n)} \to \{0,1\}$ and $\pi_{n,1}, ..., \pi_{n,p(\log n)} : \{0,1\}^{c \log n} \to [n]$ such that, for every $x \in \{0,1\}^n$, it holds that $x \in S$ if and only if for all $w \in \{0,1\}^{c \log n}$*

$$\Phi_x(w) \stackrel{\text{def}}{=} \phi_n(\pi_{n,1}(w), ..., \pi_{n,p(\log n)}(w), x_{\pi_{n,1}(w)}, ..., x_{\pi_{n,p(\log n)}(w)}) \tag{2.5}$$

equals 0.

(To simplify the rest of our exposition, we require that, in case of inputs in S the predicate ϕ_n evaluates to 0 (rather than to 1). On the other hand, for sake of intuition, we preferred to feed the predicate with the locations $\pi_{n,1}(w), ..., \pi_{n,p(\log n)}(w)$ rather that with the index of the condition (denoted w).)[6]

Looking at Definition 2.3, note that each value of $w \in \{0,1\}^{c \log n}$ yields a local condition that refers to polylogarithmically many locations in the input (i.e., the locations $\pi_{n,1}(w), ..., \pi_{n,p(\log n)}(w) \in [n]$). This local condition is captured by ϕ_n, and in its general form it depends both on the selected locations and on the value of the input bits at these locations. A simplified form, which suffices in many cases, uses a local condition that only depends on the values of the input bits in these locations (i.e., $\phi_n : [n]^{p(\log n)} \times \{0,1\}^{p(\log n)} \to \{0,1\}$ only depends on the $p(\log n)$-bit long suffix).

The simplified form (in which $\phi_n : \{0,1\}^{p(\log n)} \to \{0,1\}$) suffices for capturing the specific problems mentioned above. Specifically, for fixed $t \in \mathbb{N}$, when representing n-vertex graphs by their adjacency matrix, denoted $x = (x_{r,c})_{r,c \in [n]}$, the t-`CLIQUE` problem is captured by $\Phi_x(i_1, ..., i_t) = \wedge_{j<k} x_{i_j,i_k}$; that is, we use $\phi_{n^2} : \{0,1\}^{t^2} \to \{0,1\}$ and $\pi_{n^2,(j,k)} : \{0,1\}^{t \log n} \to [n] \times [n]$ (for $j,k \in [t]$) such that $\phi_{n^2}(\sigma_{1,1}, ..., \sigma_{t,t}) = \wedge_{j<k} \sigma_{j,k}$ and $\pi_{n^2,(j,k)}(i_1, ..., i_t) = (i_j, i_k)$. Likewise, considering n-long sequences over $[-m, m]$ (where $m = \mathrm{poly}(n)$), with some abuse of notation, the t-`SUM` problem is captured by $\Phi_x(i_1, ..., i_t) = 1$ if and only if $\sum_{j \in [t]} x_{i_j} = 0$; that is, we use $\phi_n : [m]^t \to \{0,1\}$ such that $\phi_n(z_1, ..., z_t) = \texttt{TruthValue}(\sum_{j \in [t]} z_j = 0)$ and $\pi_{n,j} : \{0,1\}^{t \log n} \to [n]$ such that $\pi_{n,j}(i_1, ..., i_t) = i_j$ (for each $j \in [t]$).

Note that the complement of every locally-characterizable set has a doubly-efficient interactive proof system. In this proof system, on input $x \in \{0,1\}^n$, letting $\ell' = c\ell = c \log n$, the prover finds an adequate $w \in \{0,1\}^{\ell'}$, sends it to the verifier, who retrieves the bits $x_{\pi_{n,1}(w)}, ..., x_{\pi_{n,p(\ell)}(w)}$ and evaluates ϕ_n on them. Indeed, in this NP-proof system, the prover runs in time $2^{\ell'} \cdot \widetilde{O}(|x|) = \mathrm{poly}(|x|)$, whereas the verifier runs in $\mathrm{poly}(\log |x|)$-time (given direct access to the input). On the other hand, the set itself (i.e., every locally-characterizable set) has a doubly-efficient interactive

[5] In addition, the notion used in property testing does not restrict the number of local conditions (i.e., the domain of Φ_x), although such an upper bound can be assumed without loss of generality (cf. [20, Exer. 1.21]).

[6] Note that w can be extracted from the sequence of locations if we extend the latter with c dummy elements.

proof systems (since computing the function $\sum_{w\in\{0,1\}^{\ell'}}\Phi_x(w)$, where Φ_x is as in Eq. (2.5), is in $\mathcal{SC}$, and so the proof system of [42] can be used).[7] Here we present a much simpler construction.

2.2.2 Proof of Theorem 2.2

Let S be a locally characterizable set, and c,p as well as $\phi_n,\pi_{n,j,k},\Phi_x$ be as in Definition 2.3. Letting $\ell=\log n$, we associate $[n]$ with $\{0,1\}^\ell$, and derive from each $\pi_{n,j}:\{0,1\}^{c\ell}\to[n]$ Boolean formulae $\pi_{n,j,1},...,\pi_{n,j,\ell}:\{0,1\}^{c\ell}\to\{0,1\}$ such that $\pi_{n,j,k}(w)$ is the k^{th} bit of $\pi_{n,j}(w)$. We may assume, without loss of generality, that the depth of each of the formulae (i.e., ϕ_n and the $\pi_{n,j,k}$'s) is logarithmic in their size (which is $\mathrm{poly}(\ell)$).[8]

Next, for a finite field $\mathcal{F}$ of size $\mathrm{poly}(n)>n^c$ and $H=\{0,1\}\subset\mathcal{F}$, we construct arithmetic formulae $\widehat{\phi}_n:\mathcal{F}^{(\ell+1)\cdot p(\ell)}\to\mathcal{F}$ and $\widehat{\pi}_{n,j,k}:\mathcal{F}^{c\ell}\to\mathcal{F}$ such that $\widehat{\phi}_n$ (resp., $\widehat{\pi}_{n,j,k}$) agrees with ϕ_n on $H^{\ell\cdot p(\ell)+p(\ell)}$ (resp., with $\pi_{n,j,k}$ on $H^{c\ell}$). These arithmetic formulae are constructed by replacing each Boolean gate by the corresponding arithmetic gate, while preserving the topology of the formulae. The crucial point is that these arithmetic formulae preserve the depth of the Boolean counterparts, which equals $O(\log\mathrm{poly}(\ell))$ (where $\ell=\log n$), and so the degrees of the functions that they compute is upper-bounded by $D=\mathrm{poly}(\log n)\ll|\mathcal{F}|$. (Note that $\mathcal{F}$ is chosen so that the latter inequality holds.) Now, letting $\widehat{\pi}_{n,j}:\mathcal{F}^{c\ell}\to\mathcal{F}^\ell$ such that $\widehat{\pi}_{n,j}(\overline{z})=(\widehat{\pi}_{n,j,1}(\overline{z}),...,\widehat{\pi}_{n,j,\ell}(\overline{z}))$, we consider the function $\widehat{\Phi}_x:\mathcal{F}^{c\ell}\to\mathcal{F}$ (i.e., an extension of Φ_x) such that

$$\widehat{\Phi}_x(\overline{z}) \stackrel{\text{def}}{=} \widehat{\phi}_n(\widehat{\pi}_{n,1}(\overline{z}),...,\widehat{\pi}_{n,p(\ell)}(\overline{z}),X_1,...,X_{p(\ell)}) \tag{2.6}$$

$$\text{where } X_j=\textstyle\sum_{\alpha\in\{0,1\}^\ell}\mathtt{EQ}(\widehat{\pi}_{n,j}(\overline{z}),\alpha)\cdot x_\alpha \tag{2.7}$$

and, as in Eq. (2.3), $\mathtt{EQ}(\overline{y},\alpha)=\prod_{i\in[\ell]}(y_i\alpha_i+(1-y_i)(1-\alpha_i))$. That is, the value of $\widehat{\Phi}_x(\overline{z})$ is obtained by feeding to $\widehat{\phi}_n:\mathcal{F}^{(\ell+1)\cdot p(\ell)}\to\mathcal{F}$ the $(p(\ell)\cdot\ell+p(\ell))$-long sequence consisting of the $p(\ell)\cdot\ell$-long sequence $(\widehat{\pi}_{n,1}(\overline{z}),...,\widehat{\pi}_{n,p(\log n)}(\overline{z}))\in(\mathcal{F}^\ell)^{p(\ell)}$ and the $p(\ell)$-long sequence whose j^{th} location contains the field element $\sum_{\alpha\in\{0,1\}^\ell}\mathtt{EQ}(\widehat{\pi}_{n,j}(\overline{z}),\alpha)\cdot x_\alpha$.

Observe that $x\in S$ if and only if the sum $\sum_{w\in\{0,1\}^{c\ell}}\widehat{\Phi}_x(w)$ equals 0, where we rely on $\mathcal{F}$ being larger than $2^{c\ell}$. Hence, verifying the claim $x\in S$ reduces to verifying $\sum_{w\in\{0,1\}^{c\ell}}\widehat{\Phi}_x(w)=0$, which can be performed by invoking the Sum-Check protocol.

Note that, in this case, the Sum-Check protocol performs $c\ell$ iterations (i.e., it is applied only to the outer sum $\sum_{w\in\{0,1\}^{c\ell}}\widehat{\Phi}_x(w)$), and the verifier evaluates the residual expression, which amounts to evaluating the $\widehat{\pi}_{n,j}$'s and $\widehat{\phi}_n$ as well as computing $\mathrm{poly}(\ell)$ sums of 2^ℓ terms each (which correspond to the internal sum $\sum_{\alpha\in\{0,1\}^\ell}\mathtt{EQ}(\widehat{\pi}_{n,j}(\overline{z}),\alpha)\cdot x_\alpha$). The prover's computation is dominated by computing a sum of $2^{c\ell}$ terms, where each term requires a computation of the type conducted by the verifier. Recalling that $2^\ell=n$, it follows that the verifier runs in almost-linear-time (i.e., it runs in $\widetilde{O}(n)$-time), whereas the prover runs in polynomial-time (i.e., it runs in $\widetilde{O}(n^{c+1})$-time). This completes the proof of Theorem 2.2.

Comment: Applied to the t-no-SUM problem, the foregoing generic construction yields a proof system of complexity that is comparable to that of the tailored proof system presented in [24,

[7] Alternatively, if the algorithm guaranteed by Definition 2.3 uses linear space (i.e., $O(\log n)$ space), then computing $\sum_{w\in\{0,1\}^{\ell'}}\Phi_x(w)$ is in log-space uniform $\mathcal{NC}$, and one can use the proof system of [27].

[8] Note that the transformation to this form can be performed in polynomial time, whereas the relevant formulae are of $\mathrm{poly}(\log n)$-size.

Sec. 4]. In case of t-no-CLIQUE, the generic construction yields a verifier that runs in time that is almost linear in the size of the adjacency matrix, whereas the tailored proof system (presented in Section 2.1) yields a verifier that runs in time that is almost linear in the number of edges.[9] Furthermore, the prover's complexity in this generic construction is n times slower than that of the tailored proof system (as optimized in [24, Sec. 3]).

Remark 2.4 (a relaxation of Definition 2.3): *Theorem 2.2 holds also when relaxing the notion of locally-characterizable sets such that the* $\mathrm{poly}(\log n)$*-sized formulae are required to be generated in* $\widetilde{O}(n)$*-time, rather than in* $\mathrm{poly}(\log n)$*-time. Actually, the foregoing proof remains intact even if the said formulae may depend on x itself, but we consider the class as defined in Definition 2.3 more natural.*

Remark 2.5 (counting the number of violated conditions): *The foregoing interactive proof system is applicable also to the task of verifying the number of violated conditions. The same applies also to the problem-tailored proof system presented in Section 2.1. Note that the number of violated conditions in* t-no-CLIQUE *is the number of t-cliques in the graph.*

2.2.3 A round complexity versus computational complexity trade-off

By using a set H of arbitrary size (rather than $H \equiv \{0,1\}$), we obtain a general trade-off between the computational complexity and the number of communication rounds in doubly-efficient interactive proof system for locally-characterizable sets. Specifically, the computational complexity (for each party) increases by a factor of $\widetilde{O}(|H|)$, whereas the number of rounds is decreased by a factor of $\log |H|$.

Theorem 2.6 (Theorem 2.2, generalized): *Every locally-characterizable set has a simple doubly-efficient interactive proof system. Specifically, on input of length n and using a parameter $h \leq n$, we get public-coin interactive proof systems of round-complexity $O(\log_h n)$, verification time $\widetilde{O}(h \cdot n)$, and proving time $\widetilde{O}(h \cdot n^{c+1})$.*

In particular, setting $h = n^\epsilon$ for any constant $\epsilon > 0$, yields a constant-round interactive proof of verification time $\widetilde{O}(n^{1+\epsilon})$. On the other hand, using $h = \log n$ maintains the computational complexity bounds of Theorem 2.2 (i.e., $\widetilde{O}(n)$-time verification and $\widetilde{O}(n^{c+1})$-time prover strategy) while using $o(\log n)$ rounds of communication.

Proof: Letting $d = \log h$, we use $H = [2^d] \equiv \{0,1\}^d$ but maintain $\ell = \log n$. Fixing $x \in \{0,1\}^n$, let $\widehat{\Phi}_x : \mathcal{F}^{c\ell} \to \mathcal{F}$ be the low degree polynomial derived in the proof of Theorem 2.2, and recall that $\sum_{w \in \{0,1\}^{c\ell}} \widehat{\Phi}_x(w) = 0$ if and only if x is in the (corresponding locally characterizable) set. Wishing to replace the sum over $\{0,1\}^{c\ell}$ by a sum over $H^{c\ell/d}$, we wish to map H to d-long binary sequences (and extend this mapping to a mapping of the form $\mathcal{F} \mapsto \mathcal{F}^d$). This is done by the mapping $\psi : H \to \{0,1\}^d$, or rather by its low degree extension $\widehat{\psi} : \mathcal{F} \to \mathcal{F}^d$, which is defined as follows

$$\widehat{\psi}(z) = \sum_{\alpha \in H} \left(\prod_{\beta \in H \setminus \{\alpha\}} \frac{\beta - z}{\beta - \alpha} \right) \cdot \psi(\alpha), \tag{2.8}$$

[9] The improved complexity is due to the fact that, in the case of t-no-CLIQUE, where the graph $G = ([n], E)$ is represented by an n-by-n matrix $(x_{j,k})_{j,k \in [n]}$, the sum $\sum_{\alpha \in [n] \times [n]} \mathtt{EQ}(\widehat{\pi}_{n,j}(\overline{z}), \alpha) \cdot x_\alpha$ can be replaced by the sum $\sum_{\alpha \in E} \mathtt{EQ}(\widehat{\pi}_{n,j}(\overline{z}), \alpha) \cdot x_\alpha$.

where $\psi(\alpha) \in \{0,1\}^d \subset \mathcal{F}^d$ is a d-long sequence that represents the binary expansion of α (where α is viewed as a d-bit long string). Indeed, $\widehat{\psi}$ is a d-long sequence of univariate polynomials (of degree $|H|-1$) over $\mathcal{F}$, and $\widehat{\psi}$ maps elements of H to d-long sequences over $\{0,1\} \subset \mathcal{F}$. Next, extend the definition of $\widehat{\psi}$ to $(c\ell/d$-long) sequences over $\mathcal{F}$ such that $\widehat{\psi}(\alpha_1,...,\alpha_{c\ell/d}) = (\widehat{\psi}(\alpha_1),...,\widehat{\psi}(\alpha_{c\ell/d}))$.[10] Now, define $\widehat{\Phi}'_x : \mathcal{F}^{c\ell/d} \to \mathcal{F}$ such that $\widehat{\Phi}'_x(\overline{z}) = \widehat{\Phi}_x(\widehat{\psi}(\overline{z}))$, and note that $\sum_{w \in H^{c\ell/d}} \widehat{\Phi}'_x(w) = 0$ if and only if x is in the set.

Invoking the Sum-Check protocol on the claim that the sum $\sum_{w \in H^{c\ell/d}} \widehat{\Phi}'_x(w)$ equals 0, yields a protocol that performs $c\ell/d$ iterations. The computational complexities of the corresponding strategies increase by a factor of $\widetilde{O}(|H|)$, due to the composition with $\widehat{\psi}$, since this composition increases the degree of some polynomials (by a factor of $|H|$) as well as the complexity of conducting certain computations. Specifically, the verifier evaluates the residual expression (i.e., $\widehat{\Phi}'_x(r)$ for some $r \in \mathcal{F}^{c\ell/d}$), which amounts to evaluating the $\widehat{\pi}_{n,j}$'s and $\widehat{\phi}_n$ as well as computing $\mathrm{poly}(\ell)$ sums of $|H|^{\ell/d} = 2^\ell$ terms each (i.e., terms of the form $\mathtt{EQ}(\widehat{\pi}_{n,j}(\widehat{\psi}(r)),\alpha)\cdot x_\alpha$ for some j and $\alpha \in \{0,1\}^\ell \equiv [n]$), but here evaluating these terms also requires evaluating $\widehat{\psi}$ (at the point r). (As before, the prover's computation is dominated by computing a sum of $|H|^{c\ell/d} = 2^{c\ell}$ terms, where each term requires a computation of the type conducted by the verifier.) ■

2.2.4 Extension to a wider class

As a motivation towards the following extension, we consider the problem of finding a dominating set of constant size t. This problem does not seem to be locally characterizable in the sense of Definition 2.3 (cf. [41]), but it definitely resides in the following class.

Definition 2.7 (locally ∀∃-characterizable sets): *A set S is* locally ∀∃-characterizable *if there exist constants c, c', a polynomial p and an almost-linear time algorithm that on input 1^n outputs $\mathrm{poly}(\log n)$-sized formulae $\phi_n : [n]^{p(\log n)} \times \{0,1\}^{p(\log n)} \to \{0,1\}$ and $\pi_{n,1},...,\pi_{n,p(\log n)} : \{0,1\}^{(c+c')\log n} \to [n]$ such that, for every $x \in \{0,1\}^n$, it holds that $x \in S$ if and only if for all $w \in \{0,1\}^{c\log n}$ there exists $w' \in \{0,1\}^{c'\log n}$ such that*

$$\Phi_x(w,w') \stackrel{\mathrm{def}}{=} \phi_n(\pi_{n,1}(w,w'),...,\pi_{n,p(\log n)}(w,w'),x_{\pi_{n,1}(w,w')},...,x_{\pi_{n,p(\log n)}(w,w')}) \tag{2.9}$$

equals 0.

Like in Definition 2.3, sometimes one may use a simplified form in which ϕ_n only depends on its $p(\log n)$-bit long suffix. This simplification suffices for the characterizing the set of graphs not having a dominating set of size $t = O(1)$. Specifically, when representing n-vertex graphs by their adjacency matrix x (augmented with 1's on the diagonal), the t-dominating set problem is captured by $\Phi_x(i_1,...,i_t,i_{t+1}) = \vee_{j\in[t]} x_{i_j,i_{t+1}}$; that is, x has no t-dominating set if and only if for every $w = (i_1,...,i_t) \in [n]^t$ there exists $w' = i_{t+1} \in [n]$ such that $\Phi_x(w,w') = 0$ (i.e., we use $\phi_{n^2} : \{0,1\}^t \to \{0,1\}$ and $\pi_{n^2,j} : \{0,1\}^{(t+1)\cdot\log n} \to [n] \times [n]$ such that $\phi_{n^2}(\sigma_1,...,\sigma_t) = \vee_j \sigma_j$ and $\pi_{n^2,j}(i_1,...,i_t,i_{t+1}) = (i_j,i_{t+1})$ for $j \in [t]$).

Every locally ∀∃-characterizable set is in $\mathcal{SC}$ (resp., in $\mathcal{NC}$) and so the existence of a doubly-efficient interactive proof system for it (and its complement) follows by [42] (resp., by [27] if the algorithm guaranteed by Definition 2.7 uses log-space). A direct construction that establishes the following result is provided in [24, Sec. 5.4].

[10] Indeed, this extended $\widehat{\psi}$ maps $\mathcal{F}^{c\ell/d}$ to $(\mathcal{F}^d)^{c\ell/d} = \mathcal{F}^{c\ell}$.

Theorem 2.8 (Theorem 2.2, extended): *Every locally ∀∃-characterizable set has a simple doubly-efficient interactive proof system. Specifically, on input of length n, the verifier runs in $\widetilde{O}(n)$-time and the strategy of the prescribed prover can be implemented in $\widetilde{O}(n^{c+c'+1})$-time, where $n^{c+c'}$ denotes the number of local conditions in the characterization. Furthermore, the interactive proof system uses public coins and has logarithmic round complexity.*

Note that the complement of every locally ∀∃-characterizable set also has a doubly-efficient interactive proof system. In this proof system, on input $x \in \{0,1\}^n$, letting $\ell = \log n$, the prover finds an adequate $w \in \{0,1\}^{c\ell}$, sends it to the verifier, and the parties engage in a doubly-efficient interactive proof of the residual claim that *for every $w' \in \{0,1\}^{c'\ell}$ it holds that $\Phi_{x,w}(w') \stackrel{\text{def}}{=} \Phi_x(w,w')$ evaluates to 0*. (Such a proof system is provided by Theorem 2.2.)

Remark 2.9 (beyond ∀∃-characterization): *The notion of a locally ∀∃-characterizable set can be further extended to allow for a constant number of* (alternating) *quantifiers; for example, a ∀∃∀-characterization corresponds to the case that for all $w \in \{0,1\}^{c \log n}$ there exists $w' \in \{0,1\}^{c' \log n}$ such that for all $w'' \in \{0,1\}^{c'' \log n}$ it holds that $\Phi_x(w,w',w'') = 0$. The proof of Theorem 2.8 extends naturally to that case* (cf. the proof of Toda's Theorem [47]).

Lastly, we note the correspondence between the foregoing local characterizations and the levels of a known hierarchy of *parametrized complexity classes* [16]. In particular, Definition 2.3 corresponds to a class denoted $\mathcal{W}[1]$, and Definition 2.7 corresponds to $\mathcal{W}[2]$. (In terms of the $\mathcal{W}$-hierarchy, our definitions are restricted in requiring that the "defining circuits" be more uniform.)

2.3 The second construction for t-no-CLIQUE

In this section we present an alternative interactive proof system for the number of t-cliques in a graph, with a focus on the case $t = O(1)$. The interactive proof proceeds in t iterations such that in the i^{th} iteration verifying the number of $(t-i+1)$-cliques in a graph is reduced to verifying the number of $(t-i)$-cliques in a related graph. The reduction proceeds in two conceptual steps.

First, we reduce verifying the number of t'-cliques in $G' = ([n'], E')$ to verifying, for each vertex $j \in [n']$, the number of t'-cliques (in G') that contain the vertex j, which equals the number of $(t'-1)$-cliques in the graph induced by the neighbors of j. Next, we let the parties reduce the latter n' claims to a single claim regarding the number of $(t'-1)$-cliques in a new graph, which has the same number of vertices as G', where the reduction is via a single-round of (randomized) interaction. That is, while the first step employs downwards reducibility, where the decreased parameter is the size of the counted cliques, the second step employs *batch verification* (cf., [42] or Chapter 4), where *the verification of n' claims is reduced to a verification of a single claim of the same type.*

Essentially, the foregoing batch verification is performed by considering an error correcting encoding of the n' graphs by a sequence of $\text{poly}(n')$ graphs, each having n' vertices. The prover sends a succinct description of the number of $(t'-1)$-cliques in these $\text{poly}(n')$ graphs, and the verifier verifies that the sum of the values associated with the n' first graphs equals the claim regarding the number of t-cliques in G'. If so, then the verifier select one of the $\text{poly}(n')$ graphs at random for the next iteration (in which it verifiers the number of $(t'-1)$-cliques in the selected graph). Indeed, the crucial point is finding a suitable encoding scheme, and this will be presented in the sequel.

We stress that the foregoing procedure does not use the sum-check protocol, and that each iteration starts and ends with an intuitive combinatorial claim to be verified. (Algebra "raises its ugly head" only in the encoding scheme, which utilizes a so-called multiplication code [36], and in the fact that cliques are indicated by products of all corresponding "edge indicators".) A concrete advantage of the current interactive proof system over the one presented in Section 2.3 is that the prover's complexity is proportional to the complexity of counting t-cliques (which is lower than $n^{2.373\cdot\lceil t/3\rceil}$) rather than being proportional to n^t. The foregoing procedure works also for varying $t = t(n)$, yielding an *alternative interactive proof system for* $\#\mathcal{P}$.

Theorem 2.10 (interactive proof systems for counting cliques of given size): *For an efficiently computable function $t : \mathbb{N} \to \mathbb{N}$, let S_t denote the set of pairs (G, N) such that the graph $G = ([n], E)$ has N distinct $t(n)$-cliques. Then, S_t has a t-round* (public coin) *interactive proof system in which the verifier's running time is $\widetilde{O}(t(n)^2 \cdot n^2)$, and the prover's running time is $\widetilde{O}(t(n)^2 \cdot n^{1+\omega_{\mathtt{mm}}\cdot\lceil(t(n)-1)/3\rceil})$, where $\omega_{\mathtt{mm}}$ is the matrix multiplication exponent. Furthermore, the prover can be implemented in $\mathrm{poly}(t(n))\cdot\widetilde{O}(n^2)$-time, when given access to an oracle for counting $(t(n)-1)$-cliques in $\mathrm{poly}(t(n)) \cdot \widetilde{O}(n)$-vertex graphs.*

Towards proving Theorem 2.10, it is useful to consider a generalization of the t-clique counting problem to vertex-weighted graphs, where the weight of a set (of vertices) $S \subseteq [n]$ is defined as the *product* of the weights of its elements.[11] That is, for a (simple) graph $G = ([n], E)$ and a sequence of vertex weights, $w = (w_1, ..., w_n) \in (\mathbb{N} \cup \{0\})^n$, we let $\mathtt{CWC}_t^G(w)$ (standing for *count weighted cliques*) denote the sum of the weights of all t-cliques in G; that is,

$$\mathtt{CWC}_t^G(w) \stackrel{\text{def}}{=} \sum_{S \in \binom{[n]}{t}:\mathtt{CL}(G_S)} \prod_{j\in S} w_j \tag{2.10}$$

where G_S is the subgraph of G induced by S and $\mathtt{CL}(G')$ holds if G' is a clique. Indeed, $\mathtt{CWC}_t^G(1^n)$ equals the number of t-cliques in G. In general, as shown in Section 2.3.2, $\mathtt{CWC}_t^G(w)$ equals the number of t-cliques in a graph G' that is obtained by a blow-up of G in which the i^{th} vertex is replaced by an independent set of size w_i (and the edge $\{i, j\}$ is replaced by a complete bipartite graph between the i^{th} and j^{th} sets).

2.3.1 A direct interactive proof for counting weighted cliques

Towards presenting an interactive proof system for the value of $\mathtt{CWC}_t^G(1^n)$, we first observe that (for every $w \in (\mathbb{N} \cup \{0\})^n$) it holds that

$$t \cdot \mathtt{CWC}_t^G(w) = \sum_{i\in[n]} w_i \cdot \mathtt{CWC}_{t-1}^G(w^{(i)}) \tag{2.11}$$

$$\text{where } w_j^{(i)} = w_j \text{ if } \{i, j\} \in E, \text{ and } w_j^{(i)} = 0 \text{ otherwise.} \tag{2.12}$$

(This is the case since each pair (S, i) such that S is a t-subset of $[n]$ and $i \in S$ contributes equally to each side of Eq. (2.11), where the contribution equals $w_i \cdot \prod_{j\in S\setminus\{i\}} w_j$ if G_S is a clique

[11] Our definition stands in contrast to the standard practice of defining the weight of a set as the sum of its elements (cf. [2]). However, such a definition was used before (see, e.g., the definition of a weighted cycle cover in [49]).

and equals 0 otherwise. In particular, note that if all vertices in $S \setminus \{i\}$ are neighbors of i, then $\prod_{j \in S \setminus \{i\}} w_j^{(i)} = \prod_{j \in S \setminus \{i\}} w_j$ and otherwise $\prod_{j \in S \setminus \{i\}} w_j^{(i)} = 0$.)

Indeed, Eq. (2.11) reduced the evaluation of $\mathtt{CWC}_t^G$ to n evaluations of $\mathtt{CWC}_{t-1}^G$; equivalently, it reduces the verification of the value of $\mathtt{CWC}_t^G$ at one point to the verification of the value of $\mathtt{CWC}_{t-1}^G$ at n points. Wishing to reduce these n value-verifications to a single one, we seek an error correcting code by which these n values can be encoded by a sequence of $\Omega(n)$ values such that verifying one of the values in the sequence suffice. As usual, the code of choice is the Reed-Solomon code (i.e., univariate polynomials of degree $n-1$), which calls for embedding the values in a finite field. Recalling that we are actually interested in the value of $\mathtt{CWC}_t^G(1^n) \leq t! \cdot \binom{n}{t} < n^t$, we embed all values in a finite field $\mathcal{F} = \mathrm{GF}(p)$, for a prime $p > n^t$, and reduce all w_j's modulo p. Now, we consider the following n polynomials $(f_j : \mathcal{F} \to \mathcal{F})_{j \in [n]}$:

$$f_j(z) \stackrel{\text{def}}{=} \sum_{i \in [n]} \mathtt{EQ}_i(z) \cdot w_j^{(i)} \tag{2.13}$$

where $\mathtt{EQ}_i : \mathcal{F} \to \mathcal{F}$ is a (degree $n-1$) polynomial such that $\mathtt{EQ}_i(i) = 1$ and $\mathtt{EQ}_i(k) = 0$ for every $k \in [n] \setminus \{i\}$ (i.e., $\mathtt{EQ}_i(z) = \prod_{j \in [n] \setminus \{i\}} (z-j)/(i-j)$). Note that $f_j(i) = w_j^{(i)}$ for every $i \in [n]$, and that each f_j can be computed in $O(n)$ field operations when given w (by using Eq. (2.12) and $\mathtt{EQ}_i(z) = \prod_{j \in [n] \setminus \{i\}} (z-j)/(i-j)$).[12] Letting $\mathtt{CWC}_t^{G,p}(w) \stackrel{\text{def}}{=} \mathtt{CWC}_t^G(w) \bmod p$, this leads to the following interactive reduction of the verification of the value of $\mathtt{CWC}_t^{G,p}$ at one point to the verification of the value of $\mathtt{CWC}_{t-1}^{G,p}$ at one (other) point.

Construction 2.11 (a generic iteration of the interactive proof system for counting cliques): *The iteration starts with a claim of the form $\mathtt{CWC}_t^{G,p}(w) = v$, where $w \in \mathcal{F}^n$ and $v \in \mathcal{F}$ are determined before* (by the previous iteration or by the main protocol).[13]

1. *The prover computes the $(t-1) \cdot (n-1)$ degree polynomial $P_{t-1} : \mathcal{F} \to \mathcal{F}$, where*

$$P_{t-1}(z) \stackrel{\text{def}}{=} \sum_{S \in \binom{[n]}{t-1} : \mathtt{CL}(G_S)} \prod_{j \in S} f_j(z), \tag{2.14}$$

   where the arithmetic is over $\mathcal{F} = \mathrm{GF}(p)$, and sends P_{t-1} to the verifier.

   Note that $P_{t-1}(z)$ can be computed by interpolation, using the values of P_{t-1} at less than tn points, and that (as shown in the proof of Proposition 2.12) *for every $k \in \mathcal{F}$ it holds that $P_{t-1}(k) = \mathtt{CWC}_{t-1}^{G,p}(w^{(k)})$, where $w_j^{(k)} = f_j(k)$ for every $j \in [n]$.*

2. *Upon receiving a polynomial $\widetilde{P}$ of degree $(t-1) \cdot (n-1)$, the verifier checks whether $t \cdot v \equiv \sum_{i \in [n]} w_i \cdot \widetilde{P}(i) \pmod p$, and rejects if equality does* not *hold. If equality holds, the verifier selects uniformly $r \in \mathcal{F}$, and sends it to the prover.*

[12] Note that, for any $k \in [n]$ it holds that $(\mathtt{EQ}_1(k), ..., \mathtt{EQ}_n(k)) = 0^{k-1}10^{n-k}$, whereas for $k \in \mathcal{F} \setminus [n]$ it holds that $\mathtt{EQ}_i(k) = \frac{\prod_{j \in [n]}(k-j)}{(k-i) \cdot \prod_{j \in [n] \setminus \{i\}}(i-j)}$. Hence, computing $(\mathtt{EQ}_1(k), ..., \mathtt{EQ}_n(k))$ reduces to computing $\prod_{j \in [n]}(k-j)$ and computing all $\prod_{k \in [m]} k$ for $m = 1, ..., n-1$, which in turn means that $(\mathtt{EQ}_1(k), ..., \mathtt{EQ}_n(k))$ can be computed in $O(n)$ field operations.

[13] Indeed, as hinted in the title of this construction, it will be applied iteratively by the main interactive proof systems, which contains little beyond these iterations (see the proof of Theorem 2.13).

The iteration ends with the claim that $\mathtt{CWC}_{t-1}^{G,p}(w') = v'$*, where* $v' = \widetilde{P}(r)$ *and* $w'_j = f_j(r)$ *for every* $j \in [n]$.

Recall that both parties can compute each of the f_j's using $O(n)$ field operations, which implies that the verifier strategy can be implemented in $\widetilde{O}(t \cdot n^2)$-time (assuming the field has size $O(n^t)$).[14] In addition, the prover needs to construct the polynomial P_{t-1}, and a straightforward way of doing so can be implemented using $tn \cdot (O(n^2) + \binom{n}{t-1}) = O(n^t)$ field operations. We shall later see that the complexity can be improved to $O(t \cdot n^{1+\omega_{\mathtt{mm}}\lceil (t-1)/3 \rceil})$, where $\omega_{\mathtt{mm}}$ is the matrix multiplication exponent, by using the ideas that underlie the best algorithm known for deciding t-Clique (which also suffices for counting t-cliques). But before getting to this improvement, we analyze the effect of Construction 2.11.

Proposition 2.12 (analysis of Construction 2.11): *Let* w, v, w' *and* v' *be as in Construction 2.11.*

1. *If* $\mathtt{CWC}_t^{G,p}(w) = v$ *and both parties follow their instructions, then* $\mathtt{CWC}_{t-1}^{G,p}(w') = v'$ *holds.*
2. *If* $\mathtt{CWC}_t^{G,p}(w) \neq v$ *and the verifier follows its instructions, then either the verifier rejects or* $\mathtt{CWC}_{t-1}^{G,p}(w') = v'$ *holds with probability at most* $tn/|\mathcal{F}|$.

Proof: Using Eq. (2.14), Eq. (2.13), and Eq. (2.10), we get for every $i \in [n]$

$$\begin{aligned} P_{t-1}(i) &= \sum_{S \in \binom{[n]}{t-1} : \mathtt{CL}(G_S)} \prod_{j \in S} f_j(i) \\ &= \sum_{S \in \binom{[n]}{t-1} : \mathtt{CL}(G_S)} \prod_{j \in S} w_j^{(i)} \\ &= \mathtt{CWC}_{t-1}^{G,p}(w^{(i)}), \end{aligned}$$

and so $t \cdot \mathtt{CWC}_t^G(w) \equiv \sum_{i \in [n]} w_i \cdot P_{t-1}(i) \pmod{p}$. More generally, by the same argument, for every $k \in \mathcal{F}$, it holds that $P_{t-1}(k) = \mathtt{CWC}_{t-1}^{G,p}(w^{(k)})$, where $w_j^{(k)} = f_j(k)$ for every $j \in [n]$.

Under the hypothesis of Item 1, $v = \mathtt{CWC}_t^{G,p}(w)$ and the polynomial $\widetilde{P}$ received by the verifier equals P_{t-1}, which implies that the verifier does not reject (since $t \cdot v = \sum_{i \in [n]} w_i \cdot \widetilde{P}(i)$ over $\mathcal{F} = \mathrm{GF}(p)$). Furthermore, in this case $v' = \widetilde{P}(r)$ equals $P_{t-1}(r) = \mathtt{CWC}_{t-1}^{G,p}(w')$, where $w'_j = f_j(r)$ for every $j \in [n]$.

Under the hypothesis of Item 2, the polynomial P_{t-1} does *not* satisfy the equation $t \cdot v = \sum_{i \in [n]} w_i \cdot P_{t-1}(i)$ over $\mathcal{F}$. If the prover sets $\widetilde{P} = P_{t-1}$, then the verifier rejects, and otherwise $\widetilde{P}$ and P_{t-1} may agree on at most $(t-1) \cdot (n-1)$ points. In the latter case, unless r is one of the agreement points, it follows that $\mathtt{CWC}_{t-1}^{G,p}(w') = P_{t-1}(r) \neq \widetilde{P}(r)$, where $w'_j = f_j(r)$ for every $j \in [n]$. ■

The interactive proof system. Iteratively invoking Construction 2.11 yields an interactive proof system for the claim $\mathtt{CWC}_t^{G,p}(w) = v$, where $w \in \mathcal{F}^n$ and $v \in \mathcal{F}$. In the i^{th} iteration we start with a claim of the form $\mathtt{CWC}_{t-i+1}^{G,p}(w^{(i-1)}) = v^{(i-1)}$, and end with a claim of the form $\mathtt{CWC}_{t-i}^{G,p}(w^{(i)}) =$

[14] Note that $\widetilde{P}$ can be evaluated using $O(tn)$ field operations.

$v^{(i)}$. Hence, after $t-2$ iterations, we reach a claim of the form $\mathtt{CWC}_2^{G,p}(w^{(t-2)}) = v^{(t-2)}$, which the verifier can verify by itself. To apply this procedure to the problem of counting t-cliques in an n-vertex graph, we initiate it by selecting a field of prime cardinality greater than n^t, and setting $w^{(0)} = (1, ..., 1) \equiv 1^n$. Hence, we get

Theorem 2.13 (interactive proof systems for counting cliques of given size): *For an efficiently computable function $t : \mathbb{N} \to \mathbb{N}$, let S_t denote the set of pairs (G, N) such that the graph $G = ([n], E)$ has N distinct $t(n)$-cliques. Then, S_t has a t-round* (public coin) *interactive proof system in which the verifier's running time is $\widetilde{O}(t(n)^2 \cdot n^2)$, and the prover's running time is dominated by $O(t(n)^2 \cdot n)$ calls to the oracle $\overline{\mathtt{CWC}}_{t(n)-1}^{G,p}$, where p is a prime number in $[n^{t(n)}, 2n^{t(n)}]$, and $\overline{\mathtt{CWC}}_t^{G,p}$ answers the query $(i, w) \in [t] \times \mathrm{GF}(p)^n$ with $\mathtt{CWC}_i^{G,p}(w)$.*

Indeed, this suggests an alternative construction of interactive proof systems for sets in co$\mathcal{NP}$. The fact that this construction is different from the celebrated construction of [35] is manifested by the (reduced) running time of the prover. Specifically, as shown in Section 2.3.2, the computation of $\mathtt{CWC}_t^{G,p}$ is $O(t^2|w|)$-time reducible to counting the number of t-cliques in $\widetilde{O}(t^2 n)$-vertex graphs, which in turn has complexity $O(n^{\omega_{\mathtt{mm}}\lceil t/3 \rceil})$, where $\omega_{\mathtt{mm}}$ is the matrix multiplication exponent (see [39]).

Proof: On input $G = ([n], E)$ and N (which is supposed to equal the number of $t(n)$-cliques in G), the parties pick a finite field $\mathcal{F}$ of prime cardinality greater than $n^{t(n)}$, and initialize $w = 1^n$ and $v = N$. (This prime can be selected at random in $[n^{t(n)}, 2n^{t(n)}]$ by either parties.) Next, they iteratively invoke Construction 2.11 for $t(n) - 2$ times (with decreasing clique-size parameter (starting at size $t(n)$ and ending at size 3)). In case the verifier did not reject (in any iteration), it is left with a claim of the form $\mathtt{CWC}_2^G(w) = v$, which it can verify by itself in $O(n^2)$ time (since $\mathtt{CWC}_2^G(w)$ equals $\sum_{\{j,k\} \in E} w_j \cdot w_k$). For sake of clarity, we spell out the protocol.

1. On input $G = ([n], E)$ and $N \leq \binom{n}{t(n)}$, the verifier selects uniformly a prime number p in $[n^{t(n)}, 2n^{t(n)}]$, and sends it to the prover. Both parties set $\mathcal{F} = \mathrm{GF}(p)$, and initiate $w^{(0)} \leftarrow 1^n$ and $v^{(0)} \leftarrow N$.

2. For $i = 1, ..., t(n) - 2$, the parties invoke Construction 2.11, while setting $t \leftarrow t(n) - i + 1$, $w \leftarrow w^{(i-1)}$ and $v \leftarrow v^{(i-1)}$. Once Construction 2.11 terminates, they set $w^{(i)} \leftarrow w'$ and $v^{(i)} \leftarrow v'$. (In all invocations, $\mathcal{F}$ is as set in Step 1.)

3. The verifier checks whether $\mathtt{CWC}_2^G(w^{(t(n)-2)}) = v^{(t(n)-2)}$ by direct computation.

All claims of the theorem follow quite immediately. In particular, in the i^{th} iteration, the verifier performs $O((t(n) - i) \cdot n^2)$ field operations, whereas the prover's computation is dominated by less than $t(n) \cdot n$ invocations of the oracle $\mathtt{CWC}_{t(n)-i}^{G,p}$. ■

Remark 2.14 (implementing $\mathtt{CWC}_{t'-1}$ using $\mathtt{CWC}_{t'}$): *We comment that, for any $G = ([n], E)$, the oracle $\mathtt{CWC}_{t'-1}^G$* (resp., $\mathtt{CWC}_{t'-1}^{G,p}$) *can be implemented using two oracle calls to $\mathtt{CWC}_{t'}^{G'}$* (resp., $\mathtt{CWC}_{t'}^{G',p}$), *where $G' = ([n+1], E')$ such that $E' = E \cup \{\{j, n+1\} : j \in [n]\}$. Specifically, $\mathtt{CWC}_{t'-1}^G(w) = \mathtt{CWC}_{t'}^{G'}(w1) - \mathtt{CWC}_{t'}^{G'}(w0)$, since $(t'-1)$-cliques in G correspond to t'-cliques in G' that contain the auxiliary vertex $n+1$.*

Hence, $\overline{\mathtt{CWC}}_t^G$ is $O(2^t n)$-time reducible to $\mathtt{CWC}_t^{G'}$, where G' has t more vertices than G. Specifically, $\mathtt{CWC}_{t-i}^G(w)$ is a linear combination (with coefficients in $\{\pm 1\}$) of the 2^i values $(\mathtt{CWC}_t^{G'}(w\alpha 0^{t-i}))_{\alpha\in\{0,1\}^i}$. Observing that $f(\beta) \stackrel{\text{def}}{=} \mathtt{CWC}_t^G(w\beta)$ depends only on the Hamming weight of $\beta \in \{0,1\}^t$, we can write $\mathtt{CWC}_{t-i}^G(w)$ as a linear combination of the $i+1$ values $(\mathtt{CWC}_t^G(w0^j 1^{i-j} 0^{t-i}))_{j\in[[i]]}$. Hence, we obtain an $O(tn)$-time reduction of $\overline{\mathtt{CWC}}_t^G$ to $\mathtt{CWC}_t^{G'}$. A alternative ($O(tn)$-time) reduction is presented next.

Remark 2.15 (implementing $\mathtt{CWC}_{t-i}$ using $\mathtt{CWC}_t$): *For any $G = ([n], E)$, the oracle $\mathtt{CWC}_{t-i}^G$ (resp., $\mathtt{CWC}_{t-i}^{G,p}$) can be implemented using one oracle call to $\mathtt{CWC}_t^{G^{(i)}}$ (resp., $\mathtt{CWC}_t^{G^{(i)},p}$), where $G^{(i)}$ consists of $t-i$ independent sets of size n that are connected by a double-cover of G, and augmented by an i-clique that is connected to all these $(t-i)\cdot n$ vertices.*[15] *That is, $G^{(i)} = ([(t-i)n+i], E' \cup A)$ such that $E' = \{\{(a-1)n+j, (b-1)n+k\} : \{j,k\} \in E \,\&\, a \neq b \in [t-i]\}$ and $A = \{\{j, (t-i)n+k\} : j \in [(t-i)n] \,\&\, k \in [i]\}$. Specifically, $(t-i)! \cdot \mathtt{CWC}_{t-i}^G(w) = \mathtt{CWC}_t^{G^{(i)}}(w^{t-i}1^i)$. Seeking to use the same number of vertices in all graphs $G^{(i)}$, we may augment $G^{(i)}$ with $i \cdot n - i$ isolated vertices.*

Discussion: The foregoing proof system for $\#\mathcal{P}$ versus the standard one

The standard (sum-check based) interactive proof system for #SAT starts with a full arithmetization of the Boolean problem [35], which yields an arithmetic problem that has no clear intuitive meaning. This is done by writing the CNF formula as an arithmetic formula over GF(2), and then considering it as an Arithmetic formula over a larger field. Alternatively, given a Boolean formula over n variables, viewed as a function from $\{0,1\}^n$ to $\{0,1\}$, we write the low-degree extension of this function.

In contrast, the foregoing interactive proof system for counting the number of cliques (in a graph) proceeds in iteration such that *in each iteration the clique counting problem is reduced to itself*. Hence, the claim made at the beginning (and the end) of each iteration has a clear intuitive meaning.[16] (As noted upfront and elaborated in Section 2.3.2, the sum of weighted t-cliques in a graph G equals the number of t-cliques in a corresponding graph obtained by a suitable blow-up of G.) Furthermore, each iteration consists of a natural downwards reduction, which decreases the size parameter by one unit, and a batch verification that reduces n claims to a single one. Both reductions have an intuitive appeal, although the batch verification relies on a suitable error correcting code, which is a "multiplication code" in a sense akin to the sense used in Meir's work [36]. To simplify the exposition, we use the Reed-Solomon code, but any code supporting a sufficient number of multiplications will do.

The difference between the foregoing construction and the standard one may be articulated using Meir's observation [36] that the standard construction uses the fact that Reed-Muller codes are tensor codes (and that one can use arbitrary tensor (of multiplication) codes). Specifically, the standard construction uses a codeword that encodes the 2^n values of the Boolean formula over all assignments to the n variables. In contrast, the foregoing construction does not use tensor codes, and in each iteration we encode n values that correspond to n instances of the problem. (Indeed, like in the standard interactive proof systems for co$\mathcal{NP}$, we actually refer to the counting problem.)

[15] The double-cover of $G = ([n], E)$ is a bipartite graph B with n vertices on each side such that $\langle 1, u\rangle$ and $\langle 2, v\rangle$ are connected in B if and only if $\{u, v\} \in E$.

[16] The same holds for the interactive proof system for the Permanent presented in [35] in which the permanent of an n-by-n matrix is first expressed as the sum of the permanents of n related $(n-1)$-by-$(n-1)$ matrices, and then batch verification is employed.

2.3.2 Reducing counting vertex-weighted cliques to the unweighted case

Recall that the prover strategy underlying the interactive proof system presented in the proof of Theorem 2.13 can be implemented in $O(t(n)^2n^2)$ time when given oracle access to $\overline{\mathtt{CWC}}_t^{G,p}$, where p is a prime in $[n^{t(n)}, 2n^{t(n)}]$. In fact, the prover uses $O(t(n) \cdot n)$ calls to each of oracles $\mathtt{CWC}_i^{G,p}$ for $i = 2, ..., t(n)-1$. We first observe that each such oracle can be implemented by using $O(t(n)^2 \log n)$ oracle calls to a corresponding oracle that is defined for weights that reside in $[O(t(n)^2 \log n)]$. Next, we show that each of these queries can be implemented by counting cliques is a simple $\widetilde{O}(t(n)^2n)$-vertex graph (with no weights). These two reductions are summarized in the following result.

Proposition 2.16 (reducing $\mathtt{CWC}_t^G$ to counting t-cliques in graphs): *For a graph $G = ([n], E)$ and a prime $p \in [n^t, 2n^t]$, let $\mathtt{CWC}_t^{G,p}(w) \stackrel{\text{def}}{=} \mathtt{CWC}_t^G(w) \bmod p$, where $\mathtt{CWC}_t^G$ be as in Eq. (2.10). Then, the computation of $\mathtt{CWC}_t^{G,p}(w)$ can be reduced in $\widetilde{O}(t^2|w|)$-time to computing the number of t-cliques in an unweighted graph having $\widetilde{O}(t^2n)$ vertices. Furthermore, the reduction uses $O(t^2 \log n)$ queries.*

Proof: We may assume, without loss of generality, that $w \in [p]^n$, since otherwise we reduce each coordinate of w modulo p. Observe that the value of $\mathtt{CWC}_t^G(w)$ is a non-negative integer that is smaller than $t!\binom{n}{t} \cdot p^t < (np)^t < n^{2t^2}$. For $m = O(t^2 \log n)$, it holds that the product of all primes in $[m]$ is larger than n^{2t^2}, and so (using the Chinese Remainder Theorem) we may compute $\mathtt{CWC}_t^G(w)$ by computing $\mathtt{CWC}_t^G(w)$ modulo each of these primes. Hence, for each prime p_i in $[m]$, we first reduce the weights modulo p_i, obtaining $w_j^{(i)} = w_j \bmod p_i$ for each $j \in [n]$, and then compute $\mathtt{CWC}_t^{G,p_i}(w^{(i)})$, which equals $\mathtt{CWC}_t^{G,p_i}(w)$. That is, $\mathtt{CWC}_t^G(w)$ is computed using the Chinese Remainder Theorem based on the values of $\mathtt{CWC}_t^{G,p_i}(w)$ for each prime $p_i \in [m]$.

Next, we efficiently reduce the computation of $\mathtt{CWC}_t^{G,p'}(w')$, where p' is a prime in $[m]$ and $w' \in [p']^n$, to the standard problem of counting t-cliques. Actually, we reduce the computation of $\mathtt{CWC}_t^G(w')$ to counting the number of t-cliques in a related graph G'. The reduction just maps the graph $G = ([n], E)$ with weights $w' = (w'_1, ..., w'_n) \in [m]^n$ to the graph $G' = (V', E')$ in which each vertex j of G is replaced by an independent set of size w'_j, and edges are replaced by complete bipartite graphs between the corresponding independent sets; that is

$$V' = \bigcup_{j \in [n]} V_j \quad \text{where the } V_j\text{'s are disjoint and } |V_j| = w'_j \tag{2.15}$$

$$E' = \bigcup_{\{j,k\} \in E} \{\{u, v\} : u \in V_j \ \&\ v \in V_k\}. \tag{2.16}$$

Indeed, the value of $\mathtt{CWC}_t^G(w')$ equals the number of t-cliques in G'.

Combining the two reductions, we obtain a $\widetilde{O}(t^2|w|)$-time reduction of computing $\mathtt{CWC}_t^{G,p}(w)$, where $p = O(n^t)$, to counting t-cliques in $O(t^2n \log n)$-vertex graphs. Indeed, the reduction makes $O(t^2 \log n)$ calls, where in the i^{th} call we obtain the number of t-cliques in the graph $G^{(i)}$ in which vertex j of G is replaced by an independent set of size $w_j^{(i)} \in [p_i]$ such that $w_j^{(i)} \equiv w_j \pmod{p_i}$. ■

Proof of Theorem 2.10: Combining Theorem 2.13 with Proposition 2.16 yields Theorem 2.10. Specifically, the proof system asserted in Theorem 2.13 is almost as asserted in Theorem 2.10, except that the prover strategy in the former system relies on $O(t(n)^2 \cdot n)$ calls to the oracle $\overline{\mathtt{CWC}}_{t(n)-1}^{G,p}$,

where p is a prime number in $[n^{t(n)}, 2n^{t(n)}]$, and $\overline{\mathtt{CWC}}_t^{G,p}$ answers the query $(i, w) \in [t] \times \mathrm{GF}(p)^n$ with $\mathtt{CWC}_i^{G,p}(w)$. Proposition 2.16 implies that each of these $O(t(n)^2 \cdot n)$ queries can be answered by making $O(t(n)^2 \log n)$ queries to an oracle that on input a $\widetilde{O}(t(n)^2 \cdot n)$-vertex graph G' and an integer $i \in [t(n) - 1]$ returns the number of i-cliques in G'. Lastly, using Remark 2.14 (and the discussion following it), we can implement each of these queries by $t(n)$ queries to an oracle that returns the number of $(t(n) - 1)$-cliques in $\widetilde{O}(t(n)^2 \cdot n)$-vertex graphs. Alternatively, using Remark 2.15, we can implement each of the i-clique queries by a single query to an oracle that returns the number of $(t(n) - 1)$-cliques in $\widetilde{O}(t(n)^3 \cdot n)$-vertex graphs. ■

Chapter 3

On the doubly-efficient interactive proof systems of GKR

In this chapter, we present a (somewhat simpler) variant of the doubly-efficient interactive proof systems of Goldwasser, Kalai, and Rothblum [27]. Recall that, as already stated as Theorem 1.1, these proof systems apply to log-space uniform sets in $\mathcal{NC}$; more generally, they apply to sets that are recognizable by log-space uniform circuits of bounded depth, where the number of rounds in the proof system is linearly related to the depth of the circuit.

Our simplification is in the utilization of the log-space uniformity condition. In the original work, the hypothesis that the circuits are constructable in log-space was used in an auxiliary proof system through which the prover established the correctness of selected parts of the encoding of the relevant circuit. In contrast, we avoid this auxiliary proof system, and apply the main proof system to a highly regular *universal* circuit that constructs and evaluates the relevant log-space uniform circuit.

3.1 Overview

We provide an (alternative) exposition of the result of Goldwasser, Kalai, and Rothblum [27], which asserts the following –

Theorem 3.1 (doubly-efficient interactive proof systems for log-space uniform $\mathcal{NC}$): *Let $d : \mathbb{N} \to \mathbb{N}$ and A be an algorithm of logarithmic space complexity that on input 1^n outputs a Boolean circuit $C_n : \{0,1\}^n \to \{0,1\}$ of bounded fan-in and depth $d(n)$. Then, the set $S = \{x : C_{|x|}(x)=1\}$ has an interactive proof system with $\mathrm{poly}(\log n) \cdot d(n)$ rounds in which the prescribed prover strategy runs in polynomial-time while the verifier runs in $\widetilde{O}(n + d(n))$-time. Furthermore, the proof system is of the public-coin type and the total communication is $\mathrm{poly}(\log n) \cdot d(n)$.*

(We note that the number of rounds is actually $O(\log n) \cdot d(n)$, but our simplified proof only yield a bound of $O(\log n)^2 \cdot d(n)$.)[1]

The core of the proof system asserted in Theorem 3.1 is an iterative process in which a claim about the values of the gates that are at distance $i-1$ from the output gate is reduced to a claim

[1] The first (resp., second) expression omits an additive term of $O(\log n)^2$ (resp., $O(\log n)^2$), which is immaterial in the typical case where $d \geq \log n$.

about the values of the gates at distance i (i.e., the gates at layer i). We stress that each of these claims refers to the values of the polynomially many gates at a specific layer of the circuit $C_{|x|}$ during the computation on input x, but these $\text{poly}(|x|)$ values are not communicated explicitly but rather only referred to. Nevertheless, in $d(|x|)$ iterations, the claim regarding the value of the output gate (i.e., the value $C_{|x|}(x)$) is reduced to a claim regarding the values of the bits of the input x, whereas the latter claim (which refers to x itself) can be verified in almost linear time.

Each of the aforementioned claims regarding the values of the gates at layer i, where $i \in \{0, 1, ..., d(|x|)\}$, is actually a claim about the value of a specified location in the corresponding encoding of the (string that describes all the) gate-values at layer i. Specifically, the encoding used is the low degree extension of the said string (viewed as a function), and the claims are claims about the evaluations of these polynomials at specific points.

The different codewords (or polynomials) are related via the structure of the circuit $C_{|x|}$, and so (as usual) this structure is represented by a (multi-variate) function that describes the adjacency relation (among the gates) of the circuit. We consider a low degree polynomial that extends this function, and the problem is allowing the verifier to evaluate this polynomial (which is defined over a $\text{poly}(|x|)$-sized domain) in almost-linear (in $|x|$) time. Here is where the uniformity condition comes into play.

Goldwasser, Kalai, and Rothblum used the uniformity condition in order to construct an interactive proof system by which the evaluation of this low degree polynomial is outsourced to the prover (who proves the validity of the provided answer) [27]. We present an alternative approach: We consider a "universal" circuit that, on input $x \in \{0,1\}^n$, first constructs the log-space uniform circuit C_n, and next emulates the computation of $C_n(x)$. We apply the aforementioned interactive process to this universal circuit rather than to C_n, with the benefit being that this universal circuit is much more uniform than C_n. Consequently, a low degree polynomial that extends the adjacency relation of that (universal) circuit can be evaluated in $\text{poly}(\log n)$-time. We stress that the universal circuit preserves the complexities of C_n up to our level of interest; that is, the universal circuit has depth $\text{poly}(\log n) \cdot d(n)$ and size $\text{poly}(n)$.

Organization. The main module of the interactive proof system asserted in Theorem 3.1 is presented in Section 3.2, while putting aside the problem of evaluating the low degree polynomial that extends the (function that represents the) adjacency relation of the circuit. Coping with this computational problem is the contents of Section 3.3. Additional details regarding Sections 3.2 and 3.3 are presented in Section 3.4 (see Sections 3.4.1 and 3.4.2, resp.).

3.2 The main module

For simplicity (and w.l.o.g.), we assume that C_n has fan-in two and uses only NAND-gates (i.e., $\text{NAND}(a,b) = \neg(a \wedge b)$). Viewing this gate as operating in a finite field that contains $\{0,1\}$, we have $\text{NAND}(a,b) = 1 - (a \cdot b)$ for $a, b \in \{0,1\}$.

By augmenting the circuit with gates that are fed by no gate (and feed no gate), we can present the circuit as having $d(n)+1$ layers of gates such that each layer has exactly $k(n) = \text{poly}(n)$ gates, where (by convention) gates that are fed nothing always evaluate to 0. As usual, the gates at layer i are only fed by gates at layer $i+1$, and the leaves (at layer $d(n)$) are input-variables or constants. Furthermore, we may assume that, for $j \in [n]$, the j^{th} leaf is fed by the j^{th} input-variable, and all other leaves are fed by the constant 0. (The output is produced at the first gate of layer zero.)

The high level protocol. On input $x \in \{0,1\}^n$, the prescribed prover computes the values of all layers. Letting $d = d(n)$ and $k = k(n)$, we denote the values at the i^{th} layer by $\alpha_i \in \{0,1\}^k$; in particular, $\alpha_d = x0^{k-n}$ and $\alpha_0 = C_n(x)0^{k-1}$. For a sufficiently large finite field, denoted $\mathcal{F}$, let $H = \{0,1\} \subset \mathcal{F}$, and $m = \log k$.[2] Viewing each α_i as a function from $H^m \equiv [k]$ to $\{0,1\}$, the prover encodes α_i by a low degree polynomial $\widehat{\alpha}_i : \mathcal{F}^m \to \mathcal{F}$ that extends it (i.e., $\widehat{\alpha}_i(\sigma) = \alpha_i(\sigma)$ for every $\sigma \in H^m$); that is,

$$\widehat{\alpha}_i(z_1, ..., z_m) = \sum_{\sigma_1,...,\sigma_m \in H} \mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) \cdot \alpha_i(\sigma_1, ..., \sigma_m), \tag{3.1}$$

where $\mathtt{EQ}$ is a low degree polynomial that extends the function that tests equality over H^m (e.g., $\mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) = \prod_{i \in [m]}(z_i\sigma_i + (1-z_i)(1-\sigma_i))$).[3] Hence, proving that $x \in S$ is equivalent to proving that $\widehat{\alpha}_0(1^m) = 1$, where $1^m \in H^m$ corresponds to the fixed (e.g., first) location of the output gate in the zero layer.

This proof is conducted in $d(n)$ iterations, where in each iteration a multi-round interactive protocol is employed. Specifically, in i^{th} iteration, the correctness of the claim $\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = v_{i-1}$, where $\overline{r}_{i-1} \in \mathcal{F}^m$ and $v_{i-1} \in \mathcal{F}$ are known to both parties, is reduced (via the interactive protocol) to the claim $\widehat{\alpha}_i(\overline{r}_i) = v_i$, where $\overline{r}_i \in \mathcal{F}^m$ and $v_i \in \mathcal{F}$ are determined (by this protocol) such that both parties get these values. We stress that, with the exception of $i = d$, the $\widehat{\alpha}_i$'s are not known (or given) to the verifier; still, the claims made at the beginning (and at the end) of each iteration are well defined (i.e., each claim refers to a predetermined low degree polynomial that extends the values assigned to the gates (of a certain layer) of the circuit in a computation of the circuit on input $x \in \{0,1\}^n$).

Once the last iteration is completed, the verifier is left with a claim of the form $\widehat{\alpha}_d(\overline{r}_d) = v_d$, which it can verify using Eq. (3.1) and its knowledge of $\alpha_d = x0^{k-n}$. Actually, some care is required here too, since the verifier needs to operate in time $\widetilde{O}(n)$ (rather than in time $\widetilde{O}(k)$, where $k = |H|^m = \text{poly}(n)$ may be significantly larger than n). For example, associating $[n]$ with $H^{m'} \equiv H^{m'}1^{m-m'}$, where $m' = \log_{|H|} n$, we use the fact that for every $\sigma_1, ..., \sigma_m \in H$ it holds that $\alpha_d(\sigma_1, ..., \sigma_m) = x_{\sigma_1 \cdots \sigma_{m'}}$ if $\sigma_{m-m'+1} \cdots \sigma_m = 1^{m-m'}$ and $\alpha_d(\sigma_1, ..., \sigma_m) = 0$ otherwise.[4] Hence,

$$\begin{aligned}
\widehat{\alpha}_d(z_1, ..., z_m) &= \sum_{\sigma_1,...,\sigma_m \in H} \mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) \cdot \alpha_d(\sigma_1, ..., \sigma_m) \\
&= \sum_{\sigma_1,...,\sigma_{m'} \in H} \mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_{m'}1^{m-m'}) \cdot \alpha_d(\sigma_1, ..., \sigma_{m'}1^{m-m'}).
\end{aligned}$$

where $\alpha_d(\sigma_1, ..., \sigma_{m'}1^{m-m'})$ equals the bit of x in location $\sigma_1 \cdots \sigma_{m'} \in H^{m'} \equiv [n]$.

[2] Actually, we can use an arbitrary fixed set $H \subset \mathcal{F}$ of $\text{poly}(\log n)$ size, and let $m = \log_{|H|} k = (\log k)/\log |H|$, as done in [21]. The advantage of this choice is that in the case that $|\mathcal{F}| = \text{poly}(|H|)$, which suffices when $d(n) = \text{poly}(\log n)$, we get $|\mathcal{F}|^m = \text{poly}(k) = \text{poly}(n)$, which means that the codewords used are of polynomial length. However, this fact is only of aesthetic value, because these codewords are not constructed explicitly by the prover, who only determines their k-bit long succinct representation. In fact, it may be methodologically better to think that it is infeasible to construct these codewords explicitly.

[3] In continuation to Footnote 2, we note that when a generic $H \subset \mathcal{F}$ is used, one may use $\mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) = \prod_{i \in [m]} \prod_{\beta \in H'}(z_i - \sigma_i + \beta)/\beta$, where $H' = \{\beta' - \beta'' : \beta', \beta'' \in H\} \setminus \{0\}$.

[4] Recall that $\alpha_d \in \{0,1\}^n$ is viewed as $\alpha_d : H^m \to \{0,1\}$, whereas $H^{m'}1^{m-m'} \equiv H^{m'} \equiv [n]$ and $H^m \setminus H^{m'}1^{m-m'} \equiv [k] \setminus [n]$.

A single iteration. The core of the iterative proof is the interactive protocol that is performed in each iteration. This protocol is based on the relation between subsequent α_i's, which is based on the structure of the circuit. Specifically, recall that the i^{th} iteration reduces a claim regarding $\widehat{\alpha}_{i-1}$ to a claim regarding $\widehat{\alpha}_i$, where these polynomials encode the values of neighboring layers in the circuit. To describe the relation between such neighboring layers, we consider the adjacency relation of the circuit. Actually, we let $\phi_i : H^m \times H^m \times H^m \to \{0,1\}$ represent the feeding relation between the i^{th} and $(i-1)^{\text{st}}$ layer of the circuit such that $\phi_i(\overline{u},\overline{v},\overline{w}) = 1$ if and only if the gate in location $\overline{u} \in H^m$ in layer $i-1$ is fed by the gates in locations $\overline{v}$ and $\overline{w}$ in layer i. Hence, for every $\overline{u} \in H^m$, we have

$$\alpha_{i-1}(\overline{u}) = \sum_{\overline{v},\overline{w} \in H^m} \phi_i(\overline{u},\overline{v},\overline{w}) \cdot (1 - \alpha_i(\overline{v}) \cdot \alpha_i(\overline{w})). \tag{3.2}$$

Recall that, by Eq. (3.1), we have $\widehat{\alpha}_{i-1}(\overline{z}) = \sum_{\overline{u} \in H^m} \mathtt{EQ}(\overline{z},\overline{u}) \cdot \alpha_{i-1}(\overline{u})$. Combining this with Eq. (3.2), while letting $\widehat{\phi}_i : \mathcal{F}^{3m} \to \mathcal{F}$ denote a (predetermined) low degree polynomial that extends ϕ_i, we have

$$\widehat{\alpha}_{i-1}(\overline{z}) = \sum_{\overline{u},\overline{v},\overline{w} \in H^m} \mathtt{EQ}(\overline{z},\overline{u}) \cdot \widehat{\phi}_i(\overline{u},\overline{v},\overline{w}) \cdot (1 - \widehat{\alpha}_i(\overline{v}) \cdot \widehat{\alpha}_i(\overline{w})). \tag{3.3}$$

For any fixed point $\overline{r}_{i-1} \in \mathcal{F}^m$, the expression at the r.h.s of Eq. (3.3) can be written as

$$\sum_{\overline{u},\overline{v},\overline{w} \in H^m} p_{\overline{r}_{i-1}}(\overline{u},\overline{v},\overline{w}),$$

where $p_{\overline{r}_{i-1}}(\overline{z}',\overline{z}'',\overline{z}''') \stackrel{\text{def}}{=} \mathtt{EQ}(\overline{r}_{i-1},\overline{z}') \cdot \widehat{\phi}_i(\overline{z}',\overline{z}'',\overline{z}''') \cdot (1 - \widehat{\alpha}_i(\overline{z}'') \cdot \widehat{\alpha}_i(\overline{z}'''))$ is a low degree polynomial. Applying the Sum-Check protocol to Eq. (3.3) allows to reduce a claim regarding the value of $\widehat{\alpha}_{i-1}$ at a specific point $\overline{r}_{i-1} \in \mathcal{F}^m$ to a claim regarding the value of the polynomial $p_{\overline{r}_{i-1}}$ at a random point $(\overline{r}',\overline{r}'',\overline{r}''')$ in F^{3m} (for details see Section 3.4.1). This is not quite what we aimed for, but it is close enough, since the verifier can easily evaluate EQ at any point. However, there are two problems:

1. The verifier needs to be able to evaluate $\widehat{\phi}_i$ in $\widetilde{O}(n)$-time. The definition of $\widehat{\phi}_i$ (as a low degree polynomial that extends ϕ_i) may imply that evaluating $\widehat{\phi}_i$ reduces to $|H|^{3m} = k^3$ evaluations of ϕ_i, but $k = \text{poly}(n)$ which may be much larger than n (let alone that $k^3 \geq n^3$). Hence, evaluating $\widehat{\phi}_i$ in $\widetilde{O}(n)$-time is not obvious.

2. After evaluating EQ and $\widehat{\phi}_i$, the verifier is left with a residual claim that refers to two evaluations of $\widehat{\alpha}_i$ rather than to a single one.

The second problem is handled by augmenting the protocol such that the prover provides the value of $\widehat{\alpha}_i$ at both points (i.e., at $\overline{r}''$ and $\overline{r}'''$) as well as on the line that connects these points, and is asked to prove the correctness of the value at a random point on this line. That is, if the prover claims that $\widehat{\alpha}_i(\overline{r}'') = v''$ and $\widehat{\alpha}_i(\overline{r}''') = v'''$, then it also provides a low degree univariate polynomial p' such that $p'(z) = \widehat{\alpha}_i((1-z) \cdot \overline{r}'' + z \cdot \overline{r}''')$, which satisfies that $p'(0) = v''$ and $p'(1) = v'''$, and the verifier selects a random $r \in \mathcal{F}$ and sends it to the prover. The claim to be proved in the next iteration is that the value of $\widehat{\alpha}_i$ at $\overline{r}_i = (1-r) \cdot \overline{r}'' + r \cdot \overline{r}'''$ equals $v_i = p'(r)$. Hence, the full protocol that is run in iteration i is as follows.

Construction 3.2 (reducing a claim about $\widehat{\alpha}_{i-1}$ to a claim about $\widehat{\alpha}_i$): *For known $\overline{r}_{i-1} \in \mathcal{F}^m$ and $v_{i-1} \in \mathcal{F}$, the* entry claim *is $\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = v_{i-1}$. The parties proceed as follows.*

1. *Applying the Sum-Check protocol to the entry claim, when expanded according to Eq. (3.3), determines $\overline{r}', \overline{r}'', \overline{r}''' \in \mathcal{F}^m$ and a value $v \in \mathcal{F}$ such that the residual claim for verification is*

$$\mathtt{EQ}(\overline{r}_{i-1}, \overline{r}') \cdot \widehat{\phi}_i(\overline{r}', \overline{r}'', \overline{r}''') \cdot (1 - \widehat{\alpha}_i(\overline{r}'') \cdot \widehat{\alpha}_i(\overline{r}''')) = v. \tag{3.4}$$

   (For details see Section 3.4.1.)

2. *The prover sends a univariate polynomial p' of degree m such that $p'(z) = \widehat{\alpha}_i((1-z) \cdot \overline{r}'' + z \cdot \overline{r}''')$.*

   (Note that the degree of $\widehat{\alpha}_i$ equals the degree of $\mathtt{EQ}$ in the first block of m variables, which in turn equals m.)[5]

3. *The verifier checks whether v equals*

$$\mathtt{EQ}(\overline{r}_{i-1}, \overline{r}') \cdot \widehat{\phi}_i(\overline{r}', \overline{r}'', \overline{r}''') \cdot (1 - p'(0) \cdot p'(1)), \tag{3.5}$$

   and continues only if equality holds (otherwise it rejects).

   Note that this requires evaluating $\mathtt{EQ}$ *and $\widehat{\phi}_i$.*

4. *The verifier selects a random $r \in \mathcal{F}$, and sends it to the prover. Both parties set $\overline{r}_i = (1-r) \cdot \overline{r}'' + r \cdot \overline{r}'''$ and $v_i = p'(r)$.*

The exit claim *is $\widehat{\alpha}_i(\overline{r}_i) = v_i$.*

The prescribed prover strategy as defined in Construction 3.2 can be implemented in poly(n)-time. Assuming that $\widehat{\phi}_i$ can be evaluated in poly($\log n$)-time, the verifier's strategy in Construction 3.2 can be implemented in poly($\log n$)-time, provided that $\log |\mathcal{F}| \leq \text{poly}(\log n)$ (whereas we will use $|\mathcal{F}| = \text{poly}(\log n) \cdot d(n) \leq \text{poly}(n)$). Recall that after the last iteration, the resulting claim is directly checked by the verifier in time $\widetilde{O}(n)$. Typically (i.e., when $d(n) \ll n$), this final verification dominates the running time of the verifier.

One can readily verify that if the entry claim is correct, then the exit claim is correct, whereas if the entry claim is false, then with probability at least $1 - O((D+m)/|\mathcal{F}|)$ the exit claim is false, where D is the degree of $\widehat{\phi}_i$. (Indeed, it is tempting to think that $D = O(m)$, since this would have been the case if $\widehat{\phi}_i$ is selected to be the low degree extension of ϕ_i.) Recall that the soundness error of the entire protocol is upper-bounded by the probability that there exists an iteration in which the entry claim is false but the exist claim is true. Hence, the total soundness error is $O(d(n) \cdot D/|\mathcal{F}|)$.

We are still left with the problem of evaluating $\widehat{\phi}_i$ in poly($\log n$)-time. In Section 3.3, we show that (1) it suffices to solve this problem for low degree polynomials that extends the functions ϕ_i's that describe "highly uniform" circuits, and (2) that, in that case, suitable polynomials of degree $D = \text{poly}(\log n)$ can be used to extend the ϕ_i's.

[5] In continuation to Footnote 2, we mention that in the case of general H, the degree equals $|H'| \cdot m$, where $H' = (H - H) \setminus \{0\}$.

3.3 Evaluating the polynomial $\widehat{\phi_i}$

The general problem that we face is evaluating a low degree polynomial that extends a function $\phi : H^m \to \{0,1\}$ that is computable in $O(\log n)$ space, where $|H|^m = \text{poly}(n)$. (In our setting we are interested in ϕ_i's, which range over H^{3m}, which can be treated as slices of a single $\phi : H^{4m} \to \{0,1\}$ such that $\phi_i(\overline{z}) = \phi(i,\overline{z})$, where $i \in [m]$ is viewed as an element of H^m.)[6] This problem is solved in [27] by using an auxiliary multi-round interactive protocol, but here we bypass it by making several observations.

Recall that the function ϕ that we deal with here describes the structure of arbitrary log-space uniform circuits. The first observation is that we may use instead a function that describes the structure of a highly-uniform *universal circuit*. For starters, consider a universal circuit U_n that, given a description of a poly(n)-sized circuit $C_n : \{0,1\}^n \to \{0,1\}$ of depth $d = d(n)$ and an input $x \in \{0,1\}^n$, computes $C_n(x)$; that is, $U_n(\langle C_n \rangle, x) = C_n(x)$. By using a suitable description, we may get a highly uniform circuit U_n of poly(n)-size and depth $O(\log n) \cdot d(n)$. The idea is applying the main module to U_n rather than to C_n.

The problem with the foregoing idea is that after the last iteration (i.e., the last application of Construction 3.2), the verifier needs to evaluate the low degree extension of the input to the circuit, whereas in our application the input to U_n consists of both $\langle C_n \rangle$ and x, which means that the verifier needs to compute $\langle C_n \rangle$. By the log-space uniformity condition, this can be done in poly(n)-time, but we wish the verifier to perform this step in $\widetilde{O}(n)$-time. So, instead, we consider a *modified "universal" circuit* U'_n, which is tailored to the log-space algorithm A that constructs the circuit family $\{C_n\}$. On input $x \in \{0,1\}^n$, the circuit U'_n first constructs C_n (see details below), and then computes $U_n(\langle C_n \rangle, x)$. Now, applying the main module to U'_n rather than to U_n, the verifier only needs to evaluate a low degree extension of x (as in the case that we use C_n itself).

Turning to the construction of C_n by U'_n, following [27], we use the observation that log-space uniform circuits (here $\{C_n\}_{n\in\mathbb{N}}$) can be constructed by an *highly uniform* $\mathcal{NC}^2$ circuit. This observation relies on the standard construction of $\mathcal{NC}^2$ circuits for log-space computation, a construction that is based on constructing the digraph of instantaneous configurations of the computation of a log-space machine (on a fixed input) and raising it to a power larger than its dimension (by repeated squaring). For details see Section 3.4.2. Recalling that U_n is also highly uniform (see Section 3.4.2 for details) and combining these two constructions, we obtain a highly uniform construction of U'_n. (The circuit U'_n has size poly(n) and depth $O(\log n)^2 + O(\log n) \cdot d(n) \leq O(\log n) \cdot d(n)$, since $d(n) \geq \log n$ w.l.o.g.)

So far, we avoided specifying what we meant by a highly uniform circuit. A notion that suffices for our purposes is presented now. Considering circuits of size $s(n) = \text{poly}(n)$ and letting $\ell = 3\log s(n)$, we consider the predicate $\phi : \{0,1\}^\ell \to \{0,1\}$ that describes the adjacency relation of the circuit. We say that the circuit is highly uniform if ϕ can be computed by poly(ℓ)-sized formula that can be constructed in poly(ℓ)-time. The reader may verify that the $\mathcal{NC}$ circuits that we used above (for constructing the log-space uniform circuits C_n) are highly uniform; in fact, they are even "more uniform" than that. Likewise for the circuit U_n, and consequently for U'_n. Hence, the predicate $\phi : \{0,1\}^\ell \to \{0,1\}$ that describes the adjacency relation of U'_n can be computed by a poly$(\log n)$-size formula that can be constructed in poly$(\log n)$-time. Using $H = \{0,1\}$ and

[6] Indeed, using $\phi : H^{3m+\log m} \to \{0,1\}$ and associating $[m]$ with $H^{\log m}$ would suffice. When $|H| = \Omega(\log n)$, as in Footnote 2, we have $m = O(\log n)/\log\log n < |H|$, and so using $\phi : H^{3m+1} \to \{0,1\}$ (and assuming $[m] \subseteq H$) suffices.

recalling that $m = \log n$, the following result implies that there exists a low degree polynomial over $\mathcal{F}^m$ that agrees with ϕ on H^m and can be evaluated in $\text{poly}(\log n)$-time.

Proposition 3.3 (low degree polynomials that agree with small Boolean formulae):[7] *Suppose that $\phi : H^m \to \{0,1\}$, where $H = \{0,1\}$, can be computed by formula of size $\text{poly}(m)$ that can be constructed in $\text{poly}(m)$-time. Then, there exists a polynomial $\widehat{\Phi} : \mathcal{F}^m \to \mathcal{F}$ of degree $\text{poly}(m)$ that agrees with ϕ on H^m and can be constructed in $\text{poly}(m)$-time.*

We stress that the polynomial $\widehat{\Phi} : \mathcal{F}^m \to \mathcal{F}$ is *not* the (canonical) low degree extension of ϕ; such a low degree extension has degree m and exists regardless of the size of the formula computing ϕ. In contrast, the polynomial $\widehat{\Phi}$ is derived from the formula computing ϕ, and its degree bound is derived from the depth of ϕ. (As stated above, we apply Proposition 3.3 to the function $\phi : H^{4m} \to \{0,1\}$ that describes the relation between gates in the circuit U'_n, and obtain $\widehat{\phi_i}$'s such that $\widehat{\phi_i}(\overline{z}) = \widehat{\Phi}(i, \overline{z})$, which we use in Construction 3.2.)[8]

Proof: Considering the $\text{poly}(m)$-size formula that computes $\phi : \{0,1\}^m \to \{0,1\}$, construct an arithmetic circuit $\widehat{\Phi}$ (which is actually a formula) that computes a polynomial that agrees with ϕ on H^m by mimicking the Boolean formula ϕ. Specifically, the NAND of the Boolean subformulae ψ_1 and ψ_2 is replaced by a gate that computes $1 - \widehat{\psi}_1 \cdot \widehat{\psi}_2$, where $\widehat{\psi}_1$ and $\widehat{\psi}_2$ are the corresponding arithmetic subformulae. (Indeed, we may assume, w.l.o.g., that ϕ uses only NAND-gates, and that its depth is logarithmic in its size.)[9] Note that the degree of the resulting polynomial $\widehat{\Phi}$ is upper-bounded by $\text{poly}(m)$, since its degree is exponential in the depth of ϕ, which in turn is logarithmic in the size of ϕ. ■

Wrapping-up. As stated above, we modify Construction 3.2 by replacing the polynomials $\widehat{\phi_i}$ by the polynomial $\widehat{\Phi}$ (i.e., replacing $\widehat{\phi_i}(\cdot)$ by $\widehat{\Phi}(i, \cdot)$, where $\widehat{\phi_i} : \mathcal{F}^{4m} \to \mathcal{F}$ and $i \in [m]$ is viewed as an element of $H^m \subset \mathcal{F}^m$). Doing so provides an efficient implementation of Step 3 of Construction 3.2, whereas the soundness error of the entire protocol is upper-bounded by $d(n) \cdot \text{poly}(m)/|\mathcal{F}|)$, where $\text{poly}(m)$ is the degree bound asserted in Proposition 3.3. Hence, we need to guarantee that $\text{poly}(m)/|\mathcal{F}| = o(1/d(n))$, which implies that $|\mathcal{F}| = \text{poly}(m) \cdot d(n)$ suffices. This completes the proof of Theorem 3.1, modulo some details provided in Section 3.4.

3.4 Details

In this section we provide some tedious details, which some readers may find quite straightforward.

3.4.1 Applying the Sum-Check protocol to Eq. (3.3)

Here we detail how the Sum-Check protocol is used to reduce the claim regarding the value of $\widehat{\alpha}_i$ at a given point $\overline{r}_{i-1}$ to a claim regarding $\widehat{\alpha}_{i-1}$. Recall that, towards this goal, we have defined a

[7] In continuation to Footnote 2, we mention that [21, Thm. 3] refers to H of size $\Theta(\log n)$, and its proof is more complex (due to the need to move from H to $\{0,1\}^{\log |H|}$).

[8] Recall that $\phi(i, \overline{z}) = \phi_i(\overline{z})$, where ϕ_i describes the connections between the $i-1^{\text{st}}$ and the i^{th} layers of U'_n (as in Section 3.2).

[9] Note that the transformation to this form can be performed in polynomial time.

polynomial $p_{\overline{r}_{i-1}} : \mathcal{F}^{3m} \to \mathcal{F}$ such that

$$p_{\overline{r}_{i-1}}(\overline{z}', \overline{z}'', \overline{z}''') = \mathtt{EQ}(\overline{r}_{i-1}, \overline{z}') \cdot \widehat{\phi}_i(\overline{z}', \overline{z}'', \overline{z}') \cdot (1 - \widehat{\alpha}_i(\overline{z}'') \cdot \widehat{\alpha}_i(\overline{z}''')). \tag{3.6}$$

Expanding $\widehat{\alpha}_{i-1}$ according to Eq. (3.3) and using Eq. (3.6), we have

$$\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = \sum_{\overline{u},\overline{v},\overline{w} \in H^m} p_{\overline{r}_{i-1}}(\overline{u}, \overline{v}, \overline{w}). \tag{3.7}$$

Recall that the claim that we wish to verify is $\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = v_{i-1}$; that is, the claim is that the r.h.s of Eq. (3.7) equals v_{i-1}. Using $3m$ rounds of interaction, the Sum-Check protocol reduces the latter claim to a claim regarding the value of $p_{\overline{r}_{i-1}}$ at a random point $(\overline{r}', \overline{r}'', \overline{r}''') \in \mathcal{F}^{3m}$. This is captured by Step 1 of Construction 3.2.

3.4.2 The highly uniform circuits in use

Here we detail the construction of the universal circuit U'_n, which combines a construction of $\mathcal{NC}^2$ circuits for constructing log-space uniform circuits and a construction of the universal circuit U_n. The point that we wish to establish here is that both these constructions are highly uniform (in the sense that the adjacency relation function describing each of these two poly(n)-sized circuit can be computed by a poly$(\log n)$-size formula that can be computed in poly$(\log n)$-time).

We start with the highly uniform construction of $\mathcal{NC}^2$ circuits for constructing log-space uniform circuits, denoted $\{C_n\}_{n\in\mathbb{N}}$. This construction relies on the standard construction of $\mathcal{NC}^2$ circuits for log-space computations, a construction that is based on constructing the digraph of instantaneous configurations of the computation of a log-space machine (on a fixed input) and raising it to a power larger than its dimension (by repeated squaring). In our case, the input (i.e., 1^n) is actually used only to determine the space bound, and so we can ignore it. However, we should note that the said log-space computation produces many output bits (which describe the circuit C_n, rather than a single decision bit). Without loss of generality, we can incorporate the index of the next output bit to be produced in the instantaneous configuration (since the log-space machine can maintain a corresponding counter).

Constructing the digraph of instantaneous configurations. Observe that the digraph that represents the consecutive configurations in the foregoing log-space computation can be constructed very easily (i.e., by a highly uniform $\mathcal{NC}^0$ circuit); that is, the pair $(\gamma, \gamma') \in \{0,1\}^{2 \cdot O(\log n)}$ is an edge in that digraph if and only if the log-space machine moves from the instantaneous configuration represented by γ to the configuration represented by γ' in a single step. Note that we actually consider an $\mathcal{NC}^0$ circuit that has no input (i.e., the fixed input 1^n is hardwired in this circuit) and always produces this digraph as its output. Thus, this "circuit" consists of a sequence of output gates, whose value is hardwired such that the output gate that corresponds to the entry (γ, γ') in the adjacency matrix (representing this digraph) is 1 if and only if the pair (γ, γ') satisfies a very simple relation that is only slightly more complex than testing equality.[10] It follows that the value

[10] Specifically, recall that γ represents the contents of the work-space as well as the finite state of the machine and the location of its head (on the work-tape). The bits of γ' should equal the corresponding bits of γ, except for the bit at the head's location and the bits encoding the finite state that change according to a finite function. (The location of the head on the work-tape is indicated by the corresponding location of a special symbol in γ.) Hence, the value of the entry (γ, γ') can be determined as a conjunction of $O(\log n)$ conditions, where each of these conditions refers to a constant number of bits.

of the entry (γ, γ') in the matrix can be determined in $\text{poly}(\log n)$-time as a function of γ and γ'. Hence, the circuit that produces this matrix, which consists merely of feeding the constants 0 and 1 to the $\text{poly}(n)$-many output bits, is highly uniform; that is, its adjacency relation can be expressed a formula of size $O(\log n)$ that can be constructed in time $\text{poly}(\log n)$.

This yields a (highly uniform) $\text{poly}(n)$-by-$\text{poly}(n)$ matrix M that when raised to the power $\text{poly}(n)$ yields the desired bits in the description of C_n; that is, the value of the i^{th} output bit (i.e., the i^{th} bit in the description of C_n) equals the value of the entry $(\mathtt{s}_i, \mathtt{f}_i)$ in the resulting matrix (i.e., of $M^{\text{poly}(n)}$), where $\mathtt{s}_i$ denotes the initial configuration that is used for producing the i^{th} output bit and $\mathtt{f}_i$ denotes the final configuration that actually produces this bit. As we show next, there exists a highly uniform $\mathcal{NC}^2$ circuit for raising a given $\text{poly}(n)$-by-$\text{poly}(n)$ matrix to the power $\text{poly}(n)$ (by repeated squaring), since the inner product of $\text{poly}(n)$-bit long vectors has a highly uniform $\mathcal{NC}^1$ circuit.

Constructing circuits for matrix multiplication. We now detail the highly uniform $\mathcal{NC}^1$ circuit used to compute matrix multiplication. Let A and B be n-by-n matrices, then the $(i,j)^{\text{th}}$ bit in their product is given by $\sum_{k\in[n]} A_{i,k}B_{k,j}$. Hence, we first use n^3 gates such that the gate indexed by (i,j,k) computes $A_{i,k}B_{k,j}$, which means that this gate is fed by inputs with indices $(1,i,k)$ and $(2,k,j)$, where $(1,i,k)$ (resp., $(2,k,j)$) corresponds to the $(i,k)^{\text{th}}$ (resp., $(k,j)^{\text{th}}$) bit in the input matrix A (resp., B), and then we use n^2 trees of $n-1$ addition-gates for computing each n-way sum. Specifically, the sum that corresponds to index (i,j) is computed by gates that are indexed by (i,j,α), where $\alpha \in \{0,1\}^{\leq \log n}$, such that the gates with index (i,j,α) with $\alpha \in \{0,1\}^{\log n} \equiv [n]$ are as detailed above, and the gate indexed by (i,j,α) with $|\alpha| < \log n$ is fed by the gates indexed $(i,j,\alpha 0)$ and $(i,j,\alpha 1)$. Hence, the description of the adjacency relation of this circuit is very simple; for example, for the latter gates (i.e., a gate indexed (i,j,α) that is fed by the gates indexed $(i,j,\alpha 0)$ and $(i,j,\alpha 1)$), we can use the adjacency relation

$$\phi((i,j,\alpha),(i',j',\alpha'),(i'',j'',\alpha'')) = \mathtt{EQ}(ij,i'j') \wedge \mathtt{EQ}(ij,i''j'') \wedge \mathtt{EQ}(\alpha 0,\alpha') \wedge \mathtt{EQ}(\alpha 1,\alpha'').$$

Constructing the universal circuit U_n. Recall that, for fixed depth $d = d(n)$ and size $s = s(n)$, the circuit U_n is given a description of a circuit C_n (of depth d and size s) and a string $x \in \{0,1\}^n$, and is supposed to output $C_n(x)$. The circuit U_n emulates the computation of C_n in a layer-by-layer manner. Specifically, U_n computes the value of the j^{th} gate in layer i in the emulated computation of C_n (on x), denoted $v_{i,j}$, as a function of the description of the circuit C_n and the two adequate values of gates in layer $i+1$. In particular, $v_{d,j}$ is the j^{th} bit of x if $j \leq n$ and equals 0 otherwise, and for every $i \in [d]$ and $j \in [s]$,

$$v_{i-1,j} = \bigvee_{k',k''\in[s]} (c_{(i-1,j),(i,k'),(i,k'')} \wedge \mathtt{NAND}(v_{i,k'}, v_{i,k''})),$$

where $c_{(i-1,j),(i,k'),(i,k'')}$ is a bit in the description of C_n that indicates whether or not the $(i-1,j)^{\text{th}}$ gate is fed by gates indexed the (i,k') and (i,k''). Again, the description of the adjacency relation of this circuit is very simple.

Chapter 4

Overview of the doubly-efficient interactive proof systems of RRR

In this chapter, we provide an overview of the doubly-efficient interactive proof systems of Reingold, Rothblum, and Rothblum [42]. Recall that by their result (already stated as Theorem 1.2), any set that is decidable in polynomial-time by an algorithm of space complexity $s(n) \leq n^{0.499}$, has a constant-round interactive proof system in which the prover runs polynomial time and the verifier runs in time $\widetilde{O}(n)$.

Theorem 4.1 (doubly-efficient interactive proof systems for $\mathcal{SC}$): *Suppose that S is decidable by an algorithm that runs in polynomial time and has space complexity s. Then, for any constant $\delta > 0$, the set S has an interactive proof system with $\exp(\widetilde{O}(1/\delta))$ rounds in which the prescribed prover strategy runs in polynomial-time while the verifier runs in $(\widetilde{O}(n) + s(n)^2 \cdot n^{\delta})$-time. Furthermore, the proof system is of the public-coin type and the total communication is $O(s(n)^2 \cdot n^{\delta})$.*

In the original version of [42], the verification time was bounded by $(\widetilde{O}(n) + \mathrm{poly}(s(n)) \cdot n^{\delta})$, for an unspecified constant poly. The current statement was confirmed by the authors of [42] (private communication, May 2017).

The proof of Theorem 4.1 spans 70 pages in [42], and in this chapter we merely attempt to present an overview of it. The proof is pivoted at the notion of batch verification, which is a procedure that allows for the verification of numerous claims at a significantly lower cost than required for verifying each of the claims separately. Specifically, we use batch verification for a certain class of problems (see Theorem 4.2), which extends the class of "NP-problems with unique solutions" (i.e., $\mathcal{UP}$). In fact, batch verification for $\mathcal{UP}$, which is of independent interest (see Theorem 4.4), serves as a warm-up towards Theorem 4.2.

4.1 The high level structure

Theorem 4.1 is proved by considering the instantaneous configurations of the corresponding space-bounded algorithm. The interactive proof system will refer to claims of the form *on input x, machine M moves from configuration γ to configuration γ' in t steps*. The initial claim refers to the case that $\gamma \in \{0,1\}^{s(|x|)}$ and $\gamma' \in \{0,1\}^{s(|x|)}$ are the initial and accepting configurations of M, respectively, and to $t \leq \mathrm{poly}(|x|)$.

The core of the proof of Theorem 4.1 is reducing the construction of an interactive proof system for a claim regarding t-step computations (of a space-bounded machine) to the construction of an interactive proof system for a claim regarding t/k-step computations (of such a machine), where k is a parameter (e.g., $k = n^\delta$ or so). This is done by having the prover send $\gamma_1, ..., \gamma_{k-1} \in \{0,1\}^s$ and prove that, *for every $i \in [k]$, on input x, machine M moves from configuration γ_{i-1} to configuration γ_i in t/k steps, where $\gamma_0 = \gamma$ and $\gamma_k = \gamma'$.*

Hence, we have reduced proving a single claim regarding a t-long computation of M to k claims regarding t/k-long computations of M. But have we gained anything? This depends on whether we can perform k verifications (regarding t/k-long computations) at a cost that is lower than performing each of these verifications separately. In other words, we seek a *batch verification* procedure in which verifying k claims is cheaper than verifying each of the claims separately.

To see that batch verification is not a pipe dream, consider the task of proving k claims that are each in $\mathcal{PSPACE}$. That is, for a fixed set $S \in \mathcal{PSPACE}$, given $x_1, ..., x_k \in \{0,1\}^n$, we wish to prove that $(\forall i \in [k])\ x_i \in S$. Then, the complexity of verifying the latter claim by employing the proof system of [35, 44] is only moderately larger than the complexity of verifying membership in S using this generic construction. Specifically, suppose that S has space complexity $s(n)$, and recall that verification via the generic system of [35, 44] takes time that is proportional to $s(n)^4$. Then, the time complexity of batch verification of k claims (each of length n) is proportional to $(s(n) + \log k)^4$, which is only moderately larger than $s(n)^4$.

The following result provides a batch verification procedure (or rather an interactive proof for batch verification) for any set that has an interactive proof system of a certain type, where the nature of this type will remain unspecified momentarily.[1] We shall denote the length of a single instance by n, and the number of instances by $k(n)$. For a set S and $k : \mathbb{N} \to \mathbb{N}$, we let

$$S^k = \bigcup_{n \in \mathbb{N}} \{(x_1, ..., x_{k(n)}) \in \{0,1\}^{k(n) \cdot n} : (\forall i \in [k(n)])\ x_i \in S\}. \tag{4.1}$$

Theorem 4.2 (batch verification for certain interactive proof systems): *For $c : \mathbb{N} \to \mathbb{N}$ and a constant $\ell \in \mathbb{N}$, suppose that S has an ℓ-round interactive proof system of a particular type $\mathcal{T}$ with total communication c, poly-logarithmic verification time, and polynomial-time prover strategy. Then, for every constant $\alpha > 0$ and $k : \mathbb{N} \to \mathbb{N}$, the set S^k has an $O(\ell/\alpha)$-round interactive proof system of type $\mathcal{T}$ with total communication $\widetilde{O}(k^\alpha \cdot c + k)$, poly-logarithmic verification time, and polynomial-time prover strategy.*

Using Theorem 4.2, the construction of the proof system asserted in Theorem 4.1 can be recursively presented as follows.

Construction 4.3 (recursive description of the interactive proof system): *Let $n, s, t, k \in \mathbb{N}$, where $s = s(n)$ and $k < t$, and fixed M and $x \in \{0,1\}^n$. To prove that, on input x, machine M moves in t steps from configuration $\gamma_0 \in \{0,1\}^s$ to configuration $\gamma_k \in \{0,1\}^s$, the parties proceed as follows.*

1. *The prover sends $\gamma_1, ..., \gamma_{k-1} \in \{0,1\}^s$ such that, for every $i \in [k]$, it holds that, on input x, machine M moves in t/k steps from configuration γ_{i-1} to configuration γ_i.*

   In order to support the hypothesis made in the next step, each γ_i is sent using an error correcting code. In addition, we also assume that x is presented by such a code.

[1] Readers who are too curious to wait may note that the said type is restricted to public coin systems that are "unambiguous" (akin to the notion of unique valid proofs in NP-proof systems) and support local checking of the prover-messages (akin to PCPs (or rather PCPPs)).

2. *The parties invoke an interactive proof system in order to verify the foregoing claim regarding the γ_i's, where the i^{th} claim refers to moving from γ_{i-1} to γ_i* (in t/k steps, on input x).[2] *This interactive proof system is obtained by applying batch verification for k claims* (via Theorem 4.2) *to the interactive proof system that refers to t/k-step computations, where the latter is obtained by a recursive call. Recall that Theorem 4.2 requires that the latter system be of type $\mathcal{T}$.*

   Hence, at the bottom of the recursion we have communication complexity $\mathrm{poly}(s)$*, at its j^{th} level* (from the bottom) *we have communication complexity $O(k^{j\alpha} \cdot \mathrm{poly}(s) + k^{(j-1)\alpha+1})$, where α is the constant in Theorem 4.2 and the recursion has $\log_k t$ levels. Using $k^{(\log_k t)\cdot\alpha} = t^\alpha$, this yields communication complexity $O(t^\alpha \cdot (\mathrm{poly}(s) + k))$ at the top level of the recursion, whereas verification time is poly-logarithmic, proving time is polynomial, and the number of rounds is $O(1/\alpha)^{\log_k t}$.*

In order to support the use of the derived system in subsequent recursive calls, we will show that the derived system is indeed of type $\mathcal{T}$.

Theorem 4.1 follows by combining Construction 4.3 with Theorem 4.2, while setting $\alpha = \delta/O(1)$ and $k = n^\delta$. (Actually, Construction 4.3 is invoked after letting the verifier encode x as well as the initial and accepting configurations under an error correcting code.)

In light of Construction 4.3, we focus on the proof of Theorem 4.2. We start with the simple case in which the set has an NP-proof system, which holds at the very bottom of the recursion but not at higher levels.

4.2 Warm-up: Batch verification for NP

We start by introducing two of the restrictions that make up the foregoing type $\mathcal{T}$ of proof systems. The first restriction asserts that a convincing prover strategy is essentially unique; that is, if the prover wishes to convince the verifier of the fact that $x \in S$, then it has no choice but to follow a unique (deterministic) strategy. In the case of NP-proof systems, this means that the NP-witness is unique; that is, $S \in \mathcal{UP}$.

The second restriction is that verification can be performed in poly-logarithmic time provided that the input is presented in encoded form, under a suitable error correcting code. This condition is satisfied by a variety of PCPP systems, starting from the ones based on Reed-Muller encoding (cf., [5, 17], as interpreted by [8, 14]). Readers that are unfamiliar with PCPPs (i.e., PCPs of Proximity), may consider standard PCP systems and verification that is performed in almost-linear time. Actually, for simplicity we shall just do that, while focusing on the communication complexity of the derived interactive proof system for batch verification.

Hence, our starting point is a set $S \in \mathcal{UP}$ and a corresponding PCP system (of constant query complexity). Actually, we shall assume that the PCP system uses input-oblivious queries; that is, its queries are determined non-adaptively based solely on its internal coin tosses, and only the final decision depends on its actually input.[3] Such PCP systems are known and can be derived from any PCP system.

[2] Hence, the input to the i^{th} claim is $x_i = (x, \gamma_{i-1}, \gamma_i)$.

[3] Hence, these PCP systems are not truly input-oblivious in the sense studied in [23].

Theorem 4.4 (batch verification for $\mathcal{UP}$): *For $c : \mathbb{N} \to \mathbb{N}$ and a constant $\ell \in \mathbb{N}$, suppose that $S \in \mathcal{UP}$ has a PCP system that uses input-oblivious queries and proofs of length c. Then, for every constant $\alpha > 0$ and $k : \mathbb{N} \to \mathbb{N}$, the set S^k has an $O(1/\alpha)$-round interactive proof system with total communication $O(k^\alpha \cdot c + k)$ such that prover strategy can be implemented in polynomial-time when given the NP-witness to an input in S^k.*

(We mention that subsequent work [43] achieved such batch verification using total communication $O(k^\alpha \cdot \mathrm{poly}(m))$, where m denotes the length of the NP-witness.)

Proof Sketch: For a parameter $d \leq k$ to be determined (e.g., $d = \sqrt{k}$), consider a parity-check function $F : \{0,1\}^k \to \{0,1\}^{O(d \log k)}$ such that for every $y \in \{0,1\}^k$ and $v \in \{0,1\}^{O(d \log k)}$ the Hamming ball of radius d centered at y contains at most a single pre-image of v under F. (Indeed, in the case $d = 1$ we can let $F(y)$ be the XOR of the bits of y, and in the general case we can use the parity check matrix of a Reed-Solomon code.)[4]

Construction 4.4.1 (basic construction): *On input $(x_1, ..., x_k) \in \{0,1\}^{k \cdot n}$, the proof system proceeds as follows.*

1. *Let w_i be the unique NP-witness associated with x_i, and $\pi_i \in \{0,1\}^{c(n)}$ be the PCP-oracle derived from it.* (We assume that the NP-witness is easy to extract from the PCP-oracle.)[5] *For each $j \in [c(n)]$, let $\pi_{i,j}$ denote the j^{th} bit of π_i. Then, for every $j \in [c(n)]$, the prover computes $v_j \leftarrow F(\pi_{1,j} \cdots \pi_{k,j})$, and sends $v_1, ..., v_{c(n)}$ to the verifier. Let us denote the values actually sent by $\widetilde{v}_1, ..., \widetilde{v}_{c(n)}$.*

   Pictorially, the prover forms a k-by-$c(n)$ Boolean matrix such that the i^{th} row equals π_i, and applies the parity check function to each column. These values form a very partial commitment of the prover to the values of the matrix; once these values are determined, each column is pseudo-determined in the sense that a valid revealing may either equal the original column or must differ from it on more than d coordinates.

2. *The verifier generates a sequence of queries, $j_1, ..., j_q \in [c(n)]$, for the PCP verifier, denoted V. It sends this sequence to the prover, who responds with the values of the corresponding columns. The verifier checks that* (i) *the values of these columns match the parity-check values, and that* (ii) *V would have accepted each of the inputs when given the corresponding answers. Specifically, suppose that the prover answered with $(\widetilde{\pi}_{i,j_{i'}})_{i \in [k], i' \in [q]}$. Then, the verifier performs the following two checks:*

   (a) *For every $i' \in [q]$, it checks whether $F(\widetilde{\pi}_{1,j_{i'}} \cdots \widetilde{\pi}_{k,j_{i'}}) = \widetilde{v}_{j_{i'}}$.*

   (b) *For each $i \in [k]$, it checks whether V would have accepted x_i when making the queries $j_1, ..., j_q$ and receiving the answers $\widetilde{\pi}_{i,j_1}, ..., \widetilde{\pi}_{i,j_q}$.*

[4] Suppose that F describes the operation of the parity check matrix of a linear code of distance D. Then, if $y' \neq y''$ are mapped by F to the same image, then $F(y' \oplus y'') = 0^{O(d \log k)}$, which means that $y' \oplus y''$ is in the code and so its Hamming weight must be at least D.

[5] Indeed, the NP-witness may appear as a prefix of the PCP-oracle. By the *PCP-oracle derived from an NP-witness* we mean the PCP-oracle that is outlined in the description of the PCP system, while recalling that all known PCP systems are described in this matter.

3. *The verifier selects uniformly a set R of $O(k/d)$ rows, sends their indices to the prover, who provides their contents. The verifier checks that* (i) *each of these rows equals the PCP-oracle derived from the unique NP-witness for the corresponding input, and that* (ii) *the values of these rows match the values of the columns provided in Step 2. Specifically, denoting the value of row i as sent in this step by r_i, the verifier performs the following two checks* (for each such $i \in R$)*:*

   (a) *Letting w'_i denote the purported NP-witness extracted from r_i, the verifier checks that w'_i is a valid NP-witness for x_i and that r_i is the PCP-oracle derived from w'_i.*

   (b) *For every $i' \in [q]$, it checks whether $r_{i,j_{i'}} = \widetilde{\pi}_{i,j_{i'}}$, where $r_i = r_{i,1} \cdots r_{i,c(n)}$*

The basic intuition is that Step 1 leaves the prover with the option of either cheating in Step 2 on more than d entries of one of the columns or providing the correct values for all columns. In the first case, the prover is likely to be caught cheating in Step 3, whereas in the second case it is likely that V will reject $x_i \notin S$ (when invoked in Step 2). Before presenting a more orderly analysis, let us consider the communication complexity of the foregoing system: In Step 1 the prover sends $O(d \log k) \cdot c(n)$ bits, in Step 2 it sends $q \cdot k$ bits, and in Step 3 it sends $O(k/d) \cdot c(n)$ bits. Picking $d = \sqrt{k}$ and recalling that $q = O(1)$, we get total communication of $\widetilde{O}(k^{1/2}) \cdot c(n) + O(k)$. We comment that the verification time is vastly dominated by Step 3a, whereas the prover's strategy can be implemented in polynomial-time (given the NP-witnesses).

Claim **4.4.2** (soundness of Construction 4.4.1): *Construction 4.4.1 is an interactive proof system for S^k.*

Proof: Turning to the more orderly analysis of the soundness of Construction 4.4.1, we assume without loss of generality that the values provided by the prover satisfy the conditions stated in Steps 2a and 3b (i.e., for every $i' \in [q]$ it holds that $F(\widetilde{\pi}_{1,j_{i'}} \cdots \widetilde{\pi}_{k,j_{i'}}) = \widetilde{v}_{j_{i'}}$ and $r_{i,j_{i'}} = \widetilde{\pi}_{i,j_{i'}}$ for every $i \in R$). We consider two cases when referring to the values $\widetilde{v}_1, ..., \widetilde{v}_{c(n)}$ provided by the prover in Step 1.

The first case is that, with probability at least half, over the choice of the query-sequence $(j_1, ..., j_q)$ in Step 2, the $\widetilde{\pi}_{i,j_{i'}}$'s provided by the prover satisfy $|\{i \in [k] : \widetilde{\pi}_{i,j_{i'}} \neq \pi_{i,j_{i'}}\}| > d$ for some $i' \in [q]$, where π_i denotes the PCP-oracle derived for $x_i \in S$ and is defined as the all-zero string in case $x_i \notin S$. In this case, for each query-sequence $(j_1, ..., j_q)$ that belongs to the majority, with high probability, the set of rows chosen in Step 3 contains a row i such that $r_{i,j_{i'}} = \widetilde{\pi}_{i,j_{i'}} \neq \pi_{i,j_{i'}}$ for some $i' \in [q]$, and so r_i does not satisfy the condition stated in Step 3a, which causes the verifier to reject. We stress that the uniqueness of NP-witnesses implies the uniqueness of the PCP-oracles derived from them, and so $r_i \neq \pi_i$ implies that the string extracted from r_i (so that r_i is derived from this string) is not a valid NP-witness for x_i.

Turning to the complimentary case, where with probability at least half (over the choice of $(j_1, ..., j_q)$ made in Step 2) it holds that $|\{i \in [k] : \widetilde{\pi}_{i,j_{i'}} \neq \pi_{i,j_{i'}}\}| \leq d$ for every $i' \in [q]$, we let $(\pi'_{i,j})_{i \in [k], j \in [c(n)]}$ denote the unique matrix such that for every $j \in [c(n)]$ the string $\pi'_{1,j} \cdots \pi'_{k,j}$ is the unique string that resides in the intersection of the Hamming ball of radius d centered at $\pi_{1,j} \cdots \pi_{k,j}$ and the pre-image of $\widetilde{v}_j$ under F. Hence, with probability at least half (over the choice of $(j_1, ..., j_q)$), for each $i \in [k]$, it holds that $\widetilde{\pi}_{i,j_{i'}} = \pi'_{i,j_{i'}}$ for every $i' \in [q]$, since $\widetilde{\pi}_{1,j_{i'}} \cdots \widetilde{\pi}_{k,j_{i'}}$ resides in the said Hamming ball and is in the pre-image of $\widetilde{v}_{j_{i'}}$ under F.[6] By the soundness of the PCP

[6] Actually, it suffices to establish a weaker statement asserting that, for each $i \in [k]$, it holds that $\Pr_{(j_1,...,j_q)}[(\forall i' \in [q])\, \widetilde{\pi}_{i,j_{i'}} = \pi'_{i,j_{i'}}\}] \geq 1/2$.

verifier (which is invoked in Step 2b), it follows that $x_i \in S$.

Hence, either the verifier rejects with probability at least 1/2 or all x_i's are in S. ■

Conclusion. By setting $d = \sqrt{k}$, we have established the claim of the theorem for any $\alpha > 1/2$. To establish the claim for smaller constant values of $\alpha > 0$, we use recursion. Specifically, we set $d = k^\alpha$, and employ Construction 4.4.1, except that in Step 3 we invoke Construction 4.4.1 on the $O(k/d)$ instances that correspond to the selected rows. Hence, we reduce batch verification for k instances to batch verification for $O(k/d)$ instances. However, the claim to be verified is not only that $x_i \in S$ for each selected row i (i.e., $i \in R$), but rather that there exists an NP-witness for x_i such that the PCP-oracle derived from it matches the values $\widetilde{\pi}_{i,j_1}, ..., \widetilde{\pi}_{i,j_q}$ received in Step 2 (i.e., $\widetilde{\pi}_{i,j_{i'}} = y_{i,j_{i'}}$ for every $i' \in [q]$, where $y_{i,j}$ is a variable representing the j^{th} bit of the oracle derived from the NP-witness for x_i).[7] After $1/\alpha$ recursive calls, we just use the straightforward verification that is used originally in Step 3. ■

4.3 Batch verification for unambiguous IP

The proof of Theorem 4.4 makes essential use of the hypothesis that each YES-instance has a unique proof, which in the context of $\mathcal{NP}$ means that it has a unique NP-witness. Seeking to extend Theorem 4.4 to sets that only have interactive proof systems, we seek an adequate notion of unique proving strategies, since an interactive strategy is the interactive analogue of a written proof (i.e., an NP-witness). Intuitively, we want to require that, at each round, there is at most one prover-message that may lead the verifier to accept. This condition must be phrased while accounting for the fact that the verifier strategy is probabilistic and consequently the phrase "leading the verifier to accept" should be given a probabilistic interpretation. Indeed, we require that, at each round, there is at most one prover-message that may lead the verifier to accept with probability that exceeds 1/2.

Definition 4.5 (unambiguous interactive proof systems): *Let (P,V) be an interactive proof system, where P is a deterministic prover strategy that satisfies the* (perfect) *completeness condition. The system (P,V) is called* unambiguous *if for every partial communication transcript* (of V with P) *that ends with a verifier message, there exists at most one prover-message such that, in the residual communication, the verifier accepts with probability exceeding* 1/2. *More formally:*

> *Let $\langle P', V\rangle_j(x)$ denote a random variable representing the distribution of the transcripts of the first j messages in an interaction between P' and V on common input x. Then, assuming that the verifier sends the first message, we require that for every i and every $(\alpha_1, \beta_1, ..., \alpha_{i-1}, \beta_{i-1}, \alpha_i, \beta_i)$ in the support of $\langle P, V\rangle_{2i}(x)$ and every $\widetilde{P}$, it holds that*
>
> $$\Pr\left[(\widetilde{P},V)(x)=1 \;\middle|\; \begin{array}{rcl} \langle \widetilde{P}, V\rangle_{2i-1}(x) & = & (\alpha_1, \beta_1, ..., \alpha_{i-1}, \beta_{i-1}, \alpha_i) \\ \langle \widetilde{P}, V\rangle_{2i}(x) & \neq & (\alpha_1, \beta_1, ..., \alpha_{i-1}, \beta_{i-1}, \alpha_i, \beta_i) \end{array}\right] \leq 1/2, \quad (4.2)$$
>
> *where $(\widetilde{P},V)(x)$ denotes the output of V after interacting with $\widetilde{P}$ on input x.*

[7] The corresponding NP-claim is that $(x_i, \widetilde{\pi}_{i,j_1}, ..., \widetilde{\pi}_{i,j_q}) \in S'$, which holds if there exists $y_i = (y_{i,1}, ..., y_{i,c(n)})$ such that y_i is the PCP-oracle derived from the NP-witness for $x_i \in S$ and $y_{i,j_{i'}} = \widetilde{\pi}_{i,j_{i'}}$ holds for every $i' \in [q]$.

That is, Eq. (4.2) asserts that any (i^{th} round prover) *message other than* β_i *leads the verifier to accept with probability at most* $1/2$*, whereas the message* β_i *may lead the verifier to accept with probability* 1*. In particular, if* x *is a* YES*-instance* (i.e., $\Pr[\langle P,V\rangle(x)=1]=1$)*, then* β_i *leads the verifier to accept with probability* 1*, whereas any other prover-message lead the verifier to reject with probability at least* $1/2$.[8]

Note that in the special case of NP-proof systems, Definition 4.5 means that there is at most one prover-message (which corresponds to the NP-witness) that makes the (deterministic) verifier accept. Hence, the notion of unambiguous interactive proof systems is an natural extension of the notion of an NP-proof system with unique proofs (or unique NP-witnesses). We also mention that the celebrated interactive proof systems for $\mathcal{PSPACE}$ (of [35, 44]) are unambiguous, since the Sum-Check protocol is unambiguous.[9]

When seeking batch verification for interactive proof systems, we shall restrict our attention to public-coin systems [4].[10] The essential feature of public-coin systems that we use is the fact that the sequence of verifier-messages can be efficiently generated obliviously of the prover-messages.

Lastly, we need a notion that extends the definition of PCP to the interactive context. Viewing PCP systems as NP-proof systems that support verification based on inspecting few randomly selected bits in the NP-witness, we consider (public-coin) interactive proofs systems in which the final verification is based on inspecting few randomly selected bits in the sequence of prover's messages. That is, these proof systems consists of two stages.

1. In the interaction stage, the verifier generates messages obliviously of the prover's prior messages, which are merely recorded for future use.

2. In the final verification stage, which takes place after the entire interaction is completed, the verifier decides based on inspecting few of the bits sent by the prover (and possibly based on its own coin tosses and input). Equivalently, the verifier's final decision is based on all messages sent by the verifier and few probes made into the record of the prover's messages.

A proof system that satisfies the foregoing condition is called a probabilistically checkable interactive proof (PCIP). Note that the combination of the unambiguity condition and the probabilistically checkable condition is problematic; actually, a PCIP cannot be unambiguous (in the strict sense of Definition 4.5), since changing a single bit in a prover-message is unlikely to be detected. Still, a natural relaxation of the unambiguity condition is applicable to PCIP: Rather than requiring that the successful proving strategy is unique, we require that all successful prover strategies generate messages that are close to one another (in Hamming distance). This notion is indeed akin to the

[8] We stress that β_i is determined by the sequence of verifier messages $(\alpha_1,....,\alpha_{i-1},\alpha_i)$, and that different sequences may determine different β_i's. In other words, there is a unique prover strategy P that makes V accept a YES-instance x with probability 1, whereas any other (deterministic) prover strategy $\widetilde{P}$ leads the verifier to reject x with probability at least $\widetilde{p}(x)/2$, where $\widetilde{p}(x)$ is the probability that the interaction $(P,V)(x)$ reaches a point in which $\widetilde{P}$ acts differently from P.

[9] At the beginning of each round, when one variable in the sum is stripped, there is a unique low-degree univariate polynomial that describes the residual function. If the prover sends any other polynomial, then it will be caught with high probability, no matter how it plays in the subsequent rounds.

[10] In the public coin model, at each round, the verifier tosses a predetermined number of coins and sends the outcome to the prover (see discussion at the end of Appendix A.1). This is performed obliviously of the contents of the prover-messages; indeed, the distribution of the verifier's next message is uniform over the set of strings of a predetermined length.

definition of PCPP (see [8, 14]), where the soundness condition holds only with respect to inputs that are far from the predetermined set.

Observe that unambiguous public-coin interactive proof systems can be transformed into ones that satisfy the PCIP condition, by having the prover encode its messages under an error correcting code, and letting the verifier run a PCPP to verify that the original verifier would have accepted the original messages. That is, the new verifier checks that all messages of the new prover are valid codewords, and that they encode messages that would have convinced the original verifier.

Having such a PCIP at our disposal, we employ the ideas that underlie Construction 4.4.1 to each round of interaction in the PCIP. Thus, the emulation of a typical round of interaction (of the PCIP) looks as follows, when referring to the input $(x_1, ..., x_k)$.

Construction 4.5.1 (emulation of a single round):

1. *For every $i \in [k]$, let $\beta_i \in \{0,1\}^{c(n)}$ be the* (unique) *message that the original prover would have sent in the current round regarding the input x_i* (when given the history of communication regarding this input).[11] *For each $j \in [c(n)]$, let $\beta_{i,j}$ denote the j^{th} bit of β_i. Then, for every $j \in [c(n)]$, the prover computes $v_j \leftarrow F(\beta_{1,j} \cdots \beta_{k,j})$, and sends $v_1, ..., v_{c(n)}$ to the verifier. Recall that $F : \{0,1\}^k \to \{0,1\}^{O(d \log k)}$, where d is the commitment parameter.*

   (Indeed, note the analogy to Step 1 in Construction 4.4.1.)

   We shall refer to the foregoing k messages, $\beta_1, ..., \beta_k$, as to the matrix of the current round; that is, we refer to the matrix $(\beta_{i,j})_{i \in [k], j \in [c(n)]}$.

2. *The verifier answers with a single random message, which will be used in all k copies of the PCIP emulation.*

   (Using the same verifier message in all k copies may increase the soundness error by a factor of k, but actually the same may happen when k independently selected messages are used. In any case, before employing the construction, we should reduce the soundness error of the original PCIP so to compensate for this loss.)

Note that verification steps analogous to Steps 2 and 3 of Construction 4.4.1 are not performed at this point, but rather after all rounds of interaction of the PCIP are completed. Once this happens, the following verification steps take place, where $\ell = O(1)$ denotes the number of rounds in the PCIP.

Construction 4.5.2 (final verification):

1. *The verifier generates a sequence of queries, $j_1, ..., j_q \in [c(n)]$, for the PCIP verifier, denoted V. Note that V may ask different queries to the messages of the different ℓ rounds, but for sake of simplicity* (and at the cost of increasing q by a factor of ℓ)*, we assume that it asks the same queries in each round.*

   As in Step 2 of Construction 4.4.1, the verifier sends $j_1, ..., j_q$ to the prover, who responds with the values of the corresponding columns. Note that the prover sends q columns per each of the ℓ matrices.

[11] That is, we consider k executions of the PCIP, where the i^{th} execution refers to the input x_i. We stress that the index of the current round is not reflected in the foregoing notation; an adequate notation for the messages of round r is $\beta_1^{(r)}, ..., \beta_k^{(r)}$ (since in Construction 4.5.2 the message claimed in round r of the i^{th} execution is denoted $\widetilde{\pi}_{i,1}^{(r)} \cdots \widetilde{\pi}_{i,c(n)}^{(r)}$).

For each of these ℓ matrices, the verifier checks (i) *whether the values of these columns match the parity-check values for the relevant matrix* (as provided in Step 1 of Construction 4.5.1) *and* (ii) *whether V would have accepted each of the k inputs when given the corresponding answers* (as well as its own messages as sent in Step 2 of Construction 4.5.1). *Note that the first condition refers to individual matrices, whereas the second condition refers to the sequence of ℓ matrices.*

Specifically, let $\widetilde{v}_1^{(r)}, ..., \widetilde{v}_{c(n)}^{(r)}$ denote the parity check values sent in round $r \in [\ell]$, and suppose that the prover's current answers are $(\widetilde{\pi}_{i,j_{i'}}^{(r)})_{r\in[\ell],i\in[k],i'\in[q]}$, where $(\widetilde{\pi}_{i,j}^{(r)})_{i\in[k]}$ is claimed to be the j^{th} column of the matrix of round r. Then, the verifier performs the following two checks:

(a) For every $r \in [\ell]$ and $i' \in [q]$, it checks whether $F(\widetilde{\pi}_{1,j_{i'}}^{(r)} \cdots \widetilde{\pi}_{k,j_{i'}}^{(r)}) = \widetilde{v}_{j_{i'}}^{(r)}$.

(b) For each $i \in [k]$, it checks whether V would have accepted x_i when making the queries $j_1, ..., j_q$ and receiving the answers $\widetilde{\pi}_{i,j_1}^{(1)}, ..., \widetilde{\pi}_{i,j_q}^{(1)}, ..., \widetilde{\pi}_{i,j_1}^{(\ell)}, ..., \widetilde{\pi}_{i,j_q}^{(\ell)}$ and when its own messages in the i^{th} copy are as recorded in the execution of Construction 4.5.1.

(Indeed, the current step is analogous to Step 2 of Construction 4.4.1, and the next step is analogous to Step 3 in that construction.)

2. *The verifier selects uniformly a set of $k' = O(k/d)$ rows, and the prover is required to prove that these rows contain values* (in the ℓ matrices) *that* (i) *correspond to the unambiguous transcript that makes the PCIP verifier accept, and* (ii) *match the values of the columns provided in Step 1 of Construction 4.5.1.*

   Unlike in Construction 4.4.1, the unambiguity condition cannot be verified by merely inspecting the transcripts of the relevant rows (in all ℓ matrices), *because this condition refers to possible random interactions that extend all possible prefixes of the transcript, where the crucial point is that in such random interactions the future messages of the verifier are not known. Letting $(\alpha_1, \beta_1, ..., \alpha_\ell, \beta_\ell)$ denote a transcript that corresponds to a revealed row* (in the ℓ matrices), *we need to verify a condition analogous to* Eq. (4.2), *for each $i \in [\ell]$. This is done by invoking the original PCIP for ℓ times, such that in the i^{th} invocation we continue the execution at the end of the i^{th} round as if the first $2i$ messages were $(\alpha_1, \beta_1, ..., \alpha_i, \beta_i)$. Note that in the i^{th} invocation* (which refers to an i-round long prefix), *$\ell - i$ rounds of* (additional) *interaction are performed, followed by the final verification step of the original PCIP. All these $k' \cdot \ell$ interactive processes are run in parallel.*[12]

Combining Constructions 4.5.1 and 4.5.2, we obtain a PCIP of $O(\ell)$ rounds of query complexity $q \cdot k$, with communication complexity $O(c(n) \cdot d \log k) + q \cdot k + \frac{k}{d} \cdot c(n))$, where first term is due to Construction 4.5.1, the other terms (as well as the query complexity) are due Construction 4.5.2.[13] Now, recalling that we wish to obtain communication complexity $c(n) \cdot k^\alpha$, for any $\alpha > 0$, as well as low query complexity, we face a problem since the first goal forces using $d = k^\alpha$ (just as in the proof of Theorem 4.4). The solution is to replace the sending of the k' rows in Step 2 of Construction 4.5.2

[12] The most straightforward interpretation of the above is that the j^{th} message-exchange that refers to an i-round prefix takes place in round j of the parallel execution, regardless of i. It is also natural to let this message-exchange take place in round $i + j$. Actually, the mapping of additional message-exchanges to parallel rounds is immaterial as long as it is monotone and injective.

[13] Specifically, the $O(q \cdot k)$ term (resp., the $\frac{k}{d} \cdot c(n)$ term) is due to Step 1a (resp., Step 2) of Construction 4.5.2.

by having the prover employ the entire construction recursively so to allow for the verification of the corresponding conditions. We stress that this recursion is more complex than in the proof of Theorem 4.4, since we need to verify an unambiguity claim regarding a PCIP whose (constant) number of rounds varies throughout the recursive calls.

Another problem that we face is that the resulting query complexity is $\mathrm{poly}(q) \cdot k$, whereas Theorem 4.2 asserts polylogarithmic complexity. The solution is to use a *query reduction step*; that is, after deriving a PCIP of $O(\ell)$ rounds of query complexity $\mathrm{poly}(q) \cdot k$ and communication complexity $O(c(n) \cdot k^{\alpha}) + \mathrm{poly}(q) \cdot k$, we reduce the query complexity to $\mathrm{poly}(\log n)$. Indeed, query reduction is a standard PCP technique, but one should verify that it preserves the unambiguity condition.

Conclusion: Deriving Theorem 4.2. The foregoing outline completes our high-level description of the proof of Theorem 4.2, and it implicitly specifies the type $\mathcal{T}$ of interactive proof systems to which this theorem refers. To spell it out, type $\mathcal{T}$ consists of public-coin unambiguous PCIPs (of proximity).

Chapter 5

Epilogue

We have seen four fundamentally different constructions of doubly-efficient interactive proof systems. Here we try to abstract their essence.

The first construction, presented in Sections 2.1 and 2.2, adapts the principles that underlie the (Sum-Check based) construction of Lund, Fortnow, Karloff, and Nisan [35] to a natural subclass of seemingly hard problems in $\mathcal{P}$. Specifically, this construction is based on an arithmetization of sets in the class (of locally characterizable sets), which reduces the verification of membership in a set to the evaluation of an arithmetic expression, and on the application of the Sum-Check protocol to the arithmetic expression. The doubly-efficient features of this proof system is due to the definition of *locally characterizable sets*; specifically, we rely on the fact that such sets are characterized in terms of few (i.e., polynomially many) local conditions, which can expressed by small Boolean formulae.

The second construction, presented in Section 2.3, avoids an arithmetization of the original problem (as well as breaking it down to sub-problems that have non-apparent relation to the original problem).[1] Instead, the verification of the solution for a parameterized problem (e.g., counting t-cliques) is reduced to the verification of a solution for a similar problem that refers to a smaller value of the parameter (e.g., counting $(t-1)$-cliques). In other words, the verification is via a downwards (w.r.t the parameter) self-reduction, which *maintains the flavor of the original computational problem*. While we use this approach towards obtaining a doubly-efficient interactive proof system for counting t-cliques, for any constant t, we stress that this approach is applicable also to varying t, leading to an alternative proof system for $\#\mathcal{P} \supseteq \mathrm{co}\mathcal{NP}$.

The third construction, presented in Chapter 3, refers to the verification of (the outcome of) computations conducted by (log-space uniform) Boolean circuits. This construction reduces the verification of a claim regarding some intermediate values in such a computation to a claim regarding other intermediate values in the same computation. Specifically, *the verification of a claim regarding the values of gates at a certain distance from the output of the circuit is reduced to the verification of a claim regarding the values of the gates that are one unit farther from the output*. Each reduction step employs the Sum-Check protocol, after applying an adequate arithmetization, which means that the underlying principles of Lund, Fortnow, Karloff, and Nisan [35] are employed at each reduction step (which corresponds to the adjacent levels of the circuit). The doubly-efficient features of this proof system is due to the fact that the reduction between adjacent levels in the (log-space uniform) circuit can be expressed by few (i.e., polynomially many) local conditions, which can expressed by small Boolean formulae.

[1] Here we refer to the sub-problems considered in Chapters 3 and 4.

The last construction, presented in Chapter 4, is based on different principles. This construction, which refers to polynomial-time computations of bounded-space, proceeds by a "cut-and-batch" approach: The verification of a claim regarding a t-step computation is reduced to the verification of k claims regarding different t/k-step computations, but *these k verification tasks are conducted at a cost that is significantly smaller than conducting such k verifications separately.* Indeed, the pivot of this approach is efficient methods for *batch verification*. These methods are applied not only to claims of the foregoing type, but also to more complex claims that refer to the execution of other interactive proof systems (which arise from the recursive construction).

As seen in Chapter 4, the construction of efficient batch verification refers to numerous refined notions that refer to PCPs and interactive proof systems, including PCPs of Proximity [8, 14], input-oblivious queries (to such PCPs), unambiguous interactive proofs (Definition 4.5), and PCIP (probabilistically checkable *interactive* proofs). These notions are of independent interest and the latter three (which are relatively new) beg for further study. Other begging questions refer to the possibility of efficient batch verification for all sets in $\mathcal{NP}$ (rather than only $\mathcal{UP}$) and the possibility of using "private coins" to improve the efficiency of some ingredients of the construction, let alone the problems mentioned in Chapter 1.

Speculations. Does the early discovery of fantastic results promote or hinder further progress?

Ever since 1990, I held the opinion that the amazing results of Lund, Fortnow, Karloff, and Nisan [35] and Shamir [44] have actually discouraged further study of interactive proof systems, and that this is most unfortunate. Yet, one thing that helps when confronting such celebrated results is discovering aspects that they did not address and providing interesting results that refer to these aspects. I believe that this has been happening with the notion of *doubly-efficient* interactive proof systems and the discovery of the results surveyed in this monograph.

The emergence of the study of doubly-efficient interactive proof systems does call to question ingrained attitudes like the view that (in the context of interactive proof systems per se)[2] perfect completeness and public-coin format come for free. Indeed, it was shown by Vadhan [48], already in 2000, that the known transformations of general interactive proof systems to public-coin systems and/or to systems with perfect completeness may not preserve the complexity of the prescribed prover strategy, but this fact becomes crucial in the context of *doubly-efficient* interactive proof systems.

We stress that a renewed study of interactive proof systems need not be confined to doubly-efficient ones. If we look closely, we may discover quite a few interesting open problems (see, e.g., [34]).

[2] Unlike in the context of zero-knowledge (or when considering "relatively efficient" provers, as in Section A.2.1).

Acknowledgments

I am grateful to Guy Rothblum for many useful discussions regarding the subject. In particular, the expositions in Chapters 3 and 4 are based on his oral presentations of [27] and [42], respectively. Furthermore, Chapter 2 reproduces text from our joint works [24, 25].

I also wish to thank Ran Canetti and Ron Rothblum for discussion of various related topics. In particular, Theorem 1.6 was inspired by a discussion with Ran. Finally, I wish to thank the FnT reviewer for many useful comments.

Appendix: Defining Interactive Proofs and Arguments

In this appendix we review the basic definition of interactive proof systems as well as the definition of argument systems. The latter definition is reviewed within the context of a general discussion regarding the power of the prover. (The text was adapted from [19, Sec. 9.1], which provides further discussions as well as an exposition of the main known results.)

A.1 The basic definition of interactive proofs

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

We formalize the interaction between parties by referring to the *strategies* that the parties employ.[3] A strategy for a party is a *function mapping the party's view of the interaction so far to a description of this party's next move*; that is, such a strategy describes (or rather prescribes) the *party's next move* (i.e., its next message or its final decision) *as a function of the common input* (i.e., the aforementioned assertion), *the party's internal coin tosses*, *and all messages it has received so far*. Note that this formulation presumes (implicitly) that each party records the outcomes of its past coin tosses as well as all the messages it has received, and determines its moves based on these. Thus, an interaction between two parties, employing strategies A and B respectively, is determined by the common input, denoted x, and the randomness of both parties, denoted r_A and r_B. Assuming that A takes the first move (and B takes the last move), the corresponding (t-round) interaction transcript (on common input x and randomness r_A and r_B) is $\alpha_1, \beta_1, ..., \alpha_t, \beta_t$, where $\alpha_i = A(x, r_A, \beta_1, ..., \beta_{i-1})$ and $\beta_i = B(x, r_B, \alpha_1, ..., \alpha_i)$. The corresponding final decision of A is defined as $A(x, r_A, \beta_1, ..., \beta_t)$.

We say that a party employs a probabilistic polynomial-time strategy if its next move can be computed in a number of steps that is *polynomial in the length of the common input*. In particular,

[3] An alternative formulation refers to the interactive machines that capture the behavior of each of the parties (see, e.g., [18, Sec. 4.2.1.1]). Such an interactive machine invokes the corresponding strategy, while handling the communication with the other party and keeping a record of all messages received so far.

this means that, on common input x, the strategy may only consider a polynomial in $|x|$ many messages, which are each of poly($|x|$) length.[4] Intuitively, if the other party exceeds an *a priori* (polynomial in $|x|$) bound on the total length of the messages that it is allowed to send, then the execution is suspended. Thus, referring to the aforementioned strategies, we say that A is a probabilistic polynomial-time strategy if, for every i and $r_A, \beta_1, ..., \beta_i$, the value of $A(x, r_A, \beta_1, ..., \beta_i)$ can be computed in time polynomial in $|x|$. Again, in proper use, it must hold that $|r_A|, t$ and the $|\beta_i|$'s are all polynomial in $|x|$.

Definition A.1 (Interactive Proof systems – IP)[5]: *An* interactive proof system for a set S *is a two-party game, between a* verifier *executing a* probabilistic polynomial-time strategy*, denoted V, and a* prover *that executes a* (computationally unbounded) *strategy, denoted P, satisfying the following two conditions:*

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

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

The error probability (in the soundness condition) can be reduced by successive applications of the proof system. (This is easy to see in the case of sequential repetitions, but holds also for parallel repetitions; see [19, Exer. 9.1].) In particular, repeating the proving process for k times, reduces the probability that the verifier is fooled (i.e., accepts a false assertion) to 2^{-k}, and we can afford doing so for any $k = \text{poly}(|x|)$.

The round complexity of an interactive proof system is $\rho : \mathbb{N} \to \mathbb{N} \cup \{0\}$ if, on input x, the prover sends at most $\rho(|x|)$ messages. An interactive proof system is said to be of the public coin type if, at each round, the verifier sends a (fixed) projection of its randomness; specifically, the verifier strategy V has the form $V(x, r, \gamma_1, ..., \gamma_i) = r_{\ell_x(i-1)+1} \cdots r_{\ell_x(i)}$, where $\ell_x : \mathbb{N} \cup \{0\} \to \mathbb{N} \cup \{0\}$ satisfies $\ell_x(i) \in [0, |r|]$ and $\ell_x(i-1) \leq \ell_x(i)$ (actually, w.l.o.g, $\ell_x(i-1) < \ell_x(i)$).[6]

Note that Definition A.1 allows no error in the completeness condition. This version is often called an interactive proof system of perfect completeness. A more liberal definition allows for bounded error probability also in the completeness condition; in that case, one typically uses an error bound of 1/3 in both conditions. With the exception of doubly-efficient interactive proof systems for sets in $\mathcal{BPP} \setminus \mathcal{P}$, all interactive proof systems mentioned in this survey are of the perfect completeness type, and the upper bound stated in Theorem 1.4 applies also to the more liberal (two-sided error probability) version. (We mention that the known transformation of general interactive proof systems to ones with perfect completeness does not apply to doubly-efficient interactive proof systems, since the resulting prover strategy is not efficient and this seems inherent [48].)

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

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

[6] Actually, ℓ_x may depend on $|x|$ only. Equivalently, the verifier can be thought of as tossing coins on-line per each round, such that in the i^{th} round it tosses $\ell_x(i) - \ell_x(i-1)$ coins.

A.2 On computationally bounded provers: an overview

Recall that our definition of interactive proofs (i.e., Definition A.1) makes no reference to the computational abilities of the potential prover. This fact has two conflicting consequences:

1. The completeness condition does not provide any upper bound on the complexity of the corresponding proving strategy (which convinces the verifier to accept valid assertions).

2. The soundness condition guarantees that, regardless of the computational effort spend by a cheating prover, the verifier cannot be fooled to accept invalid assertions (with probability exceeding the soundness error).

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

A.2.1 How powerful should the prover be?

Suppose that a set S is in $\mathcal{IP}$. This means that there exists a verifier V that can be convinced to accept any input in S but cannot be fooled to accept any input not in S (except with small probability). One may ask how powerful should a prover be such that it can convince the verifier V to accept any input in S. It is known (see [19, Prop. 9.5]) that an optimal prover strategy (for convincing any fixed verifier V) can be implemented in polynomial-space, and that we cannot expect any better for a generic set in $\mathcal{PSPACE} = \mathcal{IP}$ (because the emulation of the interaction of V with any optimal prover strategy yields a decision procedure for the set). Still, we may seek better upper-bounds on the complexity of some prover strategy that convinces a *specific* verifier, which in turn corresponds to a specific set S. More interestingly, considering all possible verifiers that give rise to interactive proof systems for S, we wish to upper-bound the computational power that suffices for convincing any of these verifiers (to accept any input in S).

We stress that, unlike the case of computationally-sound proof systems (see Section A.2.2), we do not restrict the power of the prover in the soundness condition, but rather consider the minimum complexity of provers meeting the completeness condition. Specifically, we are interested in *relatively efficient* provers that meet the completeness condition. The term "relatively efficient prover" has been given three different interpretations, which are briefly surveyed next.

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

   This interpretation is adequate and in fact crucial for applications in which such an auxiliary input is available to the otherwise polynomial-time parties. Typically, such auxiliary input is available in cryptographic applications in which parties wish to prove in (zero-knowledge)

that they have correctly conducted some computation (see [19, Apdx. C.4.3.2]). In these cases, the NP-proof is just the transcript of the computation by which the claimed result has been generated, and thus the auxiliary input is available to the party that plays the role of the prover.

2. A prover is considered *relatively efficient* if it can be implemented by a probabilistic polynomial-time oracle machine with oracle access to the set S itself. Note that the prover in the standard proof system for Graph Non-Isomorphism has this property (see [19, Const. 9.3]).

   This interpretation generalizes the notion of self-reducibility of NP-proof systems. Recall that by self-reducibility of an NP-set (or rather of the corresponding NP-proof system) we mean that the search problem of finding an NP-witness is polynomial-time reducible to deciding membership in the set (cf. [19, Def. 2.14]). Here we require that implementing the prover strategy (in the relevant interactive proof) be polynomial-time reducible to deciding membership in the set.

3. A prover is considered *relatively efficient* if it can be implemented by a probabilistic machine that runs in time that is polynomial in the complexity of deciding the set.[7] This interpretation relates the time-complexity of convincing a "lazy person" (i.e., a verifier) to the time-complexity of determining the truth (i.e., deciding membership in the set).

   Hence, in contrast to the first interpretation, which is adequate in settings where assertions are generated along with their NP-proofs, the current interpretation is adequate in settings in which the prover is given only the assertion and has to find a proof to it by itself (before trying to convince a lazy verifier of its validity). See [37].

   Indeed, this interpretation generalizes one of the two efficiency conditions that underlie the definition of doubly-efficient interactive proof systems. Specifically, for set in $\mathcal{P}$ (or even in $\mathcal{BPP}$), such a *relatively efficient* prover is required to run in polynomial-time.

A.2.2 Computational-soundness

Relaxing the soundness condition such that it only refers to relatively-efficient ways of trying to fool the verifier (rather than to all possible ways) yields a fundamentally different notion of a proof system. The verifier's verdict in such a system is not absolutely sound, but is rather sound *provided that the potential cheating prover does not exceed the presumed complexity limits*. As in Section A.2.1, the notion of "relative efficiency" can be given different interpretations, the most popular one being that the cheating prover strategy can be implemented by a (non-uniform) family of polynomial-size circuits. The latter interpretation coincides with the first interpretation used in Section A.2.1 (i.e., a probabilistic polynomial-time strategy that is given an auxiliary input (of polynomial length)). Specifically, in this case, the soundness condition is replaced by the following computational soundness condition that asserts that it is infeasible to fool the verifier into accepting false statements. Formally:

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

[7] We stress that we refer to the probabilistic or deterministic time complexity of the set, not to its non-deterministic time complexity.

As in case of standard soundness, the computational-soundness error can be reduced by repetitions. We warn, however, that unlike in the case of standard soundness (where both sequential and parallel repetitions will do), the computational-soundness error cannot *always* be reduced by parallel repetitions (see [7]). Hence, it is often preferred to define computational soundness as having negligible error (i.e., error that decreases faster than the reciprocal of any polynomial function).[8]

It is common and natural to consider proof systems in which the prover strategies considered both in the completeness and soundness conditions satisfy the same notion of relative efficiency. Protocols that satisfy these conditions with respect to the foregoing interpretation are called arguments. We mention that argument systems may be more efficient (e.g., in terms of their communication complexity) than interactive proof systems (cf. [32] versus [26]).

[8] In such a case, the definition asserts that *for every prover strategy that is implementable by a family of polynomial-size circuits $\{C_n\}$, every positive polynomial p, and every sufficiently long $x \in \{0,1\}^* \setminus S$, the probability that V accepts x when interacting with $C_{|x|}$ is less than $1/p(|x|)$.*

Bibliography

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In *46th IEEE Symposium on Foundations of Computer Science*, pages 98–117, 2015.

[2] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing Weight by Gaining Edges. In *22nd ESA*, pages 1–12, 2014

[3] Noga Alon, Oded Goldreich, Johan Håstad, and Rene Peralta. Simple Constructions of Almost k-wise Independent Random Variables. *Journal of Random Structures and Algorithms*, Vol. 3, No. 3, pages 289–304, 1992. Preliminary version in *31st FOCS*, 1990.

[4] Laszlo Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–429, 1985.

[5] Laszlo Babai, Lance Fortnow, Leonid Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In *23rd ACM Symposium on the Theory of Computing*, pages 21–31, 1991.

[6] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant N. Vasudevan. Average-Case Fine-Grained Hardness. In *48th ACM Symposium on the Theory of Computing*, pages 483–496, 2017.

[7] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science*, pages 374–383, 1997.

[8] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, Vol. 36 (4), pages 889–974, 2006. Extended abstract in *36th STOC*, 2004.

[9] Andreas Björklund and Petteri Kaski. How Proofs are Prepared at Camelot. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 391–400, 2016.

[10] Allan Borodin. On Relating Time and Space to Size and Depth. *SIAM Journal on Computing*, Vol. 6 (4), pages 733–744, 1977.

[11] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *49th ACM Symposium on the Theory of Computing*, pages 474–482, 2017.

[12] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility. In *2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 261–270, 2016.

[13] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, Vol. 201 (2), pages 216–231, 2005.

[14] Irit Dinur and Omer Reingold. Assignment-testers: Towards a combinatorial proof of the PCP-Theorem. *SIAM Journal on Computing*, Vol. 36 (4), pages 975–1024, 2006. Extended abstract in *45th FOCS*, 2004.

[15] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science A*, Vol. 141 (12), pages 109–131, 1995.

[16] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer-Verlag Monographs in Computer Science, 1999.

[17] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Approximating Clique is almost NP-complete. *Journal of the ACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.

[18] Oded Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001.

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

[20] Oded Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

[21] Oded Goldreich. On the doubly-efficient interactive proof systems of GKR. *ECCC*, TR17-101, 2017.

[22] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 3, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[23] Oded Goldreich and Or Meir. Input-Oblivious Proof Systems and a Uniform Complexity Perspective on P/poly. *TOCT*, Vol. 7 (4), pages 16:1–16:13, 2015.

[24] Oded Goldreich and Guy N. Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. *ECCC*, TR17-018, 2017.

[25] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. *ECCC*, TR18-046, 2018.

[26] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic provers. *Computational Complexity*, Vol. 11, pages 1–53, 2002.

[27] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating Computation: Interactive Proofs for Muggles. *Journal of the ACM*, Vol. 62(4), Art. 27:1-27:64, 2015. Extended abstract in *40th STOC*, pages 113–122, 2008.

[28] Shafi Goldwasser, Silvio Micali and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985. Earlier versions date to 1982.

[29] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, Volume 28, Number 4, pages 1364–1396, 1999.

[30] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *45th ACM Symposium on the Theory of Computing*, pages 565–574, 2013.

[31] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *46th ACM Symposium on the Theory of Computing*, pages 485–494, 2014.

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

[33] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In *38th IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1977.

[34] Maya Leshkowitz. Round Complexity Versus Randomness Complexity in Interactive Proofs. *ECCC*, TR17-055, 2017.

[35] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992. Extended abstract in *31st FOCS*, 1990.

[36] Or Meir. IP = PSPACE Using Error-Correcting Codes. *SIAM Journal on Computing*, Vol. 42 (1), pages 380–403, 2013.

[37] Silvio Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, Vol. 30 (4), pages 1253–1298, 2000. Preliminary version in *35th FOCS*, 1994.

[38] Joseph Naor and Moni Naor. Small-bias Probability Spaces: Efficient Constructions and Applications. *SIAM Journal on Computing*, Vol 22, pages 838–856, 1993. Preliminary version in *22nd STOC*, 1990.

[39] Jaroslav Nesetril and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, Vol. 26, No. 2, pages 415–419, 1985.

[40] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *42nd ACM Symposium on the Theory of Computing*, pages 603–610, 2010.

[41] Mihai Patrascu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In *21st SODA*, pages 1065–1075, 2010.

[42] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *48th ACM Symposium on the Theory of Computing*, pages 49–62, 2016.

[43] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient Batch Verification for UP. *ECCC*, TR18-022, 2018.

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

[45] Madhu Sudan. Invariances in Property Testing. In *Property Testing: Current Research and Surveys*. Springer, Lecture Notes in Computer Science (Vol. 6390), pages 211–227, 2010.

[46] Justin Thaler. Semi-Streaming Algorithms for Annotated Graph Streams. In *43rd International Colloquium on Automata, Languages, and Programming*, pages 59:1–59:14, 2016.

[47] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, Vol. 20 (5), pages 865–877, 1991.

[48] Salil P. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In *32nd ACM Symposium on the Theory of Computing*, pages 200–207, 2000.

[49] Leslie G. Valiant. The Complexity of Computing the Permanent. Theoretical Computer Science, Vol. 8, pages 189–201, 1979.

[50] Virginia Vassilevska Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis. In *10th International Symposium on Parameterized and Exact Computation*, pages 17–29, 2015.

[51] Ryan Williams. Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation. In *31st Conference on Computational Complexity*, pages 2:1–2:17, 2016.