Electronic Colloquium on Computational Complexity, Report No. 102 (2017)

Overview of the doubly-efficient interactive proof systems of RRR

Oded Goldreich*

June 9, 2017

Abstract

We provide an overview of the doubly-efficient interactive proof systems of Reingold, Rothblum, and Rothblum (*STOC*, 2016). Recall that by their result, 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 $\widetilde{O}(n)$.

Contents

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

ISSN 1433-8092

1 A brief introduction

Loosely speaking, doubly-efficient interactive proof systems are interactive proof systems in which the prescribed prover runs in polynomial-time, whereas the prescribed verifier runs in almost-linear-time.[1] We stress that the soundness condition of these systems is information theoretic; that is, it refers to all possible cheating strategies (and not only to feasible ones (as in argument systems)).

The notion of a doubly-efficient interactive proof systems was first defined by Goldwasser, Kalai, and Rothblum [7], who presented such systems for sets having log-space uniform circuits of polynomial size and $d(n) = n^{O(1)}$ depth. This result is actually incomparable to a recent result of Reingold, Rothblum, and Rothblum [10] that provides a (constant-round) doubly-efficient interactive proof systems for sets that are decidable in simultaneous polynomial time and $s(n) = n^{O(1)}$ space. Our focus is on this later result, which asserts the following –

Theorem 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 $(\widetilde{O}(n) + \mathrm{poly}(s(n)) \cdot n^{\delta})$-time.*

Actually, the unspecified polynomial in [10] is almost quadratic (priv. comm. with the authors of [10], May 2017). The proof of Theorem 1 spans 70 pages in [10], and in this note we merely attempt to present an overview of it.

2 High level structure

Theorem 1 is proved by considering the instantaneous configurations of a 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 that $t \leq \mathrm{poly}(|x|)$.

The core of the proof of Theorem 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 $(\forall i \in [k])\ x_i \in S$. Then, the complexity of verifying the latter claim by employing the proof

[1] Such proof systems were called *interactive proofs for muggles* [7] and *interactive proofs for delegating computation* [10]. Here, we interpret the term "almost linear (in n)" as having the form $n^{1+o(1)}$.

system of [9, 11] 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 [9, 11] takes time that is proportional to $s(n)^4$. Then, the time complexity of batch verification 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.[2] 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

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

Theorem 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 2, the construction of the proof system asserted in Theorem 1 can be recursively presented as follows.

Construction 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.

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).[3] This interactive proof system is obtained by applying Theorem 2 to the interactive proof system that refers to t/k step computation, where the latter is obatined by a recursive call. Recall that Theorem 2 requires that the latter system be of type $\mathcal{T}$.*

   Hence, at the bottom of the recursion we have communication complexity $\text{poly}(s)$*, at its j^{th} level* (from the bottom) *we have communication complexity $O(k^{j\alpha}\cdot\text{poly}(s)+k^{(j-1)\alpha+1})$, where α is the constant in Theorem 2 and the recusion has $\log_k t$ levels. Using $k^{(\log_k t)\cdot\alpha} = t^\alpha$, at the top level of the recursion, this yields communication complexity $O(t^\alpha \cdot (\text{poly}(s) + k))$, whereas verification time is poly-logarithmic, proving time is polynomial, and the number of rounds is $O(1/\alpha)^{\log_k t}$.*

[2] 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 solutions in NP-proof systems) and support local checking of the prover messages (akin to PCPs (or rather PCPPs)).

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

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 1 follows by combining Construction 3 with Theorem 2, while setting $\alpha = \delta/O(1)$ and $k = n^\delta$. (Actually, Construction 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 3, we focus on the proof of Theorem 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.

3 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 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., [2, 5], as interpreted by [3, 4]). Readers that are unfamiliar with PCPPs (i.e., PCPs of Proximity), may consider standard PCP systems and verification 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 verifieraction.

Hence, our starting point is a set $S \in \mathcal{UP}$ and a corresponding PCP system. 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.[4] Such PCP systems are known and can be derived from any PCPP system.

Theorem 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 $O(k^\alpha \cdot c + k)$.*

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 a Reed-Solomon code.)

Construction 4.1 (basic construction): *On input $(x_1, ..., x_k) \in \{0,1\}^{k \cdot n}$, the proof system proceeds as follows.*

[4] Hence, these PCP systems are not truly input-oblivious in the sense studied in [6].

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 corruptions.

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* (i) *whether the values of these columns match the parity-check values, and* (ii) *whether 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 $\widetilde{\pi}_{i,j_1}, ..., \widetilde{\pi}_{i,j_q}$.*

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* (ii) *that 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 above 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 $q = O(1)$, 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 **4.2** (soundness of Construction 4.1): *Construction 4.1 is an interactive proof system for S^k.*

[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.

Proof: Turning to the more orderly analysis of the soundness of Construction 4.1, we assume without loss of generality that the values provided by the prover satisfy the conditions stated in Steps 2a and 3b. 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 $\widetilde{\pi}_{i,j_{i'}} \neq \pi_{i,j_{i'}}$ for some $i' \in [q]$, and so $\widetilde{\pi}_i$ 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 $\widetilde{\pi}_i \neq \pi_i$ implies that the string extracted from $\widetilde{\pi}_i$ (so that $\widetilde{\pi}_i$ 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 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 4.1, except that in Step 3 we invoke Construction 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 match 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 straightfoward verification that is used originally in Step 3. ■

4 Batch verification for unambiguous IP

The proof of Theorem 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 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

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

[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 $i' \in [q]$.

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 at least 1/2.

Definition 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 at least* 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*
>
> $$\Pr\left[(\widetilde{P},V)(x)=1 \;\middle|\; \begin{array}{rcl} \langle \widetilde{P},V\rangle_{2i-1}(x) & = & (\alpha_1,\beta_1,...,\alpha_{i-1},\beta_{i-1},\alpha_i) \\ \langle \widetilde{P},V\rangle_{2i}(x) & \neq & (\alpha_1,\beta_1,....,\alpha_{i-1},\beta_{i-1},\alpha_i,\beta_i) \end{array}\right] < 1/2, \qquad (2)$$
>
> *where $(\widetilde{P},V)(x)$ denotes the output of V after interacting with $\widetilde{P}$ on input x.*

That is, Eq. (2) asserts that any message other than β_i leads the verifier to accept with probability that is smaller than 1/2, *whereas the message β_i may lead the verifier to accept with probability* 1.[8]

Note that in the special case of NP-proof systems, Definition 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 asystem with unique proofs (or unique NP-witnesses). We also mention that the celebrated interactive proof systems for $\mathcal{PSPACE}$ (of [9, 11]) 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 [1]. The essential feature of public-coin systems that we use is the fact that the next message of the verifier can be selected without looking at the previous prover messages. Furthermore, we use the fact that given the previous messages of the verifier (but not the previous messages of the prover), one can efficiently sample the distribution of the next verifier message. Needless to say, these features are trivially satisfied by public-coin interactive proof systems.

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.

[8] In particular, if x is a YES-instance (i.e., $\Pr[\langle P,V\rangle(x)=1]=1$), then the last message in $\langle P,V\rangle_{2i}(x)$ is uniquely determined by the first $2i-1$ messages.

[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.

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 verfication 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 possiblly based on its own coin tosses and input). Equivalently, the verifier 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 5), since changing a single bit in a 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 definition of PCPP (see [3, 4]), 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 verifier 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 4.1 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 5.1 (emulation of a single round):

1. *Let β_i 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).[10] *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 4.1.)

   We shall refer to the foregoing k messages as to the matrix of the current round.

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 4.1 are not performed at this point, but rather after all rounds of interaction of the PCIP are completed. Once this happens,

[10] That is, we consider k executions of the PCIP, where the i^{th} execution refers to the input x_i.

the following verification steps take place, where $\ell = O(1)$ denotes the number of rounds in the PCIP.

Construction 5.2 (final verification):

1. *The verifier generate 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 4.1, 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.

   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 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 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, 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 5.1.*

   (Indeed, the current step is analogous to Step 2 of Construction 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 5.1.*

   Unlike in Construction 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 prefices 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. (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.[11]

Combining Constructions 5.1 and 5.2, we obtain a PCIP of $O(\ell)$ rounds of query complexity $\mathrm{poly}(q) \cdot k$, with communication complexity $O(c(n) \cdot d \log k) + \mathrm{poly}(q) \cdot k + \frac{k}{d} \cdot c(n))$, where first term is due to Construction 5.1 , the other terms (as well as the query complexity) are due Construction 5.2.[12] 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). The solution is to replace the sending of the k' rows in Step 2 of Construction 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, 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 2 asserts polylogarithmic query 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 $q = \mathrm{poly}(\log n)$. Indeed, query reduction is a standard PCP technique, but one should verify that it preserves the unambiguity condition.

Conclusion: Deriving Theorem 2. The foregoing outline completes our high-level description of the proof of Theorem 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).

Acknowledgements

We are grateful to Guy Rothblum for many useful discussions regarding the Reingold-Rothblum-Rothblum proof system.

References

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

[2] 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.

[3] 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.

[11] The most straightforward interpretation of the above is that the j^{th} interaction 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 interaction take place in round $i+j$. Actually, the mapping of additional interactions to parallel rounds is immaterial as long as it is monotone and injective.

[12] Specifically, the $\mathrm{poly}(q) \cdot k$ term (resp., the $\frac{k}{d} \cdot c(n)$ term) is due to Step 1a (resp., Step 2) of Construction 5.2.

[4] 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.

[5] 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.

[6] 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.

[7] Shafi Goldwasser, Yael Tauman Kalai, 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.

[8] 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.

[9] 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.

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

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

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il