Randomized Lifting to Semi-Structured Communication Complexity via Linear Diversity

Vladimir Podolskii
Tufts University, Medford, MA, USA

Alexander Shekhovtsov
Moscow Institute of Physics and Technology, Russia

Abstract

We study query-to-communication lifting. The major open problem in this area is to prove a lifting theorem for gadgets of constant size. The recent paper [2] introduces semi-structured communication complexity, in which one of the players can only send parities of their input bits. They have shown that for any $m \geq 4$ deterministic decision tree complexity of a function f can be lifted to the so called semi-structured communication complexity of $f \circ \textsc{Ind}_m$, where $\textsc{Ind}_m$ is the Indexing gadget.

As our main contribution we extend these results to randomized setting. Our results also apply to a substantially larger set of gadgets. More specifically, we introduce a new complexity measure of gadgets, *linear diversity*. For all gadgets g with non-trivial linear diversity we show that randomized decision tree complexity of f lifts to randomized semi-structured communication complexity of $f \circ g$. In particular, this gives tight lifting results for Indexing gadget $\textsc{Ind}_m$, Inner Product gadget IP_m for all $m \geq 2$, and for Majority gadget MAJ_m for all $m \geq 4$. We prove the same results for deterministic case.

From our result it immediately follows that deterministic/randomized decision tree complexity lifts to deterministic/randomized parity decision tree complexity. For randomized case this is the first result of this type. For deterministic case, our result improves the bound in [6] for Inner Product gadget.

To obtain our results we introduce a new *secret sets* approach to simulation of semi-structured communication protocols by decision trees. It allows us to simulate (restricted classes of) communication protocols on truly uniform distribution of inputs.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Oracles and decision trees

Keywords and phrases communication complexity, decision trees, lifting

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.78

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2024/199/`

1 Introduction

In recent years numerous results emerged that lift the complexity of a function in a weak model of computation to the complexity of a modified version of the function in a stronger model of computation [10, 13, 5, 11, 4, 14]. These results proved to be extremely useful for solving open problems in various areas of computational complexity [17, 18, 16, 9, 8]. In this type of results we start with a function $f\colon \{0,1\}^n \to \{0,1\}$ that is hard for a weak computation model (like decision trees) and for a gadget $g\colon \{0,1\}^m \to \{0,1\}$ we consider a function $f \circ g^n\colon \{0,1\}^{nm} \to \{0,1\}$ in which we substitute each variable of f by an output of g applied to fresh variables. Our goal is to show that the resulting function $f \circ g^n$ is hard for a strong computation model (like communication complexity or Boolean formula complexity). We would like the result to hold for as simple g, as possible.

In this paper we are mostly interested in lifting from decision tree complexity to communication complexity, which is sometimes called query-to-communication lifting. This particular type of lifting has seen numerous results [17, 10, 12, 5, 11, 4]. In these papers

the results of this type were obtained and gradually improved for both deterministic and randomized cases and for gradually increasing set of possible gadgets. The size of the gadget is a parameter that is of importance for applications. The smallest known size of the gadget for both deterministic and randomized case is logarithmic. For deterministic case, the results for logarithmic size of gadgets were obtained in [5] and [20]. For randomized case, the first result was obtained in [11] with a gadget of polynomial size. The paper [4] reduced the size of gadget for randomized case to logarithm. Obtaining the lifting results with the gadgets of constant size remains a major open problem.

One of the possible approaches to this problem is to address lifting to restricted models of communication or even simpler computational models and to try to obtain lifting with constant-size gadget in this setting. Some progress in this direction was obtained in recent independent papers [2, 6].

The paper [6] shows lifting from deterministic decision tree complexity $\mathsf{D}^{\mathsf{dt}}(f)$ to deterministic parity decision tree complexity $\mathsf{D}^{\mathsf{dt}}_{\oplus}(f \circ g)$ for a wide range of gadgets g, including gadgets of constant size. More specifically, for each gadget g they introduce a stifling complexity measure k and they show that $\mathsf{D}^{\mathsf{dt}}_{\oplus}(f \circ g) \geq \Omega(k \cdot \mathsf{D}^{\mathsf{dt}}(f))$. In particular, from their result it follows that $\mathsf{D}^{\mathsf{dt}}_{\oplus}(f \circ \textsc{Ind}_m) \geq \Omega(\log m \cdot \mathsf{D}^{\mathsf{dt}}(f))$ and $\mathsf{D}^{\mathsf{dt}}_{\oplus}(f \circ \mathrm{IP}_m) \geq \Omega(\mathsf{D}^{\mathsf{dt}}(f))$ for any positive m, where $\textsc{Ind}_m$ and IP_m are Indexing and Inner Product gadgets that are among the most standard in this field (see Subsection 2.3 for the definition of these functions).

The paper [2] introduces semi-structured communication complexity. In this model one of the players is allowed to send only parities of their input bits. This model is restricted compared to regular communication complexity model, but is more powerful (up to a factor of 2) compared to parity decision trees, as players can easily simulate a parity decision tree with a semi-structured communication protocol. The paper [2] shows lifting from deterministic decision trees to semi-structured communication protocol with $\textsc{Ind}_k$ gadget for any $k \geq 4$.

Our results

We show that for a wide range of gadgets (including constant size) lifting is possible from randomized decision trees to randomized semi-structured communication complexity.

More specifically, we introduce a complexity measure *linear diversity* for gadgets. Informally, it is equal to the number of distinct (up to negation) non-constant linear functions (over $\mathbb{F}_2$) in Bob's variables we can obtain by fixing Alice's variables. We observe that the linear diversity of $\textsc{Ind}_m$ is m and the linear diversity of IP_m is $2^m - 1$.

We show that for any function (or relation) f for any $k \geq 2$ and for any gadget g with linear diversity k randomized semi-structured communication complexity of $f \circ g$ is greater or equal to $\Omega(\log k \cdot \mathsf{R}^{\mathsf{dt}}(f))$, where by $\mathsf{R}^{\mathsf{dt}}(f)$ we denote the minimal depth of probabilistic decision tree computing f. In particular, our result applies to gadgets of constant size. Our result gives tight bounds for both $\textsc{Ind}_m$ and IP_m gadgets. When lifting to probabilistic parity decision trees, our result also gives tight bounds for the MAJ_m gadget using a trick described in Subsection 3.4.

Similarly to [6] we extend our result to give the same lower bound for the logarithm of the size of the randomized semi-structured communication protocol and to a version of communication complexity, in which Bob is allowed to send indicator functions of subspaces of his input bits.

Although our techniques (see below) is designed specifically for randomized case, we translate our results to deterministic case as well. Compared to [2] the deterministic version of our results apply to a wide range of gadgets.

As an immediate corollary, we have the same results (both randomized and deterministic) for lifting to parity decision trees. Compared to [6] the deterministic version of our result for parity decision trees uses linear diversity complexity measure instead of stifling. We discuss the comparison between these two measures below.

Our results can be used to obtain lower bounds on randomized parity decision tree complexity of Boolean functions. We provide a couple of examples of bounds we can obtain.

Linear diversity vs. stifling

A function $g\colon \{0,1\}^m \to \{0,1\}$ is *k-stifled* if for any subset $S \subseteq [m]$ of k input variables and any bit b we can fix all other variables in such a way that g evaluates to b no matter what the values of the variables in the subset S are.

The binary logarithm of linear diversity measure is greater than stifling at least for some functions. A notable example is IP_m function that is a common gadget in lifting results. Its linear diversity is maximal, but its stifling is just 1.

Our techniques

The proof of our results builds on so-called simulation argument in the style of [11, 4]. In this argument, given a randomized communication protocol $\mathbf{\Pi}$ computing $f \circ g^n$ of cost d, we build a randomized decision tree $\mathbf{T}$ computing f of depth $O(d/\log k)$. The tree $\mathbf{T}$ simulates $\mathbf{\Pi}$, querying the necessary information. Next we describe the simulation argument and then explain the new ideas.

For convenience we introduce the following notations:

- The gadget: it is convenient to consider gadgets of the form $g\colon [k] \times \{0,1\}^m \to \{0,1\}$, where $[k]$ corresponds to the inputs of Alice and $\{0,1\}^m$ corresponds to the inputs of Bob, and the linear diversity of g is k; that is, for convenience, we assume that for any fixed first input of g the resulting function on the second input is linear.
- The collection of all gadgets: $G := g^n : [k]^n \times (\{0,1\}^m)^n \to \{0,1\}^n$.
- The input to the i-th gadget: $(x_i, y_i) \in [k] \times \{0,1\}^m$.
- The whole inputs of Alice and Bob: $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$.

Simulation argument. The main idea is that $\mathbf{T}$ on input $z \in \{0,1\}^n$ simulates $\mathbf{\Pi}$ on a random input (x, y) that is distributed uniformly on $G^{-1}(z)$. Since $f \circ G(x, y) = f(z)$, $\mathbf{T}(z)$ will output $f(z)$ as long as the simulation of $\mathbf{\Pi}$ performed correctly.

The main difficulty in this approach is to simulate $\mathbf{\Pi}$ on $(x, y) \sim G^{-1}(z)$ without knowing z. To overcome this problem, it was shown in [11, 4] that the distribution $(x, y) \sim G^{-1}(z)$ does not differ substantially (from the perspective of the players) from the uniform distribution on all inputs. Thus, the tree $\mathbf{T}$ can instead simulate $\mathbf{\Pi}$ on (x, y), where (x, y) is distributed uniformly on $[k]^n \times (\{0,1\}^m)^n$, until $\mathbf{\Pi}$ reveals too much information about some block of inputs (x_i, y_i). Once this happens for some i, the tree $\mathbf{T}$ queries z_i and proceeds with the simulation knowing the correct distribution of the pair (x_i, y_i).

However, in this approach the simulation becomes approximate. To be able to assume that the uniform distribution on $G^{-1}(z)$ does not differ too much from the uniform distribution on all inputs, we need that the size of the gadget is at least logarithmic in n (we need this to bound the error probability of the simulation). Thus, it is not clear how to use this approach for gadgets of smaller size.

The key idea of our approach is to simulate $\mathbf{\Pi}$ precisely on the distribution $(x, y) \sim G^{-1}(z)$. This allows us to work with the gadgets of constant size. To address the issue of the simulation with unknown z, we introduce the main key ingredient of our argument, the *secret sets* technique.

Secret sets. Let S_i be some subset of $\{i\} \times [m]$. Assume for now that z_i equals the XOR of variables of y on the positions in S_i. Then we can show that as long as S_i is not in a linear span of linear functions sent by Bob in $\mathbf{\Pi}$, the uniform distribution on $g^{-1}(z_i)$ is indistinguishable (by players) from the uniform distribution on the whole i-th block of input. This allows us to simulate $\mathbf{\Pi}$ as if z_i is known. Once S_i falls into a linear span of functions sent by Bob, we just query z_i and proceed with the simulation. To prove our bound, we must show that Bob needs to send many (about $\log_2 k$) messages in $\mathbf{\Pi}$ on average to force us to query z_i. Recall that S_i is actually random. The intuition is that Bob must send roughly $\log_2 k$ linear functions for their linear span to capture a random S_i.

There are two key ingredients to actually prove that the secret set technique forces Bob to send many messages. Below is the brief description of them.

Narrowing Bob's messages to blocks. For each Bob's message we assign a block which will account for it. For each message assigned in block $\{i\} \times [m]$, we remove the part of it that lies outside of $\{i\} \times [m]$. Denote by L_i this set of truncated messages that assigned to ith block.

Each L_i will admit the following property. S_i is not lying in the linear span of messages sent by Bob as long as it is not lying in the linear span of L_i. At some point, the property can become violated after Bob sends a message, but we will send additional messages for Alice and Bob to restore the property. More precisely, we pick a linear combination of messages assigned to ith block that annulates S_i. We subtract S_i from this linear combination, in which we take untruncated messages, and send this new message for Bob recursively.

Entropy. We use the idea of *fixed/unfixed* blocks that was also used in the previous lifting theorems that utilized entropy. At the beginning, we consider all S_i to be unfixed. Note that, initially, the binary entropy of S_i is $\log_2 k$, since x_i is uniformly distributed on $[k]$. We want to show that S_i rarely lies in the linear span of L_i when a new element is added to L_i. The case when the entropy of S_i is sufficiently larger than $|L_i|$ is favorable for us, since in this case S_i does not lie in the linear span of L_i with a high probability. When the entropy of S_i goes below that level, we fix S_i (Alice sends x_i). Since Alice and Bob must send a lot of messages to decrease the entropy of S_i below the desired level or to increase the size of L_i, the number of fixed S_i is low.

As another feature of our approach we would like to mention that unlike the previous papers our simulation algorithm has a very simple description: we fix Alice's input randomly and maintain the linear space generated by Bob's messages to decide on querying z_is. Correctness of the algorithm easily follows from its description and the most technical part of the proof shifts to the complexity analysis.

For the deterministic version of our result we again use the simulation argument in the style of [11, 4] and more specifically, our general strategy is very similar to the one of [2] (with the necessary generalization to a wide range of gadgets).

Organization

The rest of the paper is organized as follows. In Section 2 we give the necessary preliminary information and introduce key notions and notation. In Section 3 we give a formulation our results. In Subsection 3.5 we show some applications of our results. In Section 4 we introduce additional notation that is used in the proofs. In Subsection 5.1 we begin the proof of our main result and as a first step reformulate the theorem in the form that is convenient for the proof. In Subsection 5.2 we describe the protocol to simulate communication protocol by decision tree. In Subsection 5.3 we proof correctness of the simulation. In Subsection 5.4 we prove the bound on the number of queries in the simulation (this is the most technically heavy part of the proof).

Proofs of the remaining theorems are omitted due to the space constraint and are provided in the full version of the paper. The proofs of our results on lifting to the size of the communication protocols and to the subspace queries are achieved by a slight modification of the proof for the depth complexity. The proof of our results for the deterministic case are similar to the ones for randomized setting with necessary technical modifications.

2 Preliminaries

2.1 Standard Notation

Here we will describe some notation that we use. We denote $[n] := \{1, 2, ..., n\}$ for a non-negative integer n. The addition mod 2 is denoted by XOR or $\oplus$. Sometimes, we view $\{0,1\}^n$ as a vector space $\mathbb{F}_2^n$.

2.2 Computational Models

For functions of the form $f\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ we consider *semi-structured communication protocols* introduced in [2]. In these protocols Bob is only allowed to send the XOR of some subset of his input bits (and Alice is not restricted). A randomized semi-structured protocol is just a distribution over deterministic semi-structured protocols. We say that such a protocol computes the function f correctly, if on every input the probability of the correct output is at least $\frac{2}{3}$. The complexity $\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f)$ of f in this model is the minimal depth of a protocol computing f.

We also consider a subspace-query model. Consider communication protocols in which Bob is only allowed to send indicators of whether his input lies in an affine subspace of $\mathbb{F}_2^m$. We denote by $\mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(f)$ the minimum complexity of a randomized protocol computing f, where the protocol is taken from the restricted class.

Additionally, let us introduce a size-complexity of a protocol. A deterministic communication protocol can be represented by a tree in which in every node either Alice or Bob sends a message. We call the size-complexity of the protocol to be the number of leafs in this tree. Denote by $\mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(f)$ the minimum size-complexity of a randomized *semi-structured* protocol computing f.

We denote by $\mathsf{D}^{\mathsf{cc}}_{\to\oplus}(f)$, $\mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}(f)$ and $\mathsf{sD}^{\mathsf{cc}}_{\to\oplus}(f)$ the deterministic versions of these complexity measures.

We can define the semi-structured complexity measures for relations $f \subseteq (\mathcal{X} \times \{0,1\}^m) \times \mathcal{R}$ completely analogously. Here a deterministic protocol Π is said to compute f if for any $(x, y) \in \mathcal{X} \times \{0,1\}^m$ it outputs any z such that $(x, y, z) \in f$, if there is such z (the protocol can give any output if there is no such z).

For a function $f\colon \{0,1\}^n \to \{0,1\}$ we denote by $\mathsf{D}^{\mathsf{dt}}(f)$ the minimal depth of a decision tree computing f. We denote by $\mathsf{D}^{\mathsf{dt}}_{\oplus}(f)$ the minimal depth of a parity decision tree computing f (on each step such a decision tree can query an XOR of a subset of the input bits). Analogously to semi-structured communication complexity, we denote by $\mathsf{D}^{\mathsf{dt}}_{\oplus}(f)$, $\mathsf{sizeD}^{\mathsf{dt}}_{\oplus}(f)$, $\mathsf{sD}^{\mathsf{dt}}_{\oplus}(f)$, $\mathsf{R}^{\mathsf{dt}}_{\oplus}(f)$, $\mathsf{sizeR}^{\mathsf{dt}}_{\oplus}(f)$ and $\mathsf{sR}^{\mathsf{dt}}_{\oplus}(f)$ complexity measures defined by deterministic and randomized parity decision trees.

There is the following standard connection between parity decision trees and communication complexity protocols.

▶ **Lemma 1.** *For any function* $f\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ *we have*

$$\mathsf{D}^{\mathsf{cc}}_{\to\oplus}(f) \le 2\mathsf{D}^{\mathsf{dt}}_{\oplus}(f),$$

$$\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f) \le 2\mathsf{R}^{\mathsf{dt}}_{\oplus}(f).$$

Proof. Given a (randomized) parity decision tree for f Alice and Bob can use it to compute the function f by a (randomized) communication protocol. For this they simulate each query one by one computing XOR of their portion of the input and sending them to each other. Simulation of each query requires two bits of communication. ◀

2.3 Gadgets

We first define a class of gadgets that is of use for us.

▶ **Definition 2** (Family of linear functions). *The gadget $g : [k] \times \{0,1\}^m \to \{0,1\}$ is a family of linear functions of order k, if the following is true*

(1) $\forall i \in [k]$ $g(i,\cdot)$ *is a non-trivial linear function as a function of the second argument (that is, it is an XOR of a nonempty subset of its inputs),*

(2) $\forall i,j \in [k], i \neq j$ $g(i,\cdot) \neq g(j,\cdot)$.

For convenience from now on we will use the notation $g_i(y) := g(i,y)$.

We will use the following notion of gadget reduction.

▶ **Definition 3** (Gadget reduction). *Consider two gadgets $g : \mathcal{X} \times \{0,1\}^m \to \{0,1\}, h : \mathcal{Y} \times \{0,1\}^n \to \{0,1\}$. Then g is reducible to h, if there are mappings $\varphi : \mathcal{X} \to \mathcal{Y}, \psi : \{0,1\}^m \to \{0,1\}^n$, such that*

(1) $\forall x \in \mathcal{X}, y \in \{0,1\}^m$ $g(x,y) = h(\varphi(x), \psi(y))$

(2) ψ *is linear, that is* $\forall y_1, y_2 \in \{0,1\}^m$ $\psi(y_1 \oplus y_2) = \psi(y_1) \oplus \psi(y_2)$

Note that gadget reduction relation is transitive.

Gadget reduction is useful for us due to the following lemma.

▶ **Lemma 4.** *Assume that a gadget g reduces to a gadget h. Then for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$ we have*

$$\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) \leq \mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n),$$

$$\mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) \leq \mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n).$$

$$\mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) \leq \mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n).$$

The same inequalities are true for the deterministic complexities.

Proof. If Alice and Bob would like to compute $f \circ g^n$, they can just compute the mappings $\varphi : \mathcal{X} \to \mathcal{Y}, \psi : \{0,1\}^m \to \{0,1\}^n$ on their inputs individually and use the protocol for $f \circ h^n$. Since $\psi : \{0,1\}^m \to \{0,1\}^n$ is linear, Bob can simulate the protocol for $f \circ h^n$ sending only XORs of his input bits. To see that, denote by $e_i := \{0\}^{i-1} \times \{1\} \times \{0\}^{m-i}$. Thus, ψ is uniquely determined by $\psi(e_1), \ldots, \psi(e_m)$. Let x be Bob's input, and let $y = \psi(x)$. An arbitrary parity message of y can be represented as $\langle y, y' \rangle$ for some y'. Here, $\langle \cdot, \cdot \rangle$ is the dot product modulo 2. Observe that $\langle y, y' \rangle = \langle \psi(x), y' \rangle = \sum_i x_i \cdot \langle \psi(e_i), y' \rangle$. In other words, to compute $\langle y, y' \rangle$, it is enough to take the XOR of all those x_i for which $\langle \psi(e_i), y' \rangle$ equals one. This means that Bob can translate parity messages of y to parity messages of x. ◀

Next we define the main complexity measure for the gadgets.

▶ **Definition 5** (Linear diversity). *We let linear diversity of a function $h : \mathcal{Y} \times \{0,1\}^n \to \{0,1\}$ be the maximal k such that there is a family of linear functions $g : [k] \times \{0,1\}^m \to \{0,1\}$ of order k such that g reduces to h.*

Next we introduce a couple of standard gadgets that will be important for us.

▶ **Definition 6** (Index function). *Let* $\textsc{Ind}_m : [m] \times \{0,1\}^m \to \{0,1\}$ *be the function that on input* (i, y) *outputs* y_i.

▶ **Definition 7** (Inner product function). *Let* $IP_m : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ *be the function that on input* (x, y) *outputs* $\bigoplus_{i=1}^{n}(x_i \wedge y_i)$.

▶ **Lemma 8.** $\textsc{Ind}_m$ *is a family of linear functions of order* m. IP_m *has linear diversity* $2^m - 1$.

Proof. The statement of the lemma is almost immediate. For $\textsc{Ind}_m$, note that for any $i \in [m]$ the output of $\textsc{Ind}_m$ is y_i, which is a linear function. For IP_m, note that for any fixed $x \in \{0,1\}^n$ the output of IP_m is $\oplus_{i:x_i=1} y_i$, which is a non-trivial linear function for each $x \neq 0$. The reduction from a family of linear functions is trivial (ψ is an identity function and φ sends an integer to its binary representation). ◀

▶ **Lemma 9.** *With probability approaching* 1 *as* m *tends to infinity, a random gadget* $g : [2^{2^m}] \times \{0,1\}^m \to \{0,1\}$ *has linear diversity at least* $2^{m/2}$. *(The values of* g *on each input are chosen randomly and independently.)*

The proof of this lemma can be found in the full version of the paper.

3 Results Statement

3.1 Randomized Semi-Structured protocols

▶ **Theorem 10.** *Let* $g : [k] \times \{0,1\}^m \to \{0,1\}$ *be a family of linear functions of order* $k \geq 2$. *Then for any relation* $f \subseteq \{0,1\}^n \times \mathcal{R}$ *we have*

$$\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) \;=\; \Theta(\log_2 k \cdot \mathsf{R}^{\mathsf{dt}}(f)).$$

We note that big-O part is trivial, since Alice and Bob in communication protocol can simulate a decision tree for f and spend $O(\log_2 k)$ bits of communication for each tree query to compute the function g on the corresponding inputs.

We prove the following stronger versions of Theorem 10.

▶ **Theorem 11.** *Let* $g : [k] \times \{0,1\}^m \to \{0,1\}$ *be a family of linear functions of order* $k \geq 2$. *Then for any relation* $f \subseteq \{0,1\}^n \times \mathcal{R}$ *we have*

$$\log_2 \mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) \;=\; \Theta(\log_2 k \cdot \mathsf{R}^{\mathsf{dt}}(f)).$$

▶ **Theorem 12.** *Let* $g : [k] \times \{0,1\}^m \to \{0,1\}$ *be a family of linear functions of order* $k \geq 2$. *Then for any relation* $f \subseteq \{0,1\}^n \times \mathcal{R}$ *we have*

$$\mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) \;=\; \Theta(\log_2 k \cdot \mathsf{R}^{\mathsf{dt}}(f)).$$

For both theorems big-O part follows since $\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(h) \geq \log_2 \mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(h)$ and $\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(h) \geq \mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(h)$ for any function h.

As a corollary of these theorems and Lemma 4 we immediately obtain the following.

▶ **Corollary 13.** *Let* $h : \mathcal{Y} \times \{0,1\}^n \to \{0,1\}$ *have linear diversity at least* k *for* $k \geq 2$. *Then for any relation* $f \subseteq \{0,1\}^n \times \mathcal{R}$ *we have*

$$\mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n) \;=\; \Omega(\log_2 k \cdot \mathsf{R}^{\mathsf{dt}}(f)).$$

The same result is true for $\log \mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n)$ *and* $\mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n)$.

3.2 Deterministic Semi-structured protocols

We translate our results to deterministic case as well.

▶ **Theorem 14.** *Let $g : [k] \times \{0,1\}^m \to \{0,1\}$ be a family of linear functions of order $k \geq 2$. Then for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$ we have*

$$\begin{aligned} \mathsf{D}^{\mathsf{cc}}(f \circ g^n) &= \Theta(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f)), \\ \log_2 \mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) &= \Theta(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f)), \\ \mathsf{sD}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n) &= \Theta(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f)). \end{aligned}$$

▶ **Corollary 15.** *Let $h : \mathcal{Y} \times \{0,1\}^n \to \{0,1\}$ have linear diversity at least k for $k \geq 2$. Then for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$ we have*

$$\mathsf{D}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n) = \Omega(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f)).$$

The same result is true for $\log \mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n)$ and $\mathsf{sD}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n)$.

3.3 Parity decision trees

Finally, we prove that our results imply the same results for parity decision trees instead of semi-structured communication protocols.

▶ **Theorem 16.** *Let $h : \mathcal{Y} \times \{0,1\}^n \to \{0,1\}$ have linear diversity at least k for $k \geq 2$. Then for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$ we have*

$$\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ h^n) = \Omega(\log_2 k \cdot \mathsf{R}^{\mathsf{dt}}(f)).$$

The same is true for $\log \mathsf{sizeR}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n)$ and $\mathsf{sR}^{\mathsf{cc}}_{\to\oplus}(f \circ g^n)$. The same results are also true for the deterministic complexities.

The part of this theorem concerning $\mathsf{R}^{\mathsf{dt}}_{\oplus}$ is a direct consequence of Lemma 1, Theorem 10, and Lemma 4. The same applies to $\mathsf{D}^{\mathsf{dt}}_{\oplus}$.

The part of this theorem about $\log \mathsf{sizeR}^{\mathsf{dt}}_{\oplus}$ and $\mathsf{sR}^{\mathsf{dt}}_{\oplus}$ does not follow from previous theorems that easily, and we prove them in the full version of the paper.

3.4 More powerful gadget reduction

In this section, we describe a more general version of a gadget reduction than in Definition 3. We apply the reduction to the MAJ gadget, obtaining tight bounds for lifting to parity decision trees.

More specifically, we can consider the domain $\{0,1\}^m$ of a gadget as a vector space $\mathbb{F}_2^m$ and input variables $x_1, \ldots, x_m$ as coordinates in the standard basis. The idea is that we can switch to another basis in this vector space, consider new coordinates as variables and apply the reduction in Definition 3 after that. Since the transformation to the new variables is linear, new variables can be expressed as an XOR of old variables and vice versa. Thus, the parity decision trees in new and old variables can simulate each other (this does not hold for communication complexity setting, since switching to the new basis mixes up the variables belonging to Alice and Bob).

We illustrate this idea on a MAJ gadget. Recall that MAJ_m is a function that returns 1 iff at least $m/2$ of its m variables are equal to 1.

▶ **Lemma 17.** *For any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$ and $m \geq 4$, $\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ MAJ^n_m) = \Omega(m \cdot \mathsf{R}^{\mathsf{dt}}(f))$.*

This method can be applied to gadgets other than MAJ. We formulate this approach in a theorem.

▶ **Theorem 18.** *Let $r : \{0,1\}^m \to \{0,1\}$, $h : \mathcal{X} \times \{0,1\}^l \to \{0,1\}$, $g : \mathcal{Y} \to \{0,1\}$ – be functions such that linear diversity of h is at least $k \geq 2$. Suppose that $r \circ h^m$ reduces to g after a change of basis. Then, for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$,*

$$\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ g^n) = \Omega(\mathsf{R}^{\mathsf{dt}}(f \circ r^n) \log k)$$

By reduction we mean Definition 3. Here g depends on x, and we consider some linear change of basis $x \to y$. We also decide on some way to split variables y between Alice and Bob before applying Definition 3.

Proofs of these results are provided in the full version of the paper.

3.5 Applications

A more detailed description of applications can be found in the full version of the paper.

Recursive majority function

Let $\mathrm{MAJ}_3^{\otimes 1} := \mathrm{MAJ}_3$ be the majority function that returns 1 if at least two of its three input bits equal to 1, otherwise it returns 0.

For $k > 1$, define $\mathrm{MAJ}_3^{\otimes k}$ to be MAJ_3 in which each argument is substituted by $\mathrm{MAJ}_3^{\otimes k-1}$. Thus, $\mathrm{MAJ}_3^{\otimes k}$ takes 3^k inputs.

In [15], it is shown that

$$\Omega(2.57143^k) \leq \mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}_3^{\otimes k}) = O(2.64944^k).$$

It is easily observed that $\mathrm{MAJ}_3^{\otimes 2}$ is at least 2 linear diverse. By using it as a gadget, our lifting theorem imply that $\mathsf{R}^{\mathsf{dt}}_{\oplus}(\mathrm{MAJ}_3^{\otimes k}) = \Omega(\mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}_3^{\otimes k}))$. Thus, parity queries do not help in computing $\mathrm{MAJ}_3^{\otimes k}$ as compared to single variable queries.

The same approach can be applied to formulas having the form of complete binary AND-OR-tree. In other words, this function is obtained by repeated iteration of $f(x) = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$. It was shown in [19] that R^{dt} of this function is at least $n^{0.7537\ldots}$, where n is the number of inputs. Since f can be used as a gadget, we translate this result to parity decision trees.

Quantum Complexity

Our lifting theorem can be applied to exhibit a separation between randomized parity decision tree complexity and bounded-error quantum complexity. Let $k \leq \log n$. It was shown in [1] that there is a $\lceil k/2 \rceil$ versus $\widetilde{\Omega}(n^{1-\frac{1}{k}})$ separation between the quantum and randomized query complexity. The separation was shown for a partial function f called k-fold Forrelation.

Using our result, we can lift this separation to randomized parity query complexity. Consider the function $f \circ \textsc{Ind}_2^n$. On the one hand, its quantum complexity is the same as the quantum complexity of f (up to a factor), since a quantum protocol can extract inputs to f from $\textsc{Ind}_2$ in constant number of additional queries. On the other hand, by Theorem 10, the randomized parity query complexity of $f \circ \textsc{Ind}_2^n$ is no less than its randomized query complexity. Hence, we obtain the same separation even if our query model can ask parity queries.

An alternative method to obtain separation between randomized parity query and quantum complexity was given in [3]. They also relied on the result of [1]. They used Fourier analysis and properties of k-Fold Forrelation to derive their result. In contrast, our lifting theorem can lift any function f that exhibits the separation.

4 Notation for proving main theorems

In this section we introduce notation and facts about entropy that will be used for proving the main theorems in the subsequent sections. Recall that in a semi-structured communication protocol, Bob can send only parities of its input. In a lifting setting, Bob is given $n \cdot m$ boolean variables. For each of n variables of the initial function, there are m gadget's variables. To analyze Bob's messages, we introduce the following definitions.

4.1 Notation

▶ **Definition 19** (XOR of a subset). *For a set $S \subseteq [m]$ and $y \in \{0,1\}^m$, we define*

$$y_S := \bigoplus_{j \in S} y_j.$$

Analogously, we define y_S if $y \in (\{0,1\}^m)^n$ and $S \subseteq [n] \times [m]$.

If $y \in (\{0,1\}^m)^n$ is the Bob's input, then each of his messages can be represented as y_S for some $S \subseteq [n] \times [m]$.

▶ **Definition 20** (Subsets of $[n] \times [m]$ as a linear space). *We view subsets of $[n] \times [m]$ as vectors in a linear space over $\mathbb{F}_2$ (where addition (+) corresponds to symmetric difference).*

With this notation, for any $S, T \subseteq [n] \times [m], y \in (\{0,1\}^m)^n$, $y_{S+T} = y_S \oplus y_T$.

▶ **Definition 21** (Linear Order on $[n] \times [m]$). *We introduce a lexicographic linear order on $[n] \times [m]$. A pair $(i,j) \in [n] \times [m]$ is said to be lower than (i',j') if $i < i'$ or $i = i', j < j'$.*

▶ **Definition 22** (Principal variable of a non-empty subset). *For a non-empty set $S \subseteq [n] \times [m]$, denote by $p(S)$ its lowest element.*

▶ **Definition 23** (Block). *We refer to $\{i\} \times [m]$ as i-th block. A non-empty subset $S \subseteq [n] \times [m]$ touches i-th block if $p(S) \in \{i\} \times [m]$.*

▶ **Definition 24** (Bernoulli distribution). *Denote by $Bern(p)$ a distribution of a random variable that takes value 0 with probability $1-p$ and 1 with probability p.*

▶ **Definition 25** (Entropy). *For a random variable x, denote by $H(x)$ its binary entropy. If X is a set, then we define its entropy as $H(X) := \log_2 |X|$. If p is a number, then $H(p)$ denotes the binary entropy of the distribution $Bern(p)$.*

Recall that the binary entropy of a random variable taking n values with non-zero probabilities $p_1, ..., p_n$ equals to

$$\sum_{i=1}^{n} -p_i \log_2 p_i$$

4.2 Entropy theorems

We state well-known results that will be used for proving our main theorems.

▶ **Lemma 26** (Gibbs' inequality [7]). *Let $0 \le p \le 1, 0 < q < 1$. Then,*

$$H(p) \le p \log_2 \frac{1}{q} + (1-p) \log_2 \frac{1}{1-q}$$

▶ **Lemma 27** (Fano's Inequality [7]). *Let X, Y be random variables. Let $\varepsilon = P(X \neq Y) \le 1/2$. Then,*

$$H(X|Y) \le H(\varepsilon) + \varepsilon \log_2(|\mathcal{X}| - 1),$$

where $\mathcal{X}$ denotes the support of X.

5 Proof of Theorem 10

5.1 Reformulation of Theorem 10

As we noted in Section 3, the $\le$-direction is simple. Thus, it remains to prove the other direction.

Let $\mathbf{\Pi}$ be a randomized semi-structured protocol computing $f \circ g^n$. Denote by d the depth of $\mathbf{\Pi}$. Our goal is to construct a randomized tree $\mathbf{T}$ of depth $O(d/\log_2 k)$, that computes $f(z)$ for any given z.

The idea is to simulate $\mathbf{\Pi}$ on a randomly uniformly chosen pair (x, y) satisfying $g^n(x, y) = z$. For convenience, denote $P_z = \{(x, y) \mid g^n(x, y) = z\}$. For any pair $(x, y) \in P_z$ we have that $\mathbf{\Pi}(x, y) = f(z)$ with probability at least $2/3$. Thus, if we sample $(x, y) \sim U(P_z)$, the protocol $\mathbf{\Pi}(x, y)$ will also be equal to $f(z)$ with probability at least $2/3$.

We use the following general strategy for $\mathbf{T}$: sample $(x, y) \sim U(P_z)$, sample $\Pi \sim \mathbf{\Pi}$ (recall that $\mathbf{\Pi}$ is a random distribution on deterministic protocols), and output $\Pi(x, y)$. Note that the choice of the pair (x, y) is independent from the choice of Π, and thus it does not matter in which order we sample Π and (x, y). As a result, what is left to do is to simulate the given deterministic semi-structured protocol Π on a random pair $(x, y) \sim U(P_z)$.

We are going to prove the following intermediate theorem.

▶ **Theorem 28.** *Let Π be a deterministic semi-structured protocol with inputs from $[k]^n \times (\{0,1\}^m)^n$. Let d be equal to the depth of Π. Then, there exists a randomized decision tree $\mathbf{T}$ which, on input $z \in \{0,1\}^n$, outputs a random variable that is destributed as $\Pi(x, y)$ for $(x, y) \sim U(P_z)$. The expected number of queries to z made by $\mathbf{T}$ is $O(d/\log_2 k)$, where the expectation is taken over (x, y) for a fixed z.*

Before proceeding to the proof of Theorem 28 we show how to prove Theorem 10 based on Theorem 28.

Proof of Theorem 10. The computation of f on input z proceeds as follows. We choose a random $\Pi \sim \mathbf{\Pi}$ and run $\mathbf{T}$ provided by Theorem 28 on input z. By definition, $\mathbf{T}(z)$ has the same distribution as $\mathbf{\Pi}(x, y)$ for $(x, y) \sim U(P_z)$. Thus, $\mathbf{T}$ computes $f(z)$ with probability at least $\frac{2}{3}$.

The average number of queries made by $\mathbf{T}$ is at most $O(d/\log_2 k)$. To achieve this number of queries in the worst case, we halt $\mathbf{T}$ if it makes 10 times more queries than the expected number. By Markov's inequality, this only happens with probability at most $\frac{1}{10}$, the probability of the correct answer is still a constant greater than $1/2$ and it can be increased to $2/3$ by the standard argument for error reduction. ◀

Now we proceed to the proof of Theorem 28. For this we need to describe how $\mathbf{T}$ simulates Π. In the subsequent sections we will heavily use the notation introduced in Section 4.

5.2 Simulation algorithm for T

In this section we describe the algorithm for $\mathbf{T}$.

$\mathbf{T}$ starts by sampling a random $a \sim U([k]^n)$ and assuming that $x = a$.

After that, $\mathbf{T}$ starts the simulation of Π. Since x is fixed ($\mathbf{T}$ knows x but for Π it is a random variable), Alice's messages are easily simulated. Next, we describe how $\mathbf{T}$ simulates Bob's parity messages on the (unknown) variables y.

Recall that we can view g as a family of linear functions $\{g_1, \dots, g_k\}$ of order k (one function for each value of x). Since $g_1, ..., g_k$ are linear functions, we can represent them as $g_j(x) = x_{G_j}$ for some $G_j \subseteq [m]$.

Let $S_i := \{i\} \times G_{x_i}$. Note that all S_is are linearly independent since they are in distinct blocks. From now on we will call $S_1, ..., S_n$ *the secret sets*.

Consider an arbitrary step of simulation and assume Bob has already sent the parities of his inputs for the sets $Q_1, ..., Q_t \subseteq [n] \times [m]$ and is now supposed to send the parity of y's in the set $Q_{t+1} \subseteq [n] \times [m]$. Note, that it can be assumed that Q_{t+1} is linearly independent of $Q_1, ..., Q_t$: otherwise, the message Q_{t+1} does not reveal any new information and can be omitted from the protocol. There are two cases:

(1) There is a linear combination of Q_is that includes Q_{t+1} and equals to some linear combination of the secret sets:

$$Q_{i_1} + Q_{i_2} + ... + Q_{i_k} + Q_{t+1} = S_{j_1} + ... + S_{j_l}.$$

In this case, the value $y_{Q_{t+1}}$ is uniquely determined since

$$y_{Q_{t+1}} = y_{Q_{i_1}} + y_{Q_{i_2}} + ... + y_{Q_{i_k}} + z_{j_1} + ... + z_{j_l}.$$

The y's parities on the sets $Q_{i_1}, ..., Q_{i_k}$ are already known from the previous messages of Bob. Therefore, $\mathbf{T}$ queries $z_{j_1}, ..., z_{j_l}$ (if it hasn't already), and we calculate the value $y_{Q_{t+1}}$ that Bob sends in this vertex.

(2) Such a linear combination does not exist. In this case, $\mathbf{T}$ sends a random bit $Bern(1/2)$ as an XOR of y on the set Q_{t+1}. In terms of the protocol Π, this corresponds to proceeding to one of the left and right children with equal probabilities.

Thus, we have described how to simulate each Bob's message. Once $\mathbf{T}$ reaches a leaf of Π, it outputs the value written in this leaf.

5.3 Correctness of the simulation of Π done by T

Next, we need to show that $\mathbf{T}(z)$ indeed simulates Π on a random pair $(x, y) \sim U(P_z)$, and that $\mathbf{T}(z)$ makes $O(d/\log_2 k)$ queries on average. We start with the correctness of the simulation.

It is easy to see that for any z the projection of $U(P_z)$ onto x is the uniform distribution on $U([k]^n)$, therefore fixing $x = a \sim U([k]^n)$ correctly corresponds to the distribution of (x, y) we would like to simulate the protocol on.

Note that once x is fixed the only constraints on y are of the form $g_j(y_i) = z_i$, where $j = x_i$. In particular, any variable in y, not included in $\{i\} \times G_{x_i}$, has a $Bern(1/2)$ distribution.

Next, we justify why the algorithm can send $Bern(1/2)$ as an XOR of Q_{t+1} if no linear combination exists (Case 2 above). Indeed, the y's parities of $Q_1, ..., Q_t, S_1, ..., S_n$ define an affine subspace in our linear space. Since Q_{t+1} is linearly independent of $Q_1, ..., Q_t, S_1, ..., S_n$, each of the conditions $y_{Q_{t+1}} = 0$ and $y_{Q_{t+1}} = 1$ is satisfied by exactly half of the points of the affine subspace.

If there exists a linear combination of Q_is that includes Q_{t+1} and equals a linear combination of S_is (Case 1 above), then given the previous messages and the value of z there is only one possible value for $y_{Q_{t+1}}$, which $\mathbf{T}$ indeed sends in our simulation.

Thus, the distribution of $\mathbf{T}(z)$ matches $\Pi(x, y)$ for $(x, y) \sim U(P_z)$.

5.4 Upper bound on the average number of queries made by T

Next we show that $\mathbf{T}(z)$ makes $O(d/\log_2 k)$ queries to the variables z on average. For this we introduce some complexity measures and study their behaviour during the execution of the protocol. To make this analysis cleaner we first modify the protocol Π to add more messages to it. This does not change the output of the protocol, but it allows us to simplify the exposition of time analysis. In the next section we describe the modified protocol. Next we discuss its connection to $\mathbf{T}$. Finally, in Subsubsection 5.4.3, we derive the upper bound on the number of queries made by $\mathbf{T}$.

Since we need to show the upper bound on the number of queries of $\mathbf{T}(z)$ for any z, we fix z for the whole argument.

5.4.1 Description of the modified protocol $\overline{\Pi}_z$

Here we describe $\overline{\Pi}_z$, a refined version of Π. Note that the protocol depends on z, which is fixed throughout the whole argument. We will be interested only in the behaviour of the protocol on inputs in P_z.

We modify the protocol in two ways. First, we modify the messages sent by Bob into equivalent messages. This does not actually change the information transmitted by Bob, this modification is needed only for the purpose of the analysis of the protocol. Next, at some moments of time we add additional messages sent by the players. The information transmitted in these messages is not affecting the following messages in the protocol. One can think of this in the following way: at some points of the protocol the players pause the execution of the protocol, exchange some extra information, and then resume the execution of the protocol as if nothing happened. These extra messages are needed just to introduce new intermediate vertices in communication tree that will allow us to analyse the complexity of $\mathbf{T}$ in a cleaner way.

The pseudocode for the algorithm $\overline{\Pi}_z$ is provided in Figure 1. Next we describe the protocol and introduce some important notation related to it.

First of all, observe that if Bob sends b_1, b_2 as parities on the sets $Q_1, Q_2 \subseteq [n] \times [m]$ respectively, this is equivalent to sending b_1, $b_1 \oplus b_2$ as parities on $Q_1, Q_1 + Q_2$ respectively. More generally, applying an invertible linear transformation to Bob's messages does not change the information transmitted.

Recall that by $S_i = \{i\} \times G_{x_i}$ we denote the secret sets. Note that S_i depends on x and, thus, initially is only known to Alice. Initially, all S_i are *unrevealed*, and over the course of the simulation, they will be gradually *revealed*. We also introduce a separate notion that of S_i being *fixed*. Initially, all S_i are *unfixed* and then will change status to fixed over the course of the simulation. A revealed S_i will also be fixed, but not necessarily vice versa.

Suppose that at the current moment of simulating Π, Bob has sent parities on the sets $Q_1, ..., Q_t \subseteq [n] \times [m]$ and now he wants to send the parity on the set $Q \subseteq [n] \times [m]$. We will maintain the *principle variable invariant*: all $p(Q_i)$ are distinct and, if $i < j$, then $p(Q_i)$ is *lower* than $p(Q_j)$. The variables $p(Q_i)$ will be referred to as *principal*. In particular, from this invariant, it follows that $Q_1, ..., Q_t$ are linearly independent.

Define the procedure *sift*, which will transform Q to an equivalent message. The procedure *sift* iterates over $i = 1, \dots, t$ and replaces Q with $Q + Q_i$ if $p(Q_i) \in Q$. Note that after this procedure for any i we have that $p(Q_i) \notin Q$.

We run the procedure *sift* on Q. If it turns into an empty set, then Bob's message does not provide any new information. In this case we finish its processing and move on to further simulation of Π. If the resulting Q is not empty, then it contains a principal variable. Since after *sift*, Q does not contain principal variables of previous messages, the principal variable of Q does not coincide with the principal variable of previous messages. We insert a copy of Q into the sequence $Q_1, ..., Q_t$ in such a way that the principle variable invariant is maintained. That is, now the length of the sequence is $t + 1$. Assume that $p(Q)$ is in the i-th input block. Let $L_i := \{Q_j \cap \{i\} \times [m] \mid p(Q_j) \in \{i\} \times [m]\}$. In other words, we consider all sets whose principal variables are in the i-th block and intersect them with the i-th block. Note that sets in L_i are linearly independent. Denote by $\mathcal{L}_i$ the linear span of L_i with the zero vector (empty set) removed.

At this point, we apply the second modification to the protocol. If the binary entropy of S_i is less than $\frac{1}{10}\log_2 k + \log_2 |\mathcal{L}_i| + \frac{1}{3}$, Alice sends S_i, and S_i is considered fixed. Here the protocol views S_i to be a random variable of the distribution $(x, y) \sim U(P_z)$ conditioned on the information about (x, y) that the protocol has learnt so far. Note that we fixed z in advance, and we assume that the inputs given to the players are indeed in P_z (we are not interested in the behaviour of the protocol on other inputs). As a result, both players can compute the entropy of S_i and compare it to $\frac{1}{10}\log_2 k + \log_2 |\mathcal{L}_i| + \frac{1}{3}$ without communication.

After that Alice sends a message indicating if S_i lies in $\mathcal{L}_i$[1]. If this is not the case or if S_i has already been revealed on one of the previous steps, we finish processing the message Q and proceed with the further simulation of the protocol Π. If S_i lies in $\mathcal{L}_i$, then we consider S_i to be revealed. In this case Alice sends S_i, and we fix S_i if it was not already fixed before. Bob sends y_{S_i}. Next, let $Q_{j_1}, ..., Q_{j_l}$ be the sets whose linear combination, when intersected with the i-th block, equals the secret set S_i. The set Q must be present among them: otherwise, S_i would have been revealed at an earlier step. Without loss of generality we can assume that $Q_{j_1} = Q$. We update Q to $Q \leftarrow Q + Q_{j_2} + ... + Q_{j_l} + S_i$ and perform the same procedure with the updated Q starting with *sift* (note that from this players can compute y_Q for the new Q without additional communication since y_{S_i} is known). Note, that during this iteration we removed from Q all elements in the i-th block without introducing anything to Q in the previous blocks (all sets in the combinations had their principle variables in the i-th block). As a result, the updated Q lies within the blocks that are higher than i-th block and the whole procedure of updating Q will be finished eventually.

5.4.2 Connection between T and $\overline{\Pi}_z$

The protocol $\overline{\Pi}_z$ is useful for upper bounding the complexity of $\mathbf{T}$ due to the following lemmas.

[1] Note that this might be redundant if we just communicated the whole S_i. We choose to keep this step even if it is redundant to make the pseudocode in Figure 1. In the analysis below the redundancy of these messages is reflected in Observation 35.

Refined protocol $\overline{\Pi}_z$ ***on input*** $(x, y) \sim U(P_z)$**:**

1: Initialize: v = root of Π, $Q_1, ..., Q_t \subseteq [n] \times [m]$ – sets which parities Bob sends, initially $t = 0$
2: **while** v is not a leaf [invariant: $p(Q_i)$ is lower than $p(Q_j)$ when $i < j$]
3: Let v_0, v_1 be children of v
4: **if** Bob speaks at v **then**
5: Let $Q \subseteq [n] \times [m]$ be the set which parity Bob sends at v
6: Let $b = y_Q$
7: ▷ Bob sends b and we update $v \leftarrow v_b$ ▷ (B1)

8: **for** $i = 1..t$ **do**
9: **if** $p(Q_i) \in Q$ **then**
10: $Q \leftarrow Q + Q_i$
11: **end if**
12: **end for**
13: **if** $Q \neq \varnothing$ **then**
14: Insert a copy of Q in $Q_1, ..., Q_t$ so that invariant holds
15: Let $i \in [n]$ be the block containing $p(Q)$
16: // $L_i = \{Q_j \cap \{i\} \times [m] \mid p(Q_j) \in \{i\} \times [m]\}$
17: // $\mathcal{L}_i$ denotes linear span of L_i without null vector
18: // $S_i = \{i\} \times G_{x_i}$ – a *secret* set in ith block
19: **if** $H(S_i) < \frac{1}{10} \log_2 k + \log_2 |\mathcal{L}_i| + \frac{1}{3}$ **then**
20: ▷ Alice sends S_i; S_i is *fixed* now ▷ (A3)
21: **end if**
22: ▷ Alice communicates whether S_i is in $\mathcal{L}_i$ ▷ (A2)
23: **if** $S_i \in \mathcal{L}_i$ and S_i is not *revealed* **then**
24: ▷ Alice sends S_i; S_i is *fixed* now ▷ (A3)
25: S_i is considered to be *revealed*
26: ▷ Bob sends y_{S_i} ▷ (B2)
27: Find distinct $j_1, ..., j_l$, s.t. $Q = Q_{j_1}$ and $(Q_{j_1} + ... + Q_{j_l}) \cap \{i\} \times [m] = S_i$
28: $Q \leftarrow Q + Q_{j_2} + ... + Q_{j_l} + S_i$
29: Go to 8−th line
30: **end if**
31: **end if**

32: **else** Alice speaks at v
33: Let b be the bit she sends
34: ▷ Alice sends b and we update $v \leftarrow v_b$ ▷ (A1)
35: **end if**
36: **end while**
37: return the value of the leaf v

Figure 1 The modified (deterministic) protocol $\overline{\Pi}_z$. The original protocol Π can be recovered by ignoring lines 8–31 and the **red** text. Lines 8–31 are used to maintain the invariant and prove the estimate on the number of *revealed* sets. The classification (A1), (A2), (A3), (B1), (B2) of the actions made by Alice and Bob is used in Section 5.4.3. Note that Alice can send more than 1-bit of information for the message of type (A3).

▶ **Lemma 29.** *The following is a while-loop invariant in $\overline{\Pi}_z$: If a linear combination of $Q_1, ..., Q_t$ equals a linear combination of $S_1, ..., S_n$, then all S_is in this linear combination are revealed.*

Proof. First note that for all i such that S_i is unrevealed, S_i does not lie in $\mathcal{L}_i$. Otherwise, at the line 25 of the algorithm, S_i would have been revealed.

Assume that this invariant is violated. Let this linear combination be $Q_{j_1} + ... + Q_{j_l} = I$, where $j_1 < ... < j_l$. If there are many linear combinations that violate invariant, then choose the one with the greatest j_1. Let i be the block of the variable $p(Q_{j_1})$. Then, $I \cap \{i\} \times [m]$ must be equal to S_i, since $I \cap \{i\} \times [m] \neq \varnothing$. It follows that S_i must be revealed, since it lies in $\mathcal{L}_i$. At the line 28 of the algorithm, it is ensured that a linear combination $Q_{j_1} + ... + Q_{j_l} + S_i = I + S_i$ is added to the set of Qs. $I + S_i$ is contained in blocks strictly higher than i-th block and, by assumption, is equal to a linear combination of $S_1, ..., S_n$ containing an unrevealed secret set. Therefore, we obtained a contradiction with the maximality of j_1. ◀

Next we establish the connection between $\mathbf{T}$ and $\overline{\Pi}_z$. By construction, $\mathbf{T}(z)$ simulates Π on a random input $(x, y) \sim U(P_z)$. Fix r to be the outcome of the random bits of the tree $\mathbf{T}$. Then $\mathbf{T}_r(z)$ will simulate Π on some pair $(x_r, y_r) \in P_z$.

We show the following.

▶ **Lemma 30.** *The number of queries to z made by $\mathbf{T}_r(z)$ is less or equal than the number of revealed sets in protocol $\overline{\Pi}_z$ on input (x_r, y_r).*

Proof. By definition of $\mathbf{T}$, it queries z_i only if there is some linear combination of $S_1, ..., S_n$ containing S_i that equals a linear combination of $Q_1, ..., Q_t$. By Lemma 29, we get that z_i is queried by $\mathbf{T}$ only if S_i is revealed. Hence, the statement of the lemma follows. ◀

The lemma holds for all r. In particular, if we average over r, then we get that the expected number of queries made by $\mathbf{T}$ is bounded by the expected number of revealed sets in protocol $\overline{\Pi}_z$ on a random $(x, y) \sim U(P_z)$. Therefore, it remains to obtain an upper bound on the expected number of revealed sets.

5.4.3 Upper bound on the expected number of revealed sets in $\overline{\Pi}_z$

For each vertex v of the protocol $\overline{\Pi}_z$, define

$$P_{z,v} = \{(x, y) \mid \overline{\Pi}_z \text{ reaches vertex } v \text{ on input } (x, y) \text{ and } g^n(x, y) = z\}$$

In other words, if $X_v \times Y_v$ is the rectangle corresponding to the vertex v in the protocol $\overline{\Pi}_z$, then $P_{z,v} = X_v \times Y_v \cap (g^n)^{-1}(z)$.

Essentially, if the protocol $\overline{\Pi}_z$ has reached the vertex v, then (x, y) can be any element of the set $P_{z,v}$. Now consider the uniform distribution $U(P_z)$ and run $\overline{\Pi}_z$ on a random pair (x, y) from this distribution. Then $U(P_{z,v})$ is precisely the conditional distribution of the pair (x, y), given that $\overline{\Pi}_z$ has reached v. In what follows, we consider $(x, y) \sim U(P_{z,v})$. We will be interested in the entropy of the projection of this distribution onto x, hence we introduce the following notation

$$H^v = H(x),$$

$$H^v_i = H(x_i).$$

Let S_i be the secret set in the i-th block. The set S_i depends on x_i and, moreover, there is a one-to-one correspondence between S_is and x_is. Therefore, their entropies are equal: $H(S_i) = H(x_i) = H_i^v$. Denote by L_i^v the set L_i corresponding to the vertex v, and by $\mathcal{L}_i^v$ its linear span with the zero vector removed. That is, $|\mathcal{L}_i^v| = 2^{|L_i^v|} - 1$.

Using Fano's inequality, we can show the following.

▶ **Lemma 31.** *Let v be a vertex of the protocol $\overline{\Pi}_z$ in which Alice sends a message revealing whether S_i lies in the linear span L_i^v (this is (A2) type of message on Figure 1). Then, if $H_i^v \geq \log_2 |\mathcal{L}_i^v| + \frac{1}{10}\log_2 k + \frac{1}{3}$, then S_i does not lie in $\mathcal{L}_i^v$ with probability at least $\frac{1}{100}$.*

Proof. Define Y to be equal to S_i if $S_i \in \mathcal{L}_i^v$, and to be equal to any element in $\mathcal{L}_i^v$ otherwise. Note that with this definition, $P(S_i \notin \mathcal{L}_i^v) = P(S_i \neq Y)$. Denote this probability by ε. Then, by Fano's inequality we have

$$H(S_i) - \log_2 |\mathcal{L}_i^v| \leq H(S_i) - H(Y) \leq H(S_i|Y) \leq H(\varepsilon) + \varepsilon \log_2 k.$$

Assume that $\varepsilon < 1/100$. Then

$$H(S_i) < \frac{1}{10}\log_2 k + \log_2 |\mathcal{L}_i^v| + H(1/100).$$

Since $H(1/100) < 1/3$, we get at a contradiction with the statement of the lemma. ◀

Now let's show that if at vertex v Alice or Bob sends one-bit message, then the entropy $H(x)$ on average does not decrease significantly.

▶ **Lemma 32.** *Let $I_v : P_{z,v} \to \{0,1\}$ denote the bit of information that Alice or Bob sends at vertex v. Then*

$$H(x|I_v) \geq H(x) - H(I_v) \geq H(x) - 1.$$

Proof. We have that

$$H(x|I_v) + H(I_v) = H(x, I_v) \geq H(x)$$

and

$$H(x|I_v) \geq H(x) - H(I_v) \geq H(x) - 1.$$ ◀

In other words, the entropy H of the distribution of x drops by no more than 1 on average on one step of the protocol. Next, we introduce the deficiency of the distribution. Let

$$U_i^v = \begin{cases} \log_2 k, & \text{if } S_i \text{ is not fixed;} \\ 0, & \text{if } S_i \text{ is fixed,} \end{cases}$$

and let the deficiency be

$$D^v = \left(\sum_{i=1}^n U_i^v\right) - H^v.$$

Note that $H(S_i) = 0$ if S_i is fixed. Thus, $D^v \geq 0$ for any v, as $\sum_{i=1}^n U_i^v \geq \sum_{i=1}^n H_i^v \geq H^v$. We will omit the superscript v if the vertex is clear from the context.

Now we consider $(x, y) \sim U(P_z)$ and introduce some random variables for various types of messages of $\overline{\Pi}_z$ on the input (x, y).

For this we consider a finer classification of the messages sent by Alice, compared to the one in Figure 1:

(A1) A message that Alice sends in Π.

(A2') A message $S_i \overset{?}{\in} \mathcal{L}_i$, such that we have $H_i \geq \log_2 |\mathcal{L}_i| + 1/3 + \frac{1}{10}\log_2 k$ before sending this message. In other words, in these messages S_i was not previously fixed.

(A2") A message $S_i \overset{?}{\in} \mathcal{L}_i$, such that we have $H_i < \log_2 |\mathcal{L}_i| + 1/3 + \frac{1}{10}\log_2 k$ before sending this message. In other words, in these messages S_i was previously fixed.

(A3') A message that Alice sends to find out the exact value of S_i, and S_i was not fixed before sending this message.

(A3") Same as (A3') but S_i was fixed before sending this message.

For Bob we use the same classification of messages as in Figure 1:

(B1) A message that Bob sends in Π.

(B2) A message in which Bob communicates the value y_{S_i}

Let $A_1, A'_2, A''_2, A'_3, A''_3, B_1, B_2$ denote the number of messages of the corresponding type sent by the protocol during the whole computation. All of these are random variables depending on (x, y), which we draw randomly: $(x, y) \sim U(P_z)$.

The following inequality hold.

▶ **Observation 33.** $\mathbb{E}A'_2 \leq 100 \cdot \mathbb{E}B_1$.

Proof. When Alice sends a message of type (A2'), by Lemma 31 the secret set S_i does not lie in $\mathcal{L}_i$ with probability at least $\frac{1}{100}$. If it does not lie in $\mathcal{L}_i$, the processing of the Bob's message is over. Thus, for each query of type (B1), there will be no more than 100 queries of type (A2') on average. ◀

▶ **Observation 34.** *At the end of the computation, the number of fixed S_i is greater or equal than A'_3. In particular, there are at least A'_3 coordinates i such that U_i decreased from $\log_2 k$ to 0. During each of these decreases of U_is, D decreases by at least $\max\{0, \frac{9}{10}\log_2 k - 1/3 - \log_2 |\mathcal{L}_i|\}$ on average.*

Proof. By definition, A'_3 is precisely equal to the number of the fixed secret sets S_i. When the status of S_i changes from unfixed to fixed, U_i drops from $\log_2 k$ to 0. Since U_i changes to 0 only when H_i becomes less than $\log_2 |\mathcal{L}_i| + 1/3 + \frac{1}{10}\log_2 k$ (as ensured by if's on lines 19 and 23), sending S_i transmits no more than $\log_2 |\mathcal{L}_i| + 1/3 + \frac{1}{10}\log_2 k$ information. Thus, we get the required decrease of D by at least $\max\{0, \frac{9}{10}\log_2 k - 1/3 - \log_2 |\mathcal{L}_i|\}$ on average. ◀

▶ **Observation 35.** *Messages of types (B2), (A2"), and (A3") do not affect D.*

Proof. A message of the type (B2) does not change D since pairs $(x, y) \in P_z$ have a property that $g(x_i, y_i) = z_i$. Thus, message y_{S_i} does not decrease the entropy of the distribution.

Just before a message M of type (A2") is sent, by definition, we have that $H_i < \log_2 |\mathcal{L}_i| + 1/3 + \frac{1}{10}\log_2 k$. But in lines $19 - 21$ it is ensured that if H_i is lower than this threshold, then S_i will be fixed. Thus, H_i is not only lower than $\log_2 |\mathcal{L}_i| + 1/3 + \frac{1}{10}\log_2 k$, but also equals to 0. Thus, $(A2'')$ does not influence D.

By definition, S_i is fixed before sending a message of type (A3"). Thus, $H_i = 0$ and sending S_i does not reveal any information. ◀

▶ **Observation 36.** *At the end of the protocol's execution, $\sum_i |L_i| \leq B_1 + B_2$.*

Proof. A new set Q is generated and added to the list of Q_is either when a message of type (B1) is sent, or after a message of type (B2) is sent. ◀

Let's show how the desired bound follows from these lemmas. Note that if S_i is revealed, then it must be fixed. As a consequence, we get $B_2 \le A'_3$ since B_2 equals the number of revealed sets and A'_3 equals the number of fixed ones. Our goal is to upper bound the number of revealed sets. Thus, we only need to upper bound $\mathbb{E}A'_3$ by $\mathbb{E}(A_1 + B_1)/\log_2 k$, as $A_1 + B_1$ is exactly the number of messages sent by the protocol Π.

▶ **Lemma 37.** $\mathbb{E}A'_3 = O(\mathbb{E}(A_1 + B_1)/\log_2 k)$

Proof. By Lemma 32 messages of type (A1) and (B1) increase D by no more than 1 on average. The same is true for messages of type (A2'). As a result, due to Observation 34 and Observation 35,

$$\mathbb{E}\left(-A'_3 \cdot \left(\frac{9}{10}\log_2 k - 1/3\right) + \sum_i \log_2 |\mathcal{L}_i| + A_1 + B_1 + A'_2\right) \ge 0,$$

since D is always greater or equal than than 0. Applying Observation 33 and rearranging we get

$$\mathbb{E}\left(\sum_i \log_2 |\mathcal{L}_i| + 101 \cdot B_1 + A_1\right) \ge \mathbb{E}A'_3 \cdot \left(\frac{9}{10}\log_2 k - 1/3\right).$$

Now we consider two cases.

1. $k \ge 3$
   Note that $\sum_i \log_2 |\mathcal{L}_i| \le \sum_i |L_i| \le B_1 + B_2$. Using the fact that $B_2 \le A'_3$ and the inequality above, we obtain

   $$\mathbb{E}\left(102 \cdot B_1 + A_1\right) \ge \mathbb{E}A'_3 \cdot \left(\frac{9}{10}\log_2 k - 4/3\right).$$

   Since $\frac{9}{10} \cdot \log_2 k > 4/3$, we get $\mathbb{E}A'_3 = O\left(\frac{\mathbb{E}A_1 + B_1}{\log_2 k}\right)$.
2. $k = 2$
   Note that if we are sending a message of type (A3') and it is true that $|L_i| \le 1$, then D decreases by at least $1 - (1/10 + 1/3) \ge \frac{1}{2}$ on average. The number of is such that $|L_i| \ge 2$ does not exceed $\frac{B_1+B_2}{2}$ by Observation 36. As a result,

   $$\mathbb{E}\left(-\frac{1}{2} \cdot \left(A'_3 - \frac{B_1 + B_2}{2}\right) + A_1 + B_1 + A'_2\right) \ge 0.$$

   Using $B_2 \le A'_3$, we get

   $$\mathbb{E}\left(5/4 \cdot B_1 + 100 \cdot B_1 + A_1\right) \ge \mathbb{E}A'_3 \cdot 1/4.$$

   Thus, we again obtain $\mathbb{E}A'_3 = O(\mathbb{E}(A_1 + B_1))$. ◀

References

1 Nikhil Bansal and Makrand Sinha. k-forrelation optimally separates quantum and classical query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1303–1316, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451040.

2 Paul Beame and Sajin Koroth. On disperser/lifting properties of the index and inner-product functions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ITCS.2023.14.

3 Eric Blais, Li-Yang Tan, and Andrew Wan. An inequality for the Fourier spectrum of parity decision trees. *CoRR*, abs/1506.01055, 2015. arXiv:1506.01055.

4 Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting using low-discrepancy gadgets. *SIAM J. Comput.*, 50(1):171–210, 2021. doi:10.1137/19M1310153.

5 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudo-random properties. *Comput. Complex.*, 28(4):617–659, 2019. doi:10.1007/S00037-019-00190-7.

6 Arkadev Chattopadhyay, Nikhil S. Mande, Swagato Sanyal, and Suhail Sherif. Lifting to parity decision trees via stifling. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 33:1–33:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ITCS.2023.33.

7 Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006.

8 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory Comput.*, 16:1–30, 2020. doi:10.4086/TOC.2020.V016A013.

9 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

10 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM J. Comput.*, 47(6):2435–2450, 2018. doi:10.1137/16M1059369.

11 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. *SIAM J. Comput.*, 49(4), 2020. doi:10.1137/17M115339X.

12 Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Structure of protocols for XOR functions. *SIAM J. Comput.*, 47(1):208–217, 2018. doi:10.1137/17M1136869.

13 Bruno Loff and Sagnik Mukhopadhyay. Lifting theorems for equality. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 50:1–50:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.STACS.2019.50.

14 Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with sunflowers. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 104:1–104:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.ITCS.2022.104.

15 Frédéric Magniez, Ashwin Nayak, Miklos Santha, and David Xiao. Improved bounds for the randomized decision tree complexity of recursive majority. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 317–329, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-22006-7_27.

16 Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1246–1255. ACM, 2017. doi:10.1145/3055399.3055478.

17 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Comb.*, 19(3):403–435, 1999. doi:10.1007/S004930050062.

18 Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 406–415. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.51.

19 Miklos Santha. On the Monte Carlo Boolean decision tree complexity of read-once formulae. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 180–187. IEEE Computer Society, 1991. doi:10.1109/SCT.1991.160259.

20 Xiaodi Wu, Penghui Yao, and Henry S. Yuen. Raz-McKenzie simulation with the inner product gadget. *Electron. Colloquium Comput. Complex.*, TR17-010, 2017. URL: https://eccc.weizmann.ac.il/report/2017/010.