Electronic Colloquium on Computational Complexity, Report No. 199 (2024)

Randomized Lifting to Semi-Structured Communication Complexity via Linear Diversity

Vladimir Podolskii[1] and Alexander Shekhovtsov[2]

[1]Tufts University
[2]Moscow Institute of Physics and Technology

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 (Beame and Koroth, 2023) 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, and Majority gadget MAJ_m for all m. 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 (Chattopadhyay et al., 2023) 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.

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 (Göös et al., 2018; Loff and Mukhopadhyay, 2019; Chattopadhyay et al., 2019; Göös et al., 2020; Chattopadhyay et al., 2021; Lovett et al., 2022). These results proved to be extremely useful for solving open problems in various areas of computational complexity (Raz and McKenzie, 1999; Robere et al., 2016; Pitassi and Robere, 2017; Göös and Pitassi, 2018; Garg et al., 2020). 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.

ISSN 1433-8092

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 (Raz and McKenzie, 1999; Göös et al., 2018; Hatami et al., 2018; Chattopadhyay et al., 2019; Göös et al., 2020; Chattopadhyay et al., 2021). 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 (Chattopadhyay et al., 2019) and (Wu et al., 2017). For randomized case, the first result was obtained in (Göös et al., 2020) with a gadget of polynomial size. The paper (Chattopadhyay et al., 2021) 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 (Beame and Koroth, 2023; Chattopadhyay et al., 2023).

The paper (Chattopadhyay et al., 2023) 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 there 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 Section 2.3 for the definition of these functions).

The paper (Beame and Koroth, 2023) 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 (Beame and Koroth, 2023) 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 Section 3.4.

Similarly to (Chattopadhyay et al., 2023) 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 (Beame and Koroth, 2023) 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 (Chattopadhyay et al., 2023) 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 (Göös et al., 2020; Chattopadhyay et al., 2021). 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: $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$.

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 (Göös et al., 2020; Chattopadhyay et al., 2021) 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 Göös et al. (2020); Chattopadhyay et al. (2021) and more specifically, our general strategy is very similar to the one of Beame and Koroth (2023) (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 Section 4 we introduce additional notation that is used in the proofs. In Section 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 Section 5.2 we describe the protocol to simulate communication protocol by decision tree. In Section 5.3 we proof correctness of the simulation. In Section 5.4 we

prove the bound on the number of queries in the simulation (this is the most technically heavy part of the proof). In Section 6 we extend the result to the size of the communication protocols and to the case of subspace queries. In Section 7 we prove the deterministic versions of our results. In Section 8 we extend the deterministic result to the size of the communication protocols and to the case of subspace queries. In Section 9 we prove results for Majority gadget and provide some generalizations. In Section 10 we show some applications of our results.

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 (Beame and Koroth, 2023). 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 1 (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 2 (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 2. *Assume that a gadget* g *reduces to a gadget* h*. Then for any relation* $f \subseteq \{0,1\}^n \times \mathcal{R}$ *we have*

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

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

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

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 3 (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 4 (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 5 (Inner product function)**.** Let $\mathrm{IP}_m : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be the function that on input (x, y) outputs $\bigoplus_{i=1}^{n}(x_i \wedge y_i)$.

Lemma 3. *$\textsc{Ind}_m$ 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 4. *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 is provided in Appendix A.

3 Results Statement

3.1 Randomized Semi-Structured protocols

Theorem 5. *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 5.

Theorem 6. *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*

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

Theorem 7. *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 2 we immediately obtain the following.

Corollary 8. *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 9. *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 10. *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 11. *Let $h : \mathcal{Y} \times \{0,1\}^n \to \{0,1\}$ have linear diversity 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 5, and Lemma 2. 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 it in Section 6. The deterministic version is proved in Section 8.

3.4 More powerful gadget reduction

In this section, we describe a more general version of a gadget reduction than in Definition 2. 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 $x_1, \dots, x_m$ 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 2 after that. Since the transformation to the new variables is linear, new variables can be expressed as an XOR of old variables and vise 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 12. *For any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$ and $m \geq 12$, $\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ \mathrm{MAJ}_m^n) = \Omega(m \cdot \mathsf{R}^{\mathsf{dt}}(f))$.*

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

Theorem 13. *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 2. 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 2.

Proofs of these results are provided in Section 9.

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 6 (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 7. (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 8. (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 9. (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 10. (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 11. (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 12. (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 14 (Gibbs' inequality (Cover and Thomas, 2006))**.** *Let $0 \leq p \leq 1, 0 < q < 1$. Then,*

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

Lemma 15 (Fano's Inequality (Cover and Thomas, 2006))**.** *Let X, Y be random variables. Let $\varepsilon = P(X \neq Y) \leq 1/2$. Then,*

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

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

5 Proof of Theorem 5

5.1 Reformulation of Theorem 5

As we noted in Section 3, the $\leq$-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 16. *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 16 we show how to prove Theorem 5 based on Theorem 16.

Proof of Theorem 5. 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 16 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 16. 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 leaf of Π, it outputs the value written in this leaf.

5.3 Correctness of the simulation of Π done by $\mathbf{T}$

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 $\mathbf{T}$

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 are 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 $i = 1, \ldots, t$ 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.

Lemma 17. *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 18. *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 17, we get that z_i is queried by $\mathbf{T}$ only if S_i is revealed. Hence, the statement of the lemma follows. □

[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 23.

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).

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: $H(S_i) = H(x_i) = H^v_i$. Denote by L^v_i the set L_i corresponding to the vertex v, and by $\mathcal{L}^v_i$ its linear span with the zero vector removed. That is, $|\mathcal{L}^v_i| = 2^{|L^v_i|} - 1$.

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

Lemma 19. *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^v_i (this is (A2) type of message on Figure 1). Then, if $H^v_i \geq \log_2 |\mathcal{L}^v_i| + \frac{1}{10}\log_2 k + \frac{1}{3}$, then S_i does not lie in $\mathcal{L}^v_i$ with probability at least $\frac{1}{100}$.*

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

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

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

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

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 20. *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 21. $\mathbb{E}A_2' \leq 100 \cdot \mathbb{E}B_1$.

Proof. When Alice sends a message of type (A2'), by Lemma 19 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 22. *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 23. *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 24. *At the end of the protocol's execution, $\sum_i |L_i| \le B_1 + B_2$.*

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

Proof. By Lemma 20 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 22 and Observation 23,

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

since D is always greater or equal than than 0. Applying Observation 21 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

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

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 $|L_i| \leq 1$, then D decreases by at least $1 - (1/10 + 1/3) \geq \frac{1}{2}$ on average. The number of is such that $|L_i| \geq 2$ does not exceed $\frac{B_1+B_2}{2}$ by Observation 24. As a result,

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

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

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

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

□

6 Randomized Size and Subspace models

In this section, we will prove Theorem 6, Theorem 7, as well as the randomized part of Theorem 11.

6.1 Proof of Theorem 6

We start with the proof of Theorem 6.

For this we consider a more detailed classification of messages of type (B1) made by Bob:

(B1') A message about a parity of Q that eventually reduced to $\varnothing$ and there were no messages of type (A2') in the process.

(B1") Other messages of type (B1).

The following is a stronger version of Observation 24.

Observation 26. *At the end of the protocol's execution,* $\sum_i |L_i|$ *does not exceed* $B''_1 + B_2$.

Proof. The crucial observation for this proof is that in Bob's message of type (B1') Q eventually reduces to $\varnothing$. During the processing of these messages we might reveal several subsets S_i. This results in messages of type (B2) and into increasing of $\sum_i |L_i|$. But on the last step Q becomes an empty set and does not contribute to $\sum_i |L_i|$. Taking this into account we can replace B_1 by B''_1 in the proof of Observation 24 and the statement follows. □

We also need the following observation.

Observation 27. *Messages of types (B1') increase* D *by no more than* B_2 *in total.*

Proof. Consider a Bob's message of type (B1') that sends an XOR of the set Q. Since it is (B1'), its processing will result in $Q = \varnothing$. In particular, the statement of *if* at line 23 was always true. Let l denote the number of times a message of type (B2) was sent during the processing of Q. If $l = 0$, then Q reduced to $\varnothing$ immediately after *sift*. In this case, this message does not reveal any

new information, so D stays the same. Now consider a case $l > 0$. Note that D increases by no more than 1 when we send a message of type (B1'). Thus, since $l > 0$, if we sum up all inequalities $\Delta D \leq l$ over all messages of type (B1'), we get the desired bound. □

The following observation shows why we want to consider messages of type (B1") instead of type (B1).

Observation 28. *For messages of type (B1") the probability of both* 0 *and* 1 *message is at least* $\frac{1}{200}$.

Proof. By definition, either Q was non-empty in the end or a message of type (A2') was made. In the first case, Bob sends a random bit $Bern(1/2)$ (this was shown in Section 5.3). In the second case, by Lemma 19, S_i does not lie in $\mathcal{L}_i$ with probability at least $\frac{1}{100}$. If S_i indeed does not lie in $\mathcal{L}_i$, then the processing of Q will finish with Q being non-empty. In this case Bob's message will be $Bern(1/2)$ again. Thus, in both cases, we obtain the desired probability for sending either of the messages 0 and 1. □

However, to finish the proof of the theorem we need to also substitute A_1 by something smaller in the statement of Lemma 25 as A_1 might be too large. Consider the situation when Alice sends a message of type (A1). Let $I_v : P_{z,v} \to \{0, 1\}$ denote her message depending on her input. Let b be the bit she sends. When it is sent, the protocol gains $H(P_{z,v}) - H(P_{z,v}|I_v = b)$ information. Define AH_1 to be the sum of the information gained by the protocol when Alice was sending messages of type (A1). Note that $\mathbb{E}AH_1 \leq \mathbb{E}A_1$ since $H(I_v) \leq 1$.

Observation 29. *The sum of increases of D when a message of type (A1) is being is sent is bounded by AH_1.*

Proof. By definition, D increases by the quantity of information that a message has revealed. AH_1 equals exactly the total amount of information that was gained during the sending of messages (A1). □

Now we can prove a stronger version of Lemma 25.

Lemma 30. $\mathbb{E}A'_3 = O(\mathbb{E}(AH_1 + B''_1)/\log_2 k)$

Proof. We first comment on how to replace B_1 with B'_1, and then how to replace A_1 with AH_1.

By Observation 27, the contribution of (B1') to D can be bounded by B_2, and by Observation 26, (B1') does not contribute to $\sum |L_i|$. Thus, we can just repeat the same argument as in Lemma 25 using (B1") instead of (B1). However, a more careful analysis is need in case $k = 2$ due to the additional $+B_2$ term.

In the proof of Lemma 25, we use A_1 only to bound the growth of D. Thus, we can replace A_1 by AH_1 since AH_1 exactly equals to the contribution of Alice's messages to D as stated in Observation 29.

The full proof can be found in Appendix D.

□

Now we prove Theorem 6 using these lemmas.

Proof of Theorem 6. As in the previous section, to show an upper bound on the size of the decision tree we need to prove an upper bound on B_2. Again we use $B_2 \leq A'_3$. Thus, it is enough to upper bound the expectation of A'_3. We use Lemma 30 and upper bound B''_1 and AH_1 separately.

For a vertex v of the protocol Π, let $Size(v)$ denote the size of its subtree. After each message of type (B1"), $Size(v)$ on average decreases at least by a factor of $1/\left(\frac{99}{100}+\frac{1}{2}\cdot\frac{1}{100}\right)=\frac{200}{199}$. As a result, we get $\mathbb{E}B_1''=O(\log Size(root))$.

Consider a vertex v at which Alice makes a message of type (A1). Let b denote the bit she sends. Let $p=\mathbf{Pr}(b=0)$. Denote $q=Size(v_0)/Size(v)$, where v_0 is the $0-$child of v. Let's estimate the change of the logarithm of the size of the subtree.

$$\mathbb{E}_b\left(\log_2 Size(v)-\log_2 Size(v_b)\right)=p\log_2\frac{1}{q}-(1-p)\log_2\frac{1}{1-q}\geq H(p).$$

The last inequality is the application of Lemma 14. As a result, $\mathbb{E}AH_1=O(\log_2 Size(root))$.

As a result, we obtain the desired bound on the depth of $\mathbf{T}$. □

6.2 Proof of the part of Theorem 11 related to $\mathsf{sizeR}^{\mathsf{dt}}_{\oplus}$

To prove the bound for $\mathsf{sizeR}^{\mathsf{dt}}_{\oplus}$ in Theorem 11, we introduce the following modification of the deterministic semi-structured protocols, that we will denote $\Pi_{\oplus}$. In each communication round of $\Pi_{\oplus}$, Alice computes an arbitrary Boolean function of her input, while Bob computes the XOR of some subset of his input. Let b_1 be the value computed by Bob and b_2 the value computed by Alice. In this round protocol learns $b_1\oplus b_2$, but the values b_1 and b_2 themselves remain unknown.

Let $T_{\oplus}$ be the deterministic decision tree computing $f\circ h^n$, which queries the XORs of subsets of variables. It is clear how to transform $T_{\oplus}$ into $\Pi_{\oplus}$ protocol without increasing the protocol's size. Each query of the tree $T_{\oplus}$ is simulated by one round of $\Pi_{\oplus}$ by definition. Thus, essentially, the tree $\Pi_{\oplus}$ will be the same as the tree $T_{\oplus}$. As a result, their sizes are the same.

Therefore, this new model is stronger than the parity decision tree. Hence, it suffices to show a bound for it. Let's state this formally.

Let $\mathbf{\Pi}_{\oplus}$ be the randomized version of the protocol $\Pi_{\oplus}$ that computes $f\circ h^n$. We claim that

$$\log Size(\mathbf{\Pi}_{\oplus})\geq\Omega(\log_2 k\cdot\mathsf{R}^{\mathsf{dt}}(f)).$$

The rest of the section is devoted to the proof of this statement. It is easy to see that Lemma 2 holds for this modified version of semi-structured complexity. Therefore, we can assume that h is a family of linear functions of order k.

Similarly to the proof of Theorem 5, we will simulate this protocol on a random input from P_z. Hence, it suffices to prove the bound for the deterministic protocol $\Pi_{\oplus}$.

We will construct a semi-structured protocol Π based on $\Pi_{\oplus}$, which will copy the actions of $\Pi_{\oplus}$ as follows. When one round of $\Pi_{\oplus}$ occurs, Bob computes and sends the XOR of the required subset of variables b_1, and Alice computes and sends the value of the required Boolean function b_2. In this way, we learn the value $b_1\oplus b_2$ and follow the protocol $\Pi_{\oplus}$, producing the same output in the end. However, note that the size of Π may increase exponentially compared to the size of $\Pi_{\oplus}$ (imagine that $\Pi_{\oplus}$ is structured like a bamboo tree). But we will still show, using the way $\Pi_{\oplus}$ is simulated by Π, that the depth of the tree $\mathbf{T}$ will be small on average. Recall that the tree $\mathbf{T}$ is constructed based on the protocol Π, which in turn is based on the protocol $\Pi_{\oplus}$.

We use the same notation for the protocol Π as before. Therefore, according to Lemma 30, we need to show the bound $\mathbb{E}(AH_1+B_1'')=O(\log Size_{root}(\Pi_{\oplus}))$ to complete the argument.

We have that the protocol Π simulates $\Pi_{\oplus}$. Let v be the node of the protocol $\Pi_{\oplus}$ where the simulation is currently located. A round of the protocol $\Pi_{\oplus}$ is divided into two messages of the protocol Π of types (A1) and (B1). Let Bob send his message b_1 first, and then Alice sends b_2. Recall that b_1 and b_2 are random variables over $U(P_{z,v})$. Denote by Q the set whose XOR is computed and sent by Bob.

A key fact for proving the necessary bound is the following lemma.

Lemma 31. *Let E be the event that Q is not contained in the linear span of $Q_1, \dots, Q_t$ and $S_1, \dots, S_n$. Then, when $U(P_{z,v})$ is conditioned on E, b_1 is independent of b_2. Moreover, b_1 is distributed as Bern(1/2).*

Proof. For either of the two possible values of b_2, b_1 will be $Bern(1/2)$ because the number of points in the affine space such that $y_Q = 0$ and $y_Q = 1$ is the same. □

The rest of the proof is very similar to the proof of Theorem 6.

In the proof of Lemma 19, we have shown that the probability of E is at least $\frac{1}{100}$ when Bob sends a message of type (B1"). Therefore, according to the lemma, $b_1 \oplus b_2$ will also be $Bern(1/2)$ conditioned on E. Thus, analogously to the proof of Theorem 6 we have substantial probability to move to the left and right child of v. From this, we get that $\mathbb{E}B_1'' = O(\log Size_{root}(\Pi_\oplus))$.

Next we bound AH_1. When Alice sends her message b_2, its entropy is either $H(b_2|b_1 = 0)$ or $H(b_2|b_1 = 1)$ depending on what Bob sent. Since $\Pi_\oplus$ computes $b_1 \oplus b_2$, the entropy $H(v_{b_1 \oplus b_2}|b_1)$ is exactly equal to $H(b_2|b_1)$, where v_0, v_1 are the children of v. Therefore, analogously to the proof of Theorem 6, by Gibbs' inequality, we obtain that AH_1 increases by no more than the decrease in $\log_2 Size_v(\Pi_\oplus)$ on average. That is, $\mathbb{E}AH_1 = O(\log_2 Size_{root}(\Pi_\oplus))$.

6.3 Proof of Theorem 7

To prove Theorem 7, we use the same argument as in (Chattopadhyay et al., 2023). We modify Bob's message model. Bob will be allowed to make queries of the form $y_Q \stackrel{?}{=} b$ about his input y. The answer to such a query will either be ≪Correct≫ if $y_Q = b$ or ≪Incorrect≫ if $y_Q \neq b$. After Bob makes such a query, he then sends the message y_Q as usual. The cost of Bob's message is the total number of queries that received an ≪Incorrect≫ response. Note that all of Alice's messages still cost 1, as usual. We will refer to this complexity as *conditional complexity*.

In this model, Bob can easily communicate the membership of y in any arbitrary affine subspace. A query about membership in an arbitrary affine subspace can be expressed as the following conjunction:

$$y_{Q_1} = b_1 \wedge y_{Q_2} = b_2 \wedge \dots \wedge y_{Q_m} = b_m.$$

Bob will sequentially ask the questions $y_{Q_1} \stackrel{?}{=} b_1, y_{Q_2} \stackrel{?}{=} b_2, \dots$ until he receives an ≪Incorrect≫ response. This would indicate that y does not belong to the subspace. Note that from the perspective of conditional complexity, the total cost of such a series of messages is 1. Therefore, communicating membership in an affine subspace can be easily simulated in this model.

Now we will bound the complexity of $\mathbf{T}$.

Lemma 32. *The expected number of queries made by $\mathbf{T}$ during the simulation of Π is bounded by $O(d_c/\log_2 k)$, where d_c is the conditional complexity of Π.*

Proof. The type of messages that Alice sends does not change, so $A_1 = O(d_c)$. It remains to estimate the number of (B1")-type messages sent by Bob. According to Observation 28, we have $\mathbb{E}B_1'' = O(d_c)$. Thus, using Lemma 30, we obtain the desired bound. □

Proof of Theorem 7. As we have shown, we can communicate the membership of y in any arbitrary affine subspace in the model with conditional complexity. Therefore, the desired bound follows from Lemma 32. □

6.4 Proof of the Theorem 11 part about $\mathsf{sR}^{\mathsf{dt}}_{\oplus}$

In this section we show how to prove the part of Theorem 11 about $\mathsf{sR}^{\mathsf{dt}}_{\oplus}$.

Recall that a randomized decision tree is a distribution over deterministic decision trees. Our strategy that for every deterministic decision tree sT with subspace queries in the distribution we will construct a deterministic semi-structured protocol (with the same type of messages) whose depth will be bounded by the depth of sT. Then, using Lemma 32, we will obtain the desired bound.

A query on the membership in an arbitrary subspace can be expressed as:

$$x_{Q_1} = b_1 \wedge x_{Q_2} = b_2 \wedge \cdots \wedge x_{Q_m} = b_m.$$

Alice and Bob are going to run the semi-structured protocol Π of the same type as in subsubsection 5.4.1, that will proceed as follows. For each query Bob sequentially iterates through $i = 1, \ldots, m$, computes XOR of his variables from Q_i and sends it. If during the processing of this message players see that Bob's message is actually of type (B1"), Alice then sends her part of the XOR for Q_i. Since Bob's message is of type (B1"), the probability that the XOR of their messages equals b_i is no more than $\frac{199}{200}$, according to Lemma 19 and Lemma 31. In other words, with a probability of at least $\frac{1}{200}$, x will not belong to the affine subspace, meaning that this query can be finished. If players see that Bob's message is of type (B1'), Alice does **not** send her part of the XOR for Q_i. Indeed, in this case, since the query reduced to an empty set players already learned all the information.

If Bob has iterated through all $i = 1, \ldots, m$ and for all of his (B1")-type messages $x_{Q_i} = b_i$ holds, then Alice uses exactly 1 bit to communicate whether x lies in the affine space. She can do this because Bob has provided all the necessary information.

We have shown that, while processing this tree query, the expected number of (B1")-type messages is no more than 200. Moreover, the expected number of (A1)-type messages is no more than the number of (B1")-type messages plus 1. Thus, according to Lemma 25, we obtain the desired bound on the depth of $\mathbf{T}$.

7 Proof of the part of Theorem 9 for D^{cc}

As in the previous sections we will use the notation introduced in Section 4. Before describing the decision tree that will simulate a semi-structured communication protocol, we will introduce invariants that we will maintain during the simulation and discuss some properties of these parameters.

7.1 Invariants

During the algorithm's execution, we will maintain the variables $X, Y, \mathcal{Q}, \rho$. Here X is the set of the inputs of Alice that are consistent with the answers to the queries, Y is the set of consistent inputs of Bob. $\mathcal{Q}$ is roughly the list of sets for which Bob had already sent their parities and ρ reflects which coordinates of z are fixed.

Before describing the algorithm itself, we describe properties that these variables must satisfy.

Definition 13 (Y is $\mathcal{Q}$-fixed)**.** Let $Y \subseteq (\{0,1\}^m)^n$ and $Q_i \subseteq [n] \times [m]$. We say that Y is $\mathcal{Q}$-fixed, where $\mathcal{Q} = [Q_1, ..., Q_t]$, if the following holds:

$$\forall y, y' \in Y,\ \forall Q_i \in \mathcal{Q},\quad y_{Q_i} = y'_{Q_i}.$$

Basically, this means that the parity of variables on positions Q_i is the same for all inputs in Y. If Y is $\mathcal{Q}$-fixed and $Q_i \in \mathcal{Q}$, then we let Y_{Q_i} to be y_{Q_i} for some $y \in Y$. By Definition 13 Y_{Q_i} is well-defined.

Definition 14. ((X, Y) is ρ-fixed). For $X \subseteq [k]^n, Y \subseteq (\{0,1\}^m)^n, \rho \in \{0,1,*\}^n$, we say that (X, Y) is ρ-fixed if the following holds:

(1) $\forall i \in [n] : \rho(i) \neq *, \forall x, x' \in X,\ x_i = x'_i$,
(2) $\forall i \in [n] : \rho(i) \neq *, \forall x \in X, y \in Y,\ g(x_i, y_i) = \rho(i)$.

The first condition says that the projection of X onto the i-th block is a singleton. We let X_i to be equal to x_i, for some $x \in X$. This notation is correctly defined since x_i is the same for all $x \in X$. The second condition essentially means that the value of the function g in the i-th block is always equal to $\rho(i)$.

Definition 15 (ρ-Structured-Triple)**.** Consider $X \subseteq [k]^n, Y \subseteq (\{0,1\}^m)^n$, $\rho \in \{0,1,*\}^n$ and $Q_j \subseteq [n] \times [m]$ for $j = 1, \ldots, t$.

Define $L_i = \{Q_j \cap (\{i\} \times [m]) \mid p(Q_j) \in \{i\} \times [m]\}$. That is, L_i contains intersections of Q_j with $i-$th block, whose *principal* variable touches it.

Let $\mathcal{L}_i$ to be equal to the linear span of L_i with zero vector removed.

We say that the triple $(X, Y, \mathcal{Q} = [Q_1, ..., Q_t])$ is ρ-structured if the following holds:

(1) $X \neq \varnothing, Y \neq \varnothing$,
(2) $p(Q_i)$ is *lower* than $p(Q_j)$ if $i < j$,
(3) Y is $\mathcal{Q}$-fixed, and Y is an affine subspace of co-dimension t,
(4) (X, Y) is $\rho-$fixed,
(5) $\forall x \in X, i \in [n] :\ \rho(i) = * \leftrightarrow \{i\} \times G_{x_i} \notin \mathcal{L}_i$,
(6) For any i such that $\rho(i) \neq *$, it holds that $\{i\} \times G_{X_i}$ can be expressed as a linear combination of $Q_1, ..., Q_t$.

Note that condition (2) implies that Q_is are linear independent. Thus, by condition (3), we get that Y is essentially a solution to the system of linear equations given by $Q_1, ..., Q_t$.

Lemma 33. *Let $(X, Y, \mathcal{Q} = [Q_1, ..., Q_t])$ be ρ-structured, and let $x \in X$ be arbitrary. Let $S_i = \{i\} \times G_{x_i}$. Then, no linear combination of $Q_1, ..., Q_t$ equals a linear combination of $S_1, ..., S_n$, in which there is a set S_i such that $\rho(i) = *$.*

Proof. Assume the contrary. Let this linear combination be $Q_{j_1} + ... + Q_{j_l} = I$, where $j_1 < ... < j_l$. If there are many linear combinations, then choose 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 $I \cap \{i\} \times [m] \neq \varnothing$. By (5) it follows that $\rho(i) \neq *$, since S_i lies in $\mathcal{L}_i$. By (6), it is ensured that a linear combination $Q_{j_1} + ... + Q_{j_l} + S_i = I + S_i$ is also in the linear span of $\{Q_1, ..., Q_t\}$. $I + S_i$ is contained in blocks strictly higher than the i-th block and, by assumption, is equal to a linear combination of $S_1, ..., S_n$ containing a set S_i such that $\rho(i) = *$. Therefore, we obtained a contradiction with the maximality of j_1. □

Lemma 34 (Arbitrary Extension)**.** *Let $(X, Y, \mathcal{Q} = [Q_1, ..., Q_t])$ be ρ-structured. Then for any $z \in \{0,1\}^n$ that agrees with ρ (that is, $\rho(i) = z_i$ for all i, such that $\rho(i) \neq *$), there exist $x \in X, y \in Y$ such that*

$$\forall i, g(x_i, y_i) = z_i.$$

Proof. Consider any $x \in X$, and let $S_i = \{i\} \times G_{x_i}$. Let $F = \{i \mid \rho(i) = *\}$ be the set of free coordinates. Consider any i such that $\rho(i) = *$. By Lemma 33, it follows that S_i does not lie in the linear span of $\{Q_1, ..., Q_t\} \cup \{S_i \mid \rho(i) \neq *\}$. Thus, S_i is linearly independent of $\{Q_1, ..., Q_t\}$ and it follows that each of the conditions $y_{S_i} = 0$ and $y_{S_i} = 1$ is satisfied by some input. In particular, it follows that there exists $y \in Y$ such that $y_{S_i} = z_i$. Let $X' = \{x\}, Y' = \{y \in Y \mid y_{S_i} = z_i\}$, $\mathcal{Q}' \leftarrow \mathcal{Q} \cup \{S_i\}$, $\rho'(j) = z_j$ if $j = i$ and $\rho'(j) = \rho(j)$ otherwise. Note that ρ' contains one less $*$ than ρ. Thus, we can proceed by induction on ρ'-structured triplet $(X', Y', \mathcal{Q}')$. □

7.2 Description of the algorithm of T

Next we describe the simulation of Π by the tree T. We denote by X the set of available values for x, and by Y the set of available values for y. Let $Q_1, ..., Q_t$ be the queries made by Bob, ρ be a function such that $\rho(i) = z_i$ if we have queried z_i, and $\rho(i) = *$ if we have not yet queried it. Initially, $t = 0$, $X = [k]^n$, $Y = (\{0,1\}^m)^n$, $\rho = *^n$. We will also maintain a variable $F \subseteq [n]$, which initially equals $\varnothing$. It will be used later for complexity analysis, and it is equal to the subset of fixed coordinates. Over the course of the simulation some indices will be added to F. The pseudocode of the algorithm can be seen in Figure 1.

Suppose we are currently at vertex v of the protocol Π. Let v_0 and v_1 be the children of v.

It is easy to process the Alice's messages. Suppose Alice sends a message at a vertex v. Then X is divided into two parts: $X = X^0 \sqcup X^1$, corresponding to vertices v_0 and v_1. We descend to the vertex v_b, where X^b is at least half the size of X.

Now consider the case where Bob sends a message about y's parity of the set $Q \subseteq [m]^n$ at a vertex v. As with the randomized algorithm, we run it through the *sift* procedure. In this process, we also maintain the XOR of the messages that we subtract from it during the *sift* procedure. This XOR will be denoted by b. Thus, the value of Bob's original message is the value of transformed one XORed with b.

If after running Q through *sift* it becomes an empty set, then the value of Bob's original message can computed through his previous messages and equals b. Therefore, we simply descend to the vertex v_b.

If Q is non-empty, then we insert its copy into the sequence $Q_1, ..., Q_t$ in such a way that the invariant on *principal* variables is maintained. Let i be the block that contains $p(Q)$. Set $X^0 = \{x \in X \mid \{i\} \times G_{x_i} \in \mathcal{L}_i\}$. That is, X^0 contains all $x \in X$ such that the *secret* set $\{i\} \times G_{x_i}$ falls within the linear span L_i. Let $X^1 = X \setminus X^0$.

First of all, if $|X_1| < \frac{1}{100}|X|$, then we choose the most frequent element a of X_i and update $X := \{x \in X \mid x_i = a\}$. We add index i to F.

If the size of X^1 is at least $\frac{1}{100}$ of X or $\rho(i) \neq *$, then we replace X with X^1. We will show that X^1 cannot be empty. We will also show that the parity of y on Q can both 1 and 0 without violating any restrictions. For definiteness, we set it to 1, narrowing Y accordingly. The value of the original Bob's message equals to $b \oplus 1$. Therefore, we descend to the vertex $v_{b\oplus 1}$.

Now consider the case when the size of X^1 is less than $\frac{1}{100}$ and $\rho(i) = *$. In this case, $i \in F$ due to the *if* above, so $\{x_i \mid x \in X\}$ is a singleton. We query z_i and set $\rho(i) \leftarrow z_i$. We filter Y so that it satisfies $y_S = z_i$ to keep (X, Y) ρ-structured.

Since $\{i\} \times G_{X_i}$ falls into $\mathcal{L}_i$, there exist distinct $j_1, ..., j_l$ such that

$$(Q_{j_1} + ... + Q_{j_l}) \cap \{i\} \times [m] = S.$$

Moreover, Q must participate in this linear combination, which we will prove later. Therefore, we consider $Q = Q_{j_1}$. We set Q equal to $Q + Q_{j_1} + ... + Q_{j_l} + S$. At the same time, b needs to be

XORed with $Y_{Q_{j_1}} \oplus ... \oplus Y_{Q_{j_l}} \oplus z_i$. After this, we repeat the same procedure with Q, starting with *sift*.

7.3 Correctness of the algorithm of T

Lemma 35. $(X, Y, \mathcal{Q})$ - *ρ-structured is a while-loop invariant.*

First we show how the correctness of T follows from the above lemma before proving it. During the algorithm's execution, the invariant is maintained that $(X, Y, \mathcal{Q})$ is ρ-structured. Therefore, by Lemma 34 when the simulation reaches a leaf v, there exist $x \in X, y \in Y$ such that $\forall i\ g(x_i, y_i) = z_i$. Since $X \times Y$ are the inputs on which Π reaches v, the value of the leaf v will be equal to the $f(z)$. Hence, T outputs the correct answer.

Proof of Lemma 35. It's easy to verify that the invariant is maintained if Alice sends a message at a vertex v.

Therefore, it remains to check that the invariant holds when Bob sends a message at a vertex v.

We will show that all properties except for the 6th are met every time the algorithm enters the lines 14 or 2.

First, note that the principal coordinate invariant is maintained since we perform a *sift* on Q before we insert its copy into $Q_1, ..., Q_t$.

Second, we'll show that Y is always $\mathcal{Q}$-fixed. At line 31, Y is filtered because Q is added to $\mathcal{Q}$. At line 25, Y is filtered because S is contained in the linear span of $\mathcal{Q}$ after the set $Q + Q_{j_2} + ... + Q_{j_l} + S$ is processed.

Y is filtered to exactly comply the new sets added to $\mathcal{Q}$. Since $\mathcal{Q}$ is linearly independent, Y will have co-dimension t.

Next, note that (X, Y) is ρ-fixed. It suffices to note that when $\rho(i) \leftarrow z_i$ occurs, we appropriately filter down X and Y.

Let's show that the condition (1) is met. X cannot become empty at line 20. We check that it cannot become empty at line 31. It could only become empty if $\rho(i) \neq *$ and $|X^1| = 0$. However, this could not happen since $|X^0| = 0$ in this case due to properties (4) and (5).

Finally, let's show that (5) is met. Note that we either update $X \leftarrow X^1$, or $X \leftarrow \{x \in X^0 \mid x_i = a\}$.

We prove that in the first case, the invariant is met. If $\rho(i) = *$, then X^1 is simply the set of all $x \in X$, such that $\{i\} \times G_{x_i}$ does not lie in $\mathcal{L}_i$. Therefore, by definition, (5) is met. If $\rho(i) \neq *$, then $\{i\} \times G_{X_i}$ is already in $\mathcal{L}_i$, and hence, (5) is met.

We prove that in the second case, the invariant is met. In the second case, by the algorithm's description, $\rho(i)$ becomes not equal to $*$. Any $x \in X^0$ lies in $\mathcal{L}_i$. Therefore, (5) is met.

It remains to show that (6) is met after several runs of Q through *sift*.

Note that (6) can become violated only at line 24, where we change $\rho(i) \leftarrow z_i$. We need $\{i\} \times G_{X_i}$ to be represented by a linear combination of $Q_1, ..., Q_t$. Note that we add a new set $Q + Q_{j_2} + ... + Q_{j_l} + S$ at the line 27. Therefore, after it is processed and added to $\mathcal{Q}$, $S = \{i\} \times G_{X_i}$ will lie in the linear span of all Q_i. □

7.4 Number of queries made by T

To analyze the complexity of T, we introduce the following notation and parameters. For convenience we think of T as a tree, where we have vertices for all intermediate steps described in Figure 2. The

Tree T on input z:

1: Initialize: v = root of Π, $Q_1, ..., Q_t \subseteq [n] \times [m]$ — queries made by Bob, initially $t = 0$, $X = [k]^n$, $Y = (\{0,1\}^m)^n$, $\rho = *^n$, $F = \varnothing$.
2: **while** v is not a leaf [invariant: $(X, Y, [Q_1, ..., Q_t])$ — ρ-structured triple]
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 query that Bob communicates at v ▷ B1
6: $b \leftarrow 0$
7: **for** $i = 1..t$ **do**
8: **if** $p(Q_i) \in Q$ **then**
9: $Q \leftarrow Q + Q_i$; $b \leftarrow b \oplus Y_{Q_i}$
10: **end if**
11: **end for**
12: **if** $Q \neq \varnothing$ **then**
13: Insert Q in $Q_1, ..., Q_t$ so that condition (1) from ρ-Structured-Triple is satisfied
14: Let $i \in [n]$ — be the block, containing $p(Q)$
15: // $L_i = \{Q_j \cap \{i\} \times [m] \mid p(Q_j) \in \{i\} \times [m]\}$
16: // $\mathcal{L}_i$ denotes linear span of L_i without null vector
17: Let $X^0 = \{x \in X \mid \{i\} \times G_{x_i} \in \mathcal{L}_i\}$, $X^1 = X \setminus X^0$.
18: **if** $|X^1| < \frac{1}{100} \cdot |X|$ **then**
19: $a \leftarrow \arg\max_a |\{x \in X \mid x_i = a\}|$
20: $X = \{x \in X^0 \mid x_i = a\}$; $F \leftarrow F \cup \{i\}$ ▷ A3
21: **end if**
22: **if** $|X^1| < \frac{1}{100} \cdot |X|$ and $\rho(i) = *$ **then**
23: $S \leftarrow \{i\} \times G_a$
24: Query z_i and update $\rho(i) \leftarrow z_i$
25: $Y = \{y \in Y \mid y_S = z_i\}$
26: Find distinct $j_1, ..., j_l$ s.t. $Q = Q_{j_1}$ and $(Q_{j_1} + ... + Q_{j_l}) \cap \{i\} \times [m] = S$
27: $Q \leftarrow Q + Q_{j_2} + ... + Q_{j_l} + S$ ▷ B2
28: $b \leftarrow b \oplus Y_{Q_{j_2}} \oplus ... \oplus Y_{Q_{j_l}} \oplus z_i$
29: Go to the 7-th line of the algorithm
30: **else**
31: $X \leftarrow X^1$; $Y = \{y \in Y \mid y_Q = b \oplus 1\}$; $v \leftarrow v_1$ ▷ A2
32: **end if**
33: **else** $Q = \varnothing$
34: $v \leftarrow v_b$
35: **end if**
36: **else** Alice speaks at v
37: Let $X^0 \sqcup X^1 = X$ be partition according to Alice's function at v
38: Let $b \in \{0,1\}$ be such that $|X^b| \geq \frac{1}{2} \cdot |X|$
39: $X \leftarrow X^b$; $v \leftarrow v_b$ ▷ A1
40: **end if**
41: **end while**
42: return the value of the leaf v

Figure 2: The algorithm of the tree T on input z. The symbols $\triangleright A1, A2, A3, B1, B2$ denote types of actions. This classification is used in analyzing the number of queries made by T.

complexity of T is still measured in the number of queries to z. Suppose the computation is in a vertex v in T. Let X^v denote the set X when T reaches v. Let $x \sim U(X)$. Let

$$H^v := H(x) = \log_2 |X|,$$
$$H_i^v := H(x_i).$$

Lemma 36. *Let v correspond to the vertex of the protocol T where T queries whether $|X^1| < \frac{1}{100} \cdot |X|$. Then, if $H_i^v \geq \log_2 |\mathcal{L}_i^v| + \frac{1}{10} \log_2 k + \frac{1}{3}$, then $|X^1| \geq \frac{1}{100} \cdot |X|$.*

Proof. Consider $x \sim U(X)$. Let $S_i = \{i\} \times G_{x_i}$. Let 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, $\frac{|X^1|}{|X|} = P(S_i \notin \mathcal{L}_i^v) = P(S_i \neq Y)$, 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 lemma's statement. □

We introduce the deficiency of the distribution:

$$U_i^v = \begin{cases} \log_2 k, & i \notin F^v, \\ 0, & i \in F^v \end{cases}$$

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

Note that $D^v \geq 0$ 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 it is clear from the context.

Let A_1, A_2, A_3, B_1, B_2 represent the number of actions of the corresponding type that T performed on input z.

Observation 37. *After each action of type (A2), the entropy of X decreases by no more than $\log_2 100 \leq 7$.*

Proof. When the update $X \leftarrow X^1$ happens, there are two cases. Either $\rho(i) \neq *$ or $|X^1| \geq \frac{1}{100} \cdot |X|$.

In the first case, $x_i = x'_i$ for any $x, x' \in X$. So, $X^1 = X$ and entropy does not decrease at all. In the second case, the entropy decreases by no more than 7 by definition. □

Observation 38. *At least A_3 times U_i decreased from $\log_2 k$ to 0. During this decrease, D decreased by at least $\max\{0, \frac{9}{10} \log_2 k - 1/3 - \log_2 |\mathcal{L}_i|\}$.*

Proof. When an action of type (A3) happens, H_i becomes equal 0, and U_i changes from $\log_2 k$ to 0. Thus, D decreases by at least $\max\{0, \frac{9}{10} \log_2 k - 1/3 - \log_2 |\mathcal{L}_i|\}$. □

Observation 39. *At the end of the execution of T, $\sum_{i=1}^n |L_i|$ does not exceed $B_1 + B_2$.*

Proof. Q is being created during a message of type (B1) or a message of type (B2), so the total number of Q's is upper bounded by $B_1 + B_2$. □

Let's demonstrate how the desired bound follows from these lemmas. Note that $B_2 \le A_3$. Note that A_3 equals to the number of queries made by T. Therefore, the only thing we need to do is to upper bound A_3 by $O((A_1 + B_1)/\log_2 k)$, as the sum of $A_1 + B_1$ represents the number of actions performed by the protocol Π.

Lemma 40. *$A_3 = O((A_1 + B_1)/\log_2 k)$.*

Proof. A message of type (A1) increases D by no more than 1. A message of type (A2) increases D by no more than 7 according to Observation 37. Consequently, due to Observation 38,

$$\mathbb{E}\left(-A_3 \cdot \left(\frac{9}{10}\log_2 k - 1/3\right) + \sum_i \log_2 |\mathcal{L}_i| + A_1 + 7 \cdot A_2\right) \ge 0,$$

since D is always no less than 0. Applying Observation 39 and bounding $B_2 \le A_3$ we get

$$B_1 + A_1 + 7 \cdot A_2 \ge A_3 \cdot \left(\frac{9}{10}\log_2 k - 4/3\right).$$

If $k \ge 3$, then $\frac{9}{10}\log_2 k - 4/3 > 0$, obtaining the desired bound on A_3.

If $k = 2$, then we use the following argument.

Note that if, during an action of type (A3), it was true that $|L_i| \le 1$, then D decreases by at least $1 - (1/10 + 1/3) \ge \frac{1}{2}$. The number of i such that $|L_i| \ge 2$ does not exceed $\frac{B_1+B_2}{2}$ according to Observation 39. Consequently,

$$-\frac{1}{2} \cdot \left(A_3 - \frac{B_1 + B_2}{2}\right) + A_1 + 7 \cdot A_2 \ge 0.$$

Using $B_2 \le A_3$, we get

$$5/4 \cdot B_1 + A_1 + 7 \cdot A_2 \ge A_3 \cdot 1/4.$$

Since $A_2 \le B_1$, we obtain $A_3 = O(A_1 + B_1)$.

□

8 Deterministic Size and Subspace models

In this chapter, we will prove Theorem 9 for $\mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}$ and $\mathsf{sD}^{\mathsf{cc}}_{\to\oplus}$, as well as part of Theorem 11 about deterministic complexities.

8.1 Proof of the Theorem 9 part for $\mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}$

We begin by proving that $\log \mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n) = \Omega(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f))$.

To do this, we will consider a more detailed classification of type (B1) messages sent by Bob:

(B1') A message about the y's parity on Q, which reduces to $\varnothing$ at the end of its processing.
(B1") Other messages of type (B1).

The following lemma is an enhanced version of Observation 39.

Lemma 41. *At the end of the protocol execution, $\sum_i |L_i|$ does not exceed $B_1'' + B_2$.*

Proof. Since Q becomes $\varnothing$ after Bob's type (B1') message, (B1') does not contribute to $\sum_i |L_i|$. Therefore, the statement follows from Observation 39. □

Note that a message of type (B1') does not increase D. A message type (B1') query only triggers actions of types (B2) and (A3). However, for the proof of Lemma 40, we do not need an upper bound on B_2 and A_3. Therefore, in the proof of Lemma 40, we can use B_1'' instead of B_1.

Note that the algorithm reaches line 31 during a message of type (B1"). As indicated in the description of the algorithm, the y's parity of the set Q can be either 0 or 1. Therefore, let's modify line 31 to $v \leftarrow v_b$, where the size of the subtree v_b is smaller than that of v_{1-b}. After making this modification, we obtain that $B_1'' \le \log_2 Size_{root}$.

However, we also need to replace A_1 with something smaller in the proof of Lemma 40, as A_1 can be too large. Let us consider the situation when Alice sends a type (A1) message. Let's modify line 38, where the algorithm chooses b. According to Lemma 14, we can choose $b \in \{0,1\}$ such that $\log_2 Size_v - \log_2 Size_{v_b} \ge H - H'$, where H' denotes the entropy of x after Alice's message. Define AH_1 as the sum of $H - H'$ over all of Alice's messages.

In the proof of Lemma 40, we only use A_1 to estimate the growth of D. Therefore, we can replace A_1 with AH_1, as AH_1 precisely corresponds to the contribution of Alice's messages to D. Thus, we obtain an even stronger version of Lemma 40:

Lemma 42. $\mathbb{E}A_3 = O(\mathbb{E}(AH_1 + B_1'')/\log_2 k)$.

The proof of this lemma is fully analogous to Lemma 30, and therefore is omitted.

Now, from Lemma 42, it immediately follows that $\log \mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n) = \Omega(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f))$.

8.2 Proof of Theorem 9 for $\mathsf{sD}^{\mathsf{cc}}_{\to\oplus}$, and also Theorem 11 for $\mathsf{sD}^{\mathsf{dt}}_{\oplus}$ and $\mathsf{sizeD}^{\mathsf{dt}}_{\oplus}$

To prove the bound for $\mathsf{sizeD}^{\mathsf{dt}}_{\oplus}$ in Theorem 11, we will consider a modification of the semi-structured protocol described in Section 6 for $\mathsf{sizeR}^{\mathsf{dt}}_{\oplus}$. The subsequent proof for this model follows exactly the same reasoning as in Section 6. In the randomized model, B_1'' was bounded because Bob's message of this type had a $\frac{1}{2}$ probability of being either 0 or 1. In the deterministic model, we can choose the value directly by modifying the line 31 of the algorithm accordingly. Therefore, B_1'' is bounded in exactly the same way. AH_1 is also bounded similarly, as we described for the $\mathsf{sizeD}^{\mathsf{cc}}_{\to\oplus}$ model.

To prove that $\log \mathsf{sD}^{\mathsf{cc}}_{\to\oplus}(f \circ h^n) = \Omega(\log_2 k \cdot \mathsf{D}^{\mathsf{dt}}(f))$, let us consider the conditional complexity defined in Section 6. By modifying the line 31 of the algorithm, we can force an ≪Incorrect≫ value of the Bob's message. Repeating the proof from Section 6 for $\mathsf{sR}^{\mathsf{cc}}_{\to\oplus}$, we obtain a similar result for $\mathsf{sD}^{\mathsf{cc}}_{\to\oplus}$.

The same reasoning applies to $\mathsf{sD}^{\mathsf{dt}}_{\oplus}$. In Section 6, we simulated it using the semi-structured protocol Π and then proved bounds on AH_1 and B_1''. The same simulation and proof also work for $\mathsf{sD}^{\mathsf{dt}}_{\oplus}$.

9 Majority Gadget

In this section, we describe a more general version of a gadget reduction.

Proof of Lemma 12. For simplicity assume $m = 4 \cdot k$. Consider a $\mathrm{MAJ}_{4k}(x_1, ..., x_{4k})$ gadget. Define a new basis y, where $y_{2i+1} = x_{2i+1} + x_{2i+2}$ and $y_{2i} = x_{2i}$. Let us give variables y_{2i+1} to Alice, and variables y_{2i} to Bob for all i.

Partition variables $x_1, ..., x_{4k}$ into blocks of 4 elements each. For $i-$th block, Alice will either fix $y_{4i+1} = 1, y_{4i+3} = 0$ or $y_{4i+1} = 0, y_{4i+3} = 1$. In both of these cases, the $i-$th block of x-variables will strictly contain three 1's and one 0, or three 0's and one 1.

More precisely, if Alice fixes $y_{4i+1} = 1, y_{4i+3} = 0$, then the number of ones in the block will be determined by Bob's variable y_{4i+4}. If Alice does the other fix, then the number of ones in the block will be determined by y_{4i+2}.

Note that given such a restriction on Alice's variables, we have that

$$\mathrm{MAJ}_{4k}(x_1, ..., x_{4k}) = \mathrm{MAJ}_k(\mathrm{MAJ}_4(x_1, x_2, x_3, x_4), ..., \mathrm{MAJ}_4(x_{4k-3}, x_{4k-2}, x_{4k-1}, x_{4k})).$$

For each i, $\mathrm{MAJ}_4(x_{4i+1}, x_{4i+2}, x_{4i+3}, x_{4i+4}) = \textsc{Ind}_2(y_{4i+1}, y_{4i+2}, y_{4i+4})$ on the restricted subset of ys.

Thus, $\mathsf{R}^{\mathsf{dt}}_{\oplus}(\mathrm{MAJ}_{4k}) = \Omega(\mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}_k))$ and, more generally, for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$,

$$\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ \mathrm{MAJ}^n_{4k}) \gg \mathsf{R}^{\mathsf{cc}}_{\to\oplus}(f \circ \mathrm{MAJ}^n_k \circ \textsc{Ind}^{n\cdot k}_2) \gg \mathsf{R}^{\mathsf{dt}}(f \circ \mathrm{MAJ}^n_k) \gg k \cdot \mathsf{R}^{\mathsf{dt}}(f).$$

Here $\alpha \gg \beta$ denotes $\alpha = \Omega(\beta)$. The first inequality is due to Lemma 2 and the fact that the change of basis does not change parity decision tree complexity. The second inequality is Corollary 8 applied to $\textsc{Ind}_2$ gadget. The third inequality is explained in Appendix B. □

Proof of Theorem 13. As we have noted, the change of basis does not affect parity decision tree complexity. Thus,

$$\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ g^n) = \Omega(\mathsf{R}^{\mathsf{dt}}_{\oplus}(f \circ r^n \circ h^{n\cdot m}))$$

Since h has linear diversity ≥ 2, we can apply Theorem 11, lifting $f \circ r^n$ using h. □

10 Applications

10.1 Recursive Majority Function

Let $\mathrm{MAJ}^{\otimes 1}_3 := \mathrm{MAJ}_3$ 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

$$\mathrm{MAJ}^k_3(x_1, ..., x_{3^k}) = \mathrm{MAJ}_3(\mathrm{MAJ}^{k-1}_3(x_1, ..., x_{3^{k-1}}), \mathrm{MAJ}^{k-1}_3(x_{3^{k-1}+1}, ..., x_{2\cdot 3^{k-1}}), \mathrm{MAJ}^{k-1}_3(x_{2\cdot 3^{k-1}+1}, ..., x_{3^k}))$$

In Magniez et al. (2010), it is shown that

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

Using our results, we can show that $\mathsf{R}^{\mathsf{dt}}_{\oplus}(\mathrm{MAJ}^{\otimes k}_3) = \Omega(\mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}^{\otimes k}_3))$. Thus, parity queries do not help in computing $\mathrm{MAJ}^{\otimes k}_3$.

Due to the following lemma, we can use $\mathrm{MAJ}^{\otimes 2}_3$ as a gadget.

Lemma 43. $\mathrm{MAJ}^{\otimes 2}_3$ *has linear diversity* 2.

The proof of this lemma can be found in Appendix C.

Together with Theorem 11 this lemma implies $\mathsf{R}^{\mathsf{dt}}_{\oplus}(\mathrm{MAJ}^{\otimes k+2}_3) = \Omega(\mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}^{\otimes k}_3))$. Note that $\mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}^{\otimes k+2}_3) = O(\mathsf{R}^{\mathsf{dt}}(\mathrm{MAJ}^{\otimes k}_3))$, since we can compute $\mathrm{MAJ}^{\otimes k+2}_3$ by computing 9 times $\mathrm{MAJ}^{\otimes k}_3$. So, the desired bound is obtained.

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 Santha (1991) that R^{dt} of this function is at least $n^{0.7537\cdots}$, where n is the number of inputs. Since f can be used as a gadget, we translate this result to parity decision trees.

10.2 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 Bansal and Sinha (2020) 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 5, 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 Blais et al. (2015). They also relied on the result of Bansal and Sinha (2020). 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.

References

Nikhil Bansal and Makrand Sinha. k-forrelation optimally separates quantum and classical query complexity. *CoRR*, abs/2008.07003, 2020. URL https://arxiv.org/abs/2008.07003.

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. URL https://doi.org/10.4230/LIPIcs.ITCS.2023.14.

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

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. URL https://doi.org/10.1007/s00037-019-00190-7.

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. URL https://doi.org/10.1137/19M1310153.

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. URL https://doi.org/10.4230/LIPIcs.ITCS.2023.33.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0471241954.

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. URL https://doi.org/10.4086/toc.2020.v016a013.

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

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

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

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. URL https://doi.org/10.1137/17M1136869.

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. URL https://doi.org/10.4230/LIPIcs.STACS.2019.50.

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. URL https://doi.org/10.4230/LIPIcs.ITCS.2022.104.

Frédéric Magniez, Ashwin Nayak, Miklos Santha, and David Xiao. Improved bounds for the randomized decision tree complexity of recursive majority. *Electron. Colloquium Comput. Complex.*, TR10-192, 2010. URL https://eccc.weizmann.ac.il/report/2010/192.

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. URL https://doi.org/10.1145/3055399.3055478.

Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Comb.*, 19(3):403–435, 1999. doi: 10.1007/S004930050062. URL https://doi.org/10.1007/s004930050062.

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. URL https://doi.org/10.1109/FOCS.2016.51.

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. URL https://doi.org/10.1109/SCT.1991.160259.

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.

A Proof of Lemma 4

First, let's prove the following lemma.

Lemma 44. *Let $m \geq 10$, and let $f_1, ..., f_{2^{4/5 \cdot m}} : \{0,1\}^m \to \{0,1\}$ be functions satisfying (1) from Definition 1. Let $f : \{0,1\}^m \to \{0,1\}$ be a random function chosen uniformly from those that satisfy (1). Then $f \oplus f_i$ is non-constant $\forall i$ with probability of at least $\frac{1}{2}$.*

Proof. There are a total of $2^m - 1$ options for f. Of these, $2^{4/5 \cdot m}$ options are forbidden. Therefore, the desired probability is at least

$$1 - \frac{2^{4/5 \cdot m}}{2^m - 1} \geq 1/2.$$

□

Lemma 45. *For any i, the probability that g_i satisfies (1) is at least 2^{m-1-2^m}.*

Proof. There are a total of $2^m - 1$ linear non-constant functions, and a total of 2^{2^m} functions. Therefore, the desired probability is $(2^m - 1)/2^{2^m} \geq 2^{m-1-2^m}$ □

Proof of Lemma 4. Let's estimate the probability that g_i satisfies (1) from Definition 1. There are a total of $2^m - 1$ linear non-constant functions, and a total of 2^{2^m} functions. Therefore, the desired probability is $(2^m - 1)/2^{2^m} \geq 2^{m-1-2^m}$. Hence, the expected number of such i for which g_i is linear and non-constant is at least 2^{m-1}. By Markov's inequality, we obtain that the probability of having at least $2^{4/5 \cdot m}$ linear non-constant functions among g_i tends to 1. Denote by $I \subseteq [2^{2^m}]$ the indices of these non-constant linear functions. Let I' obtained from I by removing the largest indices if $|I| > 2^{4/5 \cdot m}$. Thus, $|I'| = 2^{4/5 \cdot m}$ with probability approaching to 1.

By Lemma 44 for each $i \in I'$, we have

$$P(g_i \neq g_j \ \forall \ j < i, j \in I' \mid g_1, ..., g_{i-1}, I) \geq \frac{1}{2}.$$

Thus, the expected number of i for which the condition is satisfied is at least $\frac{1}{2} \cdot 2^{4/5 \cdot m}$ conditioned on the event $|I| \geq 2^{4/5} \cdot m$. Applying Markov's inequaility again and the fact that $P(|I| \geq 2^{4/5} \cdot m) \to 1$, we get that with probability tending to 1 the number of i for which the condition is satisfied is at least $2^{2/3 \cdot m}$. Denote J to be the set of these indices. $\{g_i \mid i \in J\}$ satisfy (1) and (2), so linear diversity of g is at least $2^{2/3 \cdot m}$ with probability tending to 1.

□

B Finishing proof of Lemma 12

Here we explain why for any relation $f \subseteq \{0,1\}^n \times \mathcal{R}$, $\mathsf{R}^{\mathsf{dt}}(f \circ \mathrm{MAJ}_m^n) = \Omega(m \cdot \mathsf{R}^{\mathsf{dt}}(f))$ holds.

To demonstrate this, we use an approach similar to the *secret sets* technique. For simplicity, let $m = 2 \cdot k + 1$. Consider an assignment to MAJ_m where the first k inputs are equal to 1, and the remaining $k + 1$ inputs are equal 0. Assume we want MAJ_m function to output a bit $b \in \{0,1\}$. To achieve this, we randomly select a 0 in this assignment and change it to b. The new assignment

forces MAJ_m to output b. Moreover, a randomized decision tree must make $\Omega(k)$ queries on average to identify the hidden b in this assignment. By hiding each x_i in ith MAJ_n gadget, we can apply the same simulation method as in the *secret sets* technique. The randomized decision tree is forced to make at least $\Omega(k)$ queries in each coordinate on average to force us to query the value of a hidden variable. Thus,

$$\mathsf{R}^{\mathsf{dt}}(f \circ \mathrm{MAJ}_m^n) = \Omega(m \cdot \mathsf{R}^{\mathsf{dt}}(f)).$$

C Proof of Lemma 43

Proof. Consider the following 2 partial assignments of the input to $\mathrm{MAJ}_3^{\otimes 2}$:

(1) $(1, 0, b_1, 0, 0, b_2, 1, 1, 1)$
(2) $(0, 0, b_1, 1, 0, b_2, 1, 1, 1)$

Variables corresponding to b_1 and b_2 are given to Bob. Other variables are given to Alice.

In the first partial assignment the value of $\mathrm{MAJ}_3^{\otimes 2}$ equals b_1, while in the second it is equal to b_2. Thus, $\textsc{Ind}_2$ can be reduced to $\mathrm{MAJ}_3^{\otimes 2}$. Since $\textsc{Ind}_2$ has linear diversity 2, so has $\mathrm{MAJ}_3^{\otimes 2}$. □

D Full proof of Lemma 30

Proof. We consider two cases: $k \geq 100$ and $k = 2$. The case $2 < k < 100$ follows from the case $k = 2$.

Case $k \geq 100$. By Lemma 20 and Observation 29 messages of types (A1), (B1"), (A2') increase D by no more than $AH_1 + B_1'' + A_2'$ in total. Messages of type (B1') increase D by no more than B_2 in total by Observation 27. As a result, due to Observation 22 and Observation 23,

$$\mathbb{E}\left(-A_3' \cdot \left(\frac{9}{10}\log_2 k - 1/3\right) + \sum_i \log_2 |\mathcal{L}_i| + AH_1 + B_1'' + A_2' + B_2\right) \geq 0,$$

since D is always greater or equal than than 0. Applying Observation 21 and rearranging we get

$$\mathbb{E}\left(\sum_i \log_2 |\mathcal{L}_i| + 101 \cdot B_1'' + AH_1 + B_2\right) \geq \mathbb{E}A_3' \cdot \left(\frac{9}{10}\log_2 k - 1/3\right).$$

Note that $\sum_i \log_2 |\mathcal{L}_i| \leq \sum_i |L_i| \leq B_1'' + B_2$. Using the fact that $B_2 \leq A_3'$ and the inequality above, we obtain

$$\mathbb{E}\left(102 \cdot B_1'' + AH_1\right) \geq \mathbb{E}A_3' \cdot \left(\frac{9}{10}\log_2 k - 1/3 - 2\right).$$

Since $\frac{9}{10} \cdot \log_2 100 > 1/3 + 2$, we get $\mathbb{E}A_3' = O\left(\dfrac{\mathbb{E}AH_1 + B_1''}{\log_2 k}\right)$.

Thus, we obtain $\mathbb{E}A_3' = O(\mathbb{E}(AH_1 + B_1''))$. □

Case $k = 2$. To handle this case, we need a more careful approach. First, we are going to modify the 19th line of the modified protocol $\overline{\Pi}_z$ (Figure 1):

$$\textbf{if } H(S_i) < \frac{1}{10}\log_2 k + \frac{1}{3} \textbf{ or } |L_i| \geq 2 \textbf{ then}$$

With this modification, S_i will be fixed not only if S_i has low entropy, but also if Q is not the first set in the ith block. Note that $\log_2 |\mathcal{L}_i| = 0$ if $|L_i| = 1$. Thus, the first part of the statement is the same as before.

We also introduce another modification. Bob will send a message of type (B1) after the processing of Q. See Figure 3. The 7th line of Figure 1 is moved to the 31th line of Figure 3. Since, during the processing of Q, we subtract from Q sets for which we know y's XOR on them, sending y's XOR on the initial set Q, is the same as sending y's XOR on the modified Q.

The definition of deficiency D and the types of messages (A1), (A2'), (A2"), (A3'), (A3"), (B1'), (B1") are the same as before. AH_1 is also defined in the same way. However, we must introduce a new variable L_3.

Let L_3 be the number of times the left part of the *if* on the 18th line was false and the right part was true.

We show the following bound on L_3.

Observation 46. $L_3 \leq A'_2$.

Proof. Consider a moment when L_3 increases by one. Since $H(S_i) \geq \frac{1}{10}\log_2 k + \frac{1}{3}$, then at some previous step Alice sent a message of type (A2') in the ith block. Note that L_3 could increase by one no more than once for each block i because after S_i is fixed, the first part of the 18th line's *if* will be always true. Thus, L_3 cannot exceed A'_2. □

We also restate observations that were stated in Section 5.4.3 and Section 6.

Observation 47. $\mathbb{E}A'_2 \leq 100 \cdot \mathbb{E}B''_1$.

Proof. The proof is the same as the one for Observation 21. □

Observation 48. *Each message of type (B1') does not change D.*

Proof. Since Q eventually reduced to $\varnothing$, the Bob's message has not revealed any new information. □

Observation 49. *Messages of types (B2), (A2"), (A3") do not change D.*

Proof. The proof is analogous to the one for Observation 23. □

Observation 50. *Alice's messages of type (A1) increase D by at most AH_1.*

Proof. The proof is the same to the one for Observation 29. □

Lemma 51. $A'_3 = O(AH_1 + B''_1)$.

Proof. A message of type (A3') decreases D by at least $\frac{9}{10}\log_2 k - 1/3$ on average except when it has happened due to the second part being true and the first part being false in the 18th line's *if*. Thus, D has decreased by $\frac{9}{10}\log_2 k - 1/3$ on average at least $A'_3 - L_3$ times.

Messages of types (A1), (B1"), and (A2') increase D by no more than $AH_1 + B''_1 + A'_2$. Thus, since D is non-negative, we get

$$\mathbb{E}\left(-(A'_3 - L_3) \cdot \left(\frac{9}{10}\log_2 k - 1/3\right) + AH_1 + B''_1 + A'_2\right) \geq 0.$$

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: **for** $i = 1..t$ **do**
8: **if** $p(Q_i) \in Q$ **then**
9: $Q \leftarrow Q + Q_i$
10: **end if**
11: **end for**
12: **if** $Q \neq \varnothing$ **then**
13: Insert a copy of Q in $Q_1, ..., Q_t$ so that invariant holds
14: Let $i \in [n]$ be the block containing $p(Q)$
15: // $L_i = \{Q_j \cap \{i\} \times [m] \mid p(Q_j) \in \{i\} \times [m]\}$
16: // $\mathcal{L}_i$ denotes linear span of L_i without null vector
17: // $S_i = \{i\} \times G_{x_i}$ — a *secret* set in ith block
18: **if** $H(S_i) < \frac{1}{10} \log_2 k + \frac{1}{3}$ **or** $|L_i| \geq 2$ **then**
19: ▷ Alice sends S_i; S_i is *fixed* now ▷ (A3)
20: **end if**
21: ▷ Alice communicates whether S_i is in $\mathcal{L}_i$ ▷ (A2)
22: **if** $S_i \in \mathcal{L}_i$ and S_i is not *revealed* **then**
23: ▷ Alice sends S_i; S_i is *fixed* now ▷ (A3)
24: S_i is considered to be *revealed*
25: ▷ Bob sends y_{S_i} ▷ (B2)
26: Find distinct $j_1, ..., j_l$, s.t. $Q = Q_{j_1}$ and
$(Q_{j_1} + ... + Q_{j_l}) \cap \{i\} \times [m] = S_i$
27: $Q \leftarrow Q + Q_{j_2} + ... + Q_{j_l} + S_i$
28: Go to 8−th line
29: **end if**
30: **end if**

31: ▷ Bob sends b and we update $v \leftarrow v_b$ ▷ (B1)
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 3: A slight modification of Figure 1. The 18th line *if*'s condition is weakened, and Bob is now sending a message of type (B1) after the red part.

Applying $L_3 \leq A'_2 \leq 100 \cdot B''_1$,

$$\mathbb{E}\left(-A'_3 \cdot \left(\frac{9}{10}\log_2 k - 1/3\right) + AH_1 + 201 \cdot B''_1\right) \geq 0.$$

By rearranging, we get

$$\mathbb{E}\left(AH_1 + 201 \cdot B''_1\right) \geq A'_3 \cdot \left(\frac{9}{10}\log_2 k - 1/3\right).$$

□

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il