Electronic Colloquium on Computational Complexity, Report No. 37 (2024)

Lifting dichotomies

Yaroslav Alekseev[1], Yuval Filmus[2], and Alexander Smal[3]

[1]Technion — Israel Institute of Technology, Israel, tolstreg@gmail.com
[2]Technion — Israel Institute of Technology, Israel, yuvalfi@cs.technion.ac.il
[3]Technion — Israel Institute of Technology, Israel, avsmal@gmail.com

February 26, 2024

Abstract

Lifting theorems are used for transferring lower bounds between Boolean function complexity measures. Given a lower bound on a complexity measure A for some function f, we compose f with a carefully chosen *gadget* function g and get essentially the same lower bound on a complexity measure B for the *lifted* function $f \diamond g$. Lifting theorems have a number of applications in many different areas such as circuit complexity, communication complexity, proof complexity, and etc. One of the main question in the context of lifting is how to choose a suitable gadget g. Generally, to get better results, i.e., to minimize the losses when transferring lower bounds, we need the gadget to be of a constant size (number of inputs). Unfortunately, in many settings we know lifting results only for gadgets of size that grows with the size of f, and it is unclear whether it can be improved to a constant size gadget. This motivates us to identify the properties of gadgets that make lifting possible.

In this paper, we systematically study the question "For which gadgets does the lifting result hold?" in the following four settings: lifting from decision tree depth to decision tree size, lifting from conjunction DAG width to conjunction DAG size, lifting from decision tree depth to parity decision tree depth and size, and lifting from block sensitivity to deterministic and randomized communication complexities. In all the cases, we prove the complete classification of gadgets by exposing the properties of gadgets that make lifting results hold. The structure of the results shows that there is no intermediate cases——for every gadget there is either a polynomial lifting or no lifting at all. As a byproduct of our studies, we prove the log-rank conjecture for the class of functions that can be represented as $f \diamond \mathrm{OR} \diamond \mathrm{XOR}$ for some function f.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC. Alexander Smal has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 852870-ERC-SUBMODULAR.

ISSN 1433-8092

Contents

1 Introduction

For functions $f\colon \{0,1\}^n \to V$ and $g\colon \{0,1\}^m \to \{0,1\}$, a *(block-)composition* $f \diamond g\colon \{0,1\}^{n\times m} \to V$ is defined by

$$(f \diamond g)(z_1, z_2, \ldots, z_n) := f(g(z_1), g(z_2), \ldots, g(z_n)),$$

where each $z_i \in \{0,1\}^m$. Usually *lifting theorems* have the following general form:

$$B(f \diamond g) = \Omega(A(f)),$$

where A and B are two complexity measures. Note that the hidden constant in $\Omega(\cdot)$ may depend on g. In this context, we call the function g a *gadget* and say that *there is a (linear) lifting from A to B*. One of the first examples of a lifting theorem appeared in [RM99] where Raz and McKenzie proved a separation for the hierarchy of monotone circuit complexity classes within NC using a query-to-communication lifting theorem.

The need for such theorems is due to the fact that, in many cases, communication complexity lower bounds are much more difficult to prove compared to query complexity lower bounds. Lifting theorems proved to be useful in many other scenarios: proving communication complexity separations [GP18, GPW18, GPW17, CKLM19], proof complexity separations [GLM+16, HN12, GP18, dRMN+20], monotone circuit complexity separations [RM99, GGKS20], and etc.

While the lifting technique is widely used in different areas we still do not quite understand its limitations. In the query-to-communication lifting theorems, we want to lower bound the deterministic/randomized communication complexity of the lifted function by deterministic/randomized query complexity of the original function. So, these theorems usually have the following form:

$$\mathrm{D}(f \diamond g) = \mathrm{DT}(f) \cdot \Theta(\log n),$$

where the gadget g has at most logarithmic communication complexity. In all the lifting theorems of this type, the *size* (the number of inputs) of the gadget g grows with the number of inputs of f (e.g., in [LMM+22] the size of the gadget is $n^{1+\varepsilon}$, where n is a number of inputs of the lifted function). Even though some results do not depend on the gadget size (e.g., [GPW18]), in many scenarios the use of non-constant size gadgets leads to weaker results in the applications (e.g., when lifting theorems are used to prove monotone circuit lower bounds via communication complexity). It is unknown whether it is possible to prove a query-to-communication lifting theorem with a constant size gadget, but we tend to believe that such lifting exists.

At the same time, for other complexity measures there are lifting theorems that can accommodate constant size gadgets. Here are some examples of such lifting theorems:

1. Lifting decision tree depth to decision tree size (XOR gadget, see [Urq11]).
2. Lifting decision tree depth to parity decision tree depth/size (stifling gadgets, see [CMSS22]).
3. Lifting from critical block sensitivity to communication complexity (VER gadget, see [GP18]).
4. Lifting parity decision tree depth to communication complexity (XOR gadget, see [HHL18]).
5. Lifting AND decision tree depth to communication complexity (AND gadget, see [KLMY21]).
6. Lifting block sensitivity to randomized communication complexity (AND gadget, see [Zha09]).

Note that in examples 4 and 5, the communication complexity is lower bounded by *some power* of the parity decision tree complexity or the AND decision tree complexity (in the latter case, with an additional $\log n$ factor). In such cases, we say that there is a *polynomial lifting*.

As mentioned earlier, lifting theorems which lower bound communication complexity by a *linear* function of query complexity are only known for non-constant size gadgets. But in case of a polynomial lifting, there are query to communication complexity lifting theorems with constant size gadgets. E.g., in example 6 the author lifts block sensitivity to randomized communication complexity with a constant size gadget. Given that block sensitivity is polynomially related to query complexity, this gives a polynomial lifting from query to communication complexity.

All the examples of the lifting theorems with constant size gadgets that we mentioned above use specific and simple gadgets. But what happens if we plug some other gadget? The same proof might not work, but would the lifting result still be true? That is not always clear. And this reflects our lack of understanding of what properties of the gadget make lifting possible. For these theorems it is natural to pose the following question:

For which gadgets does the lifting result hold?

A systematical study of this question will help us to better understand how lifting works and what are the requirements for the gadget. Especially it would be interesting to understand for which gadgets lifting fails. This is what we need, for example, if we want to find a good candidate for query-to-communication linear lifting with constant size gadget.

One of the areas that might benefit from new lifting theorem is the study of *the log-rank conjecture* [LS88]. The log-rank conjecture states that for any function f the deterministic communication complexity of a function is polynomially related to the logarithm of the real rank of its associated communication matrix. Currently there is a exponential gap between the lower bound $\Omega\left(\log^2(\mathrm{rank}(f)\right)$ [GPW18] and the upper bound $O\left(\sqrt{\mathrm{rank}(f)} \cdot \log \mathrm{rank}(f)\right)$ [Lov16]. It seems that the general case of this problems is out of reach now. So most of research in this area has concentrated on the study of various complexity measures and special cases such as composed functions (e.g., see [KLMY21]). Therefore, lifting in this area is one of the main proof techniques.

The other area craving for new lifting theorems is proof complexity. There is a tight connection between proof complexity and lifting theorems. For example, lifting decision tree depth to parity decision tree depth/size from [CMSS22] provides a systematic way to prove tree-like Res($\oplus$) size lower bounds. It is important to mention that for proof complexity we usually need lifting theorems that hold for relations.

Finally, we think that a large number of different applications makes lifting a technique that is worth exploring on its own. A deeper understanding of how lifting works and what it requires from gadgets can lead to new applications and results.

1.1 Our results and methods

In this paper, we systematically explore the question "*For which gadgets does the lifting result hold?*" in several different settings. We prove complete classifications of gadgets in the following four settings:

- lifting from decision tree depth to decision tree size,
- lifting from certificate complexity to conjunction DAG size,

- lifting from decision tree depth to parity decision tree depth and size,
- lifting from block sensitivity to deterministic and randomized communication complexities.

All these results are formulated in the form of dichotomies (a trichotomy in one of the cases) that essentially state that there is either polynomial lifting or no lifting at all (no intermediate cases). As a byproduct of our studies, we prove the log-rank conjecture for the class of functions that can be represented as $f \diamond \mathrm{OR} \diamond \mathrm{XOR}$ for some function f.

Now we describe the results in more detail and give an overview of the proof methods.

1.1.1 Decision tree depth to size

In Section 3, we define a class of *resistant* gadgets (defined in Section 3.1) and prove a decision tree depth to decision tree size lifting theorem for this class of gadgets (see Theorem 3.2). We give two different proofs illustrating the ideas of two different approaches: proof by simulation and proof using random projections. The proofs later in the paper refer to the proofs in this simple case. The proof by simulation is constructive—it shows how given a decision tree for the lifted function $f \diamond g$ one can construct a decision tree for the original function f, such that the depth of the new tree is bounded by a logarithm of the size of the given tree. In the proof using random projections, we use probabilistic method to show that there is a projection that converts the decision tree for $f \diamond g$ into a shallow decision tree for f.

Our goal is to prove a classification theorem, so we need to find a class of gadgets such that lifting works only for gadgets in this class. It appears that the class of resistant gadgets is too small for this. We define a wider class of *weakly resistant* gadgets (defined in Section 3.2) and prove a certificate complexity to decision tree size lifting theorem (see Theorem 3.4). The proof uses the random projections method. Note that the decision tree size is upper bounded by the certificate complexity squared, so as a corollary we get a polynomial lifting from decision tree depth to decision tree size (see Corollary 3.5).

Finally, we give a complete classification of gadget functions in the context of polynomial lifting from decision tree depth to decision tree size. It is stated as a dichotomy result (see Theorem 3.9): either a gadget is weakly resistant and there is a polynomial lifting or there is no lifting at all.

In Section 3.4, we briefly discuss that some of the results above can be generalized to the case of search problems. Unfortunately, that does not give a classification theorem because decision tree depth of a search problem can be exponentially greater than its certificate complexity. However, we state a conjecture there is a decision tree depth to decision tree size polynomial lifting for weakly resistant gadgets.

1.1.2 Conjunction DAG width to size

In Section 4, we generalize the results from the previous section to decision conjunction DAGs (defined in Section 4.1). First of all, in Section 4.2, we show that similarly to decision trees where depth and size are exponentially separated, there is an exponential separation between conjunction DAG width and size. The separation is achieved for the Tribes function. In Section 4.3, we show that there is an exponential separation between decision tree size and conjunction DAG size in the case of relations. Indeed, conjunction DAGs can capture the structure of Resolution proofs, while decision trees can only capture the structure of tree-like Resolution proofs. Thus, the separation between these proof systems implies the separation between the measures under consideration. Finally, we argue that the lifting theorems from Section 3 can be generalized to the case of conjunction DAGs (see Theorem 4.1 and Theorem 4.2) using essentially the same proofs. Thus, we have a dichotomy result (see Theorem 4.3).

1.1.3 Decision tree depth to parity decision tree depth and size

Recently, Chattopadhyay et al. [CMSS22] showed that if g is *stifling* (defined in Section 5.1) then $\log \mathrm{DTSize}_{\oplus}(f \diamond g) = \Theta(\mathrm{DT}(f))$. In Section 5, we extend their result providing the complete classification of the gadgets for decision tree depth to parity decision tree depth and size lifting. In Section 5.2, we state and prove a "minimum weight lemma" (see Lemma 5.3), the technical lemma that we will use several times. This lemma states that if the certificate complexity of some function f is large enough in comparison to the parity certificate complexity of the lifted function $f \diamond g$ at some input, then there is a substitution such that the minimal parity certificate of $f \diamond g$ at this input has a large Hamming weight.

In Section 5.3, we consider the most challenging case of this setting—the case of OR gadget. In Theorem 5.4, we show that there is at least a quadratic gap in the certificate to parity certificate complexity lifting for OR gadget. If we assume that the lifting result in [CMSS22] holds for OR gadget as well then the quadratic separation is tight (see Proposition 5.5). In Section 5.3.2, we show a polynomial (cubic) lifting for OR gadget (see Theorem 5.6). The proof consists of three ingredients. First, we show that if the minimum weight of a parity certificate for the lifted function is at least the size of the parity certificate then this certificate covers the all-1 input for the inner function (see Lemma 5.7). Then we use it to upper bound the minimum weight of a parity certificate in terms of the sizes of parity certificates for 0 and 1 (see Lemma 5.8). And finally, we apply the "minimum weight lemma" (see Lemma 5.9). In the proof of Theorem 5.6, we assume that the certificate complexity of the original function is much larger than the parity certificate complexity of the lifted function, but due to the "minimum weight lemma" it would contradict the upper bound on the minimum weight of the parity certificate.

In Section 5.4, we prove a lifting from certificate complexity to parity decision tree size for AND/OR gadgets, the gadgets that affine project to both binary AND and binary OR (see Theorem 5.11). The proof is by simulation similar to the simulation proof in [CMSS22].

Finally, in Section 5.5, we prove the trichotomy result that gives us a complete classification of gadgets (see Theorem 5.14). The classification is based on the fact that if a gadget is not AND/OR then it is a disjunction or a conjunction of affine forms (see Lemma 5.13). We show that there are only three cases possible: (1) there are simultaneously a lifting from decision tree depth to parity decision tree depth and a lifting from certificate complexity to parity certificate complexity (the case of AND/OR gadgets); (2) there is a lifting from decision tree depth to parity decision tree depth but no lifting between certificate complexities (the case of gadgets that affine project to OR); (3) there is no lifting (all other gadgets, constant or affine).

In Section 5.6, we show an alternative proof for the lifting from decision tree depth to parity decision tree depth using function degree and sparsity (see Theorem 5.15). In the proof we show that the degree of a function is upper bounded by the logarithm of the sparsity of the function lifted with OR gadget, and compose it with a number of previously known inequalities. As a byproduct, we get a proof of the log-rank conjecture for the class of functions that can be represented as $f \diamond \mathrm{OR} \diamond \mathrm{XOR}$ (see Theorem 5.16).

1.1.4 Block sensitivity to communication complexity

In Section 6, we provide a complete classification of gadgets for lifting from block sensitivity to deterministic and randomized communication complexities. Note that this classification also gives us a classification of gadgets for query-to-communication polynomial lifting since block sensitivity and query complexity are polynomially related to each other in the case of total functions. However, since the proof goes through block sensitivity, we classify block sensitivity to communication complexity lifting.

We start with the lifting theorem of Zhang [Zha09] (see Theorem 6.1) that works for gadget g iff both AND and OR reduce to g via a communication complexity reduction (defined in Section 6.1). Then we show that if there is no reduction from OR to a gadget g then the communication matrix of g is (up to rearrangement) block diagonal (see Lemma 6.3). This gives us a classification of gadgets in the context of reductions from OR, AND, and XOR (see Corollary 6.4 and Corollary 6.5).

In Section 6.2, we prove a dichotomy result for randomized communication complexity (see Theorem 6.6). The proof is based on the fact that if OR does not reduce to a gadget g then $\mathrm{AND}_n \diamond g$ is essentially an instance of the equality function, and hence its randomized communication complexity is logarithmic (see Lemma 6.7). If AND does not reduce to a gadget, the situation is symmetrical. Thus, the theorem of Zhang covers all gadgets for which there is a lifting.

In Section 6.3, we prove the dichotomy result for deterministic communication complexity (see Theorem 6.9). The proof of the dichotomy depends on a block sensitivity to deterministic communication complexity lifting theorem that works for gadget g iff both AND and XOR reduce to g (see Theorem 6.8). Together with the Zhang's lifting theorem that give us a complete classification of gadgets: if at least two of three functions AND, OR, XOR reduce to gadget g then there is a block sensitivity to deterministic communication complexity polynomial lifting. Otherwise, if at least two functions from this list do not reduce to g then g is either a constant function or (essentially) one of the functions AND, OR, XOR. The proof of the lifting theorem requires a number of ingredients. First of all, we compose two results from [HHL18, ZS10] to get a parity certificate to deterministic communication complexity polynomial lifting for XOR gadget (see Corollary 6.10). Then we show two lower bounds for specific cases: we show that the deterministic communication complexity of $f \diamond \mathrm{XOR}$ is lower bounded by the size of any minimal sensitive block of the function f (see Lemma 6.11), and that the deterministic communication complexity of $f \diamond \mathrm{AND}$ is lower bounded by the block sensitivity of f at the all-0 input (see Lemma 6.12). One of the ingredients is again the "minimum weight lemma" (see Lemma 6.13). And the last one, is the lemma that shows that parity certificates of high minimum weight always intersect with (regular) certificates of sufficiently small size (see Lemma 6.14). Putting all the ingredients together we get the desired lifting result.

2 Prerequisites

2.1 Notation

All the logarithms are base 2. We use OR_n, AND_n, XOR_n to denote, respectively, logical 'and', 'or' and 'exclusive or' of n Boolean inputs. For simplicity we also define $\mathrm{OR} := \mathrm{OR}_2$, $\mathrm{AND} := \mathrm{AND}_2$, and $\mathrm{XOR} := \mathrm{XOR}_2$. We use the notation x^B for $x \in \{0,1\}^n$ and $B \subseteq [n]$ to denote x with all the coordinates in B flipped, i.e., $x_i = x_i^B \iff i \notin B$. For $g\colon \{0,1\}^k \to \{0,1\}$, we define[1] a function $g^n\colon \{0,1\}^{n\times k} \to \{0,1\}^n$ such that

$$g^n(x_1, x_2, \ldots, x_n) := (g(x_1), g(x_2), \ldots, g(x_n)).$$

For a function $f\colon \{0,1\}^n \to \{0,1\}$, an input $x \in \{0,1\}^n$, and a partial assignment $\sigma \in \{0,1,*\}^n$ that agrees with x on all non-$*$ coordinates, we use $f|_\sigma$ to denote a restriction of f to σ and $x|_\sigma$ to denote a projection of x to the $*$-coordinates of σ. For a gadget $g\colon \{0,1\}^k \to \{0,1\}$ and a *blockwise* partial assignment $\sigma \in \left(\{0,1\}^k \cup \{*^k\}\right)^n$ (i.e., in any block of variables that corresponds to one copy of g, the variables are either all set or all stars), we use $g^n(\sigma)$ to denote the partial assignment in $\{0,1,*\}^n$ induced by applying g to non-$*$ blocks of σ. We use '$\sqcup$' instead of '$\cup$' to indicate a union of disjoint sets.

2.2 Complexity measures

Throughout the text, we will consider several complexity measures.

Decision tree complexity A *decision tree* is a rooted binary tree with internal nodes labeled by input variables (represent queries), edges labeled with 0 and 1, and leaves labeled by some values. Decision tree evaluation is defined in the natural way: given an assignment for the input variables, we traverse the tree starting from the root and then sequentially choose every next edge according to the given assignments until we reach some leaf. The label of this leaf is the result of the evaluation. A decision tree computes some function f if for all possible assignments to the input variables the decision tree evaluates to the value of the function. For a function f, $\mathrm{DT}(f)$ denotes the minimal depth of a decision tree computing f, and $\mathrm{DTSize}(f)$ denotes the minimal number of leaves in a decision tree computing f.

Parity decision tree complexity A *parity decision tree* is a generalization of a decision tree where the queries are arbitrary linear combinations of the input variables. It can be described as a rooted binary tree with internal nodes labeled by linear combinations of the input variables, edges labeled with 0 and 1, and leaves labeled by some values. The evaluation is defined analogously. For a function f, $\mathrm{DT}_\oplus(f)$ denotes the minimal depth of a parity decision tree computing f, and $\mathrm{DTSize}_\oplus(f)$ denotes the minimal number of leaves in a parity decision tree computing f.

Certificate complexity A *certificate complexity* is a non-deterministic analogue of the decision tree complexity. A *certificate* for an input $x \in \{0,1\}^n$ to a function f is a set $S \subseteq [n]$ of indices such that f restricted to all inputs that match x on S is constant, i.e., $f(y) = f(x)$ whenever $y|_S = x|_S$. The certificate complexity $\mathrm{C}(f,x)$ of f at input x is the size of the smallest certificate for x. Finally, the certificate complexity of function f is defined as $\mathrm{C}(f) := \max_{x\in\{0,1\}^n} \mathrm{C}(f,x)$.

Parity certificate complexity A *parity certificate* for an input $x \in \{0,1\}^n$ to a function f is a set A of linear forms such that f is constant on the affine subspace defined by $A = A(x)$. A *parity certificate complexity* $\mathrm{C}_\oplus(f,x)$ of function f at input x is the size (number of linear forms) of the smallest parity certificate for x. The parity certificate complexity of f is defined as $\mathrm{C}_\oplus(f) := \max_{x\in\{0,1\}^n} \mathrm{C}_\oplus(f,x)$.

[1]In some papers lifting theorems are formulated for the classical composition operation '$\circ$' rather then for the block-composition '$\diamond$'. Note that for $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^k \to \{0,1\}$, $f \diamond g \equiv f \circ g^n$.

Block sensitivity A block $B \subseteq [n]$ is *sensitive* for a function f at x iff $f(x) \neq f(x^B)$. The *block sensitivity* $\mathrm{bs}(f, x)$ of f at x is the maximum number of disjoint sensitive blocks for f at x. The block sensitivity $\mathrm{bs}(f)$ of f is the maximum sensitivity $\mathrm{bs}(f, x)$ over all points $x \in \{0,1\}^n$.

Degree and sparsity of the function Every Boolean function defined on $\{0,1\}^n \to \{0,1\}$ can be considered as a function $f\colon \{-1,1\}^n \to \{-1,1\}$ by substituting 0 and 1 with 1 and -1, respectively. Every such function has a unique representation as a multilinear polynomial over $\mathbb{R}$ of the following form

$$f(x) = \sum_{S \subseteq [n]} c_S x_S, \quad \text{where } x_S = \prod_{i \in S} x_i.$$

Degree of f is defined to be the degree of the corresponding polynomial, we denote it by $\deg(f)$. *Sparsity* of f is the number of nonzero coefficients in this representation, we denote it by $\mathrm{spar}(f)$.

Deterministic communication complexity and rank For a function $f\colon X \times Y \to Z$, lets consider the following game for two players, Alice and Bob, that want to compute f. Alice and Bob are given $x \in X$ and $y \in Y$, respectively. Their goal is to compute $f(x, y)$. In order to do it, the players exchange information about their parts of the input using a simple communication channel that allows sending bit messages. Before the game the players come up with a *communication protocol* that determines their behavior on all possible inputs. The cost of a protocol is the maximum total number of bits sent by the players over all possible inputs $x \in X$, $y \in Y$. The *deterministic communication complexity* of f is the minimal cost of a deterministic communication protocol that computes f. We denote it by $\mathrm{D}(f)$.

A *communication matrix* of the function f is a matrix $M_f \in Z^{X \times Y}$ such that $(M_f)_{x,y} := f(x, y)$ for all $x \in X$, $y \in Y$. We define $\mathrm{rank}(f)$ to be the rank of matrix M_f.

Randomized communication complexity In a *randomized communication game*, Alice and Bob have access to an unlimited amount of random bits and they can use it to decide which bit to send next. We say that a *(private coin) randomized communication protocol* Π computes a function $f\colon X \times Y \to Z$ with error ε if

$$\Pr_{r_A, r_B}\big[\Pi(x, y, r_A, r_B) = f(x, y)\big] \geq 1 - \varepsilon, \quad \forall x \in X,\ y \in Y,$$

where r_A and r_B are the strings of random bits used by Alice and Bob, respectively. The cost of a randomized protocol is the maximum number of bits that can be sent. We denote by $\mathrm{R}_\varepsilon(f)$ the minimal cost of a private coin randomized communication protocol that computes f with error ε. A *randomized communication complexity* of f is defined as $\mathrm{R}(f) := \mathrm{R}_{1/3}(f)$.

3 Decision tree depth to size

We start by exploring perhaps one of the simplest scenarios, depth-to-size lifting in decision trees. To give a flavor of the techniques that we are going to use in this paper, we prove a lifting theorem for the class of *resistant* gadgets.

3.1 Resistant gadgets

Urquhart [Urq11] proved that for any function $f\colon \{0,1\}^n \to \{0,1\}$,

$$\log \mathrm{DTSize}(f \diamond \mathrm{XOR}) = \Omega(\mathrm{DT}(f)).$$

We generalize this result for the class of *resistant* gadgets.

Definition 3.1. A gadget $g\colon \{0,1\}^m \to \{0,1\}$ is *resistant* if for every $i \in [m]$ and $b \in \{0,1\}$, the function obtained by fixing the ith input to b is not constant. Equivalently, the minimum certificate complexity at inputs is larger than 1. E.g., XOR function is resistant while AND function is not.

Theorem 3.2. *For any function $f\colon \{0,1\}^n \to \{0,1\}$ and a resistant gadget $g\colon \{0,1\}^m \to \{0,1\}$,*

$$\log \mathrm{DTSize}(f \diamond g) = \Omega(\mathrm{DT}(f)).$$

We give two proofs of this theorem using two different approaches: *simulation* and *random projections*. The first proof is more clear and in some sense constructive, but the second proof can be generalized to other settings. In the following proofs, we use $x_1, \ldots, x_n$ to refer to the inputs of function f. The inputs to the lifted function $f \diamond g$ are denoted by $z_{i,j}$, where $i \in [n]$ and $j \in [m]$.

Proof by simulation. Given a decision tree T for $f \diamond g$ of size S, we construct a decision tree for f of depth at most $\log S$. We traverse T from the root to a leaf and maintain a partial assignment to z which is consistent with the input x, in the sense that the partial assignment can be extended to an input z which satisfies $x = g^n(z)$ (i.e., $x_i = g(z_{i,1}, \ldots, z_{i,m})$). This ensures that if the leaf is labelled b then $f(x) = b$.

Initially the partial assignment is empty. When at a node labelled $z_{i,j}$, there are two options. If the value of $z_{i,j}$ is already known, then we follow the edge labelled with its value. Otherwise, we set $z_{i,j}$ to the value which corresponds to the smaller subtree, query x_i, and set the remaining $z_{i,k}$'s so that $x_i = g(z_{i,1}, \ldots, z_{i,m})$ (this is possible since g is resistant).

Each time we make a query to x, the size of the current tree is at least halved (the size of the smaller child-subtree of T' is at most half the size of T'), and hence the process terminates after at most $\log S$ queries to x. $\square$

Proof using random projections. Let T be a decision tree for $f \diamond g$ of size S. We apply the following random projection. For each $i \in [n]$, choose an index $j \in [m]$ at random, and set it to a random value $b \in \{0,1\}$. Set the remaining variables to $x_i, \bar{x}_i, 0, 1$ in such a way that $g(z_i) = x_i$. This is possible since g is resistant, as we show below. The result is a decision tree T' for f.

To see that we can always construct such a projection, let h be a non-constant function, and choose two inputs w^0, w^1 such that $h(w^b) = b$. We define the projection ρ as follows: if $w^0_i = w^1_i = a$ then set $y_i = a$; otherwise, if $w^0_i = 0$ then set $y_i = x$, and if $w^1_i = 0$ then set $y_i = \bar{x}$. By construction, $\rho|_{x=b} = w^b$ and so $h|_\rho(b) = h(w^b) = b$.

For example, suppose that g is the majority function on three inputs, and that we set the first bit to 0. Then $g(0,1,x_i) = x_i$ and $g(0,x_i,x_i) = x_i$ are two possible settings, corresponding to the input pairs $010, 011$ and $000, 011$, respectively.

We would like to say that there is a choice of random projection for which T' is shallow. To do this, let us calculate the probability that a vertex at depth D in T survives in T'. The path leading to the vertex involves variables from at least D/m different blocks (each z_i is a block). For each block appearing in the path, choose an arbitrary variable appearing in the path. If we set this variable to the opposite value then the vertex disappears. This happens with probability at least $1/(2m)$ for each variable, and so the vertex survives with probability $p \leq (1 - 1/(2m))^{D/m} = 1/2^{O(D)}$.

If $pS < 1$ then the expected number of vertices at depth D surviving in T' is less than 1, and so there is a random projection for which T' has depth less than D. This happens for some $D = O(\log S)$. □

3.2 Weakly resistant gadgets

There are gadgets, such as $x \vee (y \wedge z)$, which are clearly not resistant, but for which the lifting still holds as we will show below. To capture these cases, we define a more general class of gadgets called *weakly resistant.*

Definition 3.3. A gadget $g\colon \{0,1\}^m \to \{0,1\}$ is *weakly resistant* if for every certificate α (a partial assignment which sets the value of g) there is a partial assignment $y_j = b$ which (1) conflicts with α and (2) does not make g constant.

Every resistant gadget is trivially weakly resistant: we can take any variable mentioned in α and substitute the opposite value. The aforementioned gadget $h = x \vee (y \wedge z)$ is weakly resistant (as we show below) but not resistant, since $h|_{x=1} = 1$. The gadget $x \vee y$ is not even weakly resistant due to the certificate $x = y = 0$: if we substitute $x = 1$ or $y = 1$ then the gadget becomes constant.

Let us verify that h is weakly resistant, by considering all possible certificates:

- $x = 1$: take $x = 0$.
- $y = z = 1$: take $y = 0$ or $z = 0$.
- $x = y = 0$: take $y = 1$.
- $x = z = 0$: take $z = 1$.

We now prove the following lifting theorem:

Theorem 3.4. *For any $f\colon \{0,1\}^n \to \{0,1\}$ and a weakly resistant gadget $g\colon \{0,1\}^m \to \{0,1\}$ the following holds:*
$$\log \mathrm{DTSize}(f \diamond g) = \Omega(\mathrm{C}(f)).$$

Proof. Given a decision tree T for $f \diamond g$ of size S, we construct a blockwise random projection ρ in the following way. For each $i \in [n]$, we choose a random certificate C of g. Let $y_j = b$ be the substitution promised by weak resistance. We substitute $z_{ij} = b$. Since this does not make $g(z_i)$ constant, we can complete $\rho|_{z_i}$ to a projection to x_i such that $g(\rho|_{z_i}) = x_i$ (as in the case of resistant predicates).

Let ℓ be a leaf of T. Suppose that at least D blocks in ℓ contain certificates. If there are M certificates of g, then ℓ is consistent with ρ with probability is at most $(1 - 1/M)^D$. Choose $D = \Theta(\log S)$ so that $(1 - 1/M)^D S < 1$, which implies that under some choice of ρ, all leaves of $T|_\rho$ contain less than D blocks with certificates. Since $(f \diamond g)|_\rho = f$, this shows that $\mathrm{C}(f) < D$.

Indeed, given an input x to f, consider the corresponding input $\rho(x)$ to $f \diamond g$. This input reaches some leaf ℓ, in which less than D blocks contain certificates, say the blocks in $I \subseteq [n]$. For each input x' consistent with gadget values for blocks in I, we can find an input z reaching the same leaf such that $g^n(z) = x'$. Hence the block certificates in ℓ correspond to a certificate of f. □

Corollary 3.5. *For any function $f\colon \{0,1\}^n \to \{0,1\}$ and a weakly resistant gadget $g\colon \{0,1\}^m \to \{0,1\}$ the following holds:*

$$\log \mathrm{DTSize}(f \diamond g) = \Omega\big(\sqrt{\mathrm{DT}(f)}\big).$$

Proof. [BBC+01] shows that for any function f

$$\mathrm{DT}(f) \le (\mathrm{C}(f))^2.$$

Together with Theorem 3.4, that gives us the desired bound. □

Note 3.6. This proof can also be phrased in the language of simulations. This was worked out jointly with Amit Chakrabarty, Mika Göös, Johan Håstad, Susanna de Rezende, Robert Robere, and Avishay Tal in a Dagstuhl breakout session.

Question 3.7. We do not know whether the lower bound is tight, even for $h = x \vee (y \wedge z)$. At the moment we can't even rule out $\log \mathrm{DTSize}(f \diamond h) = \Omega(\mathrm{DT}(f))$.

Sherstov [She10, Theorem 6.4] proved that

$$\max\big(\log \operatorname{rank}(f \diamond \mathrm{AND}), \log \operatorname{rank}(f \diamond \mathrm{OR})\big) \ge \deg(f).$$

Since rank lower bounds decision tree size, this implies that

$$\log \mathrm{DTSize}(f \diamond g) \ge \deg(f).$$

Note that we are not aware of any separation between $\mathrm{DT}(f)$ and $\max(\mathrm{C}(f), \deg(f))$.

3.3 Gadget classification

Now we can prove a dichotomy result. First of all, we need the following classification of gadgets.

Lemma 3.8. *For every $g\colon \{0,1\}^m \to \{0,1\}$, one of the following cases holds:*

1. *Function g is a (possibly empty) conjunction or a disjunction of literals.*
2. *Function g is weakly resistant.*

Proof. If g is not weakly resistant then there is a certificate α such that for every substitution $y_j = b$ conflicting with α, the function $g|_{y_j=b}$ is constant. Moreover we can assume that α is an inclusion-minimal certificate.

Suppose that

$$\alpha = \{y_{i_1} = b_1, \ldots, y_{i_t} = b_t\}$$

and $g|_\alpha = b$. By assumption, the function $g|_{y_{i_s}=\bar{b}_s}$ is constant. Since α is inclusion-minimal, necessarily $g|_{y_{i_s}=\bar{b}_s} = \bar{b}$. Thus if $b = 1$ then

$$g = [y_{i_1} = b_1] \wedge \cdots \wedge [y_{i_t} = b_t],$$

otherwise, if $b = 0$ then

$$g = [y_{i_1} \ne b_1] \vee \cdots \vee [y_{i_t} \ne b_t].$$ □

We can now state the dichotomy theorem.

Theorem 3.9. *For every gadget $g\colon \{0,1\}^m \to \{0,1\}$, one of the following cases holds:*

1. *There is an infinite family of functions f_n with $\mathrm{DT}(f_n) \to \infty$ and $\mathrm{DTSize}(f \diamond g) = O(\mathrm{DT}(f))$.*

2. *For every function f, we have $\log \mathrm{DTSize}(f \diamond g) = \Omega\left(\mathrm{DT}(f)^{\Omega(1)}\right)$.*

Proof. The two cases correspond to the two cases of Lemma 3.8. If g is a disjunction then we take $f_n := \mathrm{OR}_n$. Clearly $\mathrm{DT}(f_n) = n$ and $\mathrm{DTSize}(f \diamond g) = O(n)$ (if g is constant, then $\mathrm{DTSize}(f_n) = 1$). The case of a conjunction is similar.

Otherwise, g is weakly resistant, thus by Corollary 3.5 we have $\log \mathrm{DTSize}(f \diamond g) = \Omega\big(\sqrt{\mathrm{DT}(f)}\big)$. □

3.4 Generalization to search problems

A *relation* f is a subset of $\{0,1\}^n \times V$. For a gadget $g\colon \{0,1\}^m \to \{0,1\}$, we define a composition $f \diamond g \subseteq (\{0,1\}^m)^n \times V$ as a relation, such that

$$(z_1, z_2, \dots, z_n, r) \in (f \diamond g) \iff (g(z_1), g(z_2), \dots, g(z_n), r) \in f.$$

One can observe that the proofs of Theorems 3.2 and 3.4 work as well when we let f to be a relation. However, the Corollary 3.5 does not hold for the relations since $\mathrm{DT}(f)$ can be exponentially greater than $\mathrm{C}(f)$. As an example of this, one can take f to be a *falsified clause problem* corresponding to the Pigeonhole Principle Formula over an expander graph. [BSW01] showed that resolution width of any refutation for this formula at least $\Omega(n)$ (which is greater or equal than $\mathrm{DT}(f)$), but the certificate complexity is constant (since each of the clauses of the formula is constant-sized).

However, we conjecture that lifting from decision tree depth to decision tree size can also be proved for weakly resistant gadgets. Equivalently, this will mean that such lifting holds for the gadget $g(x, y, z) := x \vee (y \wedge z)$.

4 Conjunction DAG width to size

4.1 Conjunction DAGs

Conjunction DAGs were first defined formally in [GGKS20], though they appear implicitly in previous work. A *conjunction DAG* over a set of variables is a single-rooted DAG with the following additional information:

- Each internal vertex is annotated with a variable, and it has two outgoing edges, one labeled 0 and the other one labeled 1.

- Each vertex v is annotated with a partial assignment $\rho(v)$ with the following constraint. Suppose that v queries x_i, and the answer b leads to the vertex v_b. Then the partial assignment $\rho(v_b)$ is a subset of the partial assignment $\rho(v) \cup \{x_i \leftarrow b\}$. (This is not identical to the definition in [GGKS20], but morally the same.)

- The partial assignment at the root is the empty assignment.

- Each leaf is annotated with some value.

A decision DAG *computes* f if for every leaf ℓ annotated with y_ℓ, $\rho(\ell)$ is a y_ℓ-certificate of f.

Every decision tree is a decision DAG. There are three parameters of interest: the (total) size (number of vertices), the leaf size (number of leaves), and the width (maximum number of variables in any $\rho(v)$).

4.2 Size vs width

A decision tree of depth d contains at most 2^d leaves, and this is tight for the parity function, in the sense that the bound $\mathrm{DTSize}(f) \leq 2^{\mathrm{DT}(f)}$ cannot be improved when f is the parity function.

Similarly, a conjunction DAG of width d contains at most $\binom{n}{\leq d} = O(n^d)$ vertices, and this is tight for the following function (also known as the *Tribes* function)

$$f(x) = \bigvee_{i=1}^{\sqrt{n}} \bigwedge_{j=1}^{\sqrt{n}} x_{ij}.$$

We can construct a conjunction DAG of width $O(\sqrt{n})$ for f as follows. We think of the input as a matrix, where i is the row number and j is the column number. We scan each row sequentially. If the current row consists only of 1s, we stop. Otherwise, we add the first 0 to ρ, and forget all the remaining entries of the row.

Conversely, consider the set of $\sqrt{n}^{\sqrt{n}}$ inputs having a single 0 per row. No two inputs share the same certificate, and so every conjunction DAG for f must contain at least $n^{\sqrt{n}/2}$ leaves.

4.3 Separation between decision tree size and conjunction DAG size

Let $\mathrm{d}(f)$ denotes the conjunction DAG width of a function f, which is the smallest width of a conjunction DAG for f. Since $\mathrm{DT}(f) \geq \mathrm{d}(f) \geq \mathrm{C}(f) = \Omega\big(\sqrt{\mathrm{DT}(f)}\big)$, in case of functions conjunction DAG width and decision tree depth are polynomially related. In this section, we show that in case of relations decision tree size and conjunction DAG size can be far apart.

Alekhnovich et al. [AJPU07] constructed a family of CNF contradictions ϕ_n with $\mathrm{poly}(n)$ many variables and clauses which have a refutation in Resolution of size $\mathrm{poly}(n)$, but such that any refutation in tree-like Resolution (even in regular Resolution) is of size $2^{\Omega(n)}$.

We can think of a Resolution proof as a conjunction DAG which solves the *falsified clause problem*: given a truth assignment, find a falsified clause. This is a relation rather then function. Similarly, a tree-like Resolution proof is a decision tree solving the same problem.

Now we are going to show that there is a function $f\colon \{0,1\}^n \to \{0,1\}$ with a conjunction DAG of size $\mathrm{poly}(n)$ such that $\mathrm{DTSize}(f) = 2^{\Omega(n/\log n)}$. Consider the contradictions ϕ_n mentioned above. Index the clauses of ϕ_n using bitstrings of length $\ell = O(\log n)$ in some arbitrary way. Now consider a polynomial size Resolution proof of ϕ_n, and let f_i be the function mapping a truth assignment to the i-th bit of the bitstring indexing the falsified clause found by the proof. By construction, each f_i has a conjunction DAG of size $\mathrm{poly}(n)$. Given decision trees for $f_1, \ldots, f_\ell$, we can construct a decision tree solving the falsified clause problem of size $\prod_i \mathrm{DTSize}(f_i)$. Since this product must be at least $2^{\Omega(n)}$, we conclude that $\max_i \mathrm{DTSize}(f_i) = 2^{\Omega(n/\log n)}$.

4.4 Gadget classification

For the case of conjunction DAGs, we can generalize Theorem 3.2:

Theorem 4.1. *Let $f \subseteq \{0,1\} \times V$ be a relation and $g\colon \{0,1\}^m \to \{0,1\}$ be a resistant gadget. If there is a conjunction DAG of size S computing $f \diamond g$, then there is a conjunction DAG of width $O(\log S)$ computing f.*

Proof. We argue that the random projection proof of Theorem 3.2 can be adapted for this theorem as well. The difference is that instead of killing terms of high depth, we now focus on the nodes with high width and show that with high probability, after random projection there will be no such nodes in the DAG. All the calculations remain unchanged. □

Moreover, for conjunction DAGs we can prove an analogue of Theorem 3.4:

Theorem 4.2. *Let $f \subseteq \{0,1\} \times V$ be a relation and $g\colon \{0,1\}^m \to \{0,1\}$ be a weakly resistant gadget. If there is a conjunction DAG for $f \diamond g$ with S leaves, then the certificate complexity of f is at most $O(\log S)$.*

Proof. The proof of Theorem 3.4 can be applied in this setting as is. □

Note that Theorem 4.2 gives us a lifting from certificate complexity to leaf size. Unfortunately, in the case of the falsified clause problem for CNF, this theorem cannot be effectively used to prove lower bounds since leaf size is usually small as well as certificate complexity. However, this theorem still implies a dichotomy result similar to one in Theorem 3.9.

Theorem 4.3. *For every gadget $g\colon \{0,1\}^m \to \{0,1\}$, one of the following cases holds:*

1. *There exists an infinite family of functions f_n such that $f_n \diamond g$ can be computed with a conjunction DAG having $O(n)$ leaves, but $\mathrm{C}(f_n) = \Omega(n)$.*
2. *For every relation f, if we have a conjunction DAG for $f \diamond g$ with S leaves, then the certificate complexity of f is at most $O(\log S)$.*

Proof. The two cases of the theorem correspond to the two cases in Lemma 3.8. If g is a disjunction then we take $f_n := \mathrm{OR}_n$. Clearly $\mathrm{DT}(f_n) = n$ and $S = O(n)$ (if g is constant, then $\mathrm{DTSize}(f_n) = 1$). The case of a conjunction is similar.

Otherwise, gadget g is weakly resistant and by Theorem 4.2 we have $\log S = \Omega\big(\mathrm{C}(f)\big)$. □

5 Decision tree depth to parity decision tree depth and size

In this section, we are going to classify gadgets in the context of decision tree depth to parity decision tree depth and size lifting. We start by restating recent result of Chattopadhyay et al. [CMSS22]. After that we are going to state the "minimum weight lemma" (Lemma 5.3) that is a technical tool we will use multiple times throughout this section and the following one. We use this lemma to prove the lifting from certificate complexity to parity certificate complexity with OR gadget, which is the most challenging case in the classification. Finally, we will show an alternative and much simpler proof for the lifting from decision tree depth to parity decision tree depth using degree and sparsity. In some sense, this proof should give us a better exponent for the decision tree lifting. The main point of considering certificate complexity lifting is that there is a non-trivial upper bound for OR gadget showing that it is impossible to prove a linear lifting in this setting (see Theorem 5.4). As a byproduct, we get a proof of the log-rank conjecture for the class of functions that can be represented as $f \diamond \mathrm{OR} \diamond \mathrm{XOR}$.

5.1 Stifling gadgets

Chattopadhyay et al. [CMSS22] defined the following notion.

Definition 5.1. A function $g\colon \{0,1\}^m \to \{0,1\}$ is *k-stifling* if for every set of k coordinates and $b \in \{0,1\}$ there is a way to set the remaining $m-k$ coordinates so that the output is b (regardless of the value of the chosen k coordinates). A function is *stifling* if it is 1-stifling.

Chattopadhyay et al. [CMSS22] showed that if g is stifling then $\log \mathrm{DTSize}_{\oplus}(f \diamond g) = \Theta(\mathrm{DT}(f))$, where the hidden constant can depend on g. See Appendix A for an exposition of their proof.

5.2 Minimum weight lemma

We can identify a parity certificate with the set of affine equations satisfying it, which constitutes a linear subspace.

Definition 5.2. A *minimum weight* of a parity certificate C is the minimum Hamming weight of a non-zero vector in the linear part of the equation satisfying this certificate.

For example, one can consider a certificate $\{x+y=1, x+y+z=0\}$. The minimum weight of this certificate is equal to 1 and this corresponds to the equation $z=1$. On the other hand, the minimum weight of a certificate $\{x+y=1, x+z=0\}$ is equal to 2.

The following lemma is a key tool that we will use both in Section 5 and in Section 6. This lemma shows that if the certificate complexity of some function f is large enough in comparison to the parity certificate complexity of the lifted function $f \diamond g$ at some input, then there is a substitution such that the minimal parity certificate of $f \diamond g$ at this input has large Hamming weight. Informally, this means that we can make a small enough substitution, such that the preimage of the parity certificate after the substitution will contain the points that we are interested in.

Lemma 5.3 (minimal weight lemma)**.** *For functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$, suppose that $f \diamond g$ has a parity certificate complexity at most k at some point x. Let K be a parameter. If certificate complexity of f at $g^n(x)$ is greater than $k \cdot K$ then we can find a blockwise partial assignment σ to the inputs of $f \diamond g$ consistent with x, such that for some $k' \le k$:*

- *$(f \diamond g)|_\sigma$ has a parity certificate of size k' at $x|_\sigma$ whose minimum weight is at least K,*
- *$\mathrm{C}\big(f|_{g^n(\sigma)}, g^n(x)|_{g^n(\sigma)}\big) > k'K$.*

Proof. The proof is by induction on k. If $k=0$ then the premises contradict each other, so we can assume that $k>0$.

Let C be the parity certificate for $f \diamond g$ at input x. By the assumption, the size of C is at most k. If its minimum weight is at least K then we are done. Otherwise, without loss of generality C contains a nontrivial constraint $\ell = \alpha$ whose support is less than K. Choose a partial assignment that agrees with x on the support of this constraint, and complete it to a blockwise partial assignment σ consistent with x (complete only the blocks with at least one variable assigned). And let $\tau := g^n(\sigma)$.

By construction, $(f \diamond g)|_\sigma$ has a parity certificate of size at most $k-1$ at $x|_\sigma$, namely $C \setminus \{\ell = \alpha\}$. On the other hand, since we only set the values of at most K blocks, the certificate complexity of f reduces by at most K after being restricted to τ, and hence $\mathrm{C}(f|_\tau, g^n(x)|_\tau) > (k-1)K$. Now we can apply the induction hypothesis for $k-1$. □

5.3 OR gadget

5.3.1 Separation

Chattopadhyay et al. [CMSS22] showed that $C_\oplus(f \diamond g) = \Theta(C(f))$ whenever g is stifling. This no longer holds when g is binary OR. The following theorem shows that there is at least a quadratic gap for OR gadget in the certificate to parity certificate complexity lifting.

Theorem 5.4. *Let $n = m^2$, and let $f\colon \{0,1\}^n \to \{0,1\}$ be the function that accepts an $m \times m$ Boolean matrix iff it has no rows of Hamming weight 1.*

1. $C(f) = n$,
2. $C_\oplus(f \diamond \mathrm{OR}) \le 2m$.

Proof. For the first part, we will show that $C(f, 0) = n$. Indeed, $f(0) = 1$, but if we flip any bit we get a 0-input of f.

For the second part, observe first that if $(f \diamond \mathrm{OR})(x) = 0$ then one of the rows has Hamming weight 1, which can be certified using m constraints. We complete the proof by showing that given inputs for one row of the matrix $y, z \in \{0,1\}^m$, we can certify that $y \vee z$ does not have Hamming weight 1 using two constraints.

If $y \vee z$ has at least two 1s, say $(y \vee z)_i = (y \vee z)_j = 1$, then this can be certified using one of y_i, z_i and one of y_j, z_j. If $y = z = \vec{0}$ then we can certify that $y \vee z$ does not have Hamming weight 1 by two constraints $\mathrm{XOR}_n(y) = \mathrm{XOR}_n(z) = 0$. Indeed, if $y \vee z$ had Hamming weight 1, say $y_i = 1$, then $\mathrm{XOR}_n(y) = 1$. □

Note that if we assume that the lifting result of [CMSS22] holds for the OR gadget as well then the quadratic separation is tight.

Proposition 5.5. *Suppose that $\mathrm{DT}_\oplus(f \diamond \mathrm{OR}) = \Theta(\mathrm{DT}(f))$ for all f. Then every function f satisfies $C(f) = O(C_\oplus(f \diamond \mathrm{OR})^2)$.*

Proof. The assumption implies that

$$C(f) \le \mathrm{DT}(f) = \Theta(\mathrm{DT}_\oplus(f \diamond \mathrm{OR})) \overset{[ZS10]}{=} O(C_\oplus(f \diamond \mathrm{OR})^2).$$ □

5.3.2 Lifting

Given Lemma 5.3 we are going to prove the following lifting for the certificate complexity.

Theorem 5.6. *For every function $f\colon \{0,1\}^n \to \{0,1\}$, we have $C(f) = O(C_\oplus(f \diamond \mathrm{OR})^3)$.*

Throughout the proof we will think of f as having inputs $x_1, x_2 \ldots, x_n$, and of $f \diamond \mathrm{OR}$ as having inputs $z_{1,1}, z_{1,2}, z_{2,1}, z_{2,2} \ldots, z_{n,1}, z_{n,2}$, such that x_i corresponds to $z_{i,1} \vee z_{i,2}$. The *twin* of $z_{i,1}$ is $z_{i,2}$, and vice versa. We start the proof with the following lemma, which would be another technical tool in this section. Together with Lemma 5.3, the proof of the lifting theorem is essentially a combination of these two tricks.

Lemma 5.7. *Suppose that C is a parity certificate of size k whose minimum weight is at least k. Then there are k indices $i_1, \ldots, i_k$ such that for every input x with $x_{i_1} = \cdots = x_{i_k} = 1$, we can find an input z satisfying C such that $\mathrm{OR}^n(z) = x$. In particular, there is an input z satisfying C such that $\mathrm{OR}^n(z) = (1, \ldots, 1)$.*

Proof. Let $V + b$ be the affine subspace corresponding to C, where V is a linear subspace. We pick a basis $\ell_1, \ldots, \ell_k$ for V and indices $(i_1, j_1), \ldots, (i_k, j_k) \in [n] \times [2]$ such that:

- z_{i_t,j_t} is in the support of ℓ_t.
- $z_{i_s,1}, z_{i_s,2}$ are not in the support of ℓ_t for $s < t$.

We do this inductively. At step t, we can identify the complement of the span of $\ell_1, \ldots, \ell_k$ in V with all vectors in V whose support does not contain any of the variables $z_{i_1,j_1}, \ldots, z_{i_{t-1},j_{t-1}}$. Since V has minimum distance $k > t - 1$, we can find a vector ℓ_t in the complement whose support contains a variable z_{i_t,j_t} with $i_t \notin \{i_1, \ldots, i_{t-1}\}$.

Now let x be an input satisfying $x_{i_1} = \cdots = x_{i_k} = 1$. We set the twins of $z_{i_1,j_1}, \ldots, z_{i_t,j_t}$ to 1, and set $z_{i,1}, z_{i,2}$ for $i \neq i_1, \ldots, i_t$ in an arbitrary way subject to $z_{i,1} \vee z_{i,2} = x_i$. We then set $z_{i_t,j_t}, \ldots, z_{i_1,j_1}$ (in this order) in such a way that the resulting input satisfies C. □

The next lemma that we need for the proof of the lifting result gives an upper bound on the minimum weight of a parity certificate in terms of the sizes of certificates for 0 and 1.

Lemma 5.8. *Suppose that $f' \diamond \mathrm{OR}$ has a parity certificate C of size k' which certifies b, and a parity certificate C' of size k'' which certifies $\bar{b}$. Then the minimum weight of C is at most $k' + 2(k'')^2$.*

Proof. The proof is by induction on k''. If $k'' = 0$ then $f' \diamond \mathrm{OR}$ should be constant, which contradicts the existence of C, so $k'' > 0$.

Suppose for the sake of contradiction that the minimum weight of the parity certificate C is at least $k' + 2(k'')^2 + 1 \geq k'$. According to Lemma 5.7, some input z satisfying C satisfies $\mathrm{OR}^n(z) = (1, \ldots, 1)$, and so $f(1, \ldots, 1) = b$. If the minimum weight of C' was also at least k'' then the same argument would imply that $f(1, \ldots, 1) = \bar{b}$, and so the minimum weight of C' is less than k''. Consider a corresponding constraint $\ell = \alpha$, choose a partial assignment satisfying it, and complete it to a blockwise partial assignment τ in an arbitrary way. Thus τ assigns fewer than $2k''$ variables.

The assignment τ cannot contradict C since the minimum weight of C is more than $2k''$. Therefore $(f' \diamond \mathrm{OR})|_\tau$ still has a b-certificate $C|_\tau$ of size k' and a $\bar{b}$-certificate $C'|_\tau$ of size $k'' - 1$. The induction hypothesis shows that the minimum weight of $C|_\tau$ is at most $k' + 2(k'' - 1)^2 + 1$, and so the minimum weight of C is at most $k' + 2(k'' - 1)^2 + 2k'' \leq k' + 2(k'')^2$. □

The following lemma is an application of the "minimal weight lemma" to the current setting.

Lemma 5.9. *Suppose that $f \diamond \mathrm{OR}$ has a parity certificate complexity at some input z of size at most k. Let K be a parameter. Suppose that certificate complexity of f at $\mathrm{OR}^n(z)$ is greater than $k \cdot K$. Then we can find a blockwise partial assignment σ to the inputs of $f \diamond \mathrm{OR}$, consistent with z, such that for some $k' \leq k$:*

- *$(f \diamond \mathrm{OR})|_\sigma$ has a parity certificate of size k' at $z|_\sigma$ whose minimum weight is at least K.*
- *$\mathrm{C}(f|_{\mathrm{OR}^n(\sigma)}, \mathrm{OR}^n(z)|_{\mathrm{OR}^n(\sigma)}) > k'K$.*

Proof. Apply Lemma 5.3 for a function f and a gadget $g = \mathrm{OR}$. □

Now we have all the ingredients for proving Theorem 5.6.

Proof of Theorem 5.6. Let $\mathrm{C}_\oplus(f \diamond \mathrm{OR}) = k$, and let K be a parameter to be determined later. Suppose that $\mathrm{C}(f) > kK$. Apply Lemma 5.9 to get a non-constant function f', obtained by restricting f, such that $f' \diamond \mathrm{OR}$ has a parity certificate of size k' whose minimum weight is at least K (the function f' is non-constant since its certificate complexity is positive). Since $\mathrm{C}_\oplus(f' \diamond \mathrm{OR}) \leq \mathrm{C}_\oplus(f \diamond \mathrm{OR})$ and f' is non-constant, Lemma 5.8 (with $k'' \leq k$) shows that $K \leq k' + 2(k'')^2 \leq k + 2k^2$. By choosing $K = k + 2k^2 + 1$ we reach a contradiction. Therefore $\mathrm{C}(f) \leq kK = O(k^3)$. $\square$

Question 5.10. Can we improve this bound from cubic to quadratic, so this bound matches the separation provided by Proposition 5.5?

5.4 AND/OR gadgets

A function $g \colon \{0,1\}^m \to \{0,1\}$ *affine projects* to a function $h \colon \{0,1\}^p \to \{0,1\}$ if there are affine functions $\ell_1, \ldots, \ell_m \colon \mathbb{Z}_2^p \to \mathbb{Z}_2$ such that

$$g(\ell_1(z), \ldots, \ell_m(z)) = h(z).$$

A gadget $g \colon \{0,1\}^m \to \{0,1\}$ is AND/OR if it affine projects to both binary AND and binary OR. Here are some examples of AND/OR gadgets:

- $g(x,y,z) := x \vee (y \wedge z)$. The projections: $g(0,a,b) = a \wedge b$ and $g(a,b,1) = a \vee b$.
- $g(x,y,z) := [x+y+z=1]$. The projections: $g(1,\bar{a},\bar{b}) = a \wedge b$ and $g(\bar{a},\bar{b},\overline{a \oplus b}) = a \vee b$.

Theorem 5.11. *If g is an* AND/OR *gadget then for all functions $f \colon \{0,1\}^n \to \{0,1\}$*

$$\log \mathrm{DTSize}_\oplus(f \diamond g) \geq \mathrm{C}(f).$$

Proof. Let T_0 be a parity decision tree for $f \diamond g$. Let z be a point such that $\mathrm{C}(f,z) = \mathrm{C}(f)$. Since g is AND/OR, we can apply affine substitutions to T_0 and get a parity decision tree T computing function $f \diamond (h_{z_1}, \ldots, h_{z_n})$, where $h_0(a,b) := a \wedge b$ and $h_1(a,b) := a \vee b$.

Now we follow the simulation proof of Chattopadhyay et al. [CMSS22]. We start at the root. When at a node v, if the query is constant then we follow the corresponding edge, otherwise we pick some variable, without loss of generality a_i, which appears in the query. So the query is of the form $a_i \oplus \ell$. We set $b_i := 0$ if $z_i = 0$, and we set $b_i := 1$ if $z_i = 1$. Then we choose c, the answer to the query, such that it leads to the smaller subtree, and set $a_i := \ell \oplus c$. From now on, we do not see neither a_i nor b_i.

Eventually we reach a leaf. Suppose that this happens after eliminating coordinates $i_1, \ldots, i_K$, which means that T contains at least 2^K leaves. For every input z' to f which agrees with z on $i_1, \ldots, i_K$ we can find an input w' to T which satisfies $g(w') = z'$ and leads to the same leaf, showing that $z'_{i_1} = z_{i_1}, \ldots, z'_{i_K} = z_{i_K}$ is a certificate for f at z. We conclude that $K \geq \mathrm{C}(f,z) = \mathrm{C}(f)$, and so $|T_0| \geq |T| \geq 2^K$. $\square$

5.5 Gadget classification

For the proof we will need the following classical fact.

Proposition 5.12 ([Che09, Theorem 4.3])**.** *Suppose that $f \colon \{0,1\}^n \to \{0,1\}$ satisfies the following condition: whenever $f(x) = f(y) = f(z) = 1$ then $f(x \oplus y \oplus z) = 1$. Then either $f \equiv 0$, or f is an indicator of an affine subspace.*

We use this statement to classify gadgets which are not AND/OR.

Lemma 5.13. *If g is not an* AND/OR *then g is a disjunction or a conjunction of affine forms in the inputs.*

Proof. Suppose that g does not affine project to OR. We claim that for any inputs x, y, z, if $g(x) = g(y) = g(z) = 1$ then $g(x \oplus y \oplus z) = 1$. Indeed, suppose that $g(x) = g(y) = g(z) = 1$ but $g(x \oplus y \oplus z) = 0$, and consider the projection

$$h(a, b) := g(x \oplus y \oplus z \oplus a(x \oplus y) \oplus b(x \oplus z)).$$

Then $h(0, 0) = g(x \oplus y \oplus z) = 0$, $h(1, 0) = g(z) = 1$, $h(0, 1) = g(y) = 1$, and $h(1, 1) = g(x) = 1$. Thus, we can conclude that g affine projects to OR.

Applying Proposition 5.12, we see that if g does not affine project to OR then it is either the empty disjunction or a conjunction of affine forms. Similarly, if g does not affine project to AND then $\bar{g}$ does not affine project to OR, and so g is either the empty conjunction or a disjunction of affine forms. $\square$

Now we can prove the classification of gadgets.

Theorem 5.14. *For every gadget g, one of the following cases holds:*

1. *There is an infinite family of functions f_n with $\mathrm{DT}(f_n) \to \infty$ and $\mathrm{DTSize}_\oplus(f_n \diamond g) = O(1)$.*
2. *For every function f, $\mathrm{DT}_\oplus(f \diamond g) = \Omega\left(\mathrm{DT}(f)^{\Omega(1)}\right)$. There in an infinite family of functions f_n with $\mathrm{DT}(f_n) \to \infty$ and $\mathrm{DTSize}(f_n \diamond g) = O(\mathrm{DT}(f))$.*
3. *For every function f, $\log \mathrm{DTSize}_\oplus(f \diamond g) = \Omega\left(\mathrm{DT}(f)^{\Omega(1)}\right)$ and also $\mathrm{C}_\oplus(f \diamond g) = \Omega\left(\mathrm{C}(f)^{\Omega(1)}\right)$.*

Proof. If g is AND/OR then by Theorem 5.11 we have $\log \mathrm{DTSize}_\oplus(f \diamond g) \geq \mathrm{C}(f) = \Omega\big(\sqrt{\mathrm{DT}(f)}\big)$. Since $\mathrm{C}_\oplus(f \diamond g) = \Omega\big(\sqrt{\mathrm{DT}_\oplus(f \diamond g)}\big)$ [ZS10] and $\mathrm{DT}_\oplus(f \diamond g) \geq \log \mathrm{DTSize}_\oplus(f \diamond g)$, we get the third case of the theorem.

Otherwise, by Lemma 5.13, gadget g is either a conjunction of affine forms or a disjunction of affine forms. Suppose, without loss of generality, that g is a disjunction of affine forms. If $g \equiv 0$ or g is affine then we can take $f_n = \oplus_n$ to get the first case of the theorem. If g is not affine then it affine projects to OR, and thus we get the second case: we need to apply Theorem 5.6 together with the polynomial relations between (parity) certificate complexity and (parity) decision tree complexity; the infinite family of functions is $f_n = \mathrm{OR}_n$. $\square$

5.6 Lifting for OR gadget and the log-rank conjecture

In this section, we provide a proof of the following inequality:

Theorem 5.15. *For any function $f \colon \{0, 1\}^n \to \{0, 1\}$ and any gadget $g \colon \{0, 1\}^m \to \{0, 1\}$,*

$$\mathrm{DT}(f) \leq O\left(\mathrm{DT}_\oplus(f \diamond \mathrm{OR})^{O(1)}\right).$$

This inequalty is a corollary of Theorem 5.6 using the fact that DT is polynomially related to C, and $\mathrm{DT}_\oplus$ is polynomially related to $\mathrm{C}_\oplus$. We present an alternative proof of this statement that gives as a byproduct a proof of the log-rank conjecture for a subclass of Boolean functions that can be represented as $f \diamond \mathrm{OR} \diamond \mathrm{XOR}$ for arbitrary function f.

In this section, we will need a composition of a function and a gadget (XOR in this case) in the communication complexity context. For a function $f\colon \{0,1\}^n \to \{0,1\}$ we define a function $f_\oplus\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, such that $f_\oplus(x,y) := f(x_1 \oplus y_1, \ldots, x_n \oplus y_n)$, and associate it with the following communication problem: Alice and Bob are given x and y, respectively, and their goal is to compute $f(x,y)$. For a subclass of such XOR functions, we will prove the following version of log-rank conjecture:

Theorem 5.16. *For any function $f\colon \{0,1\}^n \to \{0,1\}$,*

$$\mathrm{D}((f \diamond \mathrm{OR})_\oplus) \le \mathrm{poly}(\log \mathrm{rank}((f \diamond \mathrm{OR})_\oplus).$$

The proof depends on the following lemma.

Lemma 5.17. *For any function $f\colon \{0,1\}^n \to \{0,1\}$,*

$$\begin{aligned}\mathrm{DT}(f) \overset{(1)}{\le} 2\deg(f)^4 \overset{(2)}{\le} O((\log \mathrm{spar}(f \diamond \mathrm{OR}))^4) \overset{(3)}{=} O((\log \mathrm{rank}((f \diamond \mathrm{OR})_\oplus))^4)\\ \overset{(4)}{\le} O(\mathrm{D}((f \diamond \mathrm{OR})_\oplus)^4) \overset{(5)}{\le} O(\mathrm{DT}_\oplus(f \diamond \mathrm{OR})^4),\end{aligned}$$

Proof.

(1) See [Bd02].

(2) We need to prove that
$$\deg(f) \le \log \mathrm{spar}(f \diamond \mathrm{OR}).$$
We can assume that $f\colon \{-1,1\}^n \to \{-1,1\}$ has degree n. If not, we can fix a monomial of the maximum degree and substitute 1 into all other variables. This substitution does not change the degree while the sparsity can only decrease. So, the function f has the following multilinear representation:
$$f(x) = \alpha_{[n]} \prod_{i=1}^{n} x_i + \sum_{S \subsetneq [n]} \alpha_S x_S, \qquad \text{where } \alpha_{[n]} \ne 0.$$
In order to get the multilinear representation of the function $f \diamond \mathrm{OR}$ one can substitute each x_i with $\frac{y_{i1}y_{i2} - y_{i1} - y_{i2} - 1}{2}$. After the substitution, for any combination of indices $j_1, j_2, \ldots, j_n \in \{1,2\}$, the coefficient of the monomial $\prod_{k=1}^{n} y_{kj_k}$ is equal to $\alpha_{[n]} \cdot (-1/2)^n$. Indeed, such monomials can only be produced by the substitution into the maximum degree monomial, and such substitution results in the coefficient $\alpha_{[n]} \cdot (-1/2)^n$.

Note that $\alpha_{[n]} \ne 0$ and hence there is a monomial for every 2^n possible combinations of indexes $j_1, j_2, \ldots, j_n$. That gives us the desired bound:
$$\log \mathrm{spar}(f \diamond \mathrm{OR}) \ge n = \deg(f).$$

(3) See [BC99].

(4) See [LS93].

(5) Each time we query a linear function $\ell(x \oplus y) = \ell(x) \oplus \ell(y)$, Alice and Bob can send to each other $\ell(x)$ and $\ell(y)$ to compute $\ell(x \oplus y)$. □

Theorem 5.15 follows immediately. For Theorem 5.16 we need one more inequality in the chain.

Proof of Theorem 5.16. We need to show that $\mathrm{DT}_{\oplus}(f \diamond \mathrm{OR}) \leq 2\,\mathrm{DT}(f)$. Indeed, the decision tree for f can be transformed into a decision tree for $f \diamond \mathrm{OR}$ by replacing every query with two queries of corresponding variables. Together with Lemma 5.17 that finishes the proof. □

6 Block sensitivity to communication complexity

In this section, we provide a complete classification of gadgets for lifting from block sensitivity to deterministic and randomized communication complexities. Note that this classification will also give us a classification of gadgets for query-to-communication polynomial lifting since block sensitivity and query complexity are polynomially related to each other in the case of total functions. However, since the proof goes through block sensitivity, we will classify block sensitivity to communication complexity lifting.

6.1 Reductions in communication complexity

Let $f_i \colon \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ for $i = 1, 2$ be two-party functions. We say that f_1 reduces to f_2, denoted $f_1 \leq f_2$, if the communication matrix of f_1 is a submatrix of the communication matrix of f_2. Equivalently, $f_1 \leq f_2$ iff there exist one-to-one mappings π_A and π_B such that

$$f_1(x, y) = f_2(\pi_A(x), \pi_B(y)), \quad \forall (x, y) \in \mathcal{X}_1 \times \mathcal{Y}_1.$$

Zhang [Zha09] proved the following theorem:

Theorem 6.1 (Zhang)**.** *If a two-party gadget $g \colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ satisfies $\mathrm{AND}, \mathrm{OR} \leq g$, then for every function $f \colon \{0,1\}^n \to Q$, the function $f \diamond g$ has (constant error) randomized communication complexity $\Omega(\mathrm{bs}(f))$.*

For completeness we present the proof that is a reduction from unique set disjointnes.

Definition 6.2. The *disjointness* function $\mathsf{Disj}_n \colon \{0,1\}^n \times \{0,1\}^n$ is defined as follows:

$$\mathsf{Disj}_n(x, y) := \bigvee_{i=1}^{n} x_i \wedge y_i.$$

Unique set disjointness, $\mathsf{PromiseDisj}_n$, is a promise version of disjointness with the promise that there is always at most one i such that $x_i \wedge y_i = 1$.

Proof of Theorem 6.1. Razborov [Raz92] showed that $\mathrm{R}(\mathsf{PromiseDisj}_m) = \Omega(m)$. We will show a reduction from this problem to computing $f \diamond g$. To illustrate the proof, let us start with the case in which $f(\vec{0}) = 0$ and $f(e_i) = 1$ for all $i \in [n]$, where e_i is the i'th unit vector. Then

$$\mathsf{PromiseDisj}_n(x, y) = f(x_1 \wedge y_1, \dots, x_n \wedge y_n).$$

Since AND $\leq g$, we can solve unique set disjointness using a protocol for $f \diamond g$. Therefore $f \diamond g$ has communication complexity $\Omega(n)$. Note that this case only required AND $\leq g$; we need OR $\leq g$ in order to handle inputs changing from 1 to 0.

To handle the general case, let $X^\wedge(0), X^\wedge(1) \in \mathcal{X}$ and $Y^\wedge(0), Y^\wedge(1) \in \mathcal{Y}$ witness AND $\leq g$, and let $X^\vee(0), X^\vee(1) \in \mathcal{X}$ and $Y^\vee(0), Y^\vee(1) \in \mathcal{Y}$ witness OR $\leq g$.

Let z be a point witnessing $\mathrm{bs}(f)$, and let $B_1, \dots, B_m$ be the corresponding blocks, where $m = \mathrm{bs}(f)$. Given inputs x, y to unique set disjointness, we construct the following inputs X, Y to $f \diamond g$:

- If i does not belong to any of the blocks, then we fix X_i, Y_i so that $g(X_i, Y_i) = z_i$.
- If $i \in B_j$ and $z_j = 0$, put $X_i = X^\wedge(x_j)$, $Y_i = Y^\wedge(y_j)$.
- If $i \in B_j$ and $z_j = 1$, put $X_i = X^\vee(\bar{x}_j)$, $Y_i = Y^\vee(\bar{y}_j)$.

The reader can check that if x, y are disjoint then applying g to X, Y yields z, and if they intersect only at j then it yields $z \oplus B_j$. Hence $f \diamond g$ can be used to compute unique set disjointness of length m, and consequently we obtain a lower bound of $\Omega(m)$. $\square$

For the classification of gadgets, we will need the following lemma that shows that if OR $\not\leq g$ then the communication matrix of g is block diagonal.

Lemma 6.3. *If $g\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ satisfies that OR $\not\leq g$ then there are partitions of $\mathcal{X}$ and $\mathcal{Y}$ into disjoint sets*

$$\mathcal{X} = \mathcal{X}_0 \sqcup \mathcal{X}_1 \sqcup \cdots \sqcup \mathcal{X}_k \quad \text{and} \quad \mathcal{Y} = \mathcal{Y}_0 \sqcup \mathcal{Y}_1 \sqcup \cdots \sqcup \mathcal{Y}_k,$$

such that

- *If $x \in X_i, y \in Y_j$, where $i \neq j$, or $i = 0$, or $j = 0$, then $g(x, y) = 0$.*
- *If $x \in X_i, y \in Y_i$, where $i \neq 0$, then $g(x, y) = 1$.*

Proof. We can construct such a partition step by step. Consider any point (x_0, y_0) such that $f(x_0, y_0) = 1$. If we don't have such a point, then we can take $\mathcal{X}_0 = \mathcal{X}$ and $\mathcal{Y}_0 = \mathcal{Y}$. Otherwise, we consider all points $y \in \mathcal{Y}$ such that $f(x_0, y) = 1$. Denote the set of all such points as $\mathcal{Y}'$. Now, we can consider all $x \in \mathcal{X}$ such that $f(x, y_0) = 1$. Denote the set of all such points as $\mathcal{X}'$. Observe that every point $(x, y) \in \mathcal{X}' \times \mathcal{Y}'$ satisfies $f(x, y) = 1$, otherwise we can project our gadget to points $(x_0, y_0), (x_0, y), (x, y_0), (x, y)$ and get OR.

Now, observe that for any $(x'', y') \in (\mathcal{X} \setminus \mathcal{X}') \times \mathcal{Y}'$ we have $f(x'', y_0) = 0$ by construction, and thus $f(x'', y') = 0$, since otherwise we can project the gadget to points $(x_0, y'), (x_0, y_0), (x'', y'), (x'', y_0)$ and get OR. By analogy, for any $(x', y'') \in \mathcal{X}' \times (\mathcal{Y} \setminus \mathcal{Y}')$ we have $f(x', y'') = 0$.

This means that $\mathcal{X}' \times \mathcal{Y}'$ forms a valid block of our partition, so we can use the same procedure to construct a block of $(\mathcal{X} \setminus \mathcal{X}') \times (\mathcal{Y} \setminus \mathcal{Y}')$. In the end, if we get sets $\mathcal{X}''$ and $\mathcal{Y}''$ such that any $(x, y) \in \mathcal{X}'' \times \mathcal{Y}''$ satisfies $f(x, y) = 0$, we take $\mathcal{X}_0 = \mathcal{X}''$ and $\mathcal{Y}_0 = \mathcal{Y}''$. $\square$

We say that a gadget $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is a *blow-up* of a gadget $h\colon \mathcal{Z} \times \mathcal{W} \to \{0,1\}$ if there are decompositions $\mathcal{X} = \bigsqcup_{z \in \mathcal{Z}} \mathcal{X}_z$ and $\mathcal{Y} = \bigsqcup_{w \in \mathcal{W}} \mathcal{Y}_w$, with $\mathcal{X}_z, \mathcal{Y}_w \neq \emptyset$, such that all $(x_z, y_w) \in \mathcal{X}_z \times \mathcal{Y}_w$ satisfy $g(x_z, y_w) = h(z, w)$. For example, g is a blow-up of XOR if (up to rearrangement) it has communication matrix of the following form

$$\begin{bmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & \cdots & 1 \end{bmatrix}$$

If g is a blow-up of h then $f \diamond g$ and $f \diamond h$ have the same communication complexity in all models. Indeed, on the one hand, h is a restriction of g, and so a protocol for $f \diamond g$ can be used to solve $f \diamond h$; and on the other hand, by replacing $x \in \mathcal{X}_z$ by z and $y \in \mathcal{Y}_w$ by w, we can use a protocol for $f \diamond h$ to solve $f \diamond g$.

Corollary 6.4. *If* $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *satisfies both* OR $\not\leq g$ *and* AND $\not\leq g$ *then either* g *is constant, or it is a blow-up of* XOR.

Proof. We can assume that the $\mathcal{X}_i, \mathcal{Y}_i \neq \emptyset$, except that possibly $\mathcal{X}_0 = \mathcal{Y}_0 = \emptyset$.

If $\mathcal{X}_0 \neq \emptyset$ and $k > 0$ then AND $\leq g$. Indeed, if we restrict the matrix of g so that it contains one row each from $\mathcal{X}_0, \mathcal{X}_1$ and one column each from $\mathcal{Y}_0, \mathcal{Y}_1$ then we get

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

which is AND. Consequently, either $k = 0$ or $\mathcal{X}_0 = \mathcal{Y}_0 = \emptyset$.

If $k = 0$ then $g \equiv 0$. Now suppose that $\mathcal{X}_0 = \mathcal{Y}_0 = \emptyset$. If $m \geq 3$ and we restrict the communication matrix of g so that it contains one row from each $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ and one column from each $\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3$ then we get

$$\begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 \\ 0 & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Again we get a restriction to AND. Hence, $m \leq 2$. If $m = 1$ then $g \equiv 1$, and otherwise g is a blow-up of XOR. □

Corollary 6.5. *If* $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *satisfies both* OR $\not\leq g$ *and* XOR $\not\leq g$ *then either* g *is constant, or it is a blow-up of* AND. *Similarly, if* AND $\not\leq g$ *and* XOR $\not\leq g$ *then either* g *is constant, or it is a blow-up of* OR.

Proof. We can assume that the $\mathcal{X}_i, \mathcal{Y}_i \neq \emptyset$, except that possibly $\mathcal{X}_0 = \mathcal{Y}_0 = \emptyset$. We start by proving the first claim, and so we assume that OR, XOR $\not\leq g$.

If $m \geq 2$ then XOR $\leq g$. Indeed, if we restrict the matrix of g so that it contains one row from each $\mathcal{X}_1, \mathcal{X}_2$ and one column from each $\mathcal{Y}_1, \mathcal{Y}_2$ then we obtain

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

which is XOR. If $m = 0$ then $g = 1$. If $m = 1$ and $\mathcal{X}_0 = \mathcal{Y}_0 = \emptyset$ then $g = 0$. Finally, if $m = 1$ and $\mathcal{X}_0, \mathcal{Y}_0 \neq \emptyset$ then g is a blow-up of AND.

Suppose that AND, XOR $\not\leq g$. Then OR, XOR $\not\leq \bar{g}$, where $\bar{g}$ is the negation of g. Hence either $\bar{g}$ is constant or it is a blow-up of AND. Thus, either g is constant or it is a blow-up of OR. $\square$

6.2 Gadget classification for randomized communication complexity

In this section we prove the following dichotomy result for the randomized case.

Theorem 6.6. *For every gadget g, one of the following cases holds:*

1. *There is an infinite family of functions f_n with $\mathrm{bs}(f_n) = \Omega(n)$ and $\mathrm{R}(f_n \diamond g) = O(\log n)$.*
2. *For every function f, we have $\mathrm{R}(f \diamond g) = \Omega\left(\mathrm{bs}(f)^{\Omega(1)}\right)$.*

First we need to prove the following lemma.

Lemma 6.7. *Let $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a gadget such that OR $\not\leq g$. Then*

$$\mathrm{R}(\mathrm{AND}_n \diamond g) = O(\log n).$$

Proof. Consider the decomposition $\mathcal{X} = \mathcal{X}_0 \sqcup \mathcal{X}_1 \sqcup \cdots \sqcup \mathcal{X}_k$ and $\mathcal{Y} = \mathcal{Y}_0 \sqcup \mathcal{Y}_1 \sqcup \cdots \sqcup \mathcal{Y}_k$ provided by Lemma 6.3. If one of the blocks g_i was given an input from $\mathcal{X}_0$ or $\mathcal{Y}_0$, then one of the players can share this information with the other player and finish the communication. Otherwise, they indicate that they do not have such blocks. After this step, they are restricted to inputs from $\mathcal{X} \setminus \mathcal{X}_0$ and $\mathcal{Y} \setminus \mathcal{Y}_0$. Since any two inputs from the same block $\mathcal{X}_i$ or $\mathcal{Y}_i$ are indistinguishable for our gadget, they need to solve $\mathrm{AND}_n \diamond \mathrm{EQ}_m$, where EQ_m is the *equality* function for m-bit inputs (i.e., $\mathrm{EQ}_m(x,y) = 1 \iff x = y$) for $m = \lceil \log k \rceil$. This is exactly an instance of EQ_{nm}, which can be solved with $O(\log n)$ bits of communication (note that m is a constant) by a randomized communication protocol (see [KN96]). $\square$

Now we are ready to prove the dichotomy result.

Proof of Theorem 6.6. If AND, OR $\leq g$ then Theorem 6.1 shows that $\mathrm{R}(f \diamond g) = \Omega\left(\mathrm{bs}(f)^{\Omega(1)}\right)$, so we get the second case.

Otherwise, OR $\not\leq g$ or AND $\not\leq g$. If OR $\not\leq g$, we can directly apply Lemma 6.7 and get the first case of the theorem by taking $f_n = \mathrm{AND}_n$. Otherwise, if AND $\not\leq g$, we can observe that the same proof works for $f_n = \mathrm{OR}_n$, and again we get the first case of the theorem. $\square$

6.3 Gadget classification for deterministic communication complexity

In this section, we will prove the following lifting theorem that implies the classification theorem.

Theorem 6.8. *If a two-party gadget $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ satisfies* AND, XOR $\leq g$, *then for every function $f\colon \{0,1\}^n \to \{0,1\}$, the function $f \diamond g$ has communication complexity $\Omega(\mathrm{bs}(f)^k)$ for some fixed constant $k > 0$.*

Together with Theorem 6.1 this will give us a complete classification of gadgets.

Theorem 6.9. *For every gadget g, one of the following cases holds:*

1. *There is an infinite family of functions f_n with* $\mathrm{bs}(f_n) = \Omega(n)$ *and* $\mathrm{D}(f_n \diamond g) = O(1)$.

2. *For every function f, we have* $\mathrm{D}(f \diamond g) = \Omega\left(\mathrm{bs}(f)^{\Omega(1)}\right)$.

Proof. If $\mathrm{AND}, \mathrm{OR} \le g$, or $\mathrm{AND}, \mathrm{XOR} \le g$, or $\mathrm{OR}, \mathrm{XOR} \le g$, then Theorem 6.1 and Theorem 6.8 show that $\mathrm{D}(f \diamond g) = \Omega\left(\mathrm{bs}(f)^{\Omega(1)}\right)$, so the second case holds. (While Theorem 6.8 is only stated for $\mathrm{AND}, \mathrm{XOR} \le g$, the same proof works for $\mathrm{OR}, \mathrm{XOR} \le g$.)

If none of these cases holds, then at least two of the functions AND, OR, XOR do not reduce to g. Corollaries 6.4 and 6.5 show that either g is constant (and so the first case trivially holds) or it is a blow-up of one of the functions XOR, OR, AND. By taking $f_n = \mathrm{XOR}_n, \mathrm{OR}_n, \mathrm{AND}_n$ (respectively), we get the first case of the theorem. ☐

In the rest of this section we will cover the proof of Theorem 6.8. We start with the following observation: according to [HHL18], for any $f\colon \{0,1\}^n \to \{0,1\}$ we have $\mathrm{DT}_\oplus(f) \le O(\mathrm{D}(f \diamond \mathrm{XOR})^6)$. Zhang and Shi [ZS10] showed that $\mathrm{DT}_\oplus(f) = O(\mathrm{C}_\oplus(f)^2)$. Combining these results, we can derive the following corollary:

Corollary 6.10. *For any $f\colon \{0,1\}^n \to \{0,1\}$, we have* $\mathrm{C}_\oplus(f) \le O(\mathrm{D}(f \diamond \mathrm{XOR})^{12})$.

In the following lemma, a *minimal sensitive block* of a function $f\colon \{0,1\}^n \to \{0,1\}$ at point x is a subset $I \subseteq [n]$ such that $f(x^I) \ne f(x)$ but $f(x^J) = f(x)$ for all $J \subsetneq I$. This lemma shows that the deterministic communication complexity of $f \diamond \mathrm{XOR}$ is lower bounded by the size of any minimal sensitive block of the function f.

Lemma 6.11. *For any $f\colon \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$ such that the size of one of the minimal sensitive blocks of f at x is equal to S. Then the following holds:*

$$\mathrm{D}(f \diamond \mathrm{XOR}) \ge S.$$

Proof. Let $\alpha \in \{0,1\}^n$ be the point the minimal sensitive block of size S. And let $i_1, i_2, \ldots, i_S$ be the indices of the variables in the minimal sensitive block. Consider a restriction of function f by substituting values $x_j = \alpha_j$ for the all $j \notin \{i_1, i_2, \ldots, i_S\}$. This gives us a function $f'(y_1, y_2, \ldots, y_S)$ such that

- $f'(y_1, \ldots, y_S) = f(x)$, if $y_j \ne \alpha_{i_j}$ for at least one j.
- $f'(y_1, \ldots, y_S) \ne f(x)$, if $y_j = \alpha_{i_j}$ for all j.

Thus, if we substitute XOR inside each y_j this will give us an instance of the function

$$(z_1 \oplus z'_1 \oplus \alpha_{i_1}) \vee (z_2 \oplus z'_2 \oplus \alpha_{i_2}) \vee \cdots \vee (z_S \oplus z'_S \oplus \alpha_{i_S}),$$

where each z_i and z'_i are distinct variables. This function is essentially a negation of the equality function with a shift. So, its deterministic complexity is equal to S (see [KN96] for the lower bound for EQ_n). Thus $\mathrm{D}(f \diamond \mathrm{XOR}) \ge \mathrm{D}(f' \diamond \mathrm{XOR}) \ge S$. ☐

The following lemma shows that the deterministic communication complexity of $f \diamond \mathrm{AND}$ is lower bounded by the block sensitivity of f at the all-0 input.

Lemma 6.12. *For any $f\colon \{0,1\}^n \to \{0,1\}$ such that $\mathrm{bs}(f, \vec{0}) = S$, the following holds:*

$$\mathrm{D}(f \diamond \mathrm{AND}) \ge \Omega(S).$$

Proof. The proof essentially follows the proof from [Zha09]. Let $I_1, \dots, I_S$ be the sensitive blocks at $\vec{0}$. We restrict inputs for $f \diamond \mathrm{AND}$ to get an instance of $\mathsf{PromiseDisj}_S$ in the following way:

1. For $i \notin \bigcup_{j=1}^{S} I_j$,
$$x'_i = y'_i = 0.$$

2. For $i \in I_j$,
$$x'_i = x_j, \quad y'_i = y_j.$$

Indeed, if there is a unique j such that $x_j \wedge y_j = 1$, then

$$f(x'_1 \wedge y'_1, x'_2 \wedge y'_2, \dots, x'_n \wedge y'_n) = f\left(0^{I_j}\right) \neq f(0, \dots, 0).$$

On the other hand, if $x_j \wedge y_j = 0$ for each $j \in [S]$, then

$$f(x'_1 \wedge y'_1, x'_2 \wedge y'_2, \dots, x'_n \wedge y'_n) = f(0, \dots, 0).$$

So, the protocol for $f \diamond \mathrm{AND}$ should also solve $\mathsf{PromiseDisj}_S$, and thus $\mathrm{D}(f \diamond \mathrm{AND}) \geq \Omega(S)$. □

The next lemma is an application of the "minimum weight lemma" for the identity gadget.

Lemma 6.13. *Suppose that f has a parity certificate at some point x of size at most k. Let K be a parameter. Suppose that the certificate complexity of f at x is greater than $k \cdot K$. Then we can find a partial assignment σ to at most $k \cdot K$ variables of f, consistent with x, such that:*

- *$f|_\sigma$ has a parity certificate at $x|_\sigma$ of size at most k.*
- *The minimum weight of this certificate is at least K.*

Proof. Apply Lemma 5.3 for the function f and the identity function as a gadget g. □

And the last ingredient needed for the proof of the classification theorem, the following lemma that shows that parity certificates of high minimum weight always intersect with (regular) certificates of sufficiently small size.

Lemma 6.14. *Suppose that for some function f and some input x the following holds: f has a parity certificate $\mathrm{C}_\oplus(x)$ of size k at x whose minimum weight is at least K. Then, for any point y with certificate complexity at most $K - k$, there is a non-empty intersection of the corresponding certificates $\mathrm{C}(y)$ and $\mathrm{C}_\oplus(x)$.*

Proof. Let $V + b$ be the affine subspace corresponding to $\mathrm{C}_\oplus(x)$, where V is a linear subspace. We pick a basis $\ell_1, \dots, \ell_k$ for V and indices $i_1, \dots, i_k \in [n]$ such that:

- z_{i_t} is in the support of ℓ_t.
- z_{i_s} is not in the support of ℓ_t for $s < t$.
- z_{i_t} does not belong to the support of $\mathrm{C}(y)$.

We do this inductively. At step t, we can identify the complement of the span of $\ell_1, \dots, \ell_k$ in V with all vectors in V whose support does not contain any of the variables $z_{i_1}, \dots, z_{i_{t-1}}$. Since V has minimum distance $K > t-1+K-k$, we can find a vector ℓ_t in the complement whose support contains a variable z_{i_t} with $i_t \notin \{i_1, \dots, i_{t-1}\}$ and $z_{i_t} \notin \mathrm{C}(y)$.

Now we can construct an input α which belongs to both $\mathrm{C}(y)$ and $\mathrm{C}_\oplus(x)$. First, we set the variables from $\mathrm{C}(y)$ such that the resulting input satisfies $\mathrm{C}(y)$. Then we set in an arbitrary way all the variables not occurring in $\mathrm{C}(y)$ or in $\{z_{i_1}, \dots, z_{i_k}\}$. Finally, we set $z_{i_k}, \dots, z_{i_1}$ (in this order) in such a way that the resulting input satisfies $\mathrm{C}_\oplus(x)$. □

Now we have everything we need to prove Theorem 6.8.

Proof of Theorem 6.8. Consider any function f. There is a constant S such that $\mathrm{bs}(f) = 9S^5$. If $\mathrm{C}_\oplus(f) \geq S$, then by Corollary 6.10,

$$\mathrm{D}(f \diamond g) \geq \mathrm{D}(f \diamond \mathrm{XOR}) \geq \mathrm{C}_\oplus(f)^{1/12} \geq S^{1/12} \geq \Omega\big(\mathrm{bs}(f)^{1/(12\cdot 5)}\big),$$

and so the theorem holds for $k = 1/60$. So, we can assume that $\mathrm{C}_\oplus(f) < S$. Also, we can assume that for any input, the size of the minimal sensitive block is at most S, otherwise by Lemma 6.11 we get

$$\mathrm{D}(f \diamond g) \geq \mathrm{D}(f \diamond \mathrm{XOR}) = \Omega(S) = \Omega(\mathrm{bs}(f)^{1/5}).$$

Consider any input x with maximal block sensitivity and the corresponding set of minimal sensitive blocks. The certificate complexity of x is at least $9S^5$, since the certificate complexity is at least the block sensitivity. By Lemma 6.13 (for $k = S$ and $K = 2S^3$), we can find a substitution σ such that $f|_\sigma$ has a parity certificate at $x|_\sigma$ of size at most S with minimal weight least $2S^3$. In addition, the block sensitivity of $f|_\sigma$ at x_σ is at least $9S^5 - 2S^4 \geq 7S^5$ (since we set at most $2S^4$ variables with σ), and the blocks still have size at most S. We fix the substitution σ and consider the restricted function. In what follows, we denote $\hat{f} := f|_\sigma$ and $\hat{x} := x|_\sigma$.

Consider any input y obtained from $\hat{x}$ by swapping one of the blocks. Suppose that the certificate complexity of this input is at most $2S^3 - S$. Then by Lemma 6.14 (for $k = \mathrm{C}_\oplus(\hat{f}, \hat{x}) \leq S$ and $K = 2S^3$), the certificate for y and the parity certificate for $\hat{x}$ intersect. Since $\hat{f}(\hat{x}) \neq \hat{f}(y)$, this leads to a contradiction. So, the certificate complexity of y should be greater than $2S^3 - S$.

Again, by using Lemma 6.13 (for $k = S$ and $K = S^2 - 1$), there is a substitution τ of at most S^3 variables, such that $y|_\tau$ has a parity certificate of size at most S with minimal weight at least $S^2 - 1$. In addition, the certificate complexity of $y|_\tau$ is at least $2S^3 - S^3 = S^3$.

Since the substitution τ consists of at most S^3 variables, the minimal weight of the parity certificate $\mathrm{C}_\oplus(\hat{f}, \hat{x})|_\tau$ (we apply τ to every linear form in the certificate) at least $2S^3 - S^3 \geq S^3$. Note that substitution τ can be inconsistent with $\hat{x}$, but we still can make this substitution inside $\mathrm{C}_\oplus(\hat{f}, \hat{x})$ and get a solvable certificate for $\hat{f}|_\tau$. Thus, if we find any point z, consistent with the substitution τ, such that the certificate complexity of $z|_\tau$ is at most $S^2 - S - 1$, then this certificate $\mathrm{C}(\hat{f}|_\tau, z|_\tau)$ should intersect with both $\mathrm{C}_\oplus(\hat{f}, \hat{x})|_\tau$ and $\mathrm{C}_\oplus(\hat{f}, y)|_\tau$ by Lemma 6.14 (for $k = \mathrm{C}_\oplus(\hat{x}) \leq S$ and $K = S^2 - 1$). Since all the these certificates intersect, we have $\hat{f}(\hat{x}) = \hat{f}(z) = \hat{f}(y)$, but this contradicts $\hat{f}(\hat{x}) \neq \hat{f}(y)$.

Now, consider an input z consistent with τ such that all the $*$-variables of τ are set to zeroes. We have shown above that the certificate complexity of $\hat{f}|_\tau$ at $z|_\tau$ is at least $S^2 - S$. Recall that $z|_\tau$ has a block sensitivity witness with all the blocks of size at most S. Since all the blocks together constitute a certificate, we conclude that the block sensitivity of $\hat{f}|_\tau$ at $z|_\tau$ is at least $S-1$. Since $z|_\tau$ is the all zeroes, by Lemma 6.12 we have

$$\mathrm{D}(f \diamond g) \geq \mathrm{D}(f \diamond \mathrm{AND}) = \Omega(S) = \Omega(\mathrm{bs}(f)^{1/5}). \qquad \square$$

7 Open problems

Prove matching lower and upper bounds for the certificate complexity lifting Theorem 5.4 shows that one there is a function f such that $\mathrm{C}(f) \geq \Omega(\mathrm{C}_\oplus(f \diamond \mathrm{OR})^2)$. However, Theorem 5.6 shows only that $\mathrm{C}(f) \leq O(\mathrm{C}_\oplus(f \diamond \mathrm{OR})^3)$. Can we show that $\mathrm{C}(f) \leq O(\mathrm{C}_\oplus(f \diamond \mathrm{OR})^2)$?

Generalization of current results to relations Almost all the results discussed above were proved for Boolean functions. However, the question of proving lifting dichotomies for relations is still open.

One motivation for studying relations instead of functions is the following: any tree-like Resolution refutation of a CNF formula corresponds to a *decision tree*, solving the *falsified clause problem* for this CNF (which is usually a relation rather than Boolean function). Similarly, any tree-like Res($\oplus$) refutation of some CNF formula corresponds to a *parity decision tree*, solving the falsified clause problem for this CNF. So, any lifting theorem from decision trees to parity decision trees with constant size gadgets that holds for relations, should give us a new way of proving tree-like Res($\oplus$) lower bounds.

Conjunction DAG to parity conjunction DAG lifting A parity conjunction DAG is defined similarly to a conjunction DAG, with the following two differences:

- Queries are linear forms rather than variables.
- Nodes are annotated by affine subspaces rather than partial assignments. The label $\rho(v_b)$ of a child node v_b, that is attached to the parent node v via an edge labeled b, is a subspace of $\rho(v) \cup \{\ell \leftarrow b\}$, where ℓ is the query in v.

Another possible direction of research is to prove a lifting theorem from conjunction DAG size to parity conjunction DAG size. The motivation for this kind of lifting also comes from the proof complexity. Dag-like Resolution refutation can be viewed as a conjunction DAG and dag-like Res($\oplus$) refutation can be viewed as a parity conjunction DAG. So, proving such lifting theorems for the relations with small enough gadgets can be used to prove lower bounds for Res($\oplus$) refutations, which is a long-standing open problem in proof complexity.

A Decision tree depth to parity decision size for stifling gadgets

In this section, we paraphrase the proof of Theorem 4 in [CMSS22]. Given a parity decision tree for $f \diamond g$ of size (number of leaves) S, where g is k-stifling for $k \geq 1$, we construct a decision tree for f of depth $(\log S)/k$.

Suppose that f has n inputs and that g has m inputs. We denote the input to f by $x_1, \ldots, x_n$ and the input to $f \diamond g$ by y_{ij}, where $i \in [n]$ and $j \in [m]$. The function $f \diamond g$ is computed by first computing $x_i = g(y_{i1}, \ldots, y_{im})$ and then computing $f(x_1, \ldots, x_n)$.

We will maintain a sequence of *equations* of the form

$$y_{IJ} = y_{i_1 j_1} \oplus \cdots \oplus y_{i_\ell j_\ell} \oplus b.$$

Initially, the sequence is empty. If y_{IJ} is the left-hand side of one of the equations (in this case, we say that y_{IJ} is *highlighted*), then subsequent equations will not involve y_{IJ}.

Given an input x to f, we traverse the parity decision tree for $f \diamond g$ from the root down to the leaves. When we reach a leaf, we simply output the label listed there. When at an internal vertex v labelled ℓ (which is a linear combination of y_{ij}'s), we proceed as follows. First, iteratively substitute the left-hand side of each equation by the right-hand side. Let the final result be ℓ'. If ℓ' is constant, then we descend to the corresponding child of v. Otherwise, we descend to the child of v whose subtree has fewer leaves. Suppose that this corresponds to $\ell' = b$. We choose an arbitrary y_{IJ} occurring in ℓ', say $\ell' = y_{IJ} \oplus \ell''$, and add the equation $y_{IJ} = \ell'' \oplus b$. We say that y_{IJ} is highlighted *non-trivially.*

If y_{IJ} is the k'th highlighted input among $y_{I1}, \ldots, y_{Im}$, then we query x_I, set the non-highlighted y_{Ij}'s so that $g(y_{I1}, \ldots, y_{Im}) = x_I$ no matter the value of the highlighted inputs, and add the equations $y_{IJ'} = b_{IJ'}$ corresponding to the $m-k$ inputs set in this way (where $b_{IJ'}$ is the value to which $y_{IJ'}$ is set).

By construction, each time a variable is highlighted non-trivially, the number of leaves decreases by a factor of at least 2. If we query D inputs out of $x_1, \ldots, x_n$, then we have highlighted non-trivially at least kD inputs out of y_{ij}, and so $S \geq 2^{kD}$. Consequently $D \leq (\log S)/k$.

It remains to show that the output of the decision tree is correct. To see this, it suffices to show that there is some input y such that $g(y) = x$ and y reaches the same leaf of the parity decision tree. Start by querying all x_i not queried so far, and setting the non-highlighted y_{ij} so that $g(y_{i1}, \ldots, y_{im}) = x_i$, which is possible since at most $k-1$ of these are highlighted. The system of equations now involves all y_{ij}. Since the system is triangular by construction, it has a solution. The solution corresponds to an input y which reaches the given leaf, and it satisfies $g(y) = x$ by construction.

References

[AJPU07] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory Comput.*, 3:81–102, 2007.

[BBC$^+$01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, jul 2001.

[BC99] A. Bernasconi and B. Codenotti. Spectral analysis of Boolean functions as a graph eigenvalue problem. *IEEE Transactions on Computers*, 48(3):345–351, 1999.

[Bd02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. Complexity and Logic.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *J. ACM*, 48(2):149–169, mar 2001.

[Che09] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1), dec 2009.

[CKLM19] Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudo-random properties. *Comput. Complex.*, 28(4):617–659, dec 2019.

[CMSS22] Arkadev Chattopadhyay, Nikhil S. Mande, Swagato Sanyal, and Suhail Sherif. Lifting to parity decision trees via stifling, 2022.

[dRMN$^+$20] Susanna de Rezende, Or Meir, Jakob Nordstrom, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 24–30, 11 2020.

[GGKS20] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from Resolution. *Theory Comput.*, 16:Paper No. 13, 30, 2020.

[GLM$^+$16] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016.

[GP18] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018.

[GPW17] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–143, 2017.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, 2018.

[HHL18] Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Structure of protocols for XOR functions. *SIAM J. Comput.*, 47(1):208–217, 2018.

[HN12] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: amplifying communication complexity hardness to time-space trade-offs in proof complexity [extended abstract]. In *STOC'12—Proceedings of the 2012 ACM Symposium on Theory of Computing*, pages 233–247. ACM, New York, 2012.

[KLMY21] Alexander Knop, Shachar Lovett, Sam McGuire, and Weiqiang Yuan. Log-rank and lifting for AND-functions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 197–208, New York, NY, USA, 2021. Association for Computing Machinery.

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1996.

[LMM$^+$22] 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)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 104:1–104:24, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Lov16] Shachar Lovett. Communication is bounded by root of rank. *J. ACM*, 63(1), feb 2016.

[LS88] László Lovász and Michael Saks. Lattices, mobius functions and communications complexity. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 81–90, 1988.

[LS93] László Lovász and Michael Saks. Communication complexity and combinatorial lattice theory. *Journal of Computer and System Sciences*, 47(2):322–349, 1993.

[Raz92] A. A. Razborov. On the distributional complexity of disjointness. *Theoret. Comput. Sci.*, 106(2):385–390, 1992.

[RM99] Ran Raz and Pierre Mckenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19, 09 1999.

[She10] Alexander A. Sherstov. On quantum-classical equivalence for composed communication problems. *Quantum Inf. Comput.*, 10(5-6):435–455, 2010.

[Urq11] Alasdair Urquhart. The depth of resolution proofs. *Studia Logica: An International Journal for Symbolic Logic*, 99(1/3):349–364, 2011.

[Zha09] Shengyu Zhang. On the tightness of the Buhrman–Cleve–Wigderson simulation. In *Proceedings of the 20th International Symposium on Algorithms and Computation*, ISAAC '09, page 434–440, Berlin, Heidelberg, 2009. Springer-Verlag.

[ZS10] Zhiqiang Zhang and Yaoyun Shi. On the parity complexity measures of Boolean functions. *Theoretical Computer Science*, 411(26):2612–2618, 2010.