Electronic Colloquium on Computational Complexity, Report No. 143 (2025)

Alternation Depth of Threshold Decision Lists

Vladimir Podolskii and Morgan Prior

Tufts University

Abstract

Linear decision lists are a computational model for Boolean functions built from a sequence of linear threshold function queries. Each query is evaluated in order: if a query returns true, the list outputs the value of the function, and if the answer is false, the process continues to the next query. The size of a linear decision list is the number of queries in it.

Linear decision lists form a natural and nontrivial subclass of depth-2 threshold circuits, the class of circuits that currently marks the frontier of explicit circuit lower bounds. While some techniques exist for proving lower bounds against linear decision lists, they are quite limited, leaving important open problems unresolved. Moreover, for the related model of exact linear decision lists, no strong lower bounds are known.

We initiate the study of alternation depth of decision lists. The alternation depth is defined as the number of alternations in the output values of the decision list within the sequence of its queries.

We show that linear decision lists, with both bounded and unbounded query weights, form fine hierarchies with respect to alternation depth. We establish a similar hierarchy for rectangle decision lists, the model closely related to the communication complexity with NP oracles. In all settings, we prove strong separations within these hierarchies and between them.

Next, we give a lower bound for an explicit function for exact linear decision lists up to depth $n/\log n$. Such lower bounds were not previously known and do not follow directly from existing methods. We also establish a fine depth hierarchy for exact linear decision lists.

To prove these hierarchy separations, we introduce an iterative technique, used in combination with existing techniques such as fooling sets and the analysis of blocky matrices. For the lower bound on bounded-depth exact linear decision lists, we combine the discrepancy method with an iterative analysis of blocky matrices.

1 Introduction

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a *linear threshold function* (LTF) if there exist weights $w_0, w_1, \ldots, w_n \in \mathbb{N}$ such that

$$f(x) = \operatorname{sgn}(w_0 + \sum_{i=1}^{n} w_i x_i),$$

where we let $\operatorname{sgn}(t) = 1$ for $t > 0$ and $\operatorname{sgn}(t) = 0$ otherwise. The class TC^0 of polynomial-size, constant-depth circuits with LTFs as gates is central in circuit complexity [26, 19, 9]. Proving strong lower bounds for explicit functions even for the case of depth 2 is one of the main frontiers in this area [26, 19, 5]. We denote the class of functions computable by polynomial-size, depth-2 threshold circuits by $\mathrm{LTF} \circ \mathrm{LTF}$. In attempts to prove lower bounds for $\mathrm{LTF} \circ \mathrm{LTF}$, researchers have studied weight-restricted classes. Define $\widehat{\mathrm{LTF}}$ as the class of functions in LTF with the additional restriction that for each i, $|w_i| = O(\mathrm{poly}(n))$.

ISSN 1433-8092

In many cases, the weight-restricted class is weaker than its unbounded analogue. For example, Myhill and Kautz [22] gave an explicit function that is computable by LTF not by $\widehat{\text{LTF}}$[1]. On the other hand, larger depth can make bounded weight classes more powerful: PARITY is computable in $\widehat{\text{LTF}} \circ \widehat{\text{LTF}}$ but not in LTF (see [26]). Hajnal et al. [13] proved a lower bound against $\widehat{\text{LTF}} \circ \widehat{\text{LTF}}$ using essentially the discrepancy method (see also [20]). Surprisingly, Goldmann, Håstad, and Razborov [10] showed that weight restrictions do not always weaken the class. Specifically, they proved $\widehat{\text{LTF}} \circ \widehat{\text{LTF}} = \widehat{\text{LTF}} \circ \text{LTF}$, meaning that the weights on bottom gates are irrelevant when the top weights are small. They further showed that $\widehat{\text{LTF}} \circ \text{LTF} \subsetneq \text{LTF} \circ \widehat{\text{LTF}}$, demonstrating that large weights on the top gates do increase computational power, and that $\text{LTF} \circ \widehat{\text{LTF}}$ is the largest class among depth-2 threshold circuits with restrictions on weights.

Forster et al. [8] obtained a lower bound against $\text{LTF} \circ \widehat{\text{LTF}}$ using the sign-rank method, which has since become a standard lower bound technique in complexity theory [27, 7, 24]. However, Chattopadhyay et al. [3] showed that sign-rank fails to prove lower bounds against LTF ∘ LTF: they exhibited an explicit function with large sign-rank that is computable in LTF ∘ LTF. From this, it follows that $\text{LTF} \circ \widehat{\text{LTF}} \subsetneq \text{LTF} \circ \text{LTF}$. Thus, the current picture of the threshold circuit hierarchy up to LTF ∘ LTF is

$$\widehat{\text{LTF}} \subsetneq \text{LTF} \subsetneq \widehat{\text{LTF}} \circ \widehat{\text{LTF}} = \widehat{\text{LTF}} \circ \text{LTF} \subsetneq \text{LTF} \circ \widehat{\text{LTF}} \subsetneq \text{LTF} \circ \text{LTF}.$$

The challenge of proving lower bounds against LTF ∘ LTF motivates the study of other restricted computational models related to threshold functions, such as linear decision lists [2, 5].

Linear decision lists are a specific case of the general computational model known as decision lists [28]. An S-decision list $\mathcal{L}$ of size s computing a Boolean function $f \in B_n$ is a sequence of s 3-tuples and a bit

$$(q_1, a_1, b_1), (q_2, a_2, b_2) \ldots (q_s, a_s, b_s), b_{s+1}.$$

Here, B_n is the set of all Boolean functions in n variables, each $q_i \in S \subseteq B_n$ is a query function, and a_i and b_i are Boolean constants. Given any $x \in \{0,1\}^n$, the value of $\mathcal{L}(x)$ is b_i if i is the smallest index such that $q_i(x) = a_i$; if there is no such i, then $\mathcal{L}(x) = b_{s+1}$. A *linear decision list* (LDL) is an S-decision list with $S = \text{LTF}$. It is not hard to see that the class of linear decision lists is contained in LTF ∘ LTF [2].

Several techniques exist for proving lower bounds on LDL size. One such technique is monochromatic rectangle size. To use this technique on a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we consider its *communication matrix* M_f, the $2^n \times 2^n$ matrix with entries $M_f[x, y] := f(x, y)$. It was established in [2] that if $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has no monochromatic combinatorial rectangle of size greater than $w \cdot 2^n$, then any LDL computing f must have size at least $\frac{1}{\sqrt{w}}$. In particular, a polynomial length LDL has at least one large (having size that is a polynomial fraction of the communication matrix) rectangle.

Sign-rank can also be used to prove lower bounds against the size of LDLs. More specifically, it is known that if a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is computable by an LDL of size s, then its communication matrix has sign-rank $O(s)$[2].

Despite these techniques, our ability to prove lower bounds for LDL is still rather limited. For example, it is unknown whether the bounded-weight class $\widehat{\text{LDL}}$ is strictly weaker than LDL [2, 6], or whether LDL is strictly weaker than the class of polynomial size linear decision trees.

[1] Because $\widehat{\text{LTF}}$s can be simulated by Majority (MAJ) gates without increasing circuit depth [26], the class $\widehat{\text{LTF}}$ is sometimes denoted MAJ.

[2] This follows from the fact that LDL ⊆ mpPTF (Lemma 18 in [6]) and the fact that sign-rank is a lower bound technique against the class mpPTF (Lemma 7 in [16]).

Another related model (although not a subclass of LTF $\circ$ LTF) is the *rectangle decision list*, whose query functions test membership in combinatorial rectangles. Formally, Rect be the set of all functions $f \in B_n$ such that $f(x,y) = 1 \iff (x,y) \in R$ where R is some combinatorial rectangle in M_f. A rectangle decision list is an S-decision list with $S = \text{Rect}$; we call the class of such decision lists Rect-DL. This class is related to those discussed above, since the rectangle size technique applies to it [2, 17]. This class can also be viewed in light of its connection to the communication complexity class $\text{P}^{\text{NP}^{cc}}$, which allows oracle queries to functions in NP^{cc}. In particular, it is known that Rect-DL = $\text{P}^{\text{NP}^{cc}}$ for quasipolynomial-sized Rect-DLs and $\text{P}^{\text{NP}^{cc}}$ protocol trees [23, 11].

Finally, perhaps the most intriguing subclass of LTF $\circ$ LTF of this flavor is ELDL, the class of decision lists whose query functions are exact threshold functions (ELTFs). We say a function $f : \{0,1\}^n \to \{0,1\}$ is an *exact linear threshold function* (ELTF) if there exist weights $w_0, w_1, \ldots, w_n \in \mathbb{N}$ such that $f(x) = 1 \Leftrightarrow w_0 + \sum_{i=1}^{n} w_i x_i = 0$. An ELDL is an S-decision list with $S = \text{ELTF}$. The class ELDL contains LDL [5] and is strictly more powerful: the Block-Equality function $\text{OR}_n \circ \text{EQ}_n$ requires an exponential-size LDL, but can be computed by a linear-size ELDL [2].

Proving strong lower bounds against ELDL for an explicit function is therefore a natural intermediate step toward lower bounds for LTF $\circ$ LTF. Unfortunately, known lower bound techniques against LDLs do not work for ELDLs. It was observed in [2] that monochromatic rectangle size does not work, as [17] showed that the Block-Equality function has no large monochromatic rectangles. Sign-rank also cannot prove lower bounds against ELDLs, as [3] gave an explicit function with sign-rank $2^{\Omega(n^{1/4})}$ which is computable by a linear-size ELDL.

At present, we lack techniques for proving strong lower bounds against ELDL, and more broadly for understanding threshold decision lists and depth-2 threshold circuits. This motivates the search for new methods to separate and characterize these classes.

Our results. To better understand linear decision lists and related computational models, we introduce another parameter of decision lists, their alternation depth, and study them from this perspective.

For a decision list $\mathcal{L}$ (of any type) described by the sequence

$$(q_1, a_1, b_1), (q_2, a_2, b_2), \ldots (q_s, a_s, b_s), b_{s+1},$$

we define the *alternation depth* of $\mathcal{L}$ to be $|L|$ where $L := \{i : b_i \neq b_{i-1}\}$. Intuitively, we can partition the decision list into contiguous blocks of queries where the outputs b_i are constant within each block. The alternation depth is simply the number of these blocks. For $\mathcal{L} \in \{\text{LDL}, \widehat{\text{LDL}}, \text{Rect-DL}, \text{ELDL}\}$ and $b \in \{0,1\}$, we use the notation $\mathcal{L}_{k,b}$ to denote a $\mathcal{L}$ of depth k with initial output $b_1 = b$. When the initial output is irrelevant, we use simply $\mathcal{L}_k$ to denote an $\mathcal{L}$ of depth k. When the context is clear, we often refer to "alternation depth" simply as "depth."

The main motivation for studying the depth of decision lists is the pursuit of new lower bound techniques. Existing measures such as sign-rank and monochromatic rectangle size are agnostic to depth, so to prove depth-specific results—such as separations between classes of functions computable by different depths—we must introduce new techniques.

In particular, for the size of ELDLs, no strong lower bound techniques for explicit functions are currently known. A natural starting point is to restrict attention to bounded-depth ELDLs and attempt to prove lower bounds for them, thereby enriching our arsenal of techniques.

Restricting depth is also useful for studying reductions between decision list classes. Let $\mathcal{C}$ and $\mathcal{C}'$ be classes of Boolean functions. If we would like to show that any function computable by a small $\mathcal{C}$-decision list is also computable by a small $\mathcal{C}'$-decision list, a straightforward approach

is to try to substitute each query in the $\mathcal{C}$-decision list decision list by a small $\mathcal{C}'$-decision list computing the same query. It is not difficult to see (Proposition 27 below) that this is possible whenever $\mathcal{C} \subseteq \mathcal{C}'_{1,1}$ and $\mathcal{C} \subseteq \mathcal{C}'_{1,0}$.

Another motivation for studying depth is that it arises naturally in the study of rectangle decision lists (Rect-DLs) and their connection to the communication complexity class $\mathrm{P}^{\mathrm{NP}^{cc}}$. It is not difficult to see that the depth of Rect-DL corresponds to the depth of $\mathrm{P}^{\mathrm{NP}^{cc}}$ protocols (Proposition 31 below).

We first focus on LDLs, $\widehat{\text{LDL}}$s, and Rect-DLs of bounded depth, which form natural hierarchies with respect to depth (see Figure 1). It is not hard to see that bounded-weight $\widehat{\text{LDL}}$ classes can be simulated by the corresponding LDL and Rect-DL classes. Our first results establish separations between these hierarchies. We show that the Parity function XOR_n is hard to compute by low depth LDLs, but it can easily be computed by Rect-DL$_1$s (this separation also follows from the counting argument in Proposition 29). In the other direction, the Greater-Than function is easy to compute by LDL$_1$s (one query is enough), but we show that it is hard to compute by $\widehat{\text{LDL}}$ and Rect-DL of small depth. Note that analogous results are unknown without depth restriction (aside from a counting argument separating Rect-DL from other classes).

We further show separations between classes at each level of these hierarchies, and hence between all classes within them, for depths up to $n/\log n$. These results are obtained by composing the Odd-Max-Bit function with functions exhibiting certain communication complexity properties. Intuitively, our method can be viewed as an iterated application of the fooling set technique.

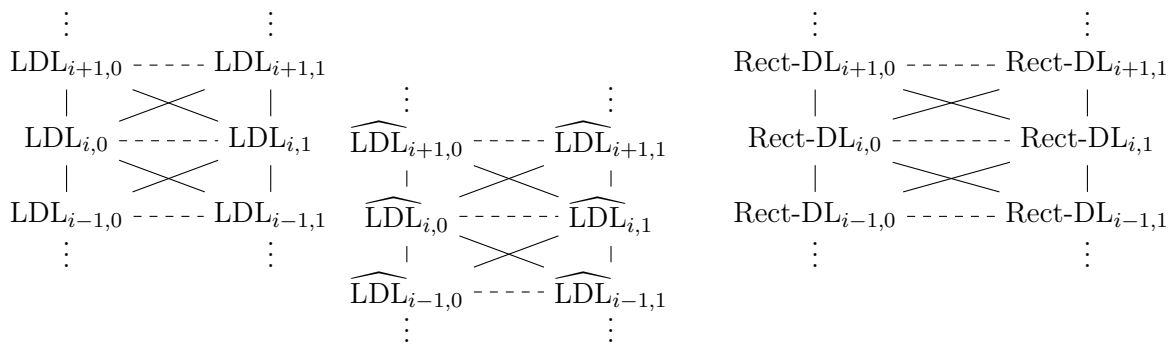

Figure 1: Hierarchy of $\widehat{\text{LDL}}$, LDL, and Rect-DL under various depths. Black lines represent inclusions (where the class at the vertically higher endpoint contains the class at the vertically lower one). Dashed lines indicate separations (in both directions). The separations within each hierarchy are provided in Lemma 38 and Theorem 40. The separations between the hierarchies are provided in Theorem 34, Theorem 36 and Corollary 37.

We then proceed to the study of ELDLs. Unlike the previous models, it remains an open problem to show strong lower bounds for explicit functions in this class. Under bounded depth, however, some lower bounds do follow from existing results. An ELDL$_1$ is essentially an OR $\circ$ ELTF, and a lower bound follows since

$$\mathrm{OR} \circ \mathrm{ELTF} \subseteq \widehat{\mathrm{LTF}} \circ \mathrm{ELTF} = \widehat{\mathrm{LTF}} \circ \widehat{\mathrm{LTF}},$$

where the last equality was implicitly shown in [10] and was explicitly discussed in [15]. More generally, for any constant k, Proposition 25 implies that an ELDL$_k$ can be expressed as $\widehat{\mathrm{LTF}} \circ$ ELTF, resulting in the lower bound.

As our first result for ELDLs we show that any function with small discrepancy under the uniform distribution cannot be computed by an ELDL of polynomial size and depth less than

$n/\log n$. To prove this result, we adapt the discrepancy technique to blocky matrices [14] and apply it recursively.

Next, we consider the depth hierarchy of ELDL classes (see Figure 2). These classes can simulate corresponding LDL classes (see Proposition 30). The strong separation from the LDL hierarchy is known: the Block-Equality function $\mathrm{OR}_n \circ \mathrm{EQ}_n$ is hard to compute by an LDL of any depth [2, 17] and is easy to compute by an ELDL_1. The separation from low-depth Rect-DLs is provided by the Greater-Than function (see Proposition 29). For the separation in the other direction, we only have a counting argument.

For ELDL hierarchy we show separations between classes on each level for depth up to $\sqrt{n}$. Here, we consider a composition of the Odd-Max-Bit function with the Disjointness function. This argument is the most technically heavy in the paper. The proof views queries as blocky matrices and iteratively identifies submatrices of the communication matrix that avoid large intersections with lower-depth queries.

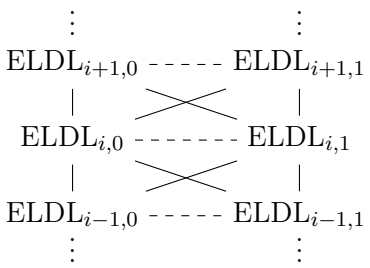

Figure 2: Hierarchy of ELDL under various depths. Solid lines represent inclusions (where the class at the vertically higher endpoint contains the class at the vertically lower one). Dashed lines indicate separations. The separations within the ELDL hierarchy are provided in Lemma 48 and Theorem 50. Separations between the ELDL hierarchy and the ones from Figure 1 are provided in Proposition 29, Lemma 32, and Theorem 34.

The remainder of the paper is organized as follows. Section 2 provides background. Section 3 presents preliminary observations about depth in decision lists. Section 4 gives our results for linear and rectangle decision lists. Section 5 presents our results for exact linear decision lists.

2 Preliminaries

2.1 Communication Complexity

Our proofs rely on notions from communication complexity, which we define below. For more background on communication complexity, see [20] or [25].

Definition 1 (Communication matrix)**.** *For a function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, its communication matrix M_f is the $2^n \times 2^n$ matrix with entries $M_f[x,y] \coloneqq f(x,y)$.*

Definition 2 (Combinatorial rectangle)**.** *Given a matrix M with rows X and columns Y, a* combinatorial rectangle *is the Cartesian product $A \times B$ for some $A \subseteq X$ and $B \subseteq Y$.*

Definition 3 (Monochromatic combinatorial rectangle)**.** *Given a matrix M, a combinatorial rectangle $A \times B$ in M is called b-monochromatic if $M[x,y] = b$ for all $(x,y) \in A \times B$.*

Definition 4 (Function class Rect)**.** *Let* Rect *be the set of all functions $f \in B_n$ such that $f(x,y) = 1 \iff (x,y) \in R$, where R is some combinatorial rectangle in the communication matrix M_f.*

Definition 5 (Discrepancy)**.** *Let $f\colon X \times Y \to \{0,1\}$ be a function, let $R \subseteq X \times Y$ be a rectangle in the communication matrix M_f, and let μ be a probability distribution over $X \times Y$.*

Then the discrepancy of R with respect to μ is

$$\mathsf{Disc}_\mu(R,f) = \left| \Pr_\mu[f(x,y) = 0 \wedge (x,y) \in R] - \Pr_\mu[f(x,y) = 1 \wedge (x,y) \in R] \right|.$$

The discrepancy of f with respect to μ is $\mathsf{Disc}_\mu(f) = \max_R \mathsf{Disc}_\mu(R,f)$.

In this paper, we are interested in discrepancy when $\mu = U$, the uniform distribution.

Definition 6 (Fooling set)**.** *For $b \in \{0,1\}$, a b-fooling set for $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a set $F \subseteq \{0,1\}^n \times \{0,1\}^n$ such that*

1. *For every $(x,y) \in F$, $f(x,y) = b$.*
2. *For all distinct pairs $(x_1,y_1),(x_2,y_2) \in F$, either $f(x_1,y_2) \neq b$ or $f(x_2,y_1) \neq b$.*

2.2 Complexity Classes

Here we provide formal definitions for complexity classes we are considering.

It will be convenient for us to define the function $\mathrm{sgn} : \mathbb{R} \to \{0,1\}$ as follows:

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0. \end{cases}$$

Definition 7 (Linear threshold function)**.** *Linear threshold function* LTF *is the class of all functions $f : \{0,1\}^n \to \{0,1\}$ such that there exist weights $w_0, w_1, \ldots, w_n \in \mathbb{N}$ such that*

$$f(x) = \mathrm{sgn}(w_0 + \sum_{i=1}^{n} w_i x_i).$$

Definition 8 (Exact linear threshold function)**.** *Exact linear threshold function* ELTF *is the class of all functions $f : \{0,1\}^n \to \{0,1\}$ such that there exist weights $w_0, w_1, \ldots, w_n \in \mathbb{N}$ such that*

$$f(x) = 1 \Leftrightarrow w_0 + \sum_{i=1}^{n} w_i x_i = 0.$$

Definition 9 (Bounded weight linear threshold function)**.** *Bounded weight linear threshold function $\widehat{\mathrm{LTF}}$ is the class of functions $f : \{0,1\}^n \to \{0,1\}$ such that there exist weights $w_0, w_1, \ldots, w_n \in \mathbb{N}$ with $|w_i| = O(poly(n))$ such that*

$$f(x) = \mathrm{sgn}(w_0 + \sum_{i=1}^{n} w_i x_i).$$

Exact bounded weight linear threshold function $\widehat{\mathrm{ELTF}}$ is the class of functions $f : \{0,1\}^n \to \{0,1\}$ such that there exist weights $w_0, w_1, \ldots, w_n \in \mathbb{N}$ with $|w_i| = O(poly(n))$ such that

$$f(x) = 1 \Leftrightarrow w_0 + \sum_{i=1}^{n} w_i x_i = 0.$$

Definition 10 (S-decision list, LDL, $\widehat{\text{LDL}}$, ELDL, Rect-DL, DL)**.** *Let B_n denote the set of all Boolean functions in n variables and $S \subset B_n$ be some function class. An S-decision list $\mathcal{L}$ of size s computing a Boolean function $f \in B_n$ is a sequence sequence of s 3-tuples and a bit*

$$(q_1, a_1, b_1), (q_2, a_2, b_2) \dots (q_s, a_s, b_s), b_{s+1},$$

where each $q_i \in S$ is a query function, and the a_i and b_i are Boolean constants. Given any $x \in \{0,1\}^n$, the value of $\mathcal{L}(x)$ is b_i if i is the smallest index such that $q_i(x) = a_i$; if there is no such i, then $\mathcal{L}(x) = b_{s+1}$.

From this definition, we introduce several complexity classes related to decision lists that are important for our results:

- *Linear decision list* (LDL) *is the class of functions computable by an S-decision list with $S = \text{LTF}$.*
- *Bounded-weight linear decision list* ($\widehat{\text{LDL}}$) *is the class of functions computable by an S-decision list with $S = \widehat{\text{LTF}}$.*
- *Exact linear decision list* (ELDL) *is the class of functions computable by an S-decision list with $S = \text{ELTF}$.*
- *Rectangle decision list* (Rect-DL) *is the class of functions computable by an S-decision list with $S = \text{Rect}$.*
- *Decision list* (DL) *is the class of functions computable by an S-decision list where S is the set of Boolean functions that query exactly one bit of x.*

We say that a query q_i *covers* an input $x \in \{0,1\}^n$, if q_i is the query producing the output for x. That is, q_i covers x if $q_i(x) = a_i$.

Remark 11. *Without loss of generality, we can assume that $b_s \neq b_{s+1}$. Indeed, if $b_s = b_{s+1}$, then performing query q_s does not change the output value and can be omitted.*

When discussing decision lists, we sometimes use the same notation both for the computational model and for the class of functions that are efficiently (polynomial-size) computable by this model. For example, we use LDL to refer both to the object of a linear decision list and to the class of Boolean functions which are computable by an LDL *of polynomial length.*

It is often convenient to think of the decision lists in Definition 10 as outputting a value only when the query function evaluates to true. The following lemma shows that, without loss of generality, we can assume this is the case:

Lemma 12. *Let f be a computable by an S-decision list of size s for some $S \in \{\text{LDL}, \widehat{\text{LDL}}, \text{Rect}, \text{ELDL}\}$. Then f is computable by an S-decision list of size $t = O(s \cdot poly(n))$ defined by*

$$(q_1, a_1, b_1), (q_2, a_2, b_2), \dots, (q_t, a_t, b_t), b_{t+1},$$

where $a_i = 1$ for all i.

Proof. Since LTF is closed under complement, the claim is clearly true for LDL and $\widehat{\text{LDL}}$: for each i such that $a_i = 0$, we can substitute q_i with $\neg q_i$ and make $a_i = 1$.

For ELDLs, let $\mathcal{E}$ be an ELDL computing f and $q \in \text{ELTF}$ be a query in $\mathcal{E}$ that outputs a value when $q(x) = 0$. Then, since $q \in \text{ELTF}$, there exist weights $w_0, w_1, \dots, w_n \in \mathbb{N}$ such that $q(x) = 1 \Leftrightarrow w_0 + \sum_{i=1}^n w_i x_i = 0$. Then for $\neg q$, we have that $\neg q(x) = 1 \iff (w_0 + \sum_{i=1}^n w_i x_i <$

$0) \vee (w_0 + \sum_{i=1}^{n} w_i x_i > 0)$. That is, $\neg q \in \text{OR} \circ \text{LTF}$. It is known that an LTF can be represented as a disjoint OR of polynomially many ELTFs [15, 4], and thus $\neg q$ can be represented as an OR of polynomially many ELTF functions.

Hence, for i such that $a_i = 0$, we can replace (q_i, a_i, b_i) with a polynomial-length sequence $(q'_1, 1, b_i), \ldots, (q'_j, 1, b_i)$, without changing the output of $\mathcal{E}$.

Finally, to see that it is true for Rect-DLs, note that the complement of a rectangle $A \times B$ can be queried by two rectangles $(X \setminus A) \times Y$ and $A \times (Y \setminus B)$, where X and Y are the sets of all rows and columns respectively; call these rectangle queries q'_1 and q'_2. Then, for each i such that $a_i = 0$, we can substitute (q_i, a_i, b_i) with $(q'_1, 1, b_i), (q'_2, 1, b_i)$. □

Definition 13 (Blocky system of rectangles)**.** *Given a Boolean matrix M, a system of rectangles $R_1, \ldots, R_s$ with $R_i = A_i \times B_i$ is* blocky *if subsets $A_1, \ldots, A_s$ are pairwise disjoint and subsets $B_1, \ldots, B_s$ are pairwise disjoint.*

Definition 14 (Blocky matrix)**.** *A Boolean matrix M is a* blocky matrix *if there is a blocky system of rectangles $R_1, \ldots, R_s$ such that $M(x, y) = 1$ iff there is i such that $(x, y) \in A_i \times B_i$. In other words, ones of the blocky matrix form a set of row- and column-disjoint monochromatic rectangles. A blocky matrix is also sometimes called an* equality matrix*.*

Lemma 15. *If $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is in* ELTF*, its communication matrix M_f is blocky.*

Proof. Let f be defined such that $f(x, y) = 1 \iff w_0 + \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{n} b_j y_j = 0$. Let $\alpha(x) \coloneqq \sum_{i=1}^{n} a_i x_i$, $\beta(y) \coloneqq \sum_{j=1}^{n} b_j y_j$, and $t \coloneqq -w_0$. Then $M_f[x, y] = 1 \iff \alpha(x) + \beta(y) = t$. Now we can group sets of rows by the value of $\alpha(x)$ in the following way:

$$A_s \coloneqq \{x : \alpha(x) = s\},$$

and we can similarly group columns by the value of $\beta(y)$:

$$B_r \coloneqq \{y : \beta(y) = r\}.$$

Since each row in A_s has a 1 exactly at cells for which the corresponding column has $\beta(y) = t - s$, all rows in the same A_s are identical. Similar reasoning shows all columns in the same B_r are identical. To see that this induces a blocky structure, observe that since $\alpha(x) = s$ and $\beta(y) = t - s$ imply $\alpha(x) + \beta(y) = t$, the rectangle $A_s \times B_{t-s}$ is a 1-monochromatic rectangle. Finally, observe that outside of the rectangles $A_s \times B_{t-s}$, we have zeros. □

Next, we introduce the notion of the alternation depth of a decision list.

Definition 16 (Alternation depth)**.** *For a decision list $\mathcal{L}$ (of any type) described by the sequence*

$$(q_1, a_1, b_1), (q_2, a_2, b_2) \ldots (q_s, a_s, b_s), b_{s+1}$$

we define the alternation depth *of $\mathcal{L}$ to be $|L|$ where $L \coloneqq \{i : b_i \neq b_{i-1}\}$.*

Intuitively, alternation depth of a decision list is the number of alternations in the outputs b_i. For a class $\mathcal{C} \in \{\text{LDL}, \widehat{\text{LDL}}, \text{Rect-DL}, \text{ELDL}\}$, we use $\mathcal{C}_k$ to denote a subclass of $\mathcal{C}$ of alternation depth at most k. If we wish to specify the first output of the decision list, we use the notation $\mathcal{C}_{k,b}$ to denote a subclass of $\mathcal{C}$ and alternation depth k whose first query outputs b for $b \in \{0, 1\}$. For brevity, when it is clear from context, we often refer to alternation depth as just depth.

We call a set of consecutive leaf nodes with the same output a *depth layer* or just a *layer* when it is clear from context. If the queries in a depth layer output 1 (0), we call it a *1-layer* (resp. *0-layer*). We use *length* of a depth layer to refer to the number of queries in it. We sometimes call a query in the ith depth layer a query of depth i.

Remark 17. AND_n *and* OR_n *are clearly in* DL_1*. Note that every Boolean function is computable by* $\widehat{\mathrm{LDL}}_1$ *of exponential size. Indeed, every Boolean function can be simulated by a formula in disjunctive normal form [19], and we can translate this formula into an* $\widehat{\mathrm{LDL}}_1$ *of exponential length.*

Next, we introduce a communication complexity version of $\mathrm{P}^{\mathrm{NP}^{cc}}$.

Definition 18 ($\mathrm{P}^{\mathrm{NP}^{cc}}$ [12])**.** $\mathrm{P}^{\mathrm{NP}^{cc}}$ *is a communication complexity class that allows oracle queries to some function in* NP*. Formally, a protocol tree for a function in* $\mathrm{P}^{\mathrm{NP}^{cc}}$ *is such that each internal node* v *is labeled with either*

1. *a 1-bit function of one player's input in the usual way, or*
2. *an "*NP *oracle query" consisting of a collection of rectangles* $S_{v,w} : w \in \{0,1\}^{k_v}$*, where the indicator of whether* $(x, y) \in \bigcup_w S_{v,w}$ *determines which child to descend to in the protocol tree.*

As usual, the output of the protocol is determined by the leaf reached.

The complexity measure for this class is the maximum over all root-to-leaf paths of the following: the length of the path plus the sum of k_v *over all type-2 nodes* v *on the path.*

Since we are considering only the communication complexity version of standard complexity classes in this paper, we will hereafter omit the indication that this is a communication complexity class, denoting it simply P^{NP}. We use *depth* of a protocol to refer to the depth of the protocol tree

Remark 19. *Without loss of generality, we can assume all nodes in the protocol tree for a* P^{NP} *protocol are type-2 (i.e., oracle queries), since type-1 nodes can be expressed as a query to a single monochromatic rectangle.*

2.3 Relevant Functions

Definition 20. *We consider several well-known Boolean functions. Here* $x, y \in \{0,1\}^n$ *and* $x_1, \ldots, x_n, y_1, \ldots, y_n \in \{0,1\}$*.*

- *The* Odd-Max-Bit *function,* $\mathrm{OMB}_n(x_1, \ldots, x_n) = \max\{i : x_i = 1\} \pmod 2$.
- *The* Equality *function,* $\mathrm{EQ}_n(x, y) = \mathbb{1}[x = y]$.
- *The* Non-Equality *function,* $\mathrm{NEQ}_n(x, y) = \mathbb{1}[x \neq y]$.
- *The* Disjointness *function,* $\mathrm{DISJ}_n(x, y) = \mathbb{1}[X \cap Y = \emptyset]$ *(where* X *and* Y *are the sets for which* x *and* y *are respective indicator strings).*
- *The* Intersection *function,* $\mathrm{INT}_n(x, y) = \mathbb{1}[X \cap Y \neq \emptyset]$ *(where* X *and* Y *are the sets for which* x *and* y *are respective indicator strings).*
- *The* Parity *function* $\mathrm{XOR}_n(x_1, \ldots, x_n) = \sum_{i=1}^n x_i \pmod 2$.
- *The* Greater-Than *function,* $\mathrm{GT}_n(x, y) = \mathbb{1}[\mathrm{int}(x) \geq \mathrm{int}(y)]$ *where* $\mathrm{int}(x)$ *is the integer given by* x *as its binary representation.*
- *The* Inner Product *function* $\mathrm{IP}_n(x, y) = \langle x, y \rangle \pmod 2 = \bigoplus_{i=1}^n x_i \wedge y_i$.

We will need the following well-known property of Disjointness, which can be found in [20].

Claim 21. *Let M be the communication matrix for* DISJ_n*. Then if R is a 1-monochromatic rectangle in M, $|R| \leq 2^n$.*

It will be more convenient for us to work with a roughly equivalent version of Odd-Max-Bit:

Definition 22 (Even-Min-Bit)**.** *For a positive integer n, the Even-Min-Bit function on n inputs, denoted* EMB_n*, is defined by*

$$\mathrm{EMB}_n(x_1, \ldots, x_n) = 1 \iff \min\{i \in [n] \mid x_i = 1\} \text{ is even.}$$

Define $\mathrm{EMB}_n(0^n) = 0$ *if n is even and* $\mathrm{EMB}_n(0^n) = 1$ *if n is odd.*

Next we introduce functions that we use for our separation results.

Definition 23 (k-Layer Non-Equality)**.** *For a positive integer n, the k-Layer Non-Equality function on $2n$ inputs, denoted* $\mathrm{NEQ}_n^{(k)}$ *(see Figure 3 in below), is the composition* $\mathrm{EMB}_k \circ \mathrm{NEQ}_{n/k}$*, defined by*

$$\mathrm{NEQ}_n^{(k)}(x_1, y_1, \ldots, x_k, y_k) = 1 \iff \min\{i \in [n] \mid x_i \neq y_i\} \text{ is even.}$$

Consistent with the definition of EMB*, we say that* $\mathrm{NEQ}_n^{(k)}(x_1, y_1, \ldots, x_k, y_k) = 0$ *in the case that $x_i = y_i$ for all i if k is even, and* $\mathrm{NEQ}_n^{(k)}(x_1, y_1, \ldots, x_k, y_k) = 1$ *in the case that $x_i = y_i$ for all i if k is odd.*

Definition 24 (k-Layer Intersection)**.** *For a positive integer n, the k-Layer Intersection function on $2n$ inputs, denoted* $\mathrm{INT}_n^{(k)}$ *(see Figure 4 below), is the composition* $\mathrm{EMB}_k \circ \mathrm{INT}_{n/k}$ *is defined by*

$$\mathrm{INT}_n^{(k)}(x_1, y_1, \ldots, x_k, y_k) = 1 \iff \min\{i \in [n] \mid x_i \cap y_i \neq \emptyset\} \text{ is even.}$$

Similar to $\mathrm{NEQ}^{(k)}$*, we say that* $\mathrm{INT}_n^{(k)}(x_1, y_1, \ldots, x_k, y_k) = 0$ *in the case that $x_i \cap y_i \neq \emptyset$ for all i if k is even, and* $\mathrm{INT}_n^{(k)}(x_1, y_1, \ldots, x_k, y_k) = 1$ *in the case that $x_i \cap y_i \neq \emptyset$ for all i if k is odd.*

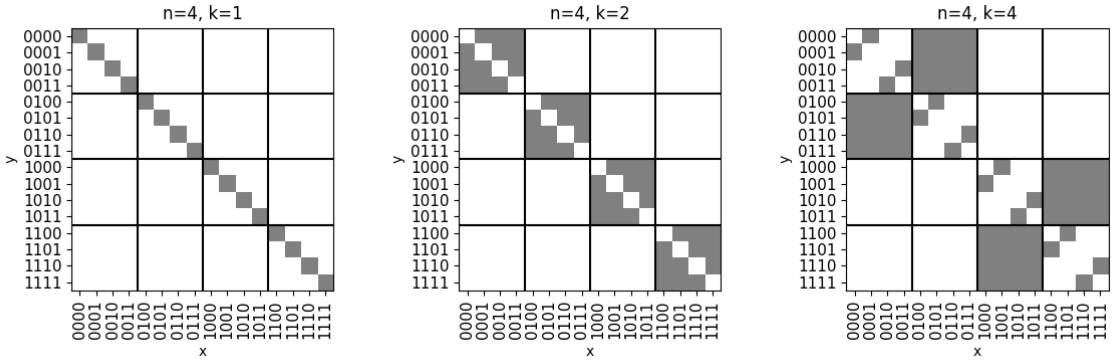

Figure 3: Communication matrix for $\mathrm{NEQ}_4^{(k)}$ shown with depths 1, 2, and 4. Zero entries are white and one entries are gray. Note that for depth = 1, we have $\mathrm{NEQ}_4^{(1)} = \mathrm{EQ}_4$.

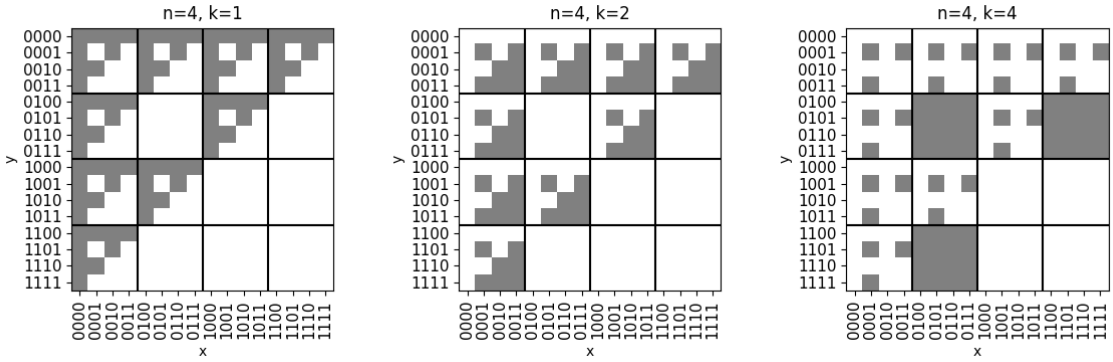

Figure 4: Communication matrix for $\mathrm{INT}_4^{(k)}$ on 4 bits, shown with depths 1, 2, and 4. Zero entries are white and one entries are gray. Note that for depth = 1, we have $\mathrm{INT}_4^{(1)} = \mathrm{DISJ}_4$.

3 Initial Observations

First we observe some depth-related connections between regular decision lists and linear decision lists.

Proposition 25. *For any constant k, we have* $\mathrm{DL}_k \subseteq \widehat{\mathrm{LTF}}$.

Proof. The idea of this argument is that we can sum up variables queried in a depth-d decision list with weights decreasing with each successive query. We construct the weights such that they only decrease substantially when we switch between depth layers of the decision list. If there are constantly many layers, the weights are polynomial. Below, we provide a formal argument.

Let $f : \{0,1\}^n \to \{0,1\}$ be computable by a depth-k decision list $\mathcal{D}$. Let M denote the length of the longest depth layer in $\mathcal{D}$. For each query j, let

$$\ell_j = \begin{cases} x_j & \text{if } a_i = 1, \\ 1 - x_j & \text{if } a_i = 0. \end{cases}$$

In other words, $\ell_j = 1$ iff the answer to the query leads to an immediate output. For each depth layer i, let $S_i = \sum_j \ell_j$, where the sum is over all queries in this layer. Clearly, $0 \le S_i \le M$ and the first layer with non-zero S_i produces the output of the decision list.

Denote by b_i for $i = 1, \ldots, k$ the output of queries on layer i, denote by b_{k+1} the output in the case when none of the queries resulted in an output. Consider the following linear inequality:

$$2\sum_{i=1}^{k}(-1)^{b_i+1}(M+1)^{k-i}S_i + (-1)^{b_{k+1}+1} \ge 0.$$

We claim that it defines an LTF computing f.

Indeed, consider the first S_i that is not equal to 0 (if there is no such S_i, the inequality is correct iff $b_{k+1} = 1$ as desired). The signs of the terms are arranged so that it is enough to check that the ith term of the sum dominates all subsequent terms in absolute value. Since $S_i \ge 1$, the absolute value of this term is at least $(M+1)^{k-i}$. For all further terms, we have $S_i \le M$, Thus, it suffices to verify that

$$(M+1)^{k-i} > (M+1)^{k-i-1}M + \ldots + (M+1)^{k-k}M.$$

Indeed, for the right-hand side we have

$$(M+1)^{k-i-1}M + \ldots + (M+1)^{k-k}M = M\frac{(M+1)^{k-i}-1}{(M+1)-1} = (M+1)^{k-i} - 1$$

and the inequality follows.

Since $M = \text{poly}(n)$ and k is constant, each weight in the constructed linear threshold function has magnitude at most $\text{poly}(n)$. □

Proposition 26. $\text{DL} \subseteq \widehat{\text{LDL}}_1$.

Proof. Consider a function f computable by a DL $\mathcal{D}$. This function outputs 1, if there is a step (q_i, a_i, b_i) in $\mathcal{D}$ with $b_i = 1$ such that $q_i(x) = a_i$ and for all $j < i$ we have $q_j(x) = \neg a_j$. By the definition of DL, each q_i is a function computing the value of exactly one input variable. For each i, the condition '$q_i(x) = a_i$ and for all $j < i$ we have $q_j(x) = \neg a_j$' is an AND of literals, and thus can be computed by an $\widehat{\text{LTF}}$. To compute f, it is enough to check if such an i exists, thus f is computable by $\text{OR} \circ \widehat{\text{LTF}}$, which is computable by $\widehat{\text{LDL}}_1$. □

The next observation shows how the notion of a depth-1 decision list is useful for reductions between various classes of decision lists.

Proposition 27. *Let $\mathcal{C}$ and $\mathcal{C}'$ be classes of functions. Suppose $\mathcal{C} \subseteq \mathcal{C}'_{1,1}$ and $\mathcal{C} \subseteq \mathcal{C}'_{1,0}$. Then any polynomial-size $\mathcal{C}$-decision list can be converted into equivalent polynomial-size $\mathcal{C}'$-decision list. Moreover, the depth of the decision list does not change.*

If $\mathcal{C}$ is closed under negation, it is enough to have $\mathcal{C} \subseteq \mathcal{C}'_{1,1}$ or $\mathcal{C} \subseteq \mathcal{C}'_{1,0}$ to reach the same conclusions.

Proof. Let $\mathcal{L}$ be a $\mathcal{C}$-decision list of polynomial size. Consider an arbitrary query $q \in \mathcal{C}$, assume that the output b is produced if the answer to the query is a. Consider a polynomial size $\mathcal{C}'_{1,a}$-decision list computing q, replace the outputs in each query by b and the default output by $\neg b$. Replace query q in $\mathcal{L}$ by the resulting decision list. It is not hard to see that the resulting decision list computes the same function.

If $\mathcal{C}$ is closed under negation and $\mathcal{C} \subseteq \mathcal{C}'_{1,1}$, then $\mathcal{C} \subseteq \mathcal{C}'_{1,0}$ and vice versa. Indeed, assume $\mathcal{C} \subseteq \mathcal{C}'_{1,1}$ and for any $f \in \mathcal{C}$ consider $\neg f \in \mathcal{C}$. Consider $\mathcal{C}'_{1,1}$-decision list computing $\neg f$. Negating all of its outputs results in a $\mathcal{C}'_{1,0}$-decision list computing f. The other direction is the same. □

Now we can use this connection to show the following.

Proposition 28. $\widehat{\text{LDL}} \subseteq \text{Rect-DL}$ *and* $\widehat{\text{LDL}}_k \subseteq \text{Rect-DL}_k$.

Proof. By Proposition 27, since $\widehat{\text{LTF}}$ is closed under negation, it is enough to show that $\widehat{\text{LTF}} \subseteq \text{Rect-DL}_{1,1}$. Consider $f \in \widehat{\text{LTF}}$ and suppose it is represented by linear inequality $\sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} b_i y_i \geq t$, where the a_i and b_i are polynomial weights. Now consider all integer pairs $A, B \geq 0$ such that $A + B \geq t$ and there are $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ such that $\sum_{i=1}^{n} a_i x_i = A$ and $\sum_{i=1}^{n} b_i y_i = B$. Since the weights a_i and b_i are polynomial in n, the absolute values of A and B are also at most polynomial, and thus there are at most polynomially many such A, B pairs.

We construct a Rect-DL computing f as follows. For each integer pair (A, B), we make the query defined by the following rectangle: $(\sum_{i=1}^{n} a_i x_i = A) \wedge (\sum_{i=1}^{n} b_i y_i = B)$. If the rectangle query is true, the Rect-DL outputs 1, and otherwise it proceeds to the next pair. If none of the queries are true, the Rect-DL outputs 0. This results in a $\text{Rect-DL}_{1,1}$ of polynomial size computing f. □

Next, we note that the converse is not true even if we drop the restriction on the weights.

Proposition 29. Rect-DL $\not\subseteq$ LDL *and* Rect-DL $\not\subseteq$ ELDL.

Proof. This observation follows by a simple counting argument. If we consider the communication matrix for a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, there are 2^{2^n} subsets of rows and 2^{2^n} subsets of columns, and hence $2^{\Omega(2^n)}$ rectangles. However, there are only $\mathcal{O}(2^{\text{poly}(n)})$ linear threshold functions [21] and exact linear threshold functions [1], and hence $\mathcal{O}(2^{\text{poly}(n)})$ linear decision lists and exact linear decision lists of polynomial size. Thus, there are not enough polynomial size linear decision lists, nor polynomial size exact linear decision lists, to compute all functions in Rect-DL. □

Another application of Proposition 27 gives a connection between LDL and ELDL.

Proposition 30. *Any* LDL *of polynomial size can be converted into an* ELDL *with polynomial size and of the same depth.*

Proof. Since LTF $\subseteq$ OR $\circ$ ELTF [15] we have that LTFs are computable by polynomial size $\text{ELTF}_{1,1}$ decision lists. Since LTF is closed under negation, the claim follows from Proposition 27. □

Next, we show that the notion of the depth of rectangular decision lists has a natural meaning in terms of communication complexity.

Proposition 31. *For a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, the following statements are equivalent:*

- *f is computable by a* Rect-DL_k *of size* $2^{\text{polylog}(n)}$.
- *f is computable by a* $\text{P}^{\text{NP}^{cc}}$ *protocol of size polylog(n) and depth* $\log k$.

Proof. We will start with a $\text{P}^{\text{NP}^{cc}}$ protocol with depth $\log k$, and show how to construct a Rect-DL of size $2^{polylog(n)}$ and depth k. Suppose f is computable by a $\text{P}^{\text{NP}^{cc}}$ protocol tree $\mathcal{T}$ of depth $\log k$. Consider the branch of $\mathcal{T}$ in which all queries return true. Let the leaf node ℓ of this branch output b. Since each query in the protocol tree is a union of rectangles, reaching node ℓ means that the conjunction of all rectangle queries along this branch must be true. Let $q_i = \bigvee_j^{2^{\text{polylog}(n)}} R_{ij}$ be the ith query in this branch. Then we have that ℓ is reached on (x, y) if and only if

$$(x,y) \in \bigwedge_{i=1}^{\log k} q_i = \bigwedge_{i=1}^{\log k} \bigvee_{j}^{2^{\text{polylog}(n)}} R_{ij}.$$

We can distribute $\bigwedge$ over $\bigvee$ to get an equivalent expression of the form OR of AND of rectangles. The size of the OR here is $\left(2^{\text{polylog}(n)}\right)^{\log k} = 2^{\text{polylog}(n)\log k}$. Note that an intersection of rectangles is a rectangle; thus, we actually have an expression of the form OR $\circ$ Rect. This is computable by an $\text{Rect-DL}_{1,b}$ of size $2^{\text{polylog}(n)\log k}$, which is $2^{\text{polylog}(n)}$ since the depth of $\text{P}^{\text{NP}^{cc}}$ communication protocol is upper bounded by its size.

Now we extend the constructed decision list to account for other branches of $\mathcal{T}$. For this, note that given that all queries of the decision list up to this point were false, we can simplify the tree $\mathcal{T}$ by removing the leaf ℓ. Indeed if all queries were false on some input, this input does not reach ℓ. Thus, in the parent of ℓ, we can assume we travel to the other branch of the tree without making a query. We remove this node of the tree and repeat the described procedure again.

On each step, we reduce the size of the tree by 1 and add at most one additional depth layer to our decision list. Thus, the resulting decision list Rect-DL has depth at most $2^{\log k} = k$ and size $2^{\text{polylog}(n)\log k}$.

For the other direction, suppose f is computable by a Rect-DL$_k$ called $\mathcal{L}$. Now, by Lemma 12, we can without loss of generality assume that $\mathcal{L}$ only outputs a value on true queries. Then we construct a tree in the following way: we binary search for the furthest layer reached in $\mathcal{L}$; call this layer v. To start, we perform an oracle query to check membership in the union of all rectangles up to the middle layer of $\mathcal{L}$. If the query outputs true, we proceed recursively into the first half of the list of $\mathcal{L}$, and otherwise, we proceed recursively into the second half of list. We can produce the output once we isolate one layer of the decision list. We need to do at most $\log k$ queries to binary search over k layers, resulting in a tree of depth $\log k$. The number of rectangles in the union on each query is at most $2^{\text{polylog}(n)}$, resulting in a protocol of size $\log k \cdot \text{polylog}(n)$, which is $\text{polylog}(n)$, since the depth of a decision list is upper bounded by its size. □

Finally, the next lemma shows that the ELDL hierarchy is separate from the hierarchies for LDL, $\widehat{\text{LDL}}$, and Rect-DL.

Lemma 32. *There exists a function f such that $f \in \text{ELDL}_{1,1}$, $f \notin \text{LDL}$, and $f \notin \text{Rect-DL}$.*

Proof. Let f be the *Block-Equality* function $\text{OR}_{n^2} \circ \text{EQ}_{n^2}(x, y) = 1$ iff $\exists i \in [n], \forall j \in [n], x_{ij} = y_{ij}$, where z_{ij} is the $(i \cdot \ell + j)$th bit of z. It is known that $\text{OR}_{n^2} \circ \text{EQ}_{n^2}$ is not in LDL nor in Rect-DL because its communication matrix has no large monochromatic rectangles [2, 17]. To see that it is in ELDL$_1$, note that for a given i, we can check whether $\forall j \in [n], x_{ij} = y_{ij}$ in a single ELTF query since EQ $\in$ ELTF. If it is true, we output 1; otherwise, we proceed to check $i+1$. Finally, if there is no such i, we output 0. Thus, we have a list of n ELTF queries, each outputting 1. This gives us an ELDL$_{1,1}$ computing $\text{OR}_{n^2} \circ \text{EQ}_{n^2}$, as desired. □

4 Bounds for LDL and Rect-DL

We start by analyzing the complexity of computing the XOR function by LDLs.

Lemma 33. *The function XOR_n can be computed by an $\widehat{\text{LDL}}$ of size n.*

Proof. First, we query $\sum_i x_i \geq n$; this identifies whether the input is the all-ones vector. The query is true, we output the value of XOR_n on the all-ones vector. Next, we query $\sum_i x_i \geq n-1$, which covers all inputs of Hamming weight $n-1$ and we output $n-1 \pmod 2$ if this query is true.

On the ith step, we query $\sum_i x_i \geq n-i+1$, and output $n-i+1 \pmod 2$. For all inputs of weight greater than $n-i+1$, one of the previous queries already produced the output. Hence, the current query will only produce an output for inputs of weight *exactly* $n-i+1$, and on this subset of inputs XOR_n is constant.

The final query is $\sum_i x_i \geq 1$. If this query is false, the input must be the all-zeros vector, in which case we output 0. □

Note that the depth of this decision list is thus also at most n. It can be reduced to roughly $n/2$ by considering vectors of large and small weights in parallel. That is, on ith iteration we can query both $\sum_i x_i \geq n-i+1$ and $\sum_i x_i \leq i-1$. If we account for the parity of n, we get an $\widehat{\text{LDL}}$ of depth $n/2$.

Next we show that any small size LDL computing XOR_n must have linear depth.

Theorem 34. *Any* LDL *of depth less than* $n/10$ *computing* XOR_n *has size* $2^{\Omega(n)}$.

Proof. Consider an LDL of depth k computing XOR_n.

We say that a query q is *generating* if it covers some input $a \in \{0,1\}^n$, such that none of the neighbors of a in the Hamming cube were covered by previous queries. We call a a *generating* input. First we observe that a generating query q can cover none of the other inputs except a. Indeed, suppose q is given by an inequality

$$w_0 + \sum_{i=1}^{n} w_i x_i \geq 0.$$

This inequality must hold for a. Since the output of XOR_n is different for each of the neighbors of a and since the neighbors were not covered by the previous queries, the inequality cannot hold for the neighbors of a. Thus, a hyperplane $w_0 + \sum_{i=1}^{n} w_i x_i = 0$ (in $\mathbb{R}^n$) corresponding to the query q intersects all edges of the Boolean cube adjacent to a. Due to the convexity of the Boolean cube, it isolates a in one of its halfspaces.

Denote the number of generating queries in our decision list by t. By the argument above, there are t generating inputs as well. We next show by induction on i that any input covered by queries on layer i must be at distance at most $i-1$ from some generating input. For the base case, note that none of the inputs are covered before layer 1, thus any query on layer 1 must be a generating query. For the induction step, consider an input a covered on layer i. It is either covered by a generating query, in which case it is a distance 0 from generating input, or it has a neighboring input b covered on some previous layer $j < i$. By the induction hypothesis, b is at distance at most $j-1$ from some generating input, and thus a is a distance at most $i-1$ from the same generating input.

Since the depth of the decision list is k, each covered input is a distance at most $k-1$ from some generating input. The number of inputs at distance at most $k-1$ from a given input is equal to $\sum_{j=0}^{k-1} \binom{n}{j}$, which we can upper bound by $\left(\frac{en}{k}\right)^k$ [18, Chapter 1].

Thus, the total number of inputs the queries of the LDL can cover is at most $t\left(\frac{en}{k}\right)^k$. At the same time, note that at least 2^{n-1} inputs must be covered, since otherwise XOR_n is not constant on the uncovered inputs. Thus, we have the following inequality:

$$t\left(\frac{en}{k}\right)^k \geq 2^{n-1}$$

or

$$t \geq 2^{n-k\log(\frac{en}{k})-1}.$$

If $k \leq n/10$ we get

$$t \geq 2^{n-k\log(\frac{en}{k})-1} = 2^{n-\frac{n}{10}\log(10e)-1} = 2^{\Omega(n)}.$$

Thus, any LDL of depth at most $n/10$ computing XOR_n must have exponentially many generating queries, and thus exponential size. □

Next we observe that XOR_n function is easy to compute by Rect-DL$_1$s and ELDL$_1$s.

Lemma 35. *The function* XOR_n *can be computed by* Rect-DL$_1$ *of size 2 and by* ELDL$_1$ *of size* $O(n)$.

Proof. For rectangle decision list, split the variables of XOR_n into two parts $y = (x_1, \ldots, x_{\frac{n}{2}})$ and $z = (x_{\frac{n}{2}+1}, \ldots, x_n)$. Note that $\mathrm{XOR}(y,z) = 1$ iff $\mathrm{XOR}(y) = 0$ and $\mathrm{XOR}(z) = 1$ or $\mathrm{XOR}(y) = 1$ and $\mathrm{XOR}(z) = 0$. For each $a, b \in \{0,1\}$, the pair of conditions $\mathrm{XOR}(y) = a$ and $\mathrm{XOR}(z) = b$ describe a combinatorial rectangle, and thus XOR is representable as an OR of two rectangles.

For ELDL, note that $\mathrm{XOR}_n(x) = 1$ iff there is an odd t such that $\sum_{i=1}^{n} x_i = t$. Each equality can be checked with one ELTF query and thus XOR_n can be represented as an OR of $n/2$ exact threshold functions. □

From Theorem 34 and Lemma 35 we get a strong separation between the LDL depth hierarchy and the Rect-DL and ELDL depth hierarchies: we exhibit a function that requires linear depth in LDL model, but is easily computable by a Rect-DL_1 and an ELDL_1.

Next we show a lower bound for GT function.

Theorem 36. *The size of any* Rect-DL_k *computing* GT *is* $\Omega(2^{n/k})$.

Proof of Theorem 36. Let $\mathcal{L}$ be a depth-d Rect-DL of size S computing GT. Suppose, without loss of generality, that the first depth layer is a 1-layer. Consider the (1-monochromatic) rectangles $R_1, \dots R_t$ in this first layer. Note that since $\mathcal{L}$ has size S, $t \le S$.

Consider all entries (a, b) in the communication matrix for GT covered by rectangles $R_1, \dots R_t$ and consider the following partial order on them: $(a, b) \le (a', b')$ iff $a \ge a'$ and $b \le b'$. Consider all maximal entries in $R_1, \dots R_t$ in this order. Clearly, each rectangle contains at most one maximal entry (see Figure 5).

Now consider the vertical gaps between successive maximal entries. That is, we sort maximal entries $(a_1, b_1), \dots, (a_t, b_t)$ by a-coordinate, $a_i < a_{i+1}$, add $(-1, -1)$ and $(2^n + 1, 2^n + 1)$ to the list. Note that $b_i < b_{i+1}$ for all i. For each two successive entries (a_i, b_i) and (a_{i+1}, b_{i+1}), consider $a_{i+1} - a_i$. Let A be the set of rows between a_{i+1} and a_i excluding a_{i+1} and a_i. By the Pigeonhole Principle, since there are at most S maximal entries, there exist two maximal entries (a_i, b_i) and (a_{i+1}, b_{i+1}) whose vertical gap is at least $\frac{2^n - S}{S+1} \ge \frac{2^n}{2S}$ (the inequality is true for $S \le O(2^n)$, and if $S \ge \Omega(2^n)$, we are done). Consider the submatrix $A \times A$. It is not hard to see that it does not intersect with any of the rectangles in the first depth layer. Indeed, for each 1-entry $(a, b) \in A \times A$ in this submatrix we have that $a < a_j$ for $j \ge i + 1$ and $b > a_i \ge b_i \ge b_j$ for $j \le i$. Thus, any such entry is either incomparable, or greater than any maximal entry, which is impossible.

We found a submatrix of the same form as the original GT that does not intersect with the first layer of the decision list. We repeat this argument on the new submatrix for each next depth layer, obtaining a submatrix of size $\frac{2^n}{(2S)^i}$ on the ith layer. If the size of the vertical submatrix is at least 2, the computation is not finished since we have a non-monochromatic submatrix that is not covered by any rectangles so far. This gives a bound of $\frac{2^n}{(2S)^k} \le 1$ for the last layer and the lower bound on the size follows. □

Propositions 28 and 36 imply the following:

Corollary 37. *Any* $\widehat{\mathrm{LDL}}_k$ *computing* GT *has size at least* $2^{\Omega(n/k)}$.

Since GT is an LTF, GT is computable by an LDL_1 of size 1. Thus, the class $\widehat{\mathrm{LDL}}_k$ is a proper subset of LDL_k and even LDL_1. It remains open whether the inclusion is still proper when depth is not restricted.

Next we proceed to the separation results within the depth hierarchies, for this we consider the function $\mathrm{NEQ}_n^{(k)}$.

We define a *cell* on the ith layer of $\mathrm{NEQ}_n^{(k)}$ to be a submatrix where the first i parts of the inputs are fixed to some values (see Figure 6). That is, $X_j = s_j$ and $Y_j = t_j$ for some fixed strings $s_j, t_j \in \{0,1\}^{\frac{n}{k}}$ for all $j \le i$. Some cells are monochromatic, more specifically, the cell becomes monochromatic once we fix a block in such a way that $s_i \ne t_i$. We call an ith layer of $\mathrm{NEQ}_n^{(k)}$ *0-dominant* if the corresponding monochromatic cells are of all zeroes (this is the case of odd i). *1-dominant* layers are defined similarly (they correspond to even i). The intuition for

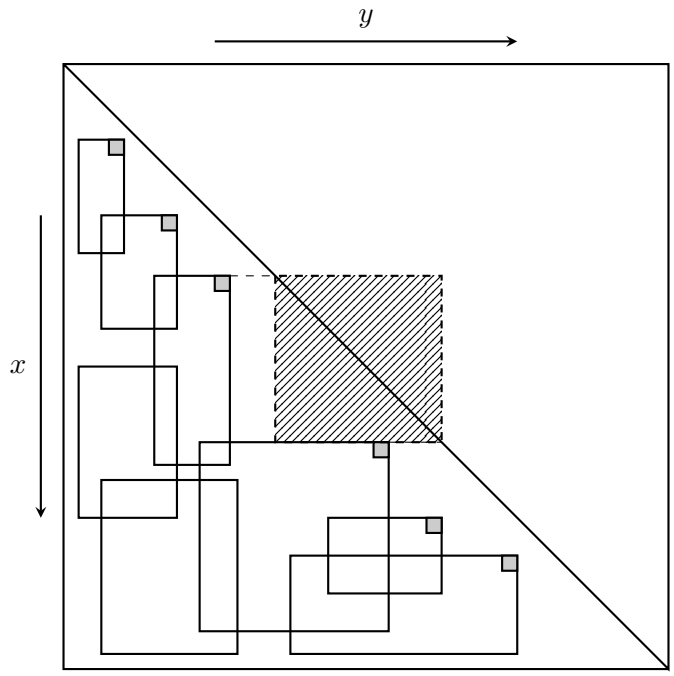

Figure 5: Step of the argument in Theorem 36: rectangles with maximal entries (note that some rectangles might not have any) and the new submatrix at the location of the maximal vertical gap.

Lemma 38, Theorem 40 and Theorem 42 is that $\mathrm{NEQ}^{(k)}$ can be computed more efficiently when *0-dominant* layers in the communication matrix "align" with 0-layers of queries in the decision list.

Lemma 38. $\mathrm{NEQ}_n^{(k)}$ *is computable by a* $\text{Rect-DL}_{k,0}$, $\widehat{\mathrm{LDL}}_{k,0}$, *and an* $\mathrm{LDL}_{k,0}$ *each of length* $2n$.

Proof. Let x_{ij} (resp. y_{ij}) with $1 \leq i \leq k$ and $1 \leq j \leq \frac{n}{k}$ denote the jth element of the string x_i (resp. y_i). We describe an $\text{Rect-DL}_{k,0}$ of length $2n$ computing $\mathrm{NEQ}^{(k)}$. We first query the rectangle defined by $x_{11} = 1$ and $y_{11} = 0$. For each element of this rectangle, clearly $x_1 \neq y_1$, so we output 0 for all elements in this rectangle. Next, we query the rectangle defined by $x_{11} = 0$ and $y_{11} = 1$, and similarly output 0 for each element in this rectangle. We proceed this way for each $j \in \frac{n}{k}$ and $b \in \{0, 1\}$, querying the rectangle $x_{1j} = b$ and $y_{1j} = 1 - b$, outputting 0 for each true query. We repeat this procedure for each $i \in [k]$, outputting 1 on each true query if i is even, and 0 otherwise. For a fixed layer i, we must do $2 \cdot \frac{n}{k}$ rectangle queries, and there are k layers, resulting in $2 \cdot \frac{n}{k} \cdot k = 2n$ queries total.

To see that it is computable by an $\widehat{\mathrm{LDL}}_{k,0}$ of the same length, note that a rectangle query defined by $(x_{ij} = 1) \wedge (y_{ij} = 0)$ can be expressed as a low-weight threshold query in the following way: $x_{ij} - y_{ij} \geq 1$. Since each of the rectangle queries in the rectangle decision list can be expressed as a low-weight threshold query, we have that $\mathrm{NEQ}^{(k)}$ is computable by an $\widehat{\mathrm{LDL}}_{k,0}$ of length $2n$. The final claim about $\mathrm{LDL}_{k,0}$ follows trivially. □

Next we prove the key lemma for the lower bound for $\mathrm{NEQ}_n^{(k)}$.

Lemma 39. *Fix a depth* k, *and suppose* $\mathrm{NEQ}_n^{(k)}$ *is computable by an* $\text{Rect-DL}_{k,1}$ $\mathcal{L}$ *of size* $< 2^{n/k}$. *Then for all* i, *there exists a non-monochromatic cell on layer* i *in* $\mathrm{NEQ}_n^{(k)}$ *such that none of its entries are covered by* $\mathcal{L}$*'s queries of depth at most* i.

Proof. We use induction on i. In the base case, where $i = 0$, the claim trivially holds as $\mathcal{L}$ restricted to queries of depth at most 0 is simply an empty Rect-DL.

$\mathrm{NEQ}_n^{(i+2)} =$

<table>
<tr><td>$\overline{\mathrm{NEQ}_n^{(i+1)}}$</td><td colspan="3">0</td><td>0</td></tr>
<tr><td rowspan="3">0</td><td>$\mathrm{NEQ}_n^{(i)}$</td><td>1</td><td>1</td><td rowspan="3">0</td></tr>
<tr><td>1</td><td>$\mathrm{NEQ}_n^{(i)}$</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>$\mathrm{NEQ}_n^{(i)}$</td></tr>
<tr><td>0</td><td colspan="3">0</td><td>$\overline{\mathrm{NEQ}_n^{(i+1)}}$</td></tr>
</table>

Figure 6: Cell structure of $\mathrm{NEQ}_n^{(k)}$ function: bigger cell $\mathrm{NEQ}_n^{(i+2)}$ consists of smaller cells $\overline{\mathrm{NEQ}_n^{(i+1)}}$ (zeroes off the diagonal indicate that this is the case of odd i), which in turn consist of smaller cells $\mathrm{NEQ}_n^{(i)}$ shown in the middle.

For the inductive step, we suppose the statement holds for $i-1$; we show that it is true for i as well. Without loss of generality, let the ith layer be a 0-dominant layer. Now consider the ith depth layer of $\mathcal{L}$. Since the size of $\mathcal{L}$ is $< 2^{n/k}$, there are at most $2^{n/k} - 1$ queries in this depth layer. Let q be any query in this depth layer. Since this a 0-dominant layer for the function, the output for q in the decision list is 1 (due to mismatch between layers of the function and the decision list).

Let C_{i-1} be the cell guaranteed by the induction hypothesis. Now let the cell $C_{s,i}$ be the set of inputs such that for all $j < i$, $x_j = y_j$ are fixed the same way as in C_{i-1} and $x_i = y_i = s$. We claim that q can intersect at most one of the $2^{\frac{n}{k}}$ $C_{s,i}$. To see this, suppose q covers both a 1-input $(X, Y) \in C_{s,i}$ and a 1-input $(X', Y') \in C_{s',i}$. This implies that $x_i = y_i = s$, and $x'_i = y'_i = s'$. But then if we consider the input (X, Y'), it is also covered by q. On the other hand, we have that $x_i = s \neq s' = y'_i$, and hence, by definition of $\mathrm{NEQ}^{(k)}$, (X, Y') is a 0-input. Input (X, Y') lies in C_{i-1} and by induction hypothesis was not covered by queries on previous depths. Thus, on query q the decision list outputs incorrect output on (X, Y'), which is a contradiction. As a result, q could not have covered both $(X, Y) \in C_{s,i}$ and $(X', Y') \in C_{s',i}$. Since this depth layer contains at most $2^{n/k} - 1$ queries and there are $2^{n/k}$ $C_{s,i}$s, one of these cells must be uncovered, as desired. □

Theorem 40. *The size of any* Rect-$\mathrm{DL}_{k,1}$ *computing* $\mathrm{NEQ}_n^{(k)}$ *is at least* $2^{n/k}$.

Proof. Suppose $\mathrm{NEQ}_n^{(k)}$ is computable by an Rect-$\mathrm{DL}_{k,1}$ $\mathcal{L}$ of size $< 2^{n/k}$. Then, by Lemma 39, there exists a cell C in layer k (this cell has just 1 entry) of the matrix $\mathrm{NEQ}_n^{(k)}$ which has not been covered by any of $\mathcal{L}$'s queries. The decision list gives an incorrect output on this cell (because of the mismatch of the outputs on layers). □

Next we provide the analogous results for LDLs.

Lemma 41. *Fix a depth k, and suppose $\mathrm{NEQ}_n^{(k)}$ is computable by an $\mathrm{LDL}_{k,1}$ $\mathcal{L}$ of size $< 2^{n/k}$. Then for all i, there exists a non-monochromatic cell on layer i in $\mathrm{NEQ}_n^{(k)}$ such that none of its entries have been covered by $\mathcal{L}$'s queries of depth at most i.*

Proof. We use induction on i. For the base case, we consider $i = 0$. For this case no parts of inputs are fixed and thus there is only one cell, the whole matrix. Clearly, none of its entries have been covered by $\mathcal{L}$'s queries of depth at most 0.

For the inductive step, we suppose the statement holds for $i-1$; we show that it is true for i as well. Without loss of generality, let the ith layer be a 0-dominant layer. Now consider the ith depth layer of $\mathcal{L}$. Since the size of $\mathcal{L}$ is $< 2^{n/k}$, there are at most $2^{n/k} - 1$ queries $L_1(x,y) \leq t_1, L_2(x,y) \leq t_2, \ldots L_v(x,y) \leq t_v$ in this depth layer, $v \leq 2^{n/k} - 1$. Since this a 0-dominant layer for the function, the leaf node for q in the decision list is 1 (due to mismatch between layers of the function and the decision list).

Let C_{i-1} be the cell guaranteed by the induction hypothesis. Let the cell $C_{s,i}$ be the set of inputs such that for all $j < i$, $x_j = y_j$ are fixed the same way as in C_{i-1} and $x_i = y_i = s$. We claim that as in the previous proof each query on this layer intersects at most one of the $C_{s,i}$ cells. Suppose, for the sake of contradiction, that a query $L_j(x,y) \leq t_j$ is true on inputs $(X,Y) \in C_{s,i}$ and $(X'Y') \in C_{s',i}$ where (X,Y) and (X',Y') are 1-inputs to $\mathrm{NEQ}_n^{(k)}$. This implies that $x_i = y_i = s$, and $x'_i = y'_i = s'$. But then if we consider the input (X,Y'), we have that $x_1 = s \neq s' = y'_1$, and hence, by definition of $\mathrm{NEQ}^{(i-1)}$, (X,Y') is a 0-input. Identical reasoning shows that (X',Y) is a 0-input. Since (X,Y) and (X',Y') are both 1-inputs, we have that $L_j(X,Y) \leq t_j$ and $L_j(X',Y') \leq t_j$. However, note that

$$L_j(X',Y) + L_j(X,Y') = L_j(X,Y) + L_j(X',Y') \leq 2t_j.$$

Thus, at least one of the following two inequalities must be true: $L_j(X',Y) \leq t_j$ or $L_j(X,Y') \leq t_j$. Then the decision tree outputs an incorrect output on this input, which is a contradiction. Hence, each query on this layer intersects at most one of the $C_{s,i}$. Since this depth layer contains at most $2^{n/k} - 1$ queries and there are $2^{n/k}$ cells $C_{s,i}$, one of these cells must be uncovered, as desired. □

Theorem 42. *The size of any $\mathrm{LDL}_{k,1}$ computing $\mathrm{NEQ}_n^{(k)}$ has length at least $2^{n/k}$.*

Proof. Suppose $\mathrm{NEQ}_n^{(k)}$ is computable by an $\mathrm{LDL}_{k,1}$ $\mathcal{L}$ of size $< 2^{n/k}$. Then, by Lemma 41, there exists a cell C on layer k (this cell has just 1 entry) in the matrix $\mathrm{NEQ}_n^{(k)}$ which has not been covered by any of $\mathcal{L}$'s queries. The decision list gives an incorrect output on this cell (because of the mismatch of the outputs on layers). □

Remark 43. *Note that similar arguments show that the negation $\neg\mathrm{NEQ}_n^{(k)} \in \widehat{\mathrm{LDL}}_{k,1}, \text{Rect-DL}_{k,1}$ and $\neg\mathrm{NEQ}_n^{(k)} \notin \widehat{\mathrm{LDL}}_{k,0}, \text{Rect-DL}_{k,0}$.*

5 ELDL lower bounds

We start with showing that large discrepancy over the uniform distribution implies a bound on depth of ELDLs.

In this argument it will be convenient to work with the notion of *advantage* of f on a subset of its inputs S:

$$\mathrm{Adv}(S,f) := \Big|\ |\{(x,y) \in S \mid f(x,y) = 1\}| - |\{(x,y) \in S | f(x,y) = 0\}|\ \Big|.$$

Advantage is closely related to discrepancy over the uniform distribution in the following way: for any combinatorial rectangle R, we have $\mathrm{Adv}(R, f) = 2^{2n}\mathsf{Disc}_U(R, f)$.

First we prove the following lemma.

Lemma 44. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function such that $\mathsf{Disc}_U(f) \le d$, where U is the uniform distribution. Then f has advantage of at most $2^{2n+1}\sqrt{d}$ on any blocky system of rectangles.*

Proof. Consider a rectangle $R = A \times B$ in M, and let $D_R \coloneqq \mathrm{Adv}(R, f)$. Note that

$$D_R = 2^{2n}\mathsf{Disc}_U(R, f) \le 2^{2n}d.$$

At the same time $D_R \le |A||B|$. Thus, at least one of the following is true for R: $|A| \ge \sqrt{D_R}$ or $|B| \ge \sqrt{D_R}$. Decompose M into two subsystems, M_{wide} and M_{tall}, where M_{wide} contains rectangle R if the former condition is true, and M_{tall} contains R if the latter condition is true. (In the case that both conditions are true, we arbitrarily select one of M_{wide} or M_{tall} for R to belong to.)

Now consider some rectangle $R : A \times B$ in M_{wide}, $\mathrm{Adv}(R) = D_R$. Since there are at least $\sqrt{D_R}$ rows in R, we have that the average advantage of a row in R is at most $\frac{\mathrm{Adv}(R)}{|A|} \le \frac{D_R}{\sqrt{D_R}} = \sqrt{D_R} \le 2^n\sqrt{d}$. Since this is true for each rectangle in M_{wide} and the rectangles are row-disjoint, we obtain that M_{wide} has average advantage of a row at most $2^n\sqrt{d}$ among all rows of the matrix. Since there are 2^n rows, we get $\mathrm{Adv}(M_{wide}) \le 2^n \cdot 2^n\sqrt{d} = 2^{2n}\sqrt{d}$. By a symmetric argument with columns, $\mathrm{Adv}(M_{tall}) \le 2^{2n}\sqrt{d}$, and hence the total $\mathrm{Adv}(M)$ is at most $2^{2n+1}\sqrt{d}$. □

Theorem 45. *Let $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and $\mathsf{Disc}_U(f) \le d$, where U is the uniform distribution on inputs. Assume that f is approximately balanced; specifically, $|f^{-1}(0)|, |f^{-1}(1)| \ge \Omega(2^{2n})$. Then the size of* ELDL *of depth k computing f is at least $\Omega(kd^{-1/2k})$.*

Proof. Consider an ELDL of size s and depth k computing f. Denote the number of queries on each layer by $s_1, \ldots, s_k$. We have $s_1 + \ldots + s_k = s$.

We prove the following statement by induction on i. Consider queries on layer i; assume that they output $b \in \{0,1\}$. We will prove that these queries together cover at most $2^{2n+i}\sqrt{d}\prod_{j=1}^{i} s_j$ entries in $f^{-1}(b)$.

For the base case, consider the first layer, and denote its output by b. By Lemma 15, a query on this layer defines a blocky matrix. Since $\mathsf{Disc}_U(f) \le d$, by Lemma 44 this blocky matrix has advantage at most $2^{2n+1}\sqrt{d}$. Since there are no previous layers, the query cannot cover any inputs (x, y) with $f(x, y) = \neg b$. Thus, the blocky matrix is monochromatic and covers at most $2^{2n+1}\sqrt{d}$ b-outputs. There are at most s_1 queries on the first layer, and together they cover at most $2^{2n+1}\sqrt{d}s_1$ b-outputs.

For the induction step, assume that the statement is true for all layers before the ith layer. Denote the output of this layer by b and consider a query on the ith layer. The corresponding blocky matrix does not have to be monochromatic since it can contain $\neg b$-entries covered on the previous layers. The number of $\neg b$-entries covered on the previous layers is upper bounded by

$$2^{2n+i-1}\sqrt{d}\prod_{j=1}^{i-1} s_j + 2^{2n+i-3}\sqrt{d}\prod_{j=1}^{i-3} s_j + 2^{2n+i-5}\sqrt{d}\prod_{j=1}^{i-5} s_j + \cdots \le$$

$$\left(2^{i-1} + 2^{i-3} + 2^{i-5} + \cdots\right) 2^{2n}\sqrt{d}\prod_{j=1}^{i-1} s_j \le (2^i - 2)2^{2n}\sqrt{d}\prod_{j=1}^{i-1} s_j$$

for all $i \geq 2$. By Lemma 44, the number of b-entries covered by this blocky matrix is at most $2^{2n+1}\sqrt{d}$ larger. Then we get the following upper bound on the number of b-entries covered by this blocky matrix:

$$2^{2n+i}\sqrt{d}\prod_{j=1}^{i-1} s_j.$$

Since there are at most s queries on the ith layer, together they cover at most $2^{2n+i}\sqrt{d}\prod_{j=1}^{i} s_j$ b-entries, as needed.

Now, assume that the last layer outputs b. Then the number of inputs on which the decision list outputs b is at most

$$2^{2n+k}\sqrt{d}\prod_{j=1}^{k} s_j + 2^{2n+k-2}\sqrt{d}\prod_{j=1}^{k-2} s_j + 2^{2n+k-4}\sqrt{d}\prod_{j=1}^{k-4} s_j + \cdots \leq 2^{2n+k+1}\sqrt{d}\prod_{j=1}^{k} s_j.$$

Since the function is approximately balanced, we have that

$$2^{2n+k+1}\sqrt{d}\prod_{j=1}^{k} s_j \geq \Omega(2^{2n})$$

using the AM-GM inequality and observing that $s_1 + \ldots + s_k = s$ gives the bound

$$s \geq \Omega(kd^{-1/2k}).$$

□

Since $\mathsf{Disc}_U(\mathrm{IP}) = 2^{-n/2}$ [20] we immediately get the following corollary.

Corollary 46. *The size of* ELDL *of depth k computing* IP *is at least $k2^{\Omega(n/k)}$.*

Remark 47. *Observe that since $2^{\Theta(n/k)} \geq \Theta(n/k)$, Corollary 46 gives a linear lower bound on the* size *of* ELDL *for* IP*. However, note that this bound on the size also follows from the standard argument based on the size of rectangles. For this, one can consider the first query and consider a square submatrix of the communication matrix of size roughly $2^n/2 \times 2^n/2$ such that the first query does not produce an output on this rectangle. Then one can argue by induction. This approach is very similar to the approach in [2].*

Next we proceed to showing that ELDLs of fixed depth form a hierarchy. The separating functions in this case are $\mathrm{INT}_n^{(k)}$.

We start with the upper bound.

Lemma 48. $\mathrm{INT}_n^{(k)}$ *is computable by an* $\mathrm{ELDL}_{k,0}$ *of length n.*

Proof. Let x_{ij} (resp. y_{ij}) with $1 \leq i \leq k$ and $1 \leq j \leq \frac{n}{k}$ denote the jth element of the string x_i (resp. y_i). We describe an $\mathrm{ELDL}_{k,0}$ of length n computing $\mathrm{INT}^{(k)}$. We first make a query defined by $x_{11} + y_{11} = 2$. For each element of this rectangle, clearly $x_1 \cap y_1 \neq \emptyset$, so by definition of $\mathrm{INT}_n^{(k)}$, we output 0 for all elements in this rectangle. We proceed this way for each $j \in \left[\frac{n}{k}\right]$, making a query defined by $x_{1j} + y_{1j} = 2$, outputting 0 for each true query. We repeat this procedure for each $i \in [k]$, outputting 1 on each true query if i is even, and 0 otherwise. For a fixed layer i, we must do $\frac{n}{k}$ rectangle queries, and there are k layers, resulting in $\frac{n}{k} \cdot k = n$ queries total. □

Remark 49. *Note that by replacing queries of the form $x_{ij} + y_{ij} = 2$ in Lemma 48 with $x_{ij}+y_{ij} \geq 2$, we have that $\mathrm{INT}_n^{(k)}$ is computable by an $\widehat{\mathrm{LDL}}_{k,0}$ of length n, and thus is computable by $\mathrm{LDL}_{k,0}$ and $\text{Rect-DL}_{k,0}$.*

The rest of the section is devoted to the proof of the following theorem.

Theorem 50. *Any $\mathrm{ELDL}_{k,1}$ computing $\mathrm{INT}_n^{(k)}$ has size $2^{\Omega(n/k)-O(k)}$.*

Note that this lower bound is exponential for $k = O(\sqrt{n})$.

For the proof of this theorem, it is convenient to change the notation slightly and talk about functions with $2nk$ input variables. That is, we will consider functions $\mathrm{INT}_{nk}^{(k)}$ and the lower bound we will prove is $2^{\Omega(n)-O(k)}$.

Note that the cells of $\mathrm{INT}_n^{(k)}$ have essentially the structure of DISJ_n function. It is helpful for this argument to restrict DISJ_n to a submatrix with the same number of ones in each row and column. To do this, we restrict the input to DISJ_n to subsets of size exactly $\frac{n}{3}$. In other words, the inputs to DISJ_n are now restricted to (X, Y) such that $|X| = |Y| = \frac{n}{3}$. The number of n-bit strings satisfying this condition is

$$\binom{n}{\frac{n}{3}} = 2^{H(1/3)n(1+o(1))}.$$

We will denote this number by N. Note also that for a fixed row (column) in the restricted submatrix, the number of ones in it is

$$\binom{\frac{2n}{3}}{\frac{n}{3}} = 2^{\frac{2n}{3}(1+o(1))}.$$

We will denote this number by D. The size of the largest 1-monochromatic rectangle is still at most 2^n (see 21).

We propagate this restriction to $\mathrm{INT}_{nk}^{(k)}$. In other words, we view the input to $\mathrm{INT}_{nk}^{(k)}$ as $((X_1, Y_1), \ldots, (X_k, Y_k))$, where each $X_i, Y_i \in \{0,1\}^n$ have $|X_i| = |Y_i| = \frac{n}{3}$. In particular, the number of rows and columns in $\mathrm{INT}_{nk}^{(k)}$ is N^k. Hereafter, in this section, when we refer to $\mathrm{INT}_{nk}^{(k)}$ or DISJ_n, we actually mean these restricted versions.

We start by establishing some useful properties of the DISJ_n function.

Lemma 51. *If we arbitrarily remove at most $N/10$ rows and at most $N/10$ columns from the communication matrix of DISJ_n, the number of ones in the resulting matrix is at least $4ND/5$.*

Moreover, we can further remove additional rows and columns (of our choice) to get a matrix with at least $N/2$ rows and at least $N/2$ columns such that any row and any column contains at least $3D/8$ ones.

Proof. Consider a communication matrix M of DISJ_n, and suppose that we remove rows with labels in A and columns with labels in B, where A and B are of size at most $N/10$. Each row and column contains D ones, thus rows in A contain $ND/10$ ones and the same is true for columns in B. After removal of these rows and columns the remaining number of ones is at least $ND - 2\frac{ND}{10} = \frac{8ND}{10}$.

To prove the second part of the lemma we repeat the following process. Check if there is a row or a column with the number of ones below $3D/8$. If such a row or column exists, remove it. If in this way we reduce the number of rows or columns below $N/2$, then we removed at most $8N/10$ rows and columns, that in total contain less than $\frac{8N}{10} \cdot \frac{3D}{8} = \frac{3ND}{10}$ ones. Thus, the resulting matrix contains more than $\frac{8ND}{10} - \frac{3ND}{10} = \frac{ND}{2}$. This is a contradiction, since this matrix has less than half rows or columns of the original one and each row and column of the original matrix contains only D ones. □

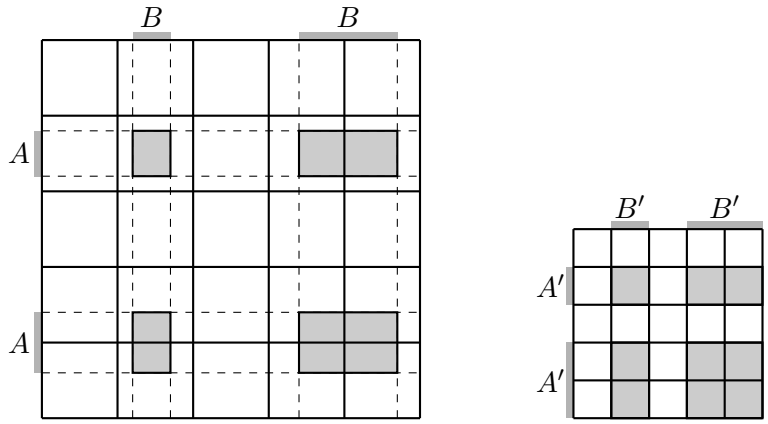

Figure 7: The notion of projection: a rectangle over the iterated matrix translates to a rectangle over the cells in DISJ matrix.

Analogously to the case of $\mathrm{NEQ}_n^{(k)}$, we define a *cell* on the ith layer of $\mathrm{INT}_{nk}^{(k)}$ to be a submatrix where the first i parts of the inputs are fixed to some values. That is, $X_j = s_j$ and $Y_j = t_j$ for some fixed strings $s_j, t_j \in \{0,1\}^n$ for all $j \leq i$. Some cells are monochromatic, more specifically the cell is monochromatic if one of its blocks is fixed in a way that $X_i \cap Y_i \neq \emptyset$. We call a layer of $\mathrm{INT}_{nk}^{(k)}$ *0-dominant* if the corresponding monochromatic cells are 0-monochromatic. *1-dominant* layers are defined similarly. In particular, the first layer is 0-dominant for INT_{nk}^{k}. We say that a row or column of a cell is *covered* by a rectangle R if it has non-empty intersection with R. When it is clear from context, we will refer to a column as covered if one of the rectangles we are currently considering intersects it.

Next we state the key lemma needed for the proof of Theorem 50.

Lemma 52. *Fix a depth k, and suppose $\mathrm{INT}_{nk}^{(k)}$ is computable by an $\mathrm{ELDL}_{k,1}$ $\mathcal{E}$ of size S. Define $\varepsilon_0 = 0$ and $\varepsilon_i = 11^i \cdot \frac{8}{3} \cdot \frac{2^{n/2}S}{D}$ for all $1 \leq i \leq k$. Then for all i, there exists a non-monochromatic cell on layer i in $\mathrm{INT}_{nk}^{(k)}$ such that $\leq \varepsilon_i$ fraction of its rows and columns have been covered by $\mathcal{E}$'s queries of depth at most i.*

The proof of Theorem 50 based on the Lemma 52 is analogous to the proofs from the previous section.

Proof of Theorem 50. Suppose $\mathrm{INT}_n^{(k)}$ is computable by an $\mathrm{ELDL}_{k,1}$ $\mathcal{E}$ of size S. Then, by Lemma 52, there exists a cell on layer k such that at most $\varepsilon_k = 11^k \cdot \frac{8}{3} \cdot \frac{2^{n/2}S}{D}$ fraction of its rows and columns have been covered by $\mathcal{E}$'s queries of depth at most k. Note that the cells on kth layer are 1×1 matrices and if $\varepsilon_k < 1$, then the only entry of this matrix is not covered, and $\mathcal{E}$ outputs an incorrect value on it (due to the mismatch between the outputs on the layers). From that, we get that the decision list is incorrect if

$$11^k \cdot \frac{8}{3} \cdot \frac{2^{n/2}S}{D} < 1$$

and given that $D = 2^{\frac{2n}{3}(1+o(1))}$, the theorem follows. □

Now it is only left to prove Lemma 52.

Proof of Lemma 52. We use induction on i. For the base case we consider $i = 0$. For this case, no parts of inputs are fixed and thus there is only one cell, the whole matrix. Clearly, $\varepsilon_0 = 0$ fraction of its rows and columns have been covered by $\mathcal{E}$'s queries of depth at most 0.

For the induction step, suppose the statement holds for the $(i-1)$st layer of $\mathrm{INT}^{(k)}_{nk}$ and consider the corresponding cell $M^{(i-1)}$ on the $(i-1)$st layer. We show that the statement is true for the ith layer as well. The cell we are going to find is a submatrix of the cell on the $(i-1)$st layer. Without loss of generality, let the ith layer be a 0-dominant layer. We view the matrix $M^{(i-1)}$ as a block-matrix, in which blocks correspond to cells of the ith layer and each block is either a constant 0 matrix, or $M^{(i)}$ matrix corresponding to the non-monochromatic cell on the next layer. That is, consider the part of input corresponding on the ith layer, (X_i, Y_i) and consider communication matrix M_{DISJ} of DISJ on these inputs. This matrix reflects the structure of the cells of layer i in $M^{(i-1)}$: each cell is labeled by specific values of (X_i, Y_i) and the cell is 0-monochromatic iff $\mathrm{DISJ}(X_i, Y_i) = 0$.

By the induction hypothesis, at most ε_{i-1} fraction of rows and columns of $M^{(i-1)}$ is covered by previous queries. First we remove block-rows and block-columns that has more than $10\varepsilon_{i-1}$ of their rows and columns respectively covered. By Markov's inequality, this way we remove at most $1/10$ fraction of all block-rows and block-columns.

As we observed, the block-matrix corresponds to Disjointness. By Lemma 51 we can further remove constant fraction of block-rows and block-columns in such a way that the number of rows and columns is still at least $N/2$ and each of the remaining block-rows and block-columns contains at least $3D/8$ non-monochromatic cells. For notation convenience denote the resulting submatrix by $M^{(i-1)}$ again.

Now consider queries of the ith depth layer of $\mathcal{E}$. Since the size of $\mathcal{E}$ is S, there are at most S queries in this layer.

Since $\mathcal{E}$ is an ELDL, each of its queries can be expressed as a blocky system of rectangles. Consider some query on the ith layer, and let M denote the corresponding blocky system (within $M^{(i-1)}$). Let $R_1, \dots R_t$ with $R_j = A_j \times B_j$ be rectangles in M. Note that since we removed all rows and columns partially covered by the queries of the previous layer, these rectangles must be 1-monochromatic. We want to show that the R_js cannot cover too much of the matrix $M^{(i-1)}$. For each rectangle R_j consider its *projection* $R'_j = A'_j \times B'_j$ to inputs (X_i, Y_i) (see Figure 7). That is, if R_j contains an entry with some specific values of (X_i, Y_i), then these values of (X_i, Y_i) are in R'_j. Basically, R'_j represent the rectangle of cells in M_{DISJ} that are intersecting with R_j. Note, that since R_js are 1-monochromatic, they cannot intersect 0-cells, and thus R'_j are 1-monochromatic rectangles in M_{DISJ}.

Note that we can decompose M into two disjoint matrices M_{wide} and M_{tall} based on the dimensions of the rectangles R'_j. In particular, the rectangle $R_j = A_j \times B_j$ is in M_{wide} if $|A'_j| < 2^{\frac{n}{2}}$ and in M_{tall} if $|B'_j| < 2^{\frac{n}{2}}$. Intuitively, M_{wide} contains all rectangles that are short and M_{tall} contains all rectangles that are narrow. Note, that by Claim 21 each rectangle is either short, or narrow.

We will argue that M_{wide} has small intersection with a random non-monochromatic cell in $M^{(i-1)}$. A symmetric argument shows that it is also true for M_{tall}, and hence for M.

Fix some column of cells c_l in $M^{(i-1)}$. We make use of the following claim:

Claim 53. *We have* $|B_1 \cap c_l| + \dots + |B_t \cap c_l| \le N^{k-i}$.

Indeed, the claim holds since M_{wide} is blocky and therefore distinct rectangles do not overlap in columns.

Pick a uniformly random multicolored cell C in c_l. Let X be a random variable for the number of C's columns which have a non-empty intersection with one of the rectangles $R_j \in M_{wide}$. We have the following upper bound on the expected number of covered columns in a non-monochromatic cell C:

$$\mathbb{E}_{C\sim c_l}[X] \leq \sum_{j=1}^{t} |B_j \cap c_l| \cdot \frac{A'_j}{\frac{3D}{8}} \leq \sum_{j=1}^{t} |B_j \cap c_l| \cdot \frac{2^{\frac{n}{2}}}{\frac{3D}{8}} \leq \frac{N^{k-i}2^{\frac{n}{2}}}{\frac{3D}{8}}. \quad (1)$$

Let X^r be the random variable denoting the fraction of columns covered by M_{wide} in a uniformly random non-monochromatic cell C in the matrix $M^{(i-1)}$. Similarly, let X^c be the random variable denoting the fraction of rows covered by M_{tall} in a uniformly random non-monochromatic cell in the matrix.

Since (1) holds for each column of cells c_l, we have $\mathbb{E}_{C\sim M^{(i-1)}}[X^r] \leq \frac{8}{3} \cdot \frac{2^{n/2}}{D}$ and the same is true for X^c. We have this bound for each query on layer i and since there are at most S queries in this layer the total expected fraction of rows and columns covered for a random cell is bounded by $\frac{8}{3} \cdot \frac{2^{n/2}S}{D}$. By Markov's inequality there is a cell on the ith layer that has both at most $10 \cdot \frac{8}{3} \cdot \frac{2^{n/2}S}{D}$ fraction of rows and at most $10 \cdot \frac{8}{3} \cdot \frac{2^{n/2}S}{D}$ fraction of columns covered. Together with rows and columns covered on the previous layers we have that for this cell at most

$$10\varepsilon_{i-1} + 10 \cdot \frac{8}{3} \cdot \frac{2^{n/2}S}{D} \leq 10\varepsilon_{i-1} + \varepsilon_{i-1} \leq \varepsilon_i$$

rows and columns covered, as needed. □

References

[1] László Babai, Kristoffer Arnsfelt Hansen, Vladimir V. Podolskii, and Xiaoming Sun. Weights of exact threshold functions. In Petr Hlinený and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 66–77. Springer, 2010. doi:10.1007/978-3-642-15155-2_8.

[2] Arkadev Chattopadhyay, Meena Mahajan, Nikhil S. Mande, and Nitin Saurabh. Lower bounds for linear decision lists. *Chic. J. Theor. Comput. Sci.*, 2020, 2020. URL: http://cjtcs.cs.uchicago.edu/articles/2020/1/contents.html.

[3] Arkadev Chattopadhyay and Nikhil Mande. A short list of equalities induces large sign rank. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 47–58, 2018. doi:10.1109/FOCS.2018.00014.

[4] Lijie Chen and R. Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via pcps of proximity. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 19:1–19:43. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/LIPIcs.CCC.2019.19, doi:10.4230/LIPICS.CCC.2019.19.

[5] Yogesh Dahiya, Vignesh K., Meena Mahajan, and Karteek Sreenivasaiah. Linear threshold functions in decision lists, decision trees, and depth-2 circuits. *Information Processing Letters*, 183:106418, 2024. URL: https://www.sciencedirect.com/science/article/pii/S0020019023000613, doi:10.1016/j.ipl.2023.106418.

[6] Mason DiCicco, Vladimir Podolskii, and Daniel Reichman. Nearest neighbor complexity and boolean circuits. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 42:1–42:23. Schloss Dagstuhl - Leibniz-Zentrum für

Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.ITCS.2025.42, doi:10.4230/LIPICS.ITCS.2025.42.

[7] Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002.

[8] Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *FSTTCS*, pages 171–182, 2001. URL: https://doi.org/10.1007/3-540-45294-X_15.

[9] Mikael Goldmann. *Communication Complexity and Lower Bounds for Threshold Circuits*, pages 85–125. Springer US, Boston, MA, 1994. doi:10.1007/978-1-4615-2696-4_3.

[10] Mikael Goldmann, Johan Håstad, and Alexander Razborov. Majority gates vs. general weighted threshold gates. *Comput. Complex.*, 2(4):277–300, December 1992.

[11] Mika Göös, Pritish Kamath, Toniann Pitassi, and Thomas Watson. Query-to-Communication Lifting for P^{NP}. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CCC.2017.12, doi:10.4230/LIPIcs.CCC.2017.12.

[12] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Comput. Complex.*, 27(2):245–304, June 2018.

[13] András Hajnal, Wolfgang Maass, Pavel Pudlak, Mario Szegedy, and Gyorgy Turan. Threshold circuits of bounded depth. *J. Comput. Syst. Sci.*, 46:129–154, 04 1993. doi:10.1109/SFCS.1987.59.

[14] Lianna Hambardzumyan, Hamed Hatami, and Pooya Hatami. Dimension-free bounds and structural results in communication complexity. *Isr. J. Math.*, 253(2):555–616, March 2023.

[15] Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 270–279, 2010. doi:10.1109/CCC.2010.33.

[16] Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Polynomial threshold functions and boolean threshold circuits. *Information and Computation*, 240:56–73, 2015. MFCS 2013. URL: https://www.sciencedirect.com/science/article/pii/S0890540114001175, doi:10.1016/j.ic.2014.09.008.

[17] Russell Impagliazzo and Ryan Williams. Communication complexity with synchronized clocks. In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 259–269, 2010. doi:10.1109/CCC.2010.32.

[18] Stasys Jukna. *Extremal Combinatorics - With Applications in Computer Science*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2011. doi:10.1007/978-3-642-17364-6.

[19] Stasys Jukna et al. *Boolean function complexity: advances and frontiers*, volume 27. Springer, 2012.

[20] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

[21] P.M. Lewis and C.L. Coates. *Threshold Logic*. Wiley, 1967.

[22] J Myhill and W H Kautz. On the size of weights required for linear-input switching functions. *IEEE Trans. Electron. Comput.*, EC-10(2):288–290, June 1961.

[23] Periklis Papakonstantinou, Dominik Scheder, and Hao Song. Overlays and limited memory communication. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 298–308, 2014. doi:10.1109/CCC.2014.37.

[24] Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *Journal of Computer and System Sciences*, 33(1):106–123, 1986. URL: https://www.sciencedirect.com/science/article/pii/0022000086900462, doi:10.1016/0022-0000(86)90046-2.

[25] Anup Rao and Amir Yehudayoff. *Communication complexity: and applications*. Cambridge University Press, 2020.

[26] Alexander A Razborov. On small depth threshold circuits. In *Scandinavian Workshop on Algorithm Theory*, pages 42–52. Springer, 1992.

[27] Alexander A Razborov and Alexander A Sherstov. The sign-rank of AC^0. *SIAM Journal on Computing*, 39(5):1833–1855, 2010.

[28] Ronald L Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, November 1987.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il