Linear Branching Programs and Directional Affine Extractors

Svyatoslav Gryaznov
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Pavel Pudlák
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Navid Talebanfard
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

A natural model of *read-once linear branching programs* is a branching program where queries are $\mathbb{F}_2$ linear forms, and along each path, the queries are linearly independent. We consider two restrictions of this model, which we call *weakly* and *strongly* read-once, both generalizing standard read-once branching programs and parity decision trees. Our main results are as follows.

- **Average-case complexity.** We define a pseudo-random class of functions which we call *directional affine extractors*, and show that these functions are hard on average for the strongly read-once model. We then present an explicit construction of such function with good parameters. This strengthens the result of Cohen and Shinkar (ITCS'16) who gave such average-case hardness for parity decision trees. Directional affine extractors are stronger than the more familiar class of affine extractors. Given the significance of these functions, we expect that our new class of functions might be of independent interest.
- **Proof complexity.** We also consider the proof system $\mathsf{Res}[\oplus]$, which is an extension of resolution with linear queries, and define the regular variant of $\mathsf{Res}[\oplus]$. A refutation of a CNF in this proof system naturally defines a linear branching program solving the corresponding search problem. If a refutation is regular, we prove that the resulting program is read-once. Conversely, we show that a weakly read-once linear BP solving the search problem can be converted to a regular $\mathsf{Res}[\oplus]$ refutation with constant blow up, where the regularity condition comes from the definition of weakly read-once BPs, thus obtaining the equivalence between these proof systems.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Computational complexity and cryptography

Keywords and phrases Boolean Functions, Average-Case Lower Bounds, AC0[2], Affine Dispersers, Affine Extractors

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.4

Related Version *Full Version*: https://arxiv.org/abs/2201.10997

Funding *Svyatoslav Gryaznov*: Supported by GAČR grant 19-05497S.
Pavel Pudlák: Supported by GAČR grant 19-27871X.
Navid Talebanfard: Supported by GAČR grant 19-27871X.

Acknowledgements We wish to thank Swastik Kopparty for useful correspondence regarding affine extractors.

1 Introduction

Circuit complexity and proof complexity are two major lines of inquiry in complexity theory (see [13, 15] for extensive introductions). The former theme attempts to identify explicit Boolean functions which are not computable by small circuits from a certain restricted class, and the latter aims to find tautologies which are not provable by short proofs in a given restricted proof system. These seemingly unrelated topics are bound together in at least two different ways: via feasible interpolation where a circuit lower bound for a concrete computational problem implies proof size lower bounds (see, e.g., [11]), and more fundamentally many proof systems have an underlying circuit class where proof lines come from. Notable examples are Frege, bounded depth Frege, and extended Frege systems where proof lines are De Morgan formulas, AC^0 circuits, and general Boolean circuits, respectively. Intuitively we expect that understanding a circuit class in terms of lower bounds and techniques should yield results in the proof complexity counterpart. This intuition has been supported by bounded depth Frege lower bounds using specialized Switching Lemmas (see, e.g., [10]), the essential ingredient of AC^0 lower bounds.

$\mathsf{AC}^0[2]$ circuits and $\mathsf{Res}[\oplus]$ proof system. It is not clear if this intuition should always hold. Lower bounds for $\mathsf{AC}^0[p]$ circuits (AC^0 circuits with Mod_p gates) have been known for a long time [19, 23] yet lower bounds for bounded depth Frege systems with modular gates still elude us. Perhaps this failure is not too surprising since our understanding of $\mathsf{AC}^0[p]$ circuits is not of the same status as our understanding of AC^0. For example, even for $\mathsf{AC}^0[2]$, that is AC^0 with parity gates, no strong average-case lower bound is known. Settling such bounds is an important challenge, since Shaltiel and Viola [22] showed that for standard worst-case to average-case hardness amplification techniques to work, the circuit class is required to compute the majority function, which is not the case for $\mathsf{AC}^0[2]$. Several works have highlighted the special case of $\mathsf{AC}^0 \circ \mathsf{Mod}_2$, where the parity gates are next to the input [21, 2, 8]. Among these works we pay special attention to the result of Cohen and Shinkar [8] who considered the depth-3 case of this problem and proved a strong average-case hardness for the special case of parity decision trees. The more general case of $\mathsf{DNF} \circ \mathsf{Mod}_2$ remains open.

In the proof complexity parallel, a special case of $\mathsf{AC}^0[2]$-Frege was suggested by Itsykson and Sokolov [12]. They considered the system $\mathsf{Res}[\oplus]$ that is an extension of resolution which reasons about disjunctions of linear equations over $\mathbb{F}_2$, which we call linear clauses. The rules of this system are:

- *the weakening rule*: from a linear clause we can derive any other linear clause which is semantically implied,
- *the resolution rule*: for every two linear clauses C and D and linear form f, we can derive $C \vee D$ from $(f = 0) \vee C$ and $(f = 1) \vee D$.

They proved exponential lower bounds for the tree-like restriction of this system. These lower bounds were later extended in [9, 18]. For DAG-like proofs, the only known results are due to Khaniki [14] who proved almost quadratic lower bounds, and to Lauria [16] for a restriction of the system when parities are on a bounded number of variables. Super-polynomial lower bounds for unrestricted DAG-like $\mathsf{Res}[\oplus]$ are widely open.

Parity decision trees and tree-like $\mathsf{Res}[\oplus]$. Given an unsatisfiable CNF $F = C_1 \wedge \ldots \wedge C_m$, the *search problem for F* is the computational problem of finding a clause C_i falsified by a given assignment to the variables. A tree-like $\mathsf{Res}[\oplus]$ refutation of F can be viewed as a parity decision tree solving the search problem for an unsatisfiable CNF [9]. Recall that the strongest average-case lower bounds for $\mathsf{AC}^0[2]$ are in fact for parity decision trees. Thus it seems that parity decision trees are at the frontier of our understanding in these two areas. Therefore a natural approach to make progress towards both general $\mathsf{Res}[\oplus]$ lower bounds and average-case hardness for $\mathsf{AC}^0[2]$ is to consider DAG-like structures more general than decision trees.

1.1 Our contributions

Motivated by strengthening tree-like Res[$\oplus$] lower bounds as well as average-case lower bounds for parity decision trees to more general models, we consider a model of read-once branching programs (BPs) with linear queries. The most natural way to interpret the property of being read-once in BPs with linear queries, is to impose that along every path, the queries are linearly independent. We consider two restrictions of this model which we call weakly read-once and strongly read-once, both of which extend parity decision trees and standard read-once branching programs.

For strongly read-once BPs, we prove average-case hardness for a new class of pseudo-random functions, and we give an explicit construction of such a function, thus strengthening the result of Cohen and Shinkar [8] and making progress towards average-case hardness for $\mathsf{DNF} \circ \mathsf{Mod}_2$. Our pseudo-random functions are defined below and might be of independent interest.

Directional affine extractors. The average-case hardness result of Cohen and Shinkar [8] is for affine extractors. An affine extractor for dimension d and bias ϵ is a function such that restricted to any affine subspace of dimension at least d it has bias at most ϵ. Explicit constructions for such functions are known (e.g., [5, 24, 4]). For our purposes it is not clear if affine extractors are sufficient. Therefore we consider a more robust concept. We say that a function $f\colon \{0,1\}^n \to \{0,1\}^n$ is a *directional affine extractor* for dimension d with bias ϵ, if for every non-zero $a \in \{0,1\}^n$, the derivative of f in the direction a, $D_a f(x) = f(x+a)+f(x)$, is an affine extractor for dimension d with bias ϵ. We give an explicit construction of a good directional affine extractor for dimension larger than $2n/3$.

For weakly read-once BPs we show a correspondence with Res[$\oplus$]. More precisely, we show that a weakly read-once BP solving the search problem for a CNF F, can be converted to a Res[$\oplus$] refutation of F while preserving the proof structure. This also justifies defining a Res[$\oplus$] counterpart to regular resolution similarly to weakly read-once BPs. Recall that in a regular resolution proof, no variable is resolved more than once along any path. It is well-known that a read-once BP solving the search problem for an unsatisfiable CNF can be converted to a regular resolution refutation of the formula. Our result should be interpreted as an extension of this result to Res[$\oplus$].

1.2 Read-once linear branching programs

The model of read-once branching programs is a natural and extensively studied model of computation for which strong lower bounds are known [20, 3]. Here we consider an extension of this model where queries are linear forms. A linear branching program[1] $\mathcal{P}$ in the variables x is a DAG with the following properties:

- it has exactly one source;
- it has two sinks labeled with 0 and 1 representing the values of the function that $\mathcal{P}$ computes;
- every inner node is labeled by a linear form q over $\mathbb{F}_2$ in x which we call *queries*;
- every inner node with a label q has two outgoing edges labeled with 0 and 1 representing the value of q.

[1] This term has already been used before in [1] with a different meaning in the context of quantum computation.

Any assignment to the input variables naturally defines a path in the program. We say that $\mathcal{P}$ computes a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ if for every $x \in \{0,1\}^n$, the path in $\mathcal{P}$ defined by x ends in the sink labeled with $f(x)$.

We now define read-once linear BPs. Given an inner node v of a linear branching program $\mathcal{P}$, we define $\mathrm{Pre}(v)$ as the span of all queries that appear on any path from the source of $\mathcal{P}$ to v, excluding the query at v. We define $\mathrm{Post}(v)$ as the span of all queries in the subprogram starting at v.

▶ **Definition 1** (Weakly and strongly read-once linear branching programs). *We say that a linear branching program $\mathcal{P}$ is* weakly read-once *if for every inner node v of $\mathcal{P}$ which queries q, it holds that $q \notin \mathrm{Pre}(v)$.*

We say that a linear branching program $\mathcal{P}$ is strongly read-once *if for every inner node v of $\mathcal{P}$, it holds that $\mathrm{Pre}(v) \cap \mathrm{Post}(v) = \{0\}$.*

It follows from both definitions that queries alongside any path in weakly or strongly read-once BP are linearly independent. Furthermore, both of these models generalize standard read-once BPs and parity decision trees. When the distinction between weakly and strongly read-once is not important, we simply write "read-once".

1.3 Regular Res[$\oplus$]

We also define a regular variant of Res[$\oplus$] using similar conditions that we require from weakly read-once BPs.

▶ **Definition 2** (Regular Res[$\oplus$]). *A Res[$\oplus$] refutation is* (weakly) regular *if for every clause C obtained by the application of the resolution rule on a literal q, the span of all literals appearing in any application of the resolution rule to C or a clause derived from C, does not contain q.*

Intuitively, the read-once (or regular) nature of this definition can be described as following: after we resolve on q, we restrict the query space U by its complimentary to q, i.e., the space W with $q \notin W$ and $\mathrm{span}(q) + W = U$. If a clause C was derived using q, there exists a linear isomorphism L that maps q to a variable y, and for every application of the resolution rule to C of a clause derived from C on a linear literal q', Lq' does not contain y.

In Section 6 we establish the connection between weakly read-once BPs and regular Res[$\oplus$] refutations.

2 Notation and basic facts

Each path in a read-once program defines an affine subspace given by the set of solutions of the system corresponding to the queries on the path. Any affine subspace can be represented by a vector space shifted by a vector from the affine space. For our purposes, we need to choose this shift carefully.

Let p be a path in a read-once linear BP $\mathcal{P}$ leading to a node v with queries $q_1, \ldots, q_k$ and answers $a_1, \ldots, a_k$ to these queries which define the affine subspace $S_p = \{x : \bigwedge_{i=1}^k q_i(x) = a_i\}$. Let V_p be the supporting vector space of S_p, i.e., $V_p = \{x : \bigwedge_{i=1}^k q_i(x) = 0\}$. Then clearly $S_p = V_p + b$ for any $b \in S_p$. Choose an arbitrary basis $q'_1, \ldots, q'_t$ for $\mathrm{Post}(v)$. Since $q'_1, \ldots, q'_t$ are independent of $q_1, \ldots, q_k$, there exists b such that $\bigwedge_{i=1}^k q_i(b) = a_i$ and $\bigwedge_{i=1}^t q'_i(b) = 0$. Then $S_p = V_p + b$ and for every $q \in \mathrm{Post}(v)$, we have $q(b) = 0$.

▶ **Definition 3** (Canonical affine subspace). *Given a path p which ends at a node v, we call S_p the* canonical affine subspace *for p. Furthermore a* canonical representation *of S_p is any $V_p + b = S_p$ where every $q \in \mathrm{Post}(v)$ vanishes on b.*

Throughout the paper we drop the word *representation* and simply say $V_p + b$ is the canonical affine subspace of p to mean that it is a canonical representation of S_p.

Since we will often use canonical affine subspaces to represent paths in BPs, we adopt the following algebraic notation. Let us denote the space of all linear forms on $\mathbb{F}_2^n$ (the dual space) as $(\mathbb{F}_2^n)^*$. Given a subspace V of $\mathbb{F}_2^n$, we define $V^\perp$ as the space of all linear forms from $(\mathbb{F}_2^n)^*$ that vanish on V (this space is sometimes called the annihilator of V), i.e.,

$$V^\perp = \{\ell \in (\mathbb{F}_2^n)^* : \forall v \in V,\ \ell(v) = 0\}.$$

Given a path p with queries $q_1, \ldots, q_k$ and its canonical affine subspace $V + b$, the space $V^\perp$ is the query space of p, i.e., $V^\perp = \mathrm{span}(q_1, \ldots, q_k)$.

It is clear that every read-once BP is a strongly read-once linear BP. We also show that every parity decision trees can be transformed into an equivalent tree that is also a strongly read-once linear BP, without increasing its size. We need the following notation. Let V and W be two subspaces. Then the sum of V and W is the subspace

$$V + W := \{v + w : v \in V,\ w \in W\}.$$

Note that $V + W = \mathrm{span}(V \cup W)$.

▶ **Lemma 4.** *Let T be a parity decision tree. Then there exists a parity decision tree T' computing the same function, which is a strongly read-once linear BP.*

Proof. Without loss of generality, we may assume that for every node v of T, the query at v is linearly independent of the queries leading to v, since otherwise we can replace v with one of its children. To prove this lemma, we will use the fact that exactly one path passes through any node of the tree. We construct T' inductively on the depth of a tree node starting from the root. Let v be a node of T labeled with the query q. Let $B = \{\beta_1, \ldots, \beta_t\}$ be a basis of $\mathrm{Pre}(v) + \mathrm{span}(q)$ and B' its extension to a basis of the whole query space $(\mathbb{F}_2^n)^*$. We can rewrite every query in the subtree rooted at v in the new basis $B \cup B'$. Since the program is a tree, every query from B has the unique value. Thus, we can safely substitute them in every query in the v-subtree, possible changing the labels of the outgoing edges. After this transformation, every query in $\mathrm{Post}(v)$, except for the query at v, will be expressed in terms of B'. By construction, $B \cup B'$ is a basis and $\mathrm{Pre}(v)$ and $\mathrm{Post}(v)$ are expressed using different basis vectors, which implies $\mathrm{Pre}(v) \cap \mathrm{Post}(v) = \{0\}$. ◀

Throughout the paper we adopt the following notation.

- Given a vector $c \in \{0,1\}^n$ the *support of c* is defined as

  $$\mathrm{supp}(c) := \{i : c_i \neq 0\}.$$

- Let σ be a partial assignment to the variables $x_1, \ldots, x_n$. Then

  $$\mathrm{dom}(\sigma) := \{i : \sigma(x_i) \text{ is defined}\}.$$

- We say that $a \in \{0,1\}^n$ is *consistent* with a partial assignment σ to $x_1, \ldots, x_n$ if for every $i \in \mathrm{dom}(\sigma)$, it holds that $\sigma(x_i) = a_i$.
- We write $a + b$ without specifying the underlying field, if it is clear from the context and often intended to be $\mathbb{F}_2$.

2.1 The trace map

The trace map $\mathrm{Tr}\colon \mathbb{F}_{p^n} \to \mathbb{F}_p$ is defined as

$$\mathrm{Tr}(x) := \sum_{i=0}^{n-1} x^{p^i}.$$

One important property that we need is that Tr is an $\mathbb{F}_p$-linear map. We also use the following fact about the trace.

▶ **Proposition 5** (cf. [17]). *For every $\mathbb{F}_p$-linear map $\pi\colon \mathbb{F}_{p^n} \to \mathbb{F}_p$ there exists $\mu \in \mathbb{F}_{p^n}$ such that for all $x \in \mathbb{F}_{p^n}$ we have*

$$\pi(x) = \mathrm{Tr}(\mu \cdot x).$$

Furthermore, π is trivial if and only if $\mu = 0$.

Since, we are interested in Boolean functions, we will only consider the case $p = 2$. Let $\phi\colon \mathbb{F}_2^n \to \mathbb{F}_{2^n}$ be any $\mathbb{F}_2$-linear isomorphism. Then $\mathrm{Tr}(\mu \cdot \phi(x))$ is a linear Boolean function of x and we have the following:

▶ **Proposition 6.** *The set of all linear Boolean functions coincides with the set of functions $\ell_\mu(x) = \mathrm{Tr}(\mu \cdot \phi(x))$, where $\mu \in \mathbb{F}_{2^n}$.*

In the rest of the paper we fix ϕ. To make the proofs more readable we use bold font to denote the corresponding elements of $\mathbb{F}_{2^n}$, e.g., $\mathbf{x}$ for $\phi(x)$.

2.2 Affine extractors and dispersers

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is an *affine disperser* for dimension d if f is not constant on any affine subspace of dimension d. Let us also recall affine extractors, which are generalizations of affine dispersers.

The *bias* of f is defined as

$$\mathrm{bias}(f) := \left| \mathop{\mathbb{E}}_{x \in U_n} [(-1)^{f(x)}] \right|,$$

where U_n is a uniform distribution on $\{0,1\}^n$. Given an affine subspace f, *the bias of f restricted to $S \subseteq \{0,1\}^n$* is defined as

$$\mathrm{bias}(f|_S) := \left| \mathop{\mathbb{E}}_{x \in U(S)} [(-1)^{f(x)}] \right|,$$

where $U(S)$ is a uniform distribution on S.

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is an *affine extractor* for dimension d with bias ϵ if for every affine subspace S of dimension d, the bias of f restricted to S, $\mathrm{bias}(f|_S)$, is at most ϵ.

3 Affine mixedness

In this section we give a criterion for functions to be worst-case hard for read-once linear BPs. Let us first recall *mixedness* from standard read-once BPs.

▶ **Definition 7.** *A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is d-*mixed[2] *if for every $I \subseteq [n]$ of size at most $n-d$ and every two distinct partial assignments σ and τ with $\mathrm{dom}(\sigma) = \mathrm{dom}(\tau) = I$, it holds that $f|_\sigma \neq f|_\tau$.*

▶ **Theorem 8** (Folklore; see [13] for a proof)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a d-mixed Boolean function. Then any read-once branching program computing f has size at least $2^{n-d} - 1$.*

Explicit constructions of d-mixed functions with $d = o(n)$ and thus $2^{n-o(n)}$ size lower bounds for read-once BPs were given in [20, 3]. We generalize this notion for linear branching programs. We need the following equivalent definition of d-mixedness.

▶ **Lemma 9.** *A Boolean function f is d-mixed if and only if for every partial assignments σ of size at most $n-d$ and every $c \neq 0$ with $\mathrm{supp}(c) \subseteq \mathrm{dom}(\sigma)$, there exists x consistent with σ such that $f(x) \neq f(x+c)$.*

Proof. ($\Leftarrow$) Let σ and τ be two distinct partial assignments with domain I of size at most $n-d$. Define $c_i = \tau(x_i) + \sigma(x_i)$ for $i \in I$ and $c_i = 0$ otherwise. By assumption there exists x consistent with σ such that $f(x) \neq f(x+c)$. It follows from the definition of c that $x+c$ is consistent with τ. Define $J = [n] \setminus I$ and $z = x_J = (x+c)_J$. Then $f|_\sigma(z) = f(x) \neq f(x+c) = f|_\tau(z)$.

($\Rightarrow$) Let σ be a partial assignment with a domain of size at most $n-d$ and let c be given such that $\mathrm{supp}(c) \subseteq \mathrm{dom}(\sigma)$. Define $\tau(x_i) = \sigma(x_i) + c_i$ for $i \in \mathrm{dom}(\sigma)$. By assumption $f|_\sigma \neq f|_\tau$, hence there exists z such that $f|_\sigma(z) \neq f|_\tau(z)$. Define x to take the same value as σ on $\mathrm{dom}(\sigma)$ and equal to z otherwise. Then $f(x) = f|_\sigma(z) \neq f|_\tau(z) = f(x+c)$. ◀

▶ **Definition 10.** *A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is d-*affine mixed *if for every affine subspace S of dimension at least d and every vector $c \notin V$, where V is the supporting vector space of S, there exists $x \in S$ such that $f(x) \neq f(x+c)$.*

It follows from Lemma 9 that d-affine mixedness implies d-mixedness since a partial assignment is a special case of an affine subspace.

Now we are ready to prove a generalization of Theorem 8.

▶ **Theorem 11.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a d-affine mixed Boolean function. Then any strongly read-once linear branching program computing f has size at least $2^{n-d} - 1$.*

Proof. We prove that any such program $\mathcal{P}$ computing f starts with a complete binary tree of depth $n-d-1$. Assume for the sake of contradiction that there are two paths p and q of length at most $n-d-1$, which meet for the first time at a node v. Let $V+a$ and $W+b$ be their corresponding canonical affine subspaces. Both of them have dimension at least $d+1$.

We start by proving $V^\perp = W^\perp$ which implies $V = W$. Suppose that it is not the case. Without loss of generality, there exists $\ell \in W^\perp \setminus V^\perp$. By the read-once property $\ell \notin \mathrm{Post}(v)$.

Consider two affine subspaces $V' + a_1$ and $V' + a_2$ obtained by intersecting $V + a$ with $\ell(x) = 0$ and $\ell(x) = 1$ such that for every $\ell' \in \mathrm{Post}(v)$, $\ell'(a_1) = 0$ and $\ell'(a_2) = 0$ (recall that we can choose such a_1 and a_2 since $\mathrm{Pre}(v) \cap \mathrm{Post}(v) = \{0\}$). By construction, they have dimension at least d. Since f is d-affine mixed, there exists $z \in V'$ such that $f(z + a_1) \neq f(z + a_2)$. Consider any query ℓ' in the subprogram starting at v. The fact that

[2] This definition is commonly given for sets of size d instead of $n-d$. We deviate from this since for our generalization to affine spaces, it corresponds to dimension which is more natural.

$\ell' \in \mathrm{Post}(v)$ implies $\ell'(a_1) = \ell'(a_2) = 0$. Thus, we have $\ell'(z + a_1) = \ell'(z) = \ell'(z + a_2)$. It implies that in the subprogram starting at v both $z + a_1$ and $z + a_2$ must follow the same path contradicting $f(z + a_1) \neq f(z + a_2)$.

Now, since $V = W$, $V + b$ is the canonical affine subspace for q, and $a \neq b$ since p and q are different paths. Again, since f is d-affine mixed, there exists $z \in V$ such that $f(z + a) \neq f(z + b)$. Analogously to the previous case, for every $\ell' \in \mathrm{Post}(v)$ we have $\ell'(a) = \ell'(b) = 0$, and thus $\ell'(z + a) = \ell'(z + b)$ contradicting $f(z + a) \neq f(z + b)$. ◀

4 Affine dispersers for directional derivatives

In this section we give an explicit construction of an affine mixed function for linear dimension. In fact, we give an even more powerful construction, which allows us to get an average-case lower bound for strongly read-once linear branching programs.

For a Boolean function f its directional derivative with respect to a non-zero vector a is defined as

$$D_a f(x) \coloneqq f(x + a) + f(x).$$

▶ **Definition 12.** *A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is a* directional affine extractor *for dimension d with bias ϵ if for every non-zero a, the derivative $D_a f$ is an affine extractor for dimension d with bias ϵ.*

Similarly, f is a directional affine disperser *for dimension d if for every non-zero a, $D_a f$ is an affine disperser for dimension d.*

Observe that this notion is stronger than the one defined in the previous section: if f is a directional affine disperser for dimension d, then it is d-affine mixed.

In what follows we construct a Boolean function f in n variables that is a good directional affine extractor for dimensions bigger than $\frac{2}{3}n$.

It is a well-known fact that the inner product function is an affine extractor. IP is a member of the class of bent functions, which are all affine extractors. A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is called a *bent function* if all Fourier coefficients of its ± 1 representation $f_{\pm}(x) \coloneqq (-1)^{f(x)}$ have the same absolute value.

▶ **Lemma 13** (Folklore; for a proof see, e.g., [8, 7])**.** *Let f be a bent function on n variables and $c \geq 1$ be an integer. Then, f is an affine extractor for dimension $k = n/2 + c$ with bias at most 2^{-c}. In particular, f is an affine disperser for dimension $n/2 + 1$.*

We apply this result to prove that the following function is an affine extractor.

▶ **Lemma 14.** *Let $a_0, a_1, a_2, a_3 \in \mathbb{F}_{2^k}$ with $a_0 \neq 0$. Let $g\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ be the function defined as*

$$g(x, y) = \mathrm{Tr}(a_0 \cdot \phi(x) \cdot \phi(y) + a_1 \cdot \phi(x) + a_2 \cdot \phi(y) + a_3).$$

Then g is an affine extractor for dimension $k + c$ with bias at most 2^{-c}. In particular, g is an affine disperser for dimension $k + 1$.

Intuitively, the functions $\mathrm{Tr}(x \cdot y)$ and $\mathrm{IP}(x, y)$ behave similarly: the Fourier transform of functions with domain $\mathbb{F}_{2^n}$ is sometimes defined in terms of the trace map. Here we prove the statement directly using the usual definition of the Fourier transform for Boolean functions.

Proof. Let $g_\pm$ be the ± 1 representation of g. By Lemma 13, it is enough to prove that all Fourier coefficients of $g_\pm$ have the same absolute value. Recall that given $\alpha \in \{0,1\}^{2k}$ the α-character χ_α is defined as $\chi_\alpha(x,y) = (-1)^{\alpha\cdot(x,y)}$, where $\alpha \cdot (x,y)$ is the inner product. Fourier coefficient $\widehat{g_\pm}(\alpha)$ can be computed as follows.

$$\widehat{g_\pm}(\alpha) = \sum_{x,y\in\{0,1\}^k} g_\pm(x,y)\cdot\chi_\alpha(x,y) = \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot\mathbf{x}\cdot\mathbf{y}+a_1\cdot\mathbf{x}+a_2\cdot\mathbf{y}+a_3)}\cdot\chi_\alpha(x,y).$$

Split α into two equal parts: $\alpha = (\alpha_1,\alpha_2)$. Then $\alpha\cdot(x,y) = \alpha_1\cdot x+\alpha_2\cdot y$. By Proposition 6, there exist $\mu_1,\mu_2 \in \mathbb{F}_{2^k}$ such that $\alpha_1\cdot x = \mathrm{Tr}(\mu_1\cdot\mathbf{x})$ and $\alpha_2\cdot y = \mathrm{Tr}(\mu_2\cdot\mathbf{y})$. Also define

$$\begin{aligned}
b_1 &:= a_0^{-1}\cdot(a_1+\mu_1),\\
b_2 &:= a_0^{-1}\cdot(a_2+\mu_2),\\
b_3 &:= a_3 + a_0^{-1}\cdot(a_1+\mu_1)\cdot(a_2+\mu_2).
\end{aligned}$$

Then we can express $\widehat{g_\pm}(\alpha)$ in terms of b_i:

$$\begin{aligned}
\widehat{g_\pm}(\alpha) &= \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot\mathbf{x}\cdot\mathbf{y}+a_1\cdot\mathbf{x}+a_2\cdot\mathbf{y}+a_3)}\cdot(-1)^{\mathrm{Tr}(\mu_1\cdot\mathbf{x})+\mathrm{Tr}(\mu_2\cdot\mathbf{y})}\\
&= \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot\mathbf{x}\cdot\mathbf{y}+a_1\cdot\mathbf{x}+a_2\cdot\mathbf{y}+a_3+\mu_1\cdot\mathbf{x}+\mu_2\cdot\mathbf{y})}\\
&= \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot(\mathbf{x}+b_2)\cdot(\mathbf{y}+b_1)+b_3)}\\
&= (-1)^{\mathrm{Tr}(b_3)}\cdot\sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot(\mathbf{x}+b_2)\cdot(\mathbf{y}+b_1))}.
\end{aligned}$$

Since x and y iterate through all vectors from $\{0,1\}^k$, $a_0\cdot(\mathbf{x}+b_2)$ and $\mathbf{y}+b_1$ take all possible values from $\mathbb{F}_{2^k}$. It follows that

$$\widehat{g_\pm}(\alpha) = (-1)^{\mathrm{Tr}(b_3)}\widehat{g_\pm}(0).$$ ◀

We are now ready to present our directional affine extractor.

▶ **Theorem 15.** *Let $f\colon \{0,1\}^k\times\{0,1\}^k\times\{0,1\}^k \to \{0,1\}$ be the function defined by*

$$f(x,y,z) = \mathrm{Tr}(\phi(x)\cdot\phi(y)\cdot\phi(z)).$$

Then f is a directional affine extractor for dimension $2k+c$ with bias $\epsilon \le 2^{-c}$. In particular, f is a directional affine disperser for dimension $2k+1$.

Proof. Consider the directional derivative of f in the non-zero direction $a = (a_1,a_2,a_3)$:

$$\begin{aligned}
D_af(x,y,z) &= f(x+a_1,y+a_2,z+a_3)+f(x,y,z)\\
&= \mathrm{Tr}(\phi(x+a_1)\cdot\phi(y+a_2)\cdot\phi(z+a_3)) + \mathrm{Tr}(\mathbf{x}\cdot\mathbf{y}\cdot\mathbf{z}).
\end{aligned}$$

By linearity of Tr and ϕ we have

$$\begin{aligned}
D_af(x,y,z) &= \mathrm{Tr}((\mathbf{x}+\mathbf{a_1})\cdot(\mathbf{y}+\mathbf{a_2})\cdot(\mathbf{z}+\mathbf{a_3})+\mathbf{x}\cdot\mathbf{y}\cdot\mathbf{z})\\
&= \mathrm{Tr}(\mathbf{a_1}\cdot\mathbf{y}\cdot\mathbf{z}+\mathbf{a_2}\cdot\mathbf{x}\cdot\mathbf{z}+\mathbf{a_3}\cdot\mathbf{x}\cdot\mathbf{y}+\ell(\mathbf{x},\mathbf{y},\mathbf{z})),
\end{aligned}$$

where ℓ is an affine function.

Without loss of generality we may assume that $a_3 \neq 0$. Let S be an affine subspace with dimension at least $2k + c$. We need to show that the bias of f restricted to S is at most ϵ. Given $z_0 \in \{0,1\}^k$ define $S_{z_0} := \{(x, y) : (x, y, z_0) \in S\}$. For every z_0 the affine subspace S_{z_0} is either empty or has dimension at least $k + c$. Consider the restriction of $D_a f$ to $z = z_0$.

$$h_{z_0}(x, y) := D_a f(x, y, z_0) = \mathrm{Tr}(\mathbf{a_3} \cdot \mathbf{x} \cdot \mathbf{y} + \ell'_{z_0}(\mathbf{x}, \mathbf{y})),$$

where ℓ'_{z_0} is an affine function. By Lemma 14, h_{z_0} is an affine extractor for dimension $k + c$ with bias $\epsilon \leq 2^{-c}$. In particular, if S_{z_0} is non-empty, then $\mathrm{bias}(h_{z_0}|_{S_{z_0}}) \leq \epsilon$.

Thus, the bias of $D_a f$ restricted to S can easily be bounded as follows:

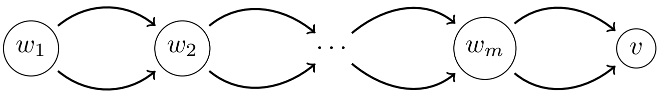
$$\begin{aligned}
\mathrm{bias}(D_a f|_S) &= \left| \frac{1}{|S|} \sum_{(x,y,z) \in S} (-1)^{D_a f(x,y,z)} \right| \\
&= \left| \frac{1}{|S|} \sum_{z_0 \in \{0,1\}^n} \sum_{(x,y,z_0) \in S} (-1)^{D_a f(x,y,z_0)} \right| \\
&= \left| \frac{1}{|S|} \sum_{z_0 \in \{0,1\}^n} \sum_{(x,y) \in S_{z_0}} (-1)^{h_{z_0}(x,y)} \right| \\
&\leq \frac{1}{|S|} \sum_{z_0 \in \{0,1\}^n} \left| \sum_{(x,y) \in S_{z_0}} (-1)^{h_{z_0}(x,y)} \right| \\
&\leq \frac{1}{|S|} \sum_{z_0 \in \{0,1\}^n} \epsilon \cdot |S_{z_0}| = \epsilon.
\end{aligned}$$

◀

5 Average-case lower bound

We consider a canonical form of strongly read-once linear branching programs. We adopt the terminology of [6] and say that a read-once linear branching program is *full* if for every inner node v of the program, all the paths leading to v have the same query space.

A *multipath* $(w_1, \ldots, w_m, v)$ is a linear branching program of the form



That is, the program ignores the answers to the queries at w_i for every i. Given a program $\mathcal{P}$, we say that a subset of nodes is an *antichain* if none of its nodes is a descendant of another. For example, the set of nodes at a fixed depth and the set of leaves form an antichain. The following lemma and its proof are easy extensions of Lemma 3.7 in [6].

▶ **Lemma 16.** *Every weakly read-once or strongly read-once linear branching program $\mathcal{P}$ of size s in n variables has an equivalent full weakly read-once or strongly read-once linear branching program $\mathcal{P}'$, respectively, of size at most $3n \cdot s$. Furthermore, the size of every antichain in $\mathcal{P}'$ is at most $2s$.*

Proof. We construct $\mathcal{P}'$ inductively. Consider the nodes of $\mathcal{P}$ in topological order. It is clear that the start node satisfies the fullness property. Let v be a node of $\mathcal{P}$ and $p_1, \ldots, p_k$ the paths that meet at v, and $V_1 + a_1, \ldots, V_k + a_k$ their canonical affine subspaces. For every $i \in [k]$ choose a set of linearly independent queries Q_i such that $V_i^{\perp} + \mathrm{span}(Q_i) = \mathrm{Pre}(v)$.

For every $i \in [k]$ do the following. Let $Q_i = \{q_1, \dots, q_m\}$. Replace the edge $u_i \to v$ with a multipath $(w_1, \dots, w_m, v)$ and an edge $u_i \to w_1$, where w_i are labeled with q_i. After this transformation, every path to v will have the query space $\mathrm{Pre}(v)$.

Since a branching program of size s has at most $2s$ edges and we replaced every edge with a multipath of length at most n, the size of the constructed full read-once linear branching program $\mathcal{P}'$ is at most $s + 2s \cdot n \leq 3n \cdot s$.

Consider an antichain A in $\mathcal{P}'$. We map every node in A to nodes in $\mathcal{P}$. Each node in A is either originally in $\mathcal{P}$ or it was created by a multipath. In the former case we map it to itself, and in the latter case we map it to the parent node from which it was created. Since the out-degree in $\mathcal{P}$ is 2 and A is an antichain, at most 2 nodes are mapped to the same node. This proves the result. ◀

Denote by $\mathrm{dist}(f, g)$ the relative distance between Boolean function f and g.

▶ **Theorem 17.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a directional affine extractor for dimension d with bias $\epsilon < \frac{1}{2}$. Then for every $g\colon \{0,1\}^n \to \{0,1\}$ computed by a strongly read-once linear branching program $\mathcal{P}$ of size at most $\epsilon \cdot 2^{n-d-1}$, it holds that $\mathrm{dist}(f, g) \geq \frac{1-\sqrt{2\epsilon}}{2}$.*

Proof. Let s denote the size of $\mathcal{P}$. We first convert $\mathcal{P}$ into a full program. By Lemma 16, the size of every antichain is at most $2s$. We then construct an equivalent program $\mathcal{P}'$ in which every path has length at least $n - d$. We can achieve this by extending every leaf of low depth by a multipath of an appropriate length.

Consider the set A of nodes in $\mathcal{P}'$ at depth exactly $n - d$. Note that every $v \in A$ is either a node at depth $n - d$ in $\mathcal{P}$, or it is uniquely defined by a leaf of $\mathcal{P}$ by a multipath. Thus A is identified by an antichain in $\mathcal{P}$ and thus $|A| \leq 2s$.

We call an input x *wrong* if $f(x) \neq g(x)$. The distance $\mathrm{dist}(f, g)$ between f and g is the fraction of wrong inputs.

▷ Claim 18. Let $v \in A$ and k the numbers of paths that meet at v. Then the number of wrong inputs that pass through v is at least

$$\frac{k \cdot 2^d}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right).$$

Proof. Since the program is full, the corresponding canonical affine subspaces for the paths that meet at v are $V + a_1, \dots, V + a_k$, for some d-dimensional vector space V, and distinct $a_1, \dots, a_k \in \{0,1\}^n$. Recall that f is a directional affine extractor with bias ϵ. Then for every $i \neq j$, it holds that $D_{a_i + a_j} f = f(x + (a_i + a_j)) + f(x)$ is an affine extractor with bias ϵ, thus

$$\begin{aligned} \left|\sum_{x \in V} (-1)^{f(x+a_i)} \cdot (-1)^{f(x+a_j)}\right| &= \left|\sum_{x \in V + a_j} (-1)^{f(x+a_i+a_j)+f(x)}\right| \\ &= \mathrm{bias}\Big(D_{a_i+a_j} f\big|_{V+a_j}\Big) \cdot |V| \leq \epsilon |V|. \end{aligned} \tag{1}$$

Every $x \in V$ produces a partition of $[k]$ into two parts $(J, [k] \setminus J)$ such that $f(x + a_i) = 0$ for $i \in J$ and $f(x + a_i) = 1$ for $i \notin J$. Let m_x be the size of the *smallest* part. By definition of canonical affine subspace and the choice of a_i, for any linear query $q \in \mathrm{Post}(v)$ we have $q(a_i) = 0$ for all $i \in [k]$. Then $x + a_1, \dots, x + a_k$ will follow the same path in the subprogram starting at v. Hence, for every $x \in V$ it holds that $f(x+a_1) = \cdots = f(x+a_k)$. It implies that at least m_x inputs of the form $x + a_i$ are wrong and the total number of wrong inputs passing through v is at least

$$m \coloneqq \sum_{x \in V} m_x.$$

Now consider the following sum

$$E \coloneqq \sum_{\substack{x \in V \\ 1 \le i < j \le k}} |f(x+a_i) - f(x+a_j)|.$$

We apply double counting to this quantity to obtain the result. On the one hand, by definitions of m_x and m, we have

$$E = \sum_{x \in V} m_x \cdot (k - m_x) = km - \sum_{x \in V} m_x^2.$$

By the Cauchy–Schwarz inequality, $\sum_{x \in V} m_x^2 \ge \left(\sum_{x \in V} m_x\right)^2 / |V| = m^2/|V|$. Thus,

$$E \le km - m^2/|V|. \tag{2}$$

On the other hand, E can be rewritten as follows.

$$\begin{aligned} E &= \sum_{\substack{x \in V \\ 1 \le i < j \le k}} \frac{1}{4}\left((-1)^{f(x+a_i)} - (-1)^{f(x+a_j)}\right)^2 \\ &= \frac{1}{4} \sum_{1 \le i < j \le k} \left(2|V| - 2\sum_{x \in V} (-1)^{f(x+a_i)} \cdot (-1)^{f(x+a_j)}\right). \end{aligned}$$

Applying (1), we obtain the following lower bound on E.

$$E \ge \frac{1}{2}\binom{k}{2}|V|(1-\epsilon). \tag{3}$$

Combining (2) and (3), we get

$$km - m^2/|V| \ge \frac{1}{2}\binom{k}{2}|V|(1-\epsilon).$$

This can be written as

$$\begin{aligned} \left(m - \frac{k|V|}{2}\right)^2 &\le \frac{1}{4}k^2|V|^2 - \frac{1-\epsilon}{2}\binom{k}{2}|V|^2 \\ &= \frac{k^2|V|^2}{4}\left(1 - (1-\epsilon)\left(1 - \frac{1}{k}\right)\right) \\ &\le \frac{k^2|V|^2}{4}\left(\epsilon + \frac{1}{k}\right). \end{aligned}$$

Thus,

$$m \ge \frac{k|V|}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right) = \frac{k \cdot 2^d}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right). \qquad \triangleleft$$

Let $k(v)$ denote the number of paths that meet at v and define $w(k)$ as

$$w(k) := \frac{k2^d}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right).$$

Then by Claim 18 the total number of bad inputs that pass through A is at least

$$\sum_{v\in A} w(k(v)) = \sum_{v\in A} \frac{k(v)2^d}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k(v)}}\right).$$

Since all paths in $\mathcal{P}'$ has length at least $n-d$, $\sum_{v\in A} k(v) = 2^{n-d}$.

The function w is convex, hence by Jensen's inequality, the total number of bad inputs passing through A is at least

$$\sum_{v\in A} w(k(v)) \geq |A| \cdot w\left(\frac{\sum_{v\in A} k(v)}{|A|}\right) = \frac{1}{2}2^n\left(1 - \sqrt{\epsilon + \frac{|A|}{2^{n-d}}}\right).$$

Since $|A| \leq 2s \leq \epsilon 2^{n-d}$, this expression is at least $\frac{1-\sqrt{2\epsilon}}{2}2^n$. ◀

Plugging in the function of Theorem 15 we get the following corollary.

▶ **Corollary 19.** *Let $f : \{0,1\}^{\frac{n}{3}} \times \{0,1\}^{\frac{n}{3}} \times \{0,1\}^{\frac{n}{3}} \to \{0,1\}$ be defined by $f(x,y,z) = \mathrm{Tr}(\phi(x)\cdot\phi(y)\cdot\phi(z))$. Then for every $g : \{0,1\}^n \to \{0,1\}$ computed by a strongly read-once linear BP of size at most $2^{\frac{n}{3}-o(n)}$, $\mathrm{dist}(f,g) \geq \frac{1}{2} - 2^{-o(n)}$.*

6 Weakly read-once BPs and Res[⊕]

In this section we prove an analogue of the correspondence between read-once BPs and regular resolution for Res[⊕] and weakly read-once BPs. The first part of the proof is an extension of a standard argument; the second part, while also easy, is more subtle.

▶ **Theorem 20.**

1. *Every Res[⊕] refutation of an unsatisfiable CNF F can be translated into a linear BP solving the corresponding search problem without increasing its size. If the refutation is regular, then the resulting program is weakly read-once.*
2. *Every weakly read-once BP of size s solving the search problem for CNF $F = C_1 \wedge \ldots \wedge C_m$ in n variables can be translated into a regular Res[⊕] refutation of F of size $O(ns)$.*

Proof.

1. Consider an application of the resolution rule in the proof DAG G. Suppose that it is applied to clauses $C_0 \vee (f = 0)$ and $C_1 \vee (f = 1)$. Then we label the outgoing edges with $f = 1$ and $f = 0$ respectively. We leave the edges corresponding to the weakening rule unlabeled.

Let u be a vertex in G and C_u the clause it is labeled with. It can be shown by induction on the depth of u that for every path to u, the linear system obtained from the equations written on the edges on this path implies $\neg C_u$. The source contains the empty clause, hence the base case holds. For the inductive step, consider any path leading to u and let v be the parent of u on this path. Consider the case when v corresponds to an application of the resolution rule and w be its other child. Let $C_0 \vee (f = b)$, $C_1 \vee (f = b + 1)$, and $C_0 \vee C_1$ be the labels of u, w, and v respectively, where $b \in \{0,1\}$. By the induction hypothesis, every path to v implies

$\neg(C_0 \vee C_1)$. In particular, it implies $\neg C_0$. By construction, the edge (v, u) is labeled with $f = b+1$. Then every path to u going through v implies $\neg C_0 \wedge (f = b+1) = \neg(C_0 \vee (f = b))$. Now consider the case when u corresponds to an application of the weakening rule and let v be it parent on this path. Let C and D be the labels of u and v. Every path to v implies $\neg D$ by the induction hypothesis and $\neg D \vDash \neg C$. Thus, every path to u through v implies $\neg C$.

In particular, every path to the sinks of G falsifies some clause of F. To obtain a linear BP, we remove labels at the inner nodes and contract all unlabeled edges. If the refutation is regular, the resulting linear BP is indeed read-once since we did not essentially change the structure of the underlying DAG.

2. A linear clause $C = \bigvee_{i=1}^{k}(f_i = a_i)$ can be viewed as a negation of a linear system $\neg C = \bigwedge_{i=1}^{k}(f_i = a_i + 1)$. We first convert P into a full BP of size $O(ns)$ using Lemma 16. Inductively, to every node v we associate a linear clause C_v such that:

1. Every assignment reaching v falsifies C_v.
2. If $\neg C_v$ represents a linear system $Bx = b$, then the row space of B is $\mathrm{Pre}(v)$.

For the base case, with each leaf v we associate the clause C_v it is labeled with. The first condition holds since P solves the search problem. To see the second property, note that any path reaching v can be expressed as a linear system on a basis for $\mathrm{Pre}(v)$ which forces every literal in C_v. This implies that single variables in C_v are in $\mathrm{Pre}(v)$.

For the inductive step, consider a node v, which queries q with outgoing neighbors u and w, in the directions $q = 0$ and $q = 1$ respectively. Observe that $\neg C_u \not\vDash q(x) = 1$ and $\neg C_w \not\vDash q(x) = 0$. Thus, there are only two cases to consider:

1. $\neg C_u \not\vDash q(x) = 0$ or $\neg C_w \not\vDash q(x) = 1$,
2. $\neg C_u \vDash q(x) = 0$ and $\neg C_w \vDash q(x) = 1$.

In the first case, we simply let C_v be C_u or C_w, depending on which condition holds. For the second case, let $B = \{\beta_1, \dots, \beta_t\}$ be a basis of $\mathrm{Pre}(v)$. Fullness implies $\mathrm{Pre}(u) = \mathrm{Pre}(w) = \mathrm{Pre}(v) + \mathrm{span}(q)$. Applying the inductive hypothesis, we can write $\neg C_u = (q(x) = 0) \wedge (B_u x = b_u)$ and $\neg C_w = (q(x) = 1) \wedge (B_w x = b_w)$, where B_u and B_w are matrices with rows in $\beta_1, \dots, \beta_t$ and b_u and b_w are some vectors. To write C_u and C_w in these forms, we might need to change the basis, which we can do by applying the weakening rule. We claim that setting C_v so that $\neg C_v$ can be written as $B_u x = b_u \wedge B_w x = b_w$ satisfies the requirements.

Consider any path to v. Such a path can be described by a system $Rx = b$ where rows in R are from B. Since every such path can be extended to both u and w, it follows that $B_u x = b_u \vDash Rx = b$ and $B_w x = b_w \vDash Rx = b$. This means that $B_u x = b_u \wedge B_w x = b_w$ is consistent and thus the derivation of C_v from C_u and C_w (possibly changing the basis) is a valid $\mathsf{Res}[\oplus]$ step. It is easy to see that conditions 1 and 2 hold for C_v.

Since for every v we create at most 2 extra clauses, the total size of the proof is at most $O(ns)$. ◀

Note that we can also define a "strongly regular" variant of $\mathsf{Res}[\oplus]$ that would be equivalent to strongly read-once programs. Since we have obtained average case lower bounds on strongly read-once BPs, it is plausible that it might be easier to prove lower bounds for this restriction of $\mathsf{Res}[\oplus]$.

7 Conclusion

Several problems are immediately suggested by our work:

- *Explicit constructions.* Give an explicit construction of directional affine extractors (or dispersers) for smaller dimension d, ideally $d = o(n)$.
- *BP lower bounds.* Prove worst-case and average-case hardness results for the weakly read-once BPs.
- *Proof complexity.* Prove a read-once linear BP lower bound for a search problem, that is for some unsatisfiable CNF $F = C_1 \wedge \ldots \wedge C_m$, show that a read-once linear BP with leaves labeled by C_is solving the corresponding search problem has to be large.

References

1 Farid M. Ablayev, Aida Gainutdinova, Marek Karpinski, Cristopher Moore, and Chris Pollett. On the computational power of probabilistic and quantum branching program. *Inf. Comput.*, 203(2):145–162, 2005. doi:10.1016/j.ic.2005.04.003.

2 Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC0-MOD2. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 251–260. ACM, 2014. doi:10.1145/2554797.2554821.

3 Alexander E. Andreev, Juri L. Baskakov, Andrea E. F. Clementi, and José D. P. Rolim. Small pseudo-random sets yield hard functions: New tight explicit lower bounds for branching programs. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 179–189. Springer, 1999. doi:10.1007/3-540-48523-6_15.

4 Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012. doi:10.1137/110826254.

5 Jean Bourgain. On the construction of affine extractors. *Geom. Funct. Anal.*, 17(1):33–57, 2007. doi:10.1007/s00039-007-0593-z.

6 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Comput. Complex.*, 24(2):333–392, 2015. doi:10.1007/s00037-015-0100-0.

7 Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ lower bounds for the boolean inner product. *J. Comput. Syst. Sci.*, 97:45–59, 2018. doi:10.1016/j.jcss.2018.04.006.

8 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 47–58. ACM, 2016. doi:10.1145/2840728.2840734.

9 Svyatoslav Gryaznov. Notes on resolution over linear equations. In *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2019. doi:10.1007/978-3-030-19955-5_15.

10 Johan Håstad. On small-depth Frege proofs for Tseitin for grids. *J. ACM*, 68(1):1:1–1:31, 2021. doi:10.1145/3425606.

11 Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 121–131. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.20.

12 Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Ann. Pure Appl. Log.*, 171(1), 2020. doi:10.1016/j.apal.2019.102722.

13 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

14 Erfan Khaniki. On proof complexity of resolution over polynomial calculus. *Electron. Colloquium Comput. Complex.*, page 34, 2020. URL: https://eccc.weizmann.ac.il/report/2020/034.

15 Jan Krajíček. *Proof Complexity.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019. doi:10.1017/9781108242066.

16 Massimo Lauria. A note about k-DNF resolution. *Inf. Process. Lett.*, 137:33–39, 2018. doi:10.1016/j.ipl.2018.04.014.

17 Rudolf Lidl and Harald Niederreiter. *Finite Fields.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1996. doi:10.1017/CBO9780511525926.

18 Fedor Part and Iddo Tzameret. Resolution with counting: Dag-like lower bounds and different moduli. *Comput. Complex.*, 30(1):2, 2021. doi:10.1007/s00037-020-00202-x.

19 A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

20 Petr Savický and Stanislav Žák. A read-once lower bound and a (1, +k)-hierarchy for branching programs. *Theor. Comput. Sci.*, 238(1-2):347–362, 2000. doi:10.1016/S0304-3975(98)00219-9.

21 Rocco A. Servedio and Emanuele Viola. On a special case of rigidity. *Electron. Colloquium Comput. Complex.*, page 144, 2012. URL: https://eccc.weizmann.ac.il/report/2012/144.

22 Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. doi:10.1137/080735096.

23 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

24 Amir Yehudayoff. Affine extractors over prime fields. *Comb.*, 31(2):245–256, 2011. doi:10.1007/s00493-011-2604-9.