Hard CNF Instances for Ideal Proof Systems

Tuomas Hakoniemi*
University of Helsinki

Nutan Limaye†
IT University of Copenhagen

Iddo Tzameret‡
Imperial College London

May 7, 2026

Abstract

Since the introduction of the Ideal Proof System (IPS) by Grochow and Pitassi (J. ACM 2018), a substantial body of work has established size lower bounds for IPS and its fragments. In particular, Forbes, Shpilka, Tzameret, and Wigderson (Theory Comput. 2021) developed the main lower-bound frameworks for restricted IPS fragments, namely functional lower bounds and the hard multiples method, while Alekseev, Grigoriev, Hirsch, and Tzameret (SIAM J. Comput. 2024) gave a general template for conditional lower bounds for full IPS.

Yet all these lower bounds apply only to purely algebraic formulas over a field, that is, non-Boolean formulas not directly expressible in propositional logic. Proving lower bounds for CNF formulas has therefore remained a central open problem in this line of work.

The current work resolves this question for IPS over read-once oblivious algebraic branching programs (roABPs) by proving lower bounds for refutations of CNF formulas in this system. Our approach is a rank-based feasible interpolation argument, following the method of Pudlák and Sgall (Proof Complexity and Feasible Arithmetic 1996) for monotone span programs, in which decomposing a given roABP refutation along a variable partition yields a low-dimensional space of polynomials from which we construct a span-program interpolant. We extend their result from Nullstellensatz refutations measured by degree to Nullstellensatz refutations measured by roABP size (i.e., roABP-IPS$_{\text{LIN}}$).

1 Introduction

This work investigates lower bounds against algebraic proof systems in the framework of the Ideal Proof System (IPS). Proof complexity studies the size of proofs certifying membership in languages such as UNSAT, the set of unsatisfiable Boolean formulas. In this setting, a proof is an efficiently verifiable witness, and for UNSAT such a proof is usually called a refutation. A central goal of the area is to prove lower bounds for increasingly strong proof systems, with the ultimate aim of showing that no proof system has polynomial-size refutations for all unsatisfiable formulas. This line of work is often called *Cook's programme*, following Cook's proposal that proof complexity lower

*Email: tuomas.hakoniemi@helsinki.fi This work has been supported by Helsinki Institute for Information Technology (HIIT).

†Email: nuli@itu.dk This project is supported by funding from Independent Research Fund Denmark (grant agreement No. 10.46540/3103-00116B) and by the Basic Algorithms Research Copenhagen (BARC), which is funded by VILLUM Foundation Grant 54451.

‡Email: iddo.tzameret@gmail.com This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101002742, EPRICOT project). It was also supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant EP/Z534158/1, Integrated Approach to Computational Complexity: Structure, Self-Reference and Lower Bounds

bounds may shed light on basic complexity-theoretic problems such as P versus NP. In particular, proving that no proof system efficiently refutes all unsatisfiable formulas would separate NP from coNP, and hence also P from NP.

Despite major progress, the main lower bound questions in proof complexity, especially those relating to strong enough proof systems such as textbook propositional logic remain open. Even proving lower bounds against propositional proofs operating with constant-depth formulas with counting modulo p gates (i.e. $\mathrm{AC}^0[p]$-Frege) is open. In order to gain better understanding of the latter proof system, algebraic proof systems have been extensively studied beginning in the 1990s (cf. [BIK+96]). Moreover, a recent important direction has been to study *algebraic proof complexity*, with the hope to bring methods from algebraic complexity theory proper into proof complexity (see [PT16] for a survey). Here one ideally aims to shed light on proof systems that goes beyond $\mathrm{AC}^0[p]$-Frege, going up to Frege (i.e. textbook propositional proofs).

The Ideal Proof System, introduced by Grochow and Pitassi [GP18], gives a particularly clean formulation of the connection between algebraic circuits and propositional proofs. IPS can be viewed as a circuit-size version of Nullstellensatz refutations [BIK+96], and hence as a proof system that directly incorporates algebraic circuit complexity into the proof complexity setting. In this way it gives a transparent reduction from circuit lower bounds to proof lower bounds as shown in [GP18]: an IPS lower bound for a CNF formula implies that the permanent polynomial does not have polynomial-size algebraic circuits, namely, $\mathsf{VP} \neq \mathsf{VNP}$. Conversely, Santhanam and Tzameret [ST25] established a partial converse, giving further evidence that IPS forms a natural bridge between algebraic complexity and proof complexity. Namely, assuming $\mathsf{VP} \neq \mathsf{VNP}$, they constructed a family of CNF formulas admitting no polynomial-size IPS refutations. A qualification is that the unsatisfiability of this family remains open, although there is evidence supporting it. The standard formulation of IPS used in this line of work is equivalent to circuit-size Nullstellensatz, and for restricted circuit classes one considers the corresponding fragments such as $\mathcal{C}\text{-IPS}_{\mathrm{LIN}}$ (see below for more details).

Forbes, Shpilka, Tzameret and Wigderson [FSTW21] introduced two approaches for turning algebraic circuit lower bounds into lower bounds for IPS: the *functional lower bound* method and the *lower bounds for multiples* method. Other approaches include the *meta-complexity* approach of [ST25], which obtains a (conditional) IPS size lower bounds on a self-referential statement (cf. [LST26]) and the *noncommutative* approach of Li, Tzameret and Wang [LTW18] (building on [Tza11]).

At the same time, the intended connection back to propositional logic has not yet been fully realised. All known unconditional lower bounds for IPS and its fragments apply to algebraic instances that are not direct encodings of propositional formulas, and in particular not of CNF formulas. Consequently, even a lower bound for IPS on such an algebraic instance does not by itself give a hard instance for propositional proof systems, such as fragments of Extended Frege: the hard instance is not a propositional formula, and hence is not an object in the usual language of propositional logic.

In fact, even moving beyond the *single-axiom framework* has remained unclear. The lower bounds obtained via the functional lower-bound method are all based on a *single* non-Boolean axiom, typically a variant of the subset-sum equation $\sum_i x_i = \beta$, where β is chosen so that the equation has no 0-1 solutions, for example $\beta = -1$. Thus, these results leave open the problem of proving IPS lower bounds for genuinely propositional instances. Andrews and Forbes [AF22] obtained lower bounds against constant-depth IPS, using the hard multiples method, for an algebraic instance with multiple axioms. However, their result is in the "placeholder" model of IPS refutations: the hard instance itself has no polynomial-size constant-depth algebraic representation, and is not a translation of a propositional formula.

The current work resolves this open problem for a natural fragment of IPS, studied in [FSTW21, HLT24, CGMS25, EGLT26], by giving the first hard CNF instances in this fragment. We work with IPS refutations in which the certificate polynomials are computed by read-once oblivious algebraic branching programs (roABPs). More precisely, we prove lower bounds for roABP-IPS$_{\text{LIN}}$ (defined below).

The argument goes back to classical results in propositional proof complexity about feasible interpolation, using an algebraic formulation by Pudlák-Sgall [PS96]. To explain our result and setup we need to start by giving more details about algebraic proof systems and the feasible interpolation method in proof complexity.

Algebraic proof systems. Algebraic proof systems certify that a given set of multivariate polynomials over a field has no common Boolean solutions. Some of the basic proof systems in this line are the Polynomial Calculus (PC) [CEI96] and its 'static' variant, Nullstellensatz [BIK+96]. In PC, proofs proceed by algebraic manipulation, adding and multiplying polynomials, until deriving the contradiction $1 = 0$. Contrastingly, in Nullstellensatz, a proof of the unsatisfiability of a set of axioms, written as polynomial equations $\{f_i(\overline{x}) = 0\}$ over a field, is a *single* formal polynomial identity expressing 1 as a combination of the axioms, that is:

$$\sum_i g_i(\overline{x}) \cdot f_i(\overline{x}) = 1\,, \tag{0.1}$$

for some polynomials $\{g_i(\overline{x})\}$. These systems measure proof size by sparsity, defined as the total number of monomials involved, which makes them comparatively weak. An alternative way to measure proof size is by algebraic circuit size. This was suggested initially by Pitassi [Pit97, Pit98], and further investigated in the work of Grigoriev and Hirsch [GH03] and subsequently Raz and Tzameret [RT08b, RT08a], eventually leading to the Ideal Proof System [GP18] described in what follows.

Ideal Proof System. The Ideal Proof System (IPS for short; Definition 3), introduced by Grochow and Pitassi [GP18], loosely speaking, is the Nullstellensatz proof system where the polynomials $g_i(\bar{x})$ in Equation (0.1) are represented by algebraic circuits. Forbes, Shpilka, Tzameret and Wigderson [FSTW21] showed that these two representations are formally equivalent. In other words, an IPS refutation of the set of axioms $\{f_i(\bar{x}) = 0\}_i$ can be defined similarly to Equation (0.1):

$$\sum_i g_i(\bar{x}) \cdot f_i(\bar{x}) + \sum_j h_j(\bar{x}) \cdot \left(x_j^2 - x_j\right) = 1, \tag{0.2}$$

for some polynomials $\{g_i(\bar{x})\}_i$, where we think of the polynomials g_i, h_j written as algebraic circuits (instead of e.g., counting the number of monomials they contain); and where $x_j^2 - x_j$ are the set of *Boolean axioms*, forcing every solution to the equations $\{x_j^2 - x_j = 0 \ : \ j \in [n]\}$ to 0-1 values. Thus, the size of the IPS refutation in Equation (0.2) is $\sum_i \text{size}\left(g_i(\bar{x})\right) + \sum_j \text{size}\left(h_j(\bar{x})\right)$, where $\text{size}(g)$ stands for the minimal size of an algebraic circuit computing the polynomial g.

Note that IPS is *not* necessarily a Cook-Reckhow propositional proof system, in the sense that there is no known deterministic polynomial-time algorithm to check whether a refutation is a correct refutation. There reason is that to verify Equation (0.2) we need to preform polynomial identity testing (PIT for short), which is a problem in $\mathsf{coRP} \subseteq \text{BPP}$, not known to be in P.

It is natural to consider IPS refutations where the polynomials $g_i(\overline{x})$ and $h_j(\overline{x})$ in Equation (0.2) are written as algebraic circuits from a prescribed circuit class $\mathcal{C}$, for example constant-depth algebraic circuits. However, when considering $\mathcal{C}$ weaker than general algebraic circuits, one has to

be a bit careful with the definition of IPS. For technical reasons the formalization in Equation (0.2) does not capture the precise definition of IPS restricted to the relevant circuit class, rather the fragment which is denoted by $\mathcal{C}$-IPS$_{\text{LIN}}$ ("LIN" here stands for the linearity of the axioms f_i and the Boolean axioms; that is, they appear as polynomials f_i^d with power $d = 1$). In this work, we focus on $\mathcal{C}$-IPS$_{\text{LIN}}$. Namely, refutations in the system $\mathcal{C}$-IPS$_{\text{LIN}}$ are defined as in Equation (0.2) where the polynomials $g_i(\overline{x}), h_j(\overline{x})$ are written as circuits in the circuit class $\mathcal{C}$. In particular we shall consider $\mathcal{C}$ to be the class of roABPs described in what follows.

Read-once oblivious algebraic branching programs. A read-once oblivious algebraic branching program (denoted, roABP) is a layered directed graph with two special vertices, the source in the first layer and the sink in the last layer, and $n-1$ layers in between, and where edges are directed from source to sink. All nodes in layer i are connected with edges only to nodes in layer $i+1$. The edges between layer i and layer $i+1$ are labeled with univariate polynomials in the variable x_i (i.e., in each layer all polynomials have the same (single) variable). Each source-to-node path computes the polynomial that is the product of all the univariate polynomials on the path, and the polynomial computed at each node is the sum of all paths incoming into the node. The program itself computes the polynomial computed at the sink node. The *width* of an roABP is the maximum number of nodes in any layer.

Note that the order in which variables are read in a roABP is fixed. In the above description we made the order to be from x_1 to x_n, namely, by increasing index order, but equivalently we can fix any other linear (i.e., total) order.

The proof system roABP-IPS$_{\text{LIN}}$ defines refutations as in Equation (0.2) in which $g_i(\overline{x})$ and $h_j(\overline{x})$ are written as roABPs. This is an interesting system, because unlike IPS, roABP-IPS$_{\text{LIN}}$ is known to be a Cook-Reckhow proof system: deterministic polynomial-time PIT algorithms for roABPs are known (cf. Raz and Shpilka [RS05]). Moreover, roABP-IPS$_{\text{LIN}}$ is a test case for lower bounds against IPS fragments, because of the relatively simple nature of roABPs, and the fact that lower bounds against this circuit class are well understood.

Feasible interpolation. Let $\varphi_0(\overline{x},\overline{z}) \wedge \varphi_1(\overline{y},\overline{z})$ be an unsatisfiable propositional formula in pairwise disjoint sequences of variables $\overline{x}$, $\overline{y}$ and $\overline{z}$. It follows that for every given total assignment to $\overline{z}$, either $\varphi_0(\overline{x},\overline{z})$ or $\varphi_1(\overline{y},\overline{z})$ is unsatisfiable (or both are); this is because, a single $\overline{z}$-assignment for which both $\varphi_0(\overline{x},\overline{z})$ and $\varphi_1(\overline{y},\overline{z})$ are satisfiable, could be merged into a satisfying assignment to $\varphi_0(\overline{x},\overline{z}) \wedge \varphi_1(\overline{y},\overline{z})$.

The basic interpolation property for propositional logic, analogous to Craig interpolation in first-order logic, guarantees the existence of a function I defined on the $\overline{z}$-variables such that:

$$I(\overline{\alpha}) = \begin{cases} 0, & \text{if } \varphi_0(\overline{x},\overline{\alpha}) \text{ is satisfiable;} \\ 1, & \text{if } \varphi_1(\overline{y},\overline{\alpha}) \text{ is satisfiable.} \end{cases}$$

Such a function I is called *an interpolant* for the conjunction $\varphi_0(\overline{x},\overline{z}) \wedge \varphi_1(\overline{y},\overline{z})$. The function is well-defined, i.e. single-valued, by the assumption that the conjunction is unsatisfiable as explained above. In a similar manner, we can consider feasible interpolation for sets of polynomial equations instead of propositional formula(s) (see bottom of Section 2.2).

In the framework of *feasible interpolation* we are interested in the computational complexity of the interpolants in terms of the sizes of the refutations of the propositional formulas. As such the framework is a method to translate computational lower bounds to proof size lower bounds. See Section 1.2 for more on the history of feasible interpolation.

Span programs. A span program [KW93] consists of a labeled matrix (M, σ) together with a target vector t, where σ is a labeling of the rows, i.e., a mapping from the rows of M to the literals in the variables $\overline{z}$. We also allow labeling a row with the constant 1. A span program is monotone, whenever there is no row labeled with a negative literal. The *size* of the span program is the number of rows (irrespective of the number of columns).

Equivalently, one may define span programs without choosing coordinates. In this formulation, a span program consists of a finite set U of pairs (ℓ, v), where ℓ is a $\overline{z}$-literal or the constant 1, and v is a vector in some vector space V, together with a target vector $t \in V$. This is equivalent to the matrix formulation: a matrix representation is obtained by choosing a basis of V and writing the vectors v as coordinate rows, while conversely the rows of a labeled matrix are simply labeled vectors in the ambient coordinate space. We use this coordinate-free formulation below because our vectors will naturally be polynomials, and no particular choice of monomial basis will be relevant.

A span program as above *computes* a Boolean function as follows: on a 0-1 assignment a to the variables $\overline{z}$, let $U(a)$ be the set of vectors v so that there is some literal ℓ with $(\ell, v) \in U$ and $\ell(a) = 1$. In other words, $U(a)$ consists of all vectors picked by the assignment $\overline{z}$, namely those vectors whose $\overline{z}$-label gets 1 under a. The span program outputs 1 if the target vector t is in the span of the vectors in $U(a)$ and 0 otherwise.

1.1 Our Results

Our main result is an exponential lower bound against roABP-IPS$_{\text{LIN}}$ for a family of CNF formulas, for any variable order and over any field. Namely, given the CNF formula, no roABP-IPS$_{\text{LIN}}$ certificate of any variable order and sub-exponential size exists.

Theorem 1 (Main lower bound (informal); see Theorem 18)**.** *There is a family of CNF formulas* Ψ_n *in* $\mathsf{poly}(n)$ *variables and* $\mathsf{poly}(n)$ *clauses that requires* roABP-IPS$_{\text{LIN}}$ *refutations of size* $2^{n^{\Omega(1)}}$ *in any variable ordering, and over any field.*

This resolves a problem left open by previous works on IPS lower bounds [FSTW21, HLT24, For24, And25, EGLT26]; see also the discussion in Andrews [And25, Sec. 1.3]. Namely, whether one can prove an IPS lower bound for an instance that is not a single polynomial equation and, more importantly, for a genuinely Boolean instance: a direct arithmetization of a propositional formula, specifically a CNF formula.

The significance of this result is twofold. First, CNF formulas are the standard benchmark instances in proof complexity. Second, as discussed above, they provide a direct connection to Frege-style propositional proof systems: a lower bound for $\mathcal{C}$-IPS on CNFs immediately implies the same lower bound for any propositional proof system simulated by $\mathcal{C}$-IPS. This implication is unavailable for purely algebraic instances, such as subset-sum equations of the form $\sum_i a_i x_i - \beta$ and their variants.

For example, a lower bound against constant-depth IPS over $\mathbb{F}_p$ for a CNF would imply a lower bound for $\mathrm{AC}^0[p]$-Frege, a longstanding open problem. Although recent work of Elbaz *et al.* [EGLT26] shows that even non-Boolean, namely purely algebraic, instances that are hard for constant-depth IPS over finite fields imply a CNF lower bound via a general translation lemma, it remains important to develop *direct* proof-size lower-bound methods for CNF formulas, in the hope that such methods may eventually yield lower bounds against constant-depth IPS refutations.

The family of hard CNF formulas Ψ is based on a lifting of the instance GEN_n introduced in Raz and McKensize [RM99] (Definition 11) and used throughout circuit and proof complexity (see

in particular Pitassi-Robere [PR18], Chan-Potechin [CP14] and Robere, Pitassi, Rossman, and Cook [RPRC16]).

The proof of Theorem 1 has three main steps:

1. Interpolation;
2. Hard monotone instance;
3. Lifting.

We describe these three steps in what follows.

Interpolation. We establish a general feasible interpolation result as follows. Let $\Psi = \phi_0(\overline{x}, \overline{z}) \wedge \phi_1(\overline{y}, \overline{z})$ be any split formula (with P_0 being a set of sparse polynomials; e.g., when Ψ is a k-CNF, for a small k). From a short roABP-IPS$_{\text{LIN}}$ refutation with a fixed variable order, refuting Ψ, we show how to extract a small span program over the input $\overline{z}$ variables separating inputs for which $\phi_0(\overline{x}, \overline{z})$ is satisfiable from those inputs for which $\phi_1(\overline{y}, \overline{z})$ is satisfiable. When the $\overline{z}$ variables are all positive in ϕ_0 or all negative in ϕ_1 we extract a *monotone* span program, hence we call it "monotone feasible interpolation".

Theorem 2 (Feasible interpolation for roABP-IPS$_{\text{LIN}}$; see Theorem 9)**.** *Let $P_0(\overline{x}, \overline{z})$ and $P_1(\overline{y}, \overline{z})$ be sets of polynomial equalities, where $\overline{x}, \overline{y}$ and $\overline{z}$ are pairwise disjoint variables and P_0 is a set of sparse polynomials of polynomial degree (e.g., a translation of k-CNF, for a constant k). Suppose that $P_0 \cup P_1$ has an* roABP-IPS$_{\text{LIN}}$ *refutation of width w in a variable ordering, where $\overline{x}$ variables precede all other variables. Then there is a span program of size $\mathsf{poly}(w \cdot |P_0(\overline{x}, \overline{z})|)$ that computes an interpolant for P_0 and P_1. Moreover, if all the polynomials in P_0 are monotone in $\overline{z}$ (see Equation (5.1)), then the span program is monotone.*

Proof idea: The proof has one central idea. Start with an roABP-IPS$_{\text{LIN}}$ refutation

$$\sum_{p \in P_0} a_p p + \sum_{q \in P_1} b_q q = 1$$

in a variable order where all $\overline{x}$-variables come first. First, put P_0 into $\overline{z}$-normal form, so each polynomial is either $p_0(\overline{x})$ or $p_0(\overline{x}) + z_j p_1(\overline{x})$, for some polynomials $p_0(\overline{x}), p_1(\overline{x})$ in the displayed variables $\overline{x}$ only. Then use the roABP structure: since each a_p is computed by a width-w roABP and all $\overline{x}$-variables appear first, cutting the roABP right after the last $\overline{x}$-layer gives

$$a_p = \sum_{i \in [w]} a_{p,i}(\overline{x})\, r_i(\overline{y}, \overline{z}).$$

The span program is built from the $\overline{x}$-parts $a_{p,i}$: for each $p_0(\overline{x}) + z_j p_1(\overline{x})$, add vectors labeled by z_j and $1 - z_j$, and for each $p_0(\overline{x})$, add a constant-labeled vector. The target vector is 1.

Why does this work? If the span program outputs 1 on an assignment $\overline{\alpha}$ to $\overline{z}$, then the selected vectors already express 1 as a linear combination of the specialized polynomials from $P_0(\overline{x}, \overline{\alpha})$, so $P_0(\overline{x}, \overline{\alpha})$ is unsatisfiable. Conversely, if it outputs 0 but $P_1(\overline{y}, \overline{\alpha})$ were satisfiable, plugging a satisfying $\overline{y}$-assignment into the original refutation would kill the P_1-part and leave a refutation of $P_0(\overline{x}, \overline{\alpha})$; by the decomposition above, this refutation is again a linear combination of the selected vectors, which contradicts the fact that the span program outputs 0. Thus the span program computes an interpolant. If P_0 is monotone in $\overline{z}$, the construction is monotone as well.

Hard monotone instance. To make use of the above feasible monotone interpolation in order to establish an actual proof size lower bound we need to find a monotone function that we know is hard for monotone span programs. Moreover, we must be able to find a suitable polynomial-size split formula Ψ, of which its interpolant is the hard monotone function. The idea is to use the function GEN_n that was shown hard for monotone span programs by Robere, Pitassi, Rossman and Cook [RPRC16, PR18]. Unlike [RPRC16, PR18], we do not need to insist on a function that is computable by small monotone circuits, rather a general (non-monotone) circuit suffices. Hence, we can use the general function GEN_n instead of the pyramid-based one in the previous works (see Section 3.1).

To form the split formula $\Psi = \phi_0(\overline{x}, \overline{z}) \wedge \phi_1(\overline{y}, \overline{z})$ whose interpolant is the hard instance for monotone span programs we define it as follows. The formula $\phi_0(\overline{x}, \overline{z})$ describes the negative instances of GEN_n and $\phi_1(\overline{y}, \overline{z})$ describes its positive instances. Also, because GEN_n is monotone, the associated formulas can be constructed to be monotone in a sense that makes monotone feasible interpolation applicable. Hence the interpolant extracted from a small refutation is a a small monotone span program separating no-instances from yes-instances of GEN_n, shown hard in [RPRC16, PR18].

Lifting. In the previous step we obtained a split formula that is hard for a specific order of variables (when the $\overline{x}$-variables come before the other variables). This does not rule out the possibility of a small refutation for a different variable ordering. To rule this out, we apply a "lifting" process on the split formula Ψ that, roughly, embeds all possible variable orders (see [FSTW21] and subsequent work on lifting in IPS; and cf. [dRGR22] for general lifting in proof complexity). We explain it informally in what follows.

Our aim is to remove the dependence on a fixed variable order. In the previous step the hard CNF $\phi = \phi_0 \wedge \phi_1$ is only shown to be hard for roABP-IPS$_{\mathrm{LIN}}$ refutations when the variables are read in one specific order, say $x_1 < \cdots < x_N$. Lifting constructs a larger CNF formula

$$\Psi(\overline{v}, \overline{u}) = \Phi_1(\overline{v}, \overline{u}) \wedge \Phi_2(\overline{u})$$

such that, for every variable order $<$ on the variables of Ψ, there is a substitution $\rho_<$ with the following effect: $\rho_<$ assigns Boolean values to the auxiliary variables $\overline{u}$, relabels the $\overline{v}$-variables by $x_1, \ldots, x_N$ according to their order under $<$, and after this substitution the lifted clause part Φ_1 becomes exactly the original hard CNF formula ϕ, while the "consistency part" Φ_2 (see below) is satisfied. Thus ϕ appears as a restricted instance of Ψ. Consequently, any roABP-IPS$_{\mathrm{LIN}}$ refutation of Ψ in order $<$ yields, under $\rho_<$, a refutation of ϕ in its hard order.

The formula $\Phi_1(\overline{v}, \overline{u})$ is the actual lift of ϕ. It uses new variables $\overline{u}$ and $\overline{v}$. The $\overline{v}$-variables are the new main variables, whose order inside the refutation may be arbitrary. The $\overline{u}$-variables are selector variables. For each clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ of ϕ, and for each triple (i, j, k) of distinct indices, the variable $u_{C,i,j,k}$ says that the first, second, and third literals of C are realised on v_i, v_j, v_k, respectively, with the same signs as in C ("realised" in the sense that there will be a relabelling of $\overline{v}$-variables by $\overline{x}$-variables to recover the original instance). This is enforced by the implication

$$u_{C,i,j,k} \to \left(v_i^{\mathrm{sg}(\ell_1)} \vee v_j^{\mathrm{sg}(\ell_2)} \vee v_k^{\mathrm{sg}(\ell_3)}\right),$$

together with clauses saying that for each original clause C, exactly one such triple is chosen. Hence Φ_1 embeds each clause of ϕ locally into the $\overline{v}$-variables.

However, these local choices could be inconsistent across different clauses. The same original variable might correspond to different $\overline{v}$-variables in different places. The role of Φ_2 is to prevent this. It is an auxiliary consistency gadget, not part of the original split $\phi_0 \wedge \phi_1$. It checks that the

selected $\overline{u}$-variables are globally consistent, meaning that local correspondences for each original clause give rise to a global one-to-one correspondence between the original variables $\overline{x}$ and the new variables $\overline{v}$. Thus Φ_2 is satisfiable exactly when the local choices in Φ_1 come from one single global relabelling.

Now fix an arbitrary variable order $<$ on Ψ, and let the induced order on the $\overline{v}$-variables be

$$v_{i_1} < v_{i_2} < \cdots < v_{i_N}.$$

This order determines the restriction $\rho_<$. First, relabel $v_{i_k} \mapsto x_k$ for each $k \in [N]$. Next, for each clause C of ϕ, set exactly one selector variable $u_{C,i,j,k}$ to 1, namely the one corresponding to the positions of the three variables of C in this $\overline{v}$-order, and set all other selector variables for C to 0. Since this choice comes from one global relabelling, it is automatically globally consistent, and therefore Φ_2 can be satisfied by a suitable assignment to the $\overline{u}$-variables. After this restriction, the auxiliary gadget disappears and the lifted formula reduces to the original hard CNF ϕ, with the surviving variables now ordered as $x_1 < \cdots < x_N$.

Therefore the restriction depends on the variable order $<$, but not on any further feature of the proof. Once the order of the $\overline{v}$-variables is fixed, the restriction is fixed as well. Hence a small roABP-IPS$_{\text{LIN}}$ refutation of Ψ in any order would yield a small roABP-IPS$_{\text{LIN}}$ refutation of ϕ in its hard order, contradicting the lower bound for a fixed order we obtained in the previous step. Hence, Ψ is hard for roABP-IPS$_{\text{LIN}}$ in every variable order.

1.2 Connection to Previous Work

Feasible interpolation in proof complexity arose from the interaction between bounded arithmetic and propositional proof systems. Early background appears in Krajíček–Pudlák [KP89] and in Razborov's work from the mid-1990s [Raz95a, Raz95b]. It was developed as a proof-complexity lower-bound method by Bonet, Pitassi, and Raz [BPR97], and formulated explicitly in Krajíček [Kra97]. These works formulated the method via communication complexity. Pudlák [Pud97] then gave a structural interpolation theorem that extracts computation from proofs without going through communication complexity. Our feasible interpolation theorem is most closely related to the monotone interpolation theorem for algebraic proof systems due to Pudlák and Sgall [PS96], that extracts a small (monotone) span program from a low degree algebraic proof.

Feasible interpolation continued to play a major role in proof complexity throughout the years. In particular, lower bounds on OBDD-based proof systems by Krajíček [Kra08] (cf. Segerlind [Seg07, Seg08] for the tree-like case), used variable ordering arguments close to our constructions. OBDD-based propositional proof systems were introduced by Atserias, Kolaitis, and Vardi [AKV04] and further investigated in several later works (cf. [BIK$^+$21, IRS17, IR22]).

The ability to apply monotone feasible interpolation for monotone span programs stems from the results of Robere, Pitassi, Rossman and Cook, and Pitassi-Robere [RPRC16, PR18].

As for IPS refutations and specifically roABP-IPS$_{\text{LIN}}$, this system was considered first in Forbes, Shpilka, Tzameret and Wigderson [FSTW21]. They proved an exponential size lower bound for this system alas for a single formula. All lower bounds against IPS and fragments are for either a single algebraic instance [FSTW21, GHT22, HLT24, BLRS25, EGLT26, CGMS25, BHLS26]; or for non-single axiom as in Andrews-Forbes [AF22], but there the instance is algebraic and the lower bound is in the "placeholder model", meaning that the hard instance against $\mathcal{C}$-IPS cannot be written by itself with a (small) circuit in the class $\mathcal{C}$.

1.3 Concluding Discussion and Open Problems

The lower bound method developed here shows that, for roABP-IPS$_{\mathrm{LIN}}$, feasible interpolation is not far from rank-based approaches as used in all previous (unconditional) IPS lower bounds (namely, by reducing IPS refutation size lower bound to an algebraic circuit lower bound, where the latter is established using a rank-argument). The key step in the interpolation theorem is to cut each coefficient roABP at the interface between the $\overline{x}$-variables and the remaining variables. This yields a decomposition through a space of dimension at most the width of the roABP, and this low-dimensional space is then converted into a span-program interpolant. Thus, at a higher level, the argument has the same structure as a rank-argument used throughout algebraic circuit complexity and previous IPS lower bounds: a small refutation gives rise to a low-dimensional object, while hardness of the interpolant implies that every such object must be large. In our case, the first ingredient is the roABP decomposition, and the second is the monotone span-program lower bound for GEN_n.

This also gives a more concrete explanation of the remark in [FSTW21] that the functional lower-bound method is reminiscent of feasible interpolation. At least for roABP-based IPS, the current work decreases the gap between these two methods.

It is also instructive to compare the present result with lower bounds for OBDD-based proof systems. The two settings are related in that both are sensitive to variable order, and both admit lower-bound arguments that exploit this feature. However, the systems themselves are orthogonal. The proof system studied here is algebraic and static, and it works with coefficients represented by roABPs, which are strictly more expressive than OBDDs. By contrast, OBDD proof systems are propositional, dynamic, and semantic. Therefore, the present lower bound should not be understood as a direct analogue of OBDD lower bounds, but rather as an algebraic proof-complexity result that shares with them only the order-sensitive aspect of the argument.

Finally, the lower bound does not necessarily reflect a general weakness of roABP-IPS$_{\mathrm{LIN}}$. The system also admits nontrivial upper bounds as we show in this work. In particular, as we show in Section 5, Tseitin formulas have polynomial-width roABP-IPS$_{\mathrm{LIN}}$ refutations in every variable order: one can start from Grigoriev's polynomial-sparsity Nullstellensatz refutation in the $\{\pm 1\}$ basis and then apply a linear change of basis, which preserves small roABP width. And further roABP-IPS$_{\mathrm{LIN}}$ simulates tree-like PC proof length (number of lines) and efficiently refutes the functional pigeonhole principle formulas.

Regarding *questions left open*, the immediate problem is to see if feasible interpolation could obtain CNF formulas hard against other fragments of IPS for which lower bounds are already known for single-axiomed algebraic instances. These fragments include multilinear formula IPS [FSTW21, HLT24] and constant-depth IPS refutations with low individual degrees as in [GHT22, HLT24, EGLT26].

On the other hand, strong enough proof systems, such as constant-depth IPS with no bounds on individual degrees simulate $\mathrm{AC}^0[p]$-Frege (when the IPS system works over the field $\mathbb{F}_p$; see [GP18]) and plausibly TC^0-Frege (when the IPS system works over characteristic 0 fields). The latter system TC^0-Frege is known not to have feasible interpolation property based on the hardness of factoring [BPR00] (i.e., non-feasible interpolation follows assuming factoring Blum integers is not in $\mathsf{P/poly}$), and similarly the former system $\mathrm{AC}^0[p]$-Frege does not have feasible interpolation unless the Diffie-Hellman function can be computed by subexponential-size circuits, i.e., 2^{n^ϵ}, for arbitrarily small $\epsilon > 0$ [BDG+99]. Hence, we cannot realistically hope for feasible interpolation to hold for (unrestricted individual degree) constant-depth IPS refutations.

1.4 Follow-ups

After our work was presented in a talk, Dmitry Sokolov [Sok26] pointed out a way to lift general Nullstellensatz degree lower bounds to roABP-IPS$_{\text{LIN}}$ refutation size lower bounds for a fixed variable order. This is based on the algebraic tiling approach of Pitassi and Robere [PR18] (cf. [RPRC16]). In this sense, one obtains CNF formulas hard against roABP-IPS$_{\text{LIN}}$ in a fixed variable order by lifting any CNF formula that is hard for Nullstellensatz degree. Using the method developed in Section 4, one can then extend the resulting lower bound to arbitrary variable orders.

2 Preliminaries

2.1 Notation

We use lower case over-lined Latin letters $\overline{x}, \overline{y}, \overline{z}$, etc. to denote sequences of (Boolean) variables. Single variables always have a subscript attached, e.g. x_i or y_{j_0}. We will use lower case over-lined Greek letters $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ etc. to denote Boolean assignments, whose domain will be clear from the context.

2.2 Feasible interpolation

Let $\varphi_0(\overline{x}, \overline{z}) \wedge \varphi_1(\overline{y}, \overline{z})$ be an unsatisfiable propositional formula in pairwise disjoint sequences of variables $\overline{x}$, $\overline{y}$ and $\overline{z}$. We call such formulas *split formulas.* It follows that for every given total assignment to $\overline{z}$, either $\varphi_0(\overline{x}, \overline{z})$ or $\varphi_1(\overline{y}, \overline{z})$ is unsatisfiable (or both are); this is because, a single $\overline{z}$-assignment for which both $\varphi_0(\overline{x}, \overline{z})$ and $\varphi_1(\overline{y}, \overline{z})$ are satisfiable, could be merged into a satisfying assignment to $\varphi_0(\overline{x}, \overline{z}) \wedge \varphi_1(\overline{y}, \overline{z})$.

The basic interpolation property for propositional logic, analogous to Craig interpolation in first-order logic, guarantees the existence of a function I defined on the $\overline{z}$-variables such that:

$$I(\overline{\alpha}) = \begin{cases} 0, \text{ if } \varphi_0(\overline{x}, \overline{\alpha}) \text{ is satisfiable;} \\ 1, \text{ if } \varphi_1(\overline{y}, \overline{\alpha}) \text{ is satisfiable.} \end{cases}$$

Note that this is identical to the following two conditions:

$$\begin{aligned} &\text{If } I(\overline{\alpha}) = 0, \text{ then } \varphi_1(\overline{y}, \overline{\alpha}) \text{ is unsatisfiable; and} \\ &\text{If } I(\overline{\alpha}) = 1, \text{ then } \varphi_0(\overline{y}, \overline{\alpha}) \text{ is unsatisfiable.} \end{aligned} \tag{2.1}$$

Such a function I is called *an interpolant* for the conjunction $\varphi_0(\overline{x}, \overline{z}) \wedge \varphi_1(\overline{y}, \overline{z})$. The function is well-defined, i.e. single-valued, by the assumption that the conjunction is unsatisfiable as explained above.

In the framework of *feasible interpolation* we are interested in the computational complexity of the interpolants in terms of the sizes of the refutations of the propositional formulas. As such the framework is a method to translate computational lower bounds to proof size lower bounds. We say that a proof system $\mathcal{P}$ admits a feasible interpolation property with respect to a circuit model $\mathcal{C}$, if for any $\mathcal{P}$-refutation of size s of a split formula $\varphi_0(\overline{x}, \overline{z}) \wedge \varphi_1(\overline{y}, \overline{z})$ there is a $\mathcal{C}$-circuit of size $\mathsf{poly}(s)$ that computes an interpolant for $\varphi_0(\overline{x}, \overline{z}) \wedge \varphi_1(\overline{y}, \overline{z})$. In the case that there exists monotone function interpolating $\varphi_0(\overline{x}, \overline{z}) \wedge \varphi_1(\overline{y}, \overline{z})$ we say that the proof system $\mathcal{P}$ admits a *monotone feasible interpolation* with respect to a monotone circuit model $\mathcal{C}$ if for any $\mathcal{P}$-refutation there is a monotone circuit from $\mathcal{C}$ that computes an interpolant.

In the current work we consider algebraic proof systems that can be used more generally to prove that a given set of polynomial equations over a field does not have 0-1 solutions. This is more

general than showing that a set of formulas in propositional logic is unsatisfiable (since propositional formulas can be encoded directly by polynomial equations). The notion of interpolation generalises naturally to this case, and we will prove our feasible interpolation result in this general setting. Namely, let $P_0(\overline{x}, \overline{z})$ and $P_1(\overline{y}, \overline{z})$ be two sets of polynomials in the sequences of variables $\overline{x}$, $\overline{y}$ and $\overline{z}$ that are mutually unsatisfiable, i.e. that have no common (Boolean) root. A Boolean function $I(\overline{z})$ is an interpolant for P_0 and P_1 if it satisfies the following for every assignment $\overline{\alpha}$:

$$I(\overline{\alpha}) = \begin{cases} 0, \text{ if } P_0(\overline{x}, \overline{\alpha}) \text{ is satisfiable;} \\ 1, \text{ if } P_1(\overline{y}, \overline{\alpha}) \text{ is satisfiable.} \end{cases}$$

This is again equivalent to the following two conditions:

$$\begin{aligned} &\text{If } I(\overline{\alpha}) = 0, \text{ then } P_1(\overline{y}, \overline{\alpha}) \text{ is unsatisfiable; and} \\ &\text{If } I(\overline{\alpha}) = 1, \text{ then } \mathsf{P}_2(\overline{y}, \overline{\alpha}) \text{ is unsatisfiable.} \end{aligned} \tag{2.2}$$

The notions of feasible and monotone feasible interpolation translate directly to the setting with general polynomial equalities.

2.3 Read-once Oblivious Algebraic Branching Programs

A variable ordering of a finite set of variables is a linear order on the set, e.g. $x_1 < \ldots < x_n$. A read-once oblivious algebraic branching program (roABP) [For14] in the variable ordering $x_1 < \ldots < x_n$ consists of a sequence of matrices $M_1, \ldots, M_n$ so that M_i is a matrix whose entries are univariate polynomials in the variable x_i. The branching program computes the polynomial

$$(M_1 \cdots M_n)_{(1,1)},$$

i.e. the upper left entry of the product of these matrices. This definition assumes that all the matrices here are of appropriate dimension so that the product is well-defined. The size measure for such a branching program is defined in terms of width: the *width* of an roABP is the maximal dimension of the matrices.

Read-once oblivious algebraic branching programs can also be defined more combinatorially as a layered directed graphs with two special vertices, the source in the first layer and the sink in the last layer, and $n-1$ layers in between. Each edge in the graph is between consecutive layers, that is directed from layer i to layer $i+1$. The edges between layer i and layer $i+1$ are labelled with univariate polynomials in the variable x_i. Each source-to-sink path computes the polynomial that is the product of all the univariate polynomials on the path, and the program itself computes the sum of all the paths in the graph. It is not hard to see that these two definitions are equivalent. In this formulation, the width of the program corresponds to the maximum size of any layer.

There is a particular representation that is useful for our purposes below (cf. [For14]). If a polynomial p is computable by a width w roABP in a variables ordering $x_1 < \ldots < x_n$, then for any $i \in [n]$ it can be written as a sum

$$\sum_{j \in [w]} p_{ij} q_{ij},$$

where each p_{ij} is a polynomial in the variables $x_1, \ldots, x_i$ and each q_{ij} is a polynomial in the variables $x_{i+1}, \ldots, x_n$.

2.4 Algebraic Proof Systems

A Nullstellensatz proof system, introduced to proof complexity in [BIK+96], is based, as the name suggests, on Hilbert's Nullstellensatz. Here a Nullstellensatz proof of $p = 0$ from a sequence $p_1 = 0, \ldots, p_m = 0$ of polynomial equalities is a sequence $q_1, \ldots, q_m$ of polynomials so that

$$p = p_1 q_1 + \ldots + p_m q_m. \tag{2.3}$$

A Nullstellensatz refutation of $p_1 = 0, \ldots, p_m = 0$ is a Nullstellensatz proof of $1 = 0$ from $p_1 = 0, \ldots, p_m = 0$. The complexity of these proofs is most commonly measured by the maximum degree of the summands, i.e. $\max_i \deg(p_i q_i)$, or by sparsity, or monomial-size, of the summands.

Polynomial Calculus [CEI96] is a dynamic variant of Nullstellensatz, where the Nullstellensatz proof is derived by local inference rules. A Polynomial Calculus proof of $p = 0$ from a sequence $p_1 = 0, \ldots, p_m = 0$ of polynomial equalities is a sequence $q_1, \ldots, q_\ell$ of polynomials so that $q_\ell = p$ and one of the following holds for every $i \in [\ell]$:

1. q_i is one of the axioms, i.e. of the polynomials p_j;
2. there is $j < i$ and a variable x so that $q_i = x q_j$;
3. there are $j, k < i$ and field elements a, b so that $q_i = a q_j + b q_k$.

A Polynomial Calculus refutation a sequence $p_1 = 0, \ldots, p_m = 0$ is again a proof of $1 = 0$ from the sequence. We say that the proof is *tree-like* if the underlying proof graph is a tree. Here the common complexity measures are again the degree and sparsity of the polynomials in the sequence. Another important measure for this work is the length of the proof, i.e. the number of polynomials in the sequence; above this is ℓ.

Ideal Proof System (IPS) [GP18] allows expressing the Nullstellensatz proof succinctly as a single algebraic circuit. Formally we have the following definition.

Definition 3 (Ideal Proof System [GP18])**.** Let $p_1 = 0, \ldots, p_m = 0$ be a sequence of polynomial equalities in variables $x_1, \ldots, x_n$. An IPS proof of $p = 0$ from $p_1 = 0, \ldots, p_m = 0$ is an algebraic circuit $C(\overline{x}, \overline{z})$ in variables $x_1, \ldots, x_n$ and new placeholder variables $z_1, \ldots, z_m$ so that

1. $C(\overline{x}, \overline{0}) = 0$;
2. $C(\overline{x}, p_1, \ldots, p_m) = p$.

An IPS refutation of the sequence $p_1 = 0, \ldots, p_m = 0$ is an IPS proof of $1 = 0$ from the sequence. The size of the IPS proof is the circuit of C.

Following [FSTW21] we call an IPS proof *linear* if the polynomial computed by the circuit C computes a polynomial that is linear in the $\overline{z}$ variables ([GP18] calls this Hilbert-like IPS); the polynomial computed is thus of the form $\sum_{i \in [m]} q_i(\overline{x}) z_i$ for some polynomials q_i. The size of this expression is roughly the size of the circuits computing the polynomials $q_i(\overline{x})$. Thus a linear IPS proof can be considered as a Nullstellensatz proof, where the size of the proof is measured by the algebraic circuit size of the polynomials $q_1, \ldots, q_m$ in Equation (2.3)

In this work we consider linear IPS proofs, where the polynomials $q_1, \ldots, q_m$ in Equation (2.3) are computed by read-once algebraic branching programs. We call these roABP-IPS$_{\text{LIN}}$ proofs. The complexity of an roABP-IPS$_{\text{LIN}}$ proof is measured by the maximum width of an roABP needed to compute any of the polynomials $q_1, \ldots, q_m$. We assume a common variable ordering for the whole proof and the complexity of the proof can and usually will depend on this choice. To emphasize the chosen variable ordering we talk about roABP-IPS$_{\text{LIN}}$ refutations in some particular variable ordering.

2.4.1 A Translation from CNFs to Polynomials

In order to consider these systems as a refutation systems for CNFs we fix a *standard translation* of a set of clauses into a set of polynomial equations. For a clause $C = \bigvee_{i\in I} x_i \vee \bigvee_{j\in J} \neg x_j$ we define its translation to be the polynomial

$$\mathrm{tr}(C) := \prod_{i\in I}(1-x_i)\prod_{j\in J} x_j$$

and for a CNF F in variables $x_1,\dots,x_n$ consisting of clauses $C_1,\dots,C_m$ we define its translation to be the set

$$\{\mathrm{tr}(C_i) = 0 : i \in [m]\} \cup \{x_i^2 - x_i = 0 : i \in [n]\}.$$

2.5 Span Programs

A span program [KW93] consists of a labeled matrix (M,σ) together with a target vector t, where σ is a labeling of the rows, i.e. a mapping from the rows of M to the literals in the variables $\overline{z}$. We also allow labeling a row with the constant 1. A span program is monotone, whenever there is no row labeled with a negative literal. The *size* of the span program is the number of rows. Equivalently define a span program as a set U of pairs (ℓ, v), where ℓ is a $\overline{z}$-literal or a constant 1 and v is a vector in some underlying vector space V together with a target vector t that lies also in the space.

A span program as above *computes* a Boolean function as follows: on a 0-1 assignment a to the variables $\overline{z}$, let $U(a)$ be the set of vectors v so that there is some literal ℓ with $(\ell, v) \in U$ and $\ell(a) = 1$. In other words, $U(a)$ consists of all vectors picked by the assignment $\overline{z}$, namely those vectors whose $\overline{z}$-label gets 1 under a. The span program outputs 1 if the target vector t is in the span of the vectors in $U(a)$ and 0 otherwise.

Remark 4. Usually span programs are defined as labeled only with literals and not with the constant 1. This inclusion is only for convenience as there are simple reductions to get rid of the constant. If the span program U is non-monotone, it suffices to replace each $(1, v)$ with the pairs (z_i, v) and $(\bar{z}_i, v)$ for some variable z_i; the size of the program only increases by a factor of two. If the span program U is monotone, replace each $(1, v)$ with the set of all the pairs (z_i, v) for each variable z_i. The obtained program computes the same function unless the function is the constant function 1 and the size increases only by a polynomial factor.

3 Feasible Interpolation Property for Fixed Variable Ordering

In this section we prove the feasible interpolation result for roABP-IPS$_{\mathrm{LIN}}$ when the order of variables queried along the roABPs is fixed.

Before proceeding to the proof we show that we can assume a certain normal form for a given set of polynomial constraints. This normal form was introduced by Pudlák and Sgall [PS96] (cf. [FGGR22] for more recent uses).

Definition 5 (Monotone normal form)**.** Let $P(\overline{x}, \overline{z})$ be a set of polynomials in the (pairwise disjoint) sets of variables $\overline{x}, \overline{z}$.

- $P(\overline{x}, \overline{z})$ is *in a $\overline{z}$-normal form* if each polynomial in the set is of the form

  $$p + z_i p',$$

  for some polynomials p and p' in the $\overline{x}$ variables and a *single* $\overline{z}$-variable z_i.

- $P(\overline{x}, \overline{z})$ is *monotone in* $\overline{z}$ if each polynomial in the set is of the form

$$\left(\prod_{i \in I} z_i\right) \cdot p, \tag{5.1}$$

where I is some subset of indices and p is a polynomial in the $\overline{x}$ variables only. Here, monotonicity is with respect to the $\overline{z}$-variables (and not necessarily the $\overline{x}$-variables), meaning that over 0-1 values to the $\overline{z}$-variables, the polynomial is monotone in the following sense: when p is fixed and has a nonnegative value, flipping a z_i from 0 to 1 can only increase the value of (5.1).

- $P(\overline{x}, \overline{z})$ is in a *monotone* $\overline{z}$*-normal form* if the size of the set of indices I in (5.1) is at most 1 for every polynomial in the set. In other words, each of its polynomials is of the form $z_i p'$ or p'.

In the below, sparsity(p), for a polynomial p, denotes the number of monomials in p, and we also call sparsity(p) the *sparse size of* p.

Lemma 6 (Normal form lemma)**.** *Let $P(\overline{x}, \overline{z})$ be a finite set of sparse polynomials over a field $\mathbb{F}$. Let $s := \sum_{p \in P} \text{sparsity}(p)$ and $n := |\overline{x}| + |\overline{z}|$, and assume that every polynomial in P has degree at most $(s+n)^{O(1)}$. Then, there is a set $P'(\overline{x}, \overline{w}, \overline{z})$ of polynomials, of sparsity $\mathsf{poly}(s+n)$ and where $\overline{w}$ are fresh variables, such that:*

1. *P' is in $\overline{z}$-normal form, namely every polynomial in P' is of the form $q(\overline{x}, \overline{w}) + z_i q'(\overline{x}, \overline{w})$, for some i, or contains no $\overline{z}$-variable;*
2. *for every assignment $\alpha \in \{0,1\}^{|\overline{z}|}$, P and P' are equisatisfiable in the following sense:*

$$\exists \overline{x} \in \{0,1\}^{|\overline{x}|}\ P(\overline{x}, \alpha) = 0 \quad \Longleftrightarrow \quad \exists \overline{x} \in \{0,1\}^{|\overline{x}|} \exists \overline{w} \in \mathbb{F}^{|\overline{w}|}\ P'(\overline{x}, \overline{w}, \alpha) = 0;$$

3. *every polynomial $p \in P$ has an* roABP-IPS$_{\text{LIN}}$ *derivation from P' of width $\mathsf{poly}(s+n)$, given any variable ordering.*

Moreover, if P is monotone in $\overline{z}$, then P' can be chosen in monotone $\overline{z}$-normal form.

Remark 7. The auxiliary variables $\overline{w}$ introduced in the lemma are algebraic extension variables; they are not constrained by Boolean axioms. Thus satisfiability of $P'(\overline{x}, \overline{w}, \alpha)$ means satisfiability by Boolean values for the original variables $\overline{x}$ and field values for the auxiliary variables $\overline{w}$.

Proof: We give two different constructions depending on whether P is monotone in $\overline{z}$ or not.

Nonmonotone case. Assume first that P is not necessarily monotone in $\overline{z}$. Introduce a fresh auxiliary variable w_i for every variable z_i. For every polynomial $p(\overline{x}, \overline{z}) \in P$, let $p^\star(\overline{x}, \overline{w})$ be obtained from p by replacing each z_i by w_i. Define

$$P'(\overline{x}, \overline{w}, \overline{z}) := \{p^\star(\overline{x}, \overline{w}) : p \in P\} \cup \{z_i - w_i : i \in [|\overline{z}|]\}.$$

Then P' is in $\overline{z}$-normal form: each $p^\star$ contains no $\overline{z}$-variable, and each $z_i - w_i$ is of the form $-w_i + z_i$.

The equisatisfiability condition is immediate. Indeed, after fixing an assignment $\alpha \in \{0,1\}^{|\overline{z}|}$, the equations $z_i - w_i = 0$ force $w_i = \alpha_i$ for every i. Hence

$$p^\star(\overline{x}, \overline{w}) = 0 \text{ for all } p \in P$$

is equivalent to

$$p(\overline{x}, \alpha) = 0 \text{ for all } p \in P.$$

Thus

$$\exists \overline{x} \in \{0,1\}^{|\overline{x}|}\ P(\overline{x}, \alpha) = 0 \quad \iff \quad \exists \overline{x} \in \{0,1\}^{|\overline{x}|} \exists \overline{w} \in \mathbb{F}^{|\overline{w}|}\ P'(\overline{x}, \overline{w}, \alpha) = 0. \tag{7.1}$$

It remains to derive every original polynomial $p \in P$ from P' by a small roABP-IPS$_{\text{LIN}}$ derivation. This is done using routine variable-by-variable substitution via a telescoping identity, as follows.

Since $p^\star$ is already an axiom of P', it suffices to derive $p - p^\star$ from the equations $z_i - w_i$. For a monomial

$$m(\overline{x}, \overline{z}) = c\overline{x}^\gamma \prod_{j=1}^{k} z_{i_j}$$

of p, where repetitions among the indices i_j are allowed, write

$$m^\star(\overline{x}, \overline{w}) = c\overline{x}^\gamma \prod_{j=1}^{k} w_{i_j}.$$

Then the telescoping identity is

$$\prod_{j=1}^{k} z_{i_j} - \prod_{j=1}^{k} w_{i_j} = \sum_{j=1}^{k} \left(\prod_{h<j} z_{i_h} \right) \left(\prod_{g>j} w_{i_g} \right) (z_{i_j} - w_{i_j}).$$

Multiplying by $c\overline{x}^\gamma$ gives a derivation of $m - m^\star$ from the equations $z_i - w_i$. Summing over all monomials of p gives a derivation of $p - p^\star$, and hence of p, from P'.

Summing over all monomials of p gives a derivation of $p - p^\star$, and hence of p, from P'. For each equation $z_i - w_i$, the corresponding coefficient is a sum of at most sparsity$(p) \cdot \deg(p)$ monomials from the telescopic identity. A monomial has width-1 roABP in any variable ordering, and a sum of T monomials has roABP width at most T by taking the sum of the width-1 roABPs. Hence every coefficient in this derivation has roABP width at most sparsity$(p) \cdot \deg(p)$, which is $\mathsf{poly}(s+n)$ by assumption. Therefore each $p \in P$ has an roABP-IPS$_{\text{LIN}}$ derivation from P' of width $\mathsf{poly}(s+n)$ in any variable ordering.

Monotone case. Assume that P is monotone in $\overline{z}$. Thus every polynomial $p \in P$ is of the form

$$p(\overline{x}, \overline{z}) = \left(\prod_{i \in I_p} z_i \right) p'(\overline{x}),$$

where $p'(\overline{x}) \in \mathbb{F}[\overline{x}]$. If $I_p = \emptyset$, we keep $p = p'(\overline{x})$ unchanged. Otherwise, for every $i \in I_p$ introduce a fresh auxiliary variable $w_{p,i}$, and replace p by the polynomials

$$p'(\overline{x}) + \sum_{i \in I_p} w_{p,i}, \qquad z_i w_{p,i} \quad (i \in I_p).$$

Let $P'(\overline{x}, \overline{w}, \overline{z})$ be the union of these polynomials over all $p \in P$. Then P' is in monotone $\overline{z}$-normal form.

The equisatisfiability condition is checked after fixing an arbitrary assignment $\alpha \in \{0,1\}^{|\overline{z}|}$ to the $\overline{z}$-variables. Consider one polynomial

$$p(\overline{x},\overline{z}) = \left(\prod_{i\in I_p} z_i\right) p'(\overline{x})$$

and the normalised equations associated with it:

$$p'(\overline{x}) + \sum_{i\in I_p} w_{p,i} = 0, \qquad z_i w_{p,i} = 0 \quad (i \in I_p).$$

We are trying to prove the following: for every fixed Boolean assignment α to the $\overline{z}$-variables, $P(\overline{x},\overline{\alpha}) = 0$ is satisfiable in Boolean $\overline{x}$ if and only if $P'(\overline{x},\overline{w},\overline{\alpha}) = 0$ is satisfiable in Boolean $\overline{x}$ and field-valued $\overline{w}$. So fix a Boolean assignment $\overline{\alpha}$ to $\overline{x}$. After substituting $\overline{z} = \overline{\alpha}$, the original equation becomes

$$\left(\prod_{i\in I_p} \alpha_i\right) p'(\overline{x}) = 0.$$

If all $\alpha_i = 1$ for $i \in I_p$, this is just $p'(\overline{x}) = 0$. On the normalised side, the equations $\alpha_i w_{p,i} = 0$ force all $w_{p,i} = 0$, and hence the first normalised equation also becomes $p'(\overline{x}) = 0$.

If some $\alpha_{i_0} = 0$, then the original equation becomes $0 \cdot p'(\overline{x}) = 0$, and hence imposes no condition on $\overline{x}$. The normalised equations impose no condition on $\overline{x}$ either: for any assignment to $\overline{x}$, set $w_{p,i_0} = -p'(\overline{x})$ and set all other $w_{p,i}$'s to 0. Then $p'(\overline{x}) + \sum_{i\in I_p} w_{p,i} = 0$, and all equations $\alpha_i w_{p,i} = 0$ hold. Thus, for each fixed α, the normalised equations associated with p impose exactly the same condition on $\overline{x}$ as the original equation $p(\overline{x},\alpha) = 0$.

Since the auxiliary variables used for different polynomials $p \in P$ are disjoint, the same argument applies independently to all polynomials in P. Therefore Equation (7.1) holds.

It remains to derive each original polynomial p from P' by a small roABP-IPS$_{\text{LIN}}$ derivation. For $p = (\prod_{i\in I_p} z_i)p'(\overline{x})$, we have the identity

$$\left(\prod_{i\in I_p} z_i\right) p'(\overline{x}) = \left(\prod_{i\in I_p} z_i\right)\left(p'(\overline{x}) + \sum_{i\in I_p} w_{p,i}\right) - \sum_{i\in I_p}\left(\prod_{j\in I_p\setminus\{i\}} z_j\right)(z_i w_{p,i}). \tag{7.2}$$

Thus p is obtained as an roABP-IPS$_{\text{LIN}}$ linear combination of the polynomials introduced for p whose coefficient polynomials are monomials in the $\overline{z}$-variables. Hence these coefficients have width-1 roABPs in every variable ordering. In particular, the width of the derivation, measured by the coefficient roABPs, is constant. Therefore every $p \in P$ has an roABP-IPS$_{\text{LIN}}$ derivation from P' of width $\mathsf{poly}(s+n)$ (in fact, of constant width).

Finally, the total sparse size of P' is polynomial in $s+n$: for each p, the construction introduces one polynomial of sparsity at most $\text{sparsity}(p) + |I_p|$ and $|I_p|$ monomial equations, and $|I_p| \le \deg(p) \le (s+n)^{O(1)}$. □

The following corollary sets how to apply Lemma 6. The argument is a simple use of composition of proofs, while making proofs are not blowing up in roABP width when the order of variables is kept $\overline{x}\cup\overline{w} < \overline{z}\cup\overline{y}$.

Corollary 8 (Normal form application)**.** *Let $P_0(\overline{x},\overline{z})$ and $P_1(\overline{y},\overline{z})$ be sets of polynomial equations, where $\overline{x},\overline{y},\overline{z}$ are pairwise disjoint sets of variables. Assume that P_0 is a set of polynomials of both degree and sparsity $\mathsf{poly}(|\overline{x}+\overline{z}|)$.*[1] *Suppose that $P_0 \cup P_1$ has an* roABP-IPS$_{\text{LIN}}$ *refutation of width w*

[1]For example, P_0 may be the standard arithmetization of a k-CNF, for a constant k.

in a variable ordering in which the $\overline{x}$-variables precede all other variables. Then, there is a set of polynomial equalities $P_0'(\overline{x}', \overline{z})$, where

- *$\overline{x}', \overline{y}, \overline{z}$ are pairwise disjoint sets of variables.*
- *P_0' is in $\overline{z}$-normal form and has sparsity $\mathsf{poly}(|\overline{x} + \overline{z}|)$.*
- *For every Boolean assignment α to the $\overline{z}$-variables,*

  $$P_0(\overline{x}, \overline{\alpha}) \text{ is satisfiable} \iff P_0'(\overline{x}', \overline{\alpha}) \text{ is satisfiable.}$$

  Consequently, the pairs (P_0, P_1) and (P_0', P_1) define the same interpolation problem, and hence have the same interpolant.
- *$P_0' \cup P_1$ has an* roABP-IPS$_{\text{LIN}}$ *refutation of width $\mathsf{poly}(w, |P_0|)$ in a variable ordering in which the $\overline{x}'$-variables precede both the $\overline{y}$- and $\overline{z}$-variables.*

Moreover, if P_0 is monotone in the $\overline{z}$-variables then P_0' is in monotone $\overline{z}$-normal form.

Proof: Apply Lemma 6 to $P_0(\overline{x}, \overline{z})$. Let $P_0'(\overline{x}, \overline{w}, \overline{z})$ be the resulting set of polynomial equations, and write $\overline{x}' := (\overline{x}, \overline{w})$. By Lemma 6, the set P_0' is in (monotone) $\overline{z}$-normal form, has sparse size polynomial in $|\overline{x}| + |\overline{z}|$, and for every assignment to the common variables $\overline{z}$, the left-hand side of the interpolation problem (i.e., P_0 and P_0') has the same satisfiability status for P_0 and for P_0'. Consequently, the pairs (P_0, P_1) and (P_0', P_1) define the same interpolation problem, and hence have the same interpolant.

It remains to show that the refutation of $P_0 \cup P_1$ can be converted into a refutation of $P_0' \cup P_1$ with only a polynomial increase in width and in an ordering in which $\overline{x}'$-variables come before all other variables. Let

$$\sum_j R_j(\overline{x}, \overline{z}, \overline{y})\, P_{0,j}(\overline{x}, \overline{z}) + \sum_t L_t(\overline{x}, \overline{z}, \overline{y})\, P_{1,t}(\overline{y}, \overline{z}) = 1 \tag{8.1}$$

be the assumed roABP-IPS$_{\text{LIN}}$ refutation of $P_0 \cup P_1$ of width w, where the $P_{0,j}$ enumerate the polynomials of P_0 and the $P_{1,t}$ enumerate the polynomials of P_1.

By Lemma 6, for every j there is an roABP-IPS$_{\text{LIN}}$ derivation of $P_{0,j}$ from P_0' of width $\mathsf{poly}(|\overline{x}| + |\overline{z}|)$. Thus we may write

$$P_{0,j}(\overline{x}, \overline{z}) = \sum_i Q_{i,j}(\overline{x}, \overline{w}, \overline{z})\, P_{0,i}'(\overline{x}, \overline{w}, \overline{z}), \tag{8.2}$$

where each coefficient $Q_{i,j}$ has an roABP of width $\mathsf{poly}(|\overline{x}| + |\overline{z}|)$ in any variable ordering.

We now substitute the identities (8.2) into the refutation (8.1). This gives

$$\begin{aligned} 1 &= \sum_j R_j(\overline{x}, \overline{z}, \overline{y}) \left(\sum_i Q_{i,j}(\overline{x}, \overline{w}, \overline{z})\, P_{0,i}'(\overline{x}, \overline{w}, \overline{z}) \right) + \sum_t L_t(\overline{x}, \overline{z}, \overline{y})\, P_{1,t}(\overline{y}, \overline{z}) \\ &= \sum_i \left(\sum_j R_j(\overline{x}, \overline{z}, \overline{y})\, Q_{i,j}(\overline{x}, \overline{w}, \overline{z}) \right) P_{0,i}'(\overline{x}, \overline{w}, \overline{z}) + \sum_t L_t(\overline{x}, \overline{z}, \overline{y})\, P_{1,t}(\overline{y}, \overline{z}). \end{aligned} \tag{8.3}$$

This is a linear IPS refutation of $P_0' \cup P_1$.

It remains only to check the width and variable ordering. Extend the original variable ordering so that all variables in $\overline{x}' = (\overline{x}, \overline{w})$ precede the variables in $\overline{y}, \overline{z}$. The original coefficients R_j and

L_t can be viewed as polynomials in the larger variable set by ignoring the new $\overline{w}$-variables; this does not increase their roABP width. The coefficients $Q_{i,j}$ have polynomial-width roABPs in this same extended ordering by Lemma 6. Products of two roABPs in the same variable ordering can be computed (while preserving the varaible ordering) by taking tensor products of the layers, so the width multiplies[2]. Sums of polynomially many such roABPs can be computed by taking their direct sum (i.e., by putting them side by side in parallel), so the width increases by at most a polynomial factor. Hence each coefficient

$$\sum_j R_j Q_{i,j}$$

in (8.3) has roABP width $\mathsf{poly}(w, |P_0|)$, and the coefficients L_t retain width at most w in a variable order in which $\overline{x}'$-variables precede both the $\overline{y}$- and $\overline{z}$-variables. Therefore, $P_0' \cup P_1$ has an roABP-IPS$_{\text{LIN}}$ refutation of width $\mathsf{poly}(w, |P_0|)$ in this variable order. □

We are now ready to prove the feasible interpolation theorem.

Theorem 9 (Feasible interpolation for roABP-IPS$_{\text{LIN}}$)**.** *Let $P_0(\overline{x}, \overline{z})$ and $P_1(\overline{y}, \overline{z})$ be sets of polynomial equalities, where $\overline{x}, \overline{y}$ and $\overline{z}$ are pairwise disjoint variables and P_0 is a set of polynomials of both degree and sparsity $\mathsf{poly}(|\overline{x}| + |\overline{z}|)$. Assume that P_0 is in $\overline{z}$-normal form. Suppose that $P_0 \cup P_1$ has a* roABP-IPS$_{\text{LIN}}$ *refutation of width w in a variable ordering where $\overline{x}$ variables precede all other variables. Then there is a span program of size $\mathsf{poly}(w \cdot |P_0(\overline{x}, \overline{z})|)$ that computes an interpolant for P_0 and P_1. Moreover, if all the polynomials in P_0 are in monotone $\overline{z}$-normal form, then the span program is monotone.*

Proof: By Corollary 8 we can assume without loss of generality that P_0 is in a (monotone) normal form and let

$$\sum_{p \in P_0} a_p p + \sum_{q \in P_1} b_q q = 1 \tag{9.1}$$

be a refutation of $P_0 \cup P_1$.

Note that, by the assumption that for every $p \in P_0$ the polynomial a_p is computable by a width w read-once oblivious algebraic branching program, it can be written in the form

$$a_p = \sum_{i \in [w]} a_{p,i}(\overline{x}) \cdot r_i(\overline{y}, \overline{z}), \tag{9.2}$$

for some polynomials $a_{p,i}$ in the $\overline{x}$ variables only and some polynomials r_i in the $\overline{y}$ and $\overline{z}$ variables only. Indeed, consider the layer of the roABP immediately after the last $\overline{x}$-variable in the given variable ordering (recall that the i-th layer in a roABP contains only edges labeled with univariate polynomials in the i-th variable in the variable linear ordering). Since the roABP has width w, this layer contains at most w nodes. For each node $i \in [w]$, let $a_{p,i}$ denote the polynomial computed from the source to node i, and let r_i denote the polynomial computed from node i to the sink. Then $a_{p,i}$ depends only on the $\overline{x}$-variables, while r_i depends only on the remaining variables, namely $\overline{y}, \overline{z}$.

[2]We use the standard closure properties of roABPs in a fixed variable order. Suppose P and Q are computed in the same variable order $x_1 < \cdots < x_n$, with matrix presentations $P = (A_1 \cdots A_n)_{1,1}$ and $Q = (B_1 \cdots B_n)_{1,1}$, where A_i and B_i are the matrices for the x_i-layer. For each i, define the x_i-layer of the product roABP to be $C_i = A_i \otimes B_i$. Then, $C_1 \cdots C_n = (A_1 \otimes B_1) \cdots (A_n \otimes B_n) = (A_1 \cdots A_n) \otimes (B_1 \cdots B_n)$, where the rightmost equality is by basic tensor calculus. Hence the output entry of the product roABP equals $(A_1 \cdots A_n)_{1,1}(B_1 \cdots B_n)_{1,1} = PQ$. If the widths of the two roABPs are w_P and w_Q, then the matrices C_i have dimension at most $w_P w_Q$, so the width is at most $w_P w_Q$.

Summing over all nodes in this layer gives the polynomial a_p. If the layer has fewer than w nodes, we pad the decomposition by setting the missing $a_{p,i}$'s equal to 0.

We now construct the span program I as follows:

1. For any polynomial $p \in P_0$ of the form $p' + z_j p''$ add to the span program the entries
$$\big(z_j, a_{p,i}(p' + p'')\big) \text{ and } (1 - z_j, a_{p,i}p'), \text{ for all } i \in [w].$$
Thus, if $z_j = 1$ the span program picks the vector $a_{p,i}(p' + z_j p'') = a_{p,i}(p' + p'')$; while if $z_j = 0$ the span program picks the vector $a_{p,i}(p' + z_j p'') = a_{p,i}p'$.

2. For any polynomial $p \in P_0$ of the form p' (i.e., no $\overline{z}$-variables appear in the polynomial) add to the span program the entries
$$(1, a_{p,i}p'), \text{ for all } i \in [w].$$

3. The target vector is the constant polynomial 1.

Note that if P_0 is monotone in the $\overline{z}$-variables then this span program is monotone, because in this case $p \in P_0$ is of the form $p' + z_j p''$ with $p' = 0$, hence $(1 - z_j, a_{p,i}p') = (1 - z_j, 0)$ can be discarded from the span program.

We claim that the constructed span program computes an interpolant for P_0 and P_1, namely, the conditions in (2.2) are met: if $I(\overline{\alpha}) = 1$ then $P_0(\overline{x}, \overline{\alpha})$ is unsatisfiable; and if $I(\overline{\alpha}) = 0$ then $P_1(\overline{y}, \overline{\alpha})$ is unsatisfiable.

We consider the two cases separately. Suppose first that the span program I outputs 1 on an input $\overline{\alpha}$. The program outputs 1, when the constant polynomial 1 is in the linear span of the vectors selected by $\overline{\alpha}$. Note that by Item 1 above these vectors correspond to polynomials in $P_0(\overline{x}, \overline{\alpha})$ multiplied by the polynomials $a_{p,i}$. Thus expressing 1 as a linear span of these polynomials constitutes a Nullstellensatz refutation of $P_0(\overline{x}, \alpha)$, and so $P_0(\overline{x}, \alpha)$ is unsatisfiable.

Otherwise, suppose that the span program I outputs 0 on an input $\overline{\alpha}$. Suppose towards a contradiction that $P_1(\overline{y}, \overline{\alpha})$ is satisfiable and let $\overline{\beta}$ be an assignment to the $\overline{y}$ variables that satisfies $P_1(\overline{y}, \overline{\alpha})$ (so P_1 vanishes under $\overline{\alpha}, \overline{\beta}$). Applying both $\overline{\alpha}$ and $\overline{\beta}$ to the given refutation (9.1) it simplifies to a refutation of $P_0(\overline{x}, \overline{\alpha})$:
$$\sum_{p \in P_0} a_p(\overline{x}, \overline{\beta}, \overline{\alpha}) p(\overline{x}, \overline{\alpha}) = 1.$$
Moreover, by Equation (9.2), for every $p \in P_0$ the polynomial $a_p(\overline{x}, \overline{\beta}, \overline{\alpha})$ is a linear combination of the polynomials $a_{p,i}(\overline{x})$, for $i \in [w]$:
$$a_p(\overline{x}, \overline{\beta}, \overline{\alpha}) = \sum_{i \in [w]} a_{p,i}(\overline{x}) \cdot r_i(\overline{\beta}, \overline{\alpha}). \tag{9.3}$$
The polynomials $a_p(\overline{x}, \overline{\beta}, \overline{\alpha}) p(\overline{x}, \overline{\alpha})$ are thus linear combinations of the vectors selected by $\overline{\alpha}$, and the span program should have output 1, concluding the proof.

Note that Equation (9.3) is the crucial equality allowing us to conclude the theorem: for this equality to hold we must be able to write the left hand side as a *linear combination* of $a_{p,i}(\overline{x})$, for $i \in [x]$. This is only possible because the variables $\overline{y}, \overline{z}$ come *after* the $\overline{x}$-variables in the variable ordering, so that the $r_i(\overline{\beta}, \overline{\alpha})$ become constant. □

Remark 10. The statement of Theorem 9 in the monotone case can be applied to a slightly more general class of formulas. This is done by observing that the monotone case in Lemma 6 does not depend on sparsity in the sense that the monotone feasible interpolation in Theorem 9 is applicable for every monotone P_0 so that the polynomials in it are computable by low-width roABPs.

3.1 Application: Lower Bounds for Fixed Variable Ordering

In this section we use the monotone feasible interpolation theorem of the previous section to obtain lower bounds against roABP-IPS$_{\text{LIN}}$. For this we will require the following lower bounds for monotone span programs.

Definition 11 (GEN$_n$ function hard for monotone span programs [PR18])**.** For a positive integer n, let $T \subseteq [n]^3$ be a set of triples. We say that T *generates* a point $w \in [n]$ if either $w = 1$, or there exists a triple $(u, v, w) \in T$ such that T generates both u and v. The function GEN$_n$ takes as input the characteristic vector of a set $T \subseteq [n]^3$ and outputs 1 if and only if T generates the point n.

Equivalently, starting from the initially generated point 1, repeatedly add a point w whenever there is a triple $(u, v, w) \in T$ such that both u and v have already been generated. Then GEN$_n(T) = 1$ if and only if this process eventually generates n.

Theorem 12 ([RPRC16, PR18])**.** *The monotone function* GEN$_n$ *is computable in polynomial time. Furthermore, over any field any monotone span program computing f requires size $2^{n^{\Omega(1)}}$.*

The following lemma gives a way to transform monotone functions computable by small circuits into monotone propositional formulas describing the yes and no instances of the function respectively. A similar construction appeared in [RPRC16] starting from a monotone circuit. The lemma below is a non-uniform analogue of the general construction of an unsatisfiable split formula from a disjoint NP pair [Kra97]. We include the proof for completeness, and to note that the circuit itself need not be monotone as long as the function is.

Lemma 13. *For any monotone function $f \in \mathsf{NP/poly} \cap \mathsf{coNP/poly}$ there are polynomial sized 3-CNF formulas $\phi_0(\overline{x}, \overline{z})$ and $\phi_1(\overline{y}, \overline{z})$ so that*

- *$f(\overline{\alpha}) = 0$ if and only if $\phi_0(\overline{x}, \overline{\alpha})$ is satisfiable;*
- *$f(\overline{\alpha}) = 1$ if and only if $\phi_1(\overline{y}, \overline{\alpha})$ is satisfiable;*
- *the $\overline{z}$-variables appear in $\phi_0(\overline{x}, \overline{z})$ only negatively and in $\phi_1(\overline{y}, \overline{z})$ only positively.*

Proof: We construct first the formula ϕ_0. For this let $C(\overline{z}, \overline{w})$ be a polynomial size nondeterministic circuit computing the negation of f, where $\overline{z}$ are the input variables and $\overline{w}$ the nondeterministic variables; its existence is guaranteed by the fact that $f \in \mathsf{coNP/poly}$. To define ϕ_0 introduce for each gate g of the circuit C a new variable x_g, and construct ϕ_0 as follows:

- if g is a $\circ$-gate, where $\circ \in \{\wedge, \vee\}$ with children g_0 and g_1, include in ϕ_0 the clauses encoding the following formula $x_g \leftrightarrow (x_{g_0} \circ x_{g_1})$;
- if g is a $\neg$-gate with a child g_0, include in ϕ_0 the clauses encoding the formula $x_g \leftrightarrow \neg x_{g_0}$;
- for the output gate g, include in ϕ_0 the clause x_g;
- finally for each input gate g labeled with a variable z_i, include in ϕ_0 the clause $\neg z_i \vee x_g$.

The constructed CNF satisfies the last condition. We will prove the first condition. Suppose first that $f(\alpha) = 0$. Then there is some assignment β to the $\overline{w}$ variables so that $C(\alpha, \beta) = 1$. We can satisfy $\phi_0(\overline{x}, \alpha)$ by assigning the input gate variables x_g according to the assignments α and β and the intermediate variables so that they respect the required conditions. Suppose then that $\phi(\overline{x}, \alpha)$ is satisfiable, and let γ be the assignment to the gate variables corresponding to gates labeled with $\overline{z}$

variables. By rewiring we may assume that there is a single input gate labeled with any $\overline{z}$ variables, and thus $|\gamma| = |\alpha|$. As γ is a part of a satisfying assignment of $\phi_0(\overline{x}, \alpha)$ we have that $f(\gamma) = 0$. Also $\alpha \leq \gamma$ as ϕ_0 contains the clauses $\neg z_i \vee x_g$. Hence $f(\alpha) \leq f(\gamma) = 0$ as f is monotone.

The formula $\phi_1(\overline{y}, \overline{z})$ is constructed similarly using the nondeterministic circuit for f itself with the modification that for input gate g labeled with variable z_i, we include in ϕ_1 the clause $z_i \vee \neg y_g$. □

With Theorem 9 and the existing monotone span program lower bounds of [RPRC16, PR18] we are ready to prove the main theorem of this section.

Theorem 14. *There is an unsatisfiable* 3*-CNF* $\phi_0(\overline{x}, \overline{y}) \wedge \phi_1(\overline{y}, \overline{z})$ *that requires* roABP-IPS$_{\text{LIN}}$ *refutations of width* $2^{N^{\Omega(1)}}$ *in any variable ordering where* $\overline{x}$ *variables precede all the other variables, where* N *is the number of variables in the formula.*

Proof: Consider the function GEN_n, and note that it is indeed computable in polynomial time by iteratively closing the initially generated set $\{1\}$ under the triples in T. In particular, it belongs to $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Thus, let $\phi_0(\overline{x}, \overline{z})$ and $\phi_1(\overline{y}, \overline{z})$ be the formulas encoding the no and yes instances of the function GEN_n, respectively, given by Lemma 13. The obtained 3-CNF has $\mathsf{poly}(n)$ variables and $\mathsf{poly}(n)$ clauses, and all the $\overline{z}$ variables appear only negatively in $\phi_0(\overline{x}, \overline{z})$.

Suppose there is an roABP-IPS$_{\text{LIN}}$ refutation of $\phi_0(\overline{x}, \overline{z}) \wedge \phi_1(\overline{y}, \overline{z})$ of width w in some variable ordering where $\overline{z}$ variables precede the $\overline{y}$ and $\overline{z}$ variables. Note that the standard polynomial translation of $\phi_0(\overline{x}, \overline{z})$ is in monotone $\overline{z}$-normal form. Thus, by Theorem 9 there is a size $\mathsf{poly}(n \cdot w)$ monotone span program computing an interpolant I for $\phi_0(\overline{x}, \overline{z}) \wedge \phi_1(\overline{y}, \overline{z})$.

To finish the proof it suffices to note that the computed interpolant is exactly the function GEN_n. To see this note that the interpolant I satisfies

$$I(\overline{\alpha}) = \begin{cases} 0, \text{ if } \phi_0(\overline{x}, \overline{\alpha}) \text{ is satisfiable;} \\ 1, \text{ if } \phi_1(\overline{y}, \overline{\alpha}) \text{ is satisfiable;} \end{cases}$$

and so by the construction of $\phi_0(\overline{x}, \overline{z})$ and $\phi_1(\overline{y}, \overline{z})$ it satisfies

$$I(\overline{\alpha}) = \begin{cases} 0, \text{ if } \text{GEN}_n(\overline{\alpha}) = 0 \\ 1, \text{ if } \text{GEN}_n(\overline{\alpha}) = 1. \end{cases}$$

Thus the function has size $\mathsf{poly}(n \cdot w)$ monotone span programs, and so, by Theorem 12 we have that $w \geq 2^{n^{\Omega(1)}}$. As $N = \mathsf{poly}(n)$ it follows that $w \geq 2^{N^{\Omega(1)}}$. □

4 Lower Bounds for Arbitrary Variable Ordering

In this section, we build a single family of unsatisfiable CNF formulas, such that each CNF requires large roABP-IPS$_{\text{LIN}}$ refutations in every variable ordering. A natural strategy that has been employed in such settings is to use the hard instance defined in the previous section, which we showed is hard for a fixed variable ordering, and then *lift* that instance to obtain a lower bound for all variable orderings.

Background on lifting in algebraic proof complexity. The first roABP-IPS$_{\text{LIN}}$ lower bound for all variable order was proved in [FSTW21] for an algebraic instance built in the subset sum. In

their work, they first prove that the following unsatisfiable algebraic formula (a polynomial equation) is hard to refute by an roABP-IPS$_{\text{LIN}}$ refutation in a fixed variable ordering, namely $\overline{u} < \overline{v}$.

$$f(\overline{u}, \overline{v}) : \sum_{i \in [n]} u_i v_i - \beta = 0,$$

where $\beta \notin \{0, 1, \ldots, n\}$. Next, they construct another hard instance, using two new sets of variables, namely $\overline{z} = \{z_{i,j}\}$ for $i, j \in [2n]$ and $\{x_i\}$ for $i \in [2n]$. Specifically, they create the following instance:

$$g(\overline{z}, \overline{x}) : \sum_{i<j} z_{i,j} x_i x_j \ - \beta = 0,$$

where $\beta \notin \{0, 1, \ldots, \binom{n}{2}\}$. The main idea is that for any order σ of the $\overline{x}$ variables, there exists a Boolean assignment to the $\overline{z}$ variables and a projection of $\overline{x}$ variables to $\overline{u}$ and $\overline{v}$ variables that recovers the original hard instance. Hence, if the original instance can be obtained as a projection of the new instance, then the lower bound proved for the original instance carries over to the new instance. They call $g(\overline{z}, \overline{x})$ as the *lifted* instance of $f(\overline{u}, \overline{v})$. This lifting idea is used by other IPS lower bounds such as [GHT22, HLT24, EGLT26].

In all the above works, the input instances were purely algebraic and not a CNF formula.

It is worth noting that the question of proving lower bounds for all variable orders naturally occurs in the context of OBDD-based proof systems, which are propositional proof systems operating with ordered binary decision diagrams (introduced by Atserias, Kolaitis and Vardi [AKV04]). This is addressed in the works of Krajíček [Kra08], Segerlind [Seg07], and a recent work of [BIK+21]. While there are some similarities between our proof strategy and that of [Kra08], the arguments are different.

4.1 The Hard Instance

Here we describe our construction of the hard instance. We start by fixing some notation.

Let us call the hard 3-CNF formula from the previous section $\phi = \phi_0 \wedge \phi_1$. From this formula, we will build a new hard instance, which we call Ψ. Let us denote the variables of the original instance ϕ by $\overline{x} = \{x_1, x_2, \ldots, x_N\}$ and assume that it is hard in the variable ordering $x_1 < \ldots < x_N$. Let the clauses in ϕ be denoted by $C_1, C_2, \ldots, C_M$. A specific clause C in this formula is a conjunction of at most three literals, i.e., $C = (\ell_1 \vee \ell_2 \vee \ell_3)$. Moreover, for a literal ℓ in C, let $\text{var}(\ell)$ denote the variable corresponding to the literal ℓ and $\text{sg}(\ell)$ denote the sign of that literal.

The hard instance Ψ we construct has two sets of variables denoted $\overline{u}$ and $\overline{v}$, where $|\overline{v}| = N$ and $|\overline{u}| = O(M \cdot N^3)$. Our construction guarantees the following properties.

- $\Psi(\overline{u}, \overline{v}) = \Phi_1(\overline{u}, \overline{v}) \wedge \Phi_2(\overline{u})$, where Φ_1 is a CNF over the variables $\overline{u}, \overline{v}$ and Φ_2 is a CNF over $\overline{u}$ variables.
- Additionally, $\Psi(\overline{u}, \overline{v})$ is unsatisfiable. Specifically, for every Boolean assignment $\overline{\alpha}$ to $\overline{u}$ exactly one of the following two holds:
  - either $\Phi_2(\overline{\alpha})$ is not satisfied;
  - or $\Phi_2(\overline{\alpha})$ is satisfied and there exists a relabelling of $\overline{v}$ variables by $\overline{x}$ variables such that $\Phi_1(\overline{\alpha}, \overline{v})$ reduces to ϕ, the hard instance for the fixed variable order (hence, $\Phi_1(\overline{\alpha}, \overline{v})$ is unsatisfiable).

Overall, let ϕ be the hard instance from Section 3.1, which is hard in the variable order σ. The construction ensures that Ψ is unsatisfiable and that, for every ordering of the variables $\overline{u}, \overline{v}$, there is a Boolean assignment to the $\overline{u}$-variables and a relabelling of the $\overline{v}$-variables by the $\overline{x}$-variables that produces ϕ in the order σ. This yields a lower bound for all variable orders. Based on this idea, we construct Ψ in two stages.

Stage 1: local realisation of the clauses. This stage should be viewed as *a lift* of the original instance. We define the variables $\overline{u}, \overline{v}$ and the first part Φ_1 of the lifted formula. There are N variables $\overline{v} = \{v_1, \ldots, v_N\}$, which will serve as renamed copies of the original variables $\overline{x} = \{x_1, \ldots, x_N\}$. The role of the $\overline{u}$-variables is to specify, clause by clause, how a clause of the original formula ϕ is realised on the $\overline{v}$-variables.

For every clause

$$C = (\ell_1 \vee \ell_2 \vee \ell_3)$$

of ϕ, and for every triple of distinct indices $i, j, k \in [N]$, we introduce a selector variable $u_{C,i,j,k}$. Hence, there are $O(M \cdot N^3)$-many $\overline{u}$ variables. Intuitively, setting $u_{C,i,j,k} = 1$ means that, in the lifted copy of the clause C, the variables underlying the literals ℓ_1, ℓ_2, ℓ_3 are represented by v_i, v_j, v_k, respectively. We enforce this local copy of C by adding the clause

$$\neg u_{C,i,j,k} \vee v_i^{\mathrm{sg}(\ell_1)} \vee v_j^{\mathrm{sg}(\ell_2)} \vee v_k^{\mathrm{sg}(\ell_3)}. \tag{14.1}$$

Thus, whenever the selector $u_{C,i,j,k}$ is set to 1, the formula contains the clause obtained from C by replacing $\mathrm{var}(\ell_1), \mathrm{var}(\ell_2), \mathrm{var}(\ell_3)$ with v_i, v_j, v_k, respectively, while preserving the signs of the literals.

Definition 15. The formula Φ_1 is the conjunction of all clauses of the form (14.1), for all clauses C in ϕ and all triples $i \neq j \neq k \in [N]$.

At this stage, different clauses may choose their triples independently, so the same original variable might be represented by different $\overline{v}$-variables in different clauses, or two different original variables might be represented by the same $\overline{v}$-variable. The role of the second stage will be to add a consistency formula Φ_2 over the $\overline{u}$-variables, forcing the selected local realisations to come from one global relabelling of the original variables by the $\overline{v}$-variables.

Stage 2: enforcing global consistency. The purpose of the second part Φ_2 of the lifted formula is to ensure that the local choices made by the selector variables $\overline{u}$ in Stage 1 are compatible with a single global relabelling of the original variables $\overline{x}$ by the new variables $\overline{v}$.

Recall that, for every clause

$$C = (\ell_1 \vee \ell_2 \vee \ell_3)$$

and every triple of distinct indices $i, j, k \in [N]$, the variable $u_{C,i,j,k}$ says that the variables underlying the literals ℓ_1, ℓ_2, ℓ_3 are realised by v_i, v_j, v_k, respectively. Thus, if $u_{C,i,j,k} = 1$, it defines the partial map

$$\pi_{C,i,j,k} = \{\, v_i \mapsto \mathrm{var}(\ell_1),\ v_j \mapsto \mathrm{var}(\ell_2),\ v_k \mapsto \mathrm{var}(\ell_3) \,\}.$$

We say that two selector variables $u_{C,i,j,k}$ and $u_{D,a,b,c}$ are compatible if the union of their corresponding partial maps is still a one-to-one partial map from $\overline{v}$ to $\overline{x}$. Equivalently, they are incompatible if either the same $\overline{v}$-variable is mapped to two different $\overline{x}$-variables, or two different $\overline{v}$-variables are mapped to the same $\overline{x}$-variable.

The formula Φ_2 enforces the following two conditions.

1. For every clause C of φ, exactly one of the variables $u_{C,i,j,k}$ is set to 1. We add the clauses

$$\bigvee_{i,j,k\in[N] \text{ distinct}} u_{C,i,j,k}$$

and, for every two distinct triples $(i,j,k) \neq (i',j',k')$,

$$\neg u_{C,i,j,k} \vee \neg u_{C,i',j',k'}.$$

2. We rule out incompatible local choices. Namely, for every two clauses C, D of ϕ, and every two triples of distinct indices (i,j,k) and (a,b,c), if the two partial maps

$$\pi_{C,i,j,k} \qquad \text{and} \qquad \pi_{D,a,b,c}$$

are incompatible, then we add the clause

$$\neg u_{C,i,j,k} \vee \neg u_{D,a,b,c}.$$

Definition 16. Let Φ_2 be the conjunction of all clauses introduced above in Items 1 and 2

Thus, Φ_2 has size polynomial in N and M: the number of selector variables is at most MN^3, and the consistency clauses are obtained by checking pairs of selector variables.

By construction, an assignment to the $\overline{u}$-variables satisfies Φ_2 if and only if, for every clause C, it chooses exactly one local realisation of C, and all chosen local realisations are mutually compatible. Equivalently, the selected partial maps combine into one global one-to-one partial map from the relevant $\overline{v}$-variables to the original variables $\overline{x}$. Thus, whenever Φ_2 is satisfied, the clauses selected in Φ_1 form a relabelled copy of the original formula ϕ. More formally, we have the following.

Proposition 17. *The CNF $\Psi(\overline{u},\overline{v}) = \Phi_1(\overline{u},\overline{v}) \wedge \Phi_2(\overline{u})$ is unsatisfiable.*

Proof: Let $\overline{\alpha}$ be an arbitrary Boolean assignment to the $\overline{u}$-variables. If $\overline{\alpha}$ does not satisfy Φ_2, then clearly $\overline{\alpha}$ has no extension to the $\overline{v}$-variables that satisfies Ψ. Thus assume that $\overline{\alpha}$ satisfies Φ_2.

By the definition of Φ_2, for every clause C of ϕ there is a unique triple (i,j,k) such that $\overline{\alpha}(u_{C,i,j,k}) = 1$, and all the partial maps selected in this way are mutually compatible. In other words, these partial maps combine into a single one-to-one partial map from the relevant $\overline{v}$-variables to the original variables $\overline{x}$. Equivalently, there is a single one-to-one partial map π from the $\overline{v}$-variables to the original variables $\overline{x}$ such that, for every clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ of ϕ, if $u_{C,i,j,k}$ is the selector that is set to 1, then $\pi(v_i) = \mathrm{var}(\ell_1)$, $\pi(v_j) = \mathrm{var}(\ell_2)$, and $\pi(v_k) = \mathrm{var}(\ell_3)$.

Now restrict Φ_1 by the assignment $\overline{\alpha}$. Let $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ be a clause of ϕ, and let (i,j,k) be the unique triple selected for C. All clauses of Φ_1 corresponding to unselected triples are satisfied, since their selector variable is set to 0. The selected triple leaves the clause

$$v_i^{\mathrm{sg}(\ell_1)} \vee v_j^{\mathrm{sg}(\ell_2)} \vee v_k^{\mathrm{sg}(\ell_3)}.$$

By the global compatibility of the selected partial maps, these remaining clauses are exactly a relabelled copy of the clauses of ϕ, with signs preserved. Since ϕ is unsatisfiable, no assignment to the $\overline{v}$-variables satisfies $\Phi_1(\overline{\alpha},\overline{v})$. Hence no extension of $\overline{\alpha}$ satisfies Ψ. Since $\overline{\alpha}$ was arbitrary, Ψ is unsatisfiable. □

4.2 Lower Bound against roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ for All Variable Orders

We are now ready to prove the main lower bounds result.

Theorem 18 (Main lower bound)**.** *The CNF* $\Psi(\overline{u},\overline{v}) = \Phi_1(\overline{u},\overline{v}) \wedge \Phi_2(\overline{u})$ *requires* roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ *refutations of width* $2^{N^{\Omega(1)}}$ *in any variable ordering.*

Proof: Let $<$ be an arbitrary variable ordering of the variables of Ψ, and suppose that Ψ has an roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation of width w in this order. Recall that N is the number of $\overline{v}$-variables, which is the number of variables in the original hard instance $\phi = \phi_0 \wedge \phi_1$ denoted $x_1, \ldots, x_N$. Consider the order induced by $<$ on the $\overline{v}$-variables, and write it as

$$v_{i_1} < v_{i_2} < \cdots < v_{i_N}.$$

We define a restriction depending only on this induced order. First relabel v_{i_r} as x_r, for every $r \in [N]$. Next assign the $\overline{u}$-variables as follows. For every clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ of ϕ (in the $\overline{x}$-variables) set to 1 the unique $u_{C,i,j,k}$ for which v_i, v_j, v_k are the $\overline{v}$-variables relabelled as $\mathrm{var}(\ell_1), \mathrm{var}(\ell_2), \mathrm{var}(\ell_3)$, respectively. Set all other variables associated with C to 0.

This assignment to the $\overline{u}$-variables satisfies Φ_2: it selects exactly one triple for each clause, and all selected triples arise from the single global relabelling $v_{i_r} \mapsto x_r$. Therefore all compatibility clauses in Φ_2 are satisfied. Under the same restriction, every clause of Φ_1 whose selector $u_{C,i,j,k}$ is assigned 0 is satisfied by the literal $\neg u_{C,i,j,k}$. For each original clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ of ϕ, exactly one variables $u_{C,i,j,k}$ is assigned 1, and the corresponding clause (14.1) of Φ_1 reduces to

$$v_i^{\mathrm{sg}(\ell_1)} \vee v_j^{\mathrm{sg}(\ell_2)} \vee v_k^{\mathrm{sg}(\ell_3)}.$$

By the definition of the assignment to the $\overline{u}$-variables and the relabelling $v_{i_r} \mapsto x_r$, this reduced clause is precisely the original clause C. Hence the restriction of Φ_1 is exactly ϕ. Thus the restricted formula is precisely ϕ, with its variables ordered as $x_1 < \cdots < x_N$.

Applying this restriction to the assumed roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation of Ψ gives an roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation of ϕ in the order $x_1 < \cdots < x_N$. The width does not increase under substituting constants for the $\overline{u}$-variables and relabelling the remaining $\overline{v}$-variables: in the roABP, this only substitutes constants into edge labels and renames variables, while preserving the layered computation.

Hence ϕ has an roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation of width at most w in the hard order $x_1 < \cdots < x_N$. By Theorem 14, we get $w \geq 2^{N^{\Omega(1)}}$. Since the original order $<$ was arbitrary, the same lower bound holds for every variable ordering. □

5 Simulations and Upper Bounds

Here we provide information on the strength of roABP-$\mathrm{IPS}_{\mathrm{LIN}}$. We show a simulation of a (dynamic) tree-like PC refutation systems in which each proof-line is written as a roABP in a fixed global order. Then we provide polynomial-size upper bounds for Tseitin and the functional pigeonhole principle.

5.1 roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ Simulates Tree-Like Polynomial Calculus Length

In this section we show that roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ simulates the tree-like Polynomial Calculus as long as the initial polynomial equalities can be written as small-width roABPs. The simulation shows that for simple axioms (computable by constant width roABPs) a length ℓ Polynomial Calculus refutation can be transformed into an roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation of width $O(\ell)$. This result holds

irrespective of the way the intermediate polynomials are represented in the tree-like Polynomial Calculus refutation as long as they are derived using the two local rules of the system.

Proposition 19. *Let $p_1 = 0, \dots, p_m = 0$ be an unsatisfiable set of polynomial equalities, and suppose each polynomial p_i is computable by width w roABP in some fixed variable ordering.*

If there is a tree-like Polynomial Calculus refutation of $p_1 = 0, \dots, p_m = 0$ of length l, then there is an roABP-IPS$_{\text{LIN}}$ *refutation of $p_1 = 0, \dots, p_m = 0$ of width at most $\ell \cdot w$.*

Proof: Let $q_1, \dots, q_\ell$ be a tree-like Polynomial Calculus refutation of $p_1 = 0, \dots, p_m = 0$. Denote by s_i the size of the sub-tree of the derivation tree rooted at q_i. We prove by induction that q_i has an roABP-IPS$_{\text{LIN}}$ derivation of width at most $s_i \cdot w$ for each $i \in [\ell]$.

If $q_i = p_j$ for some $j \in [m]$, the claim is clear, since p_j can be computed by width w roABP. If $q_i = xq_j$ for some $j < i$ and some variable x, the claim is also clear, since if q_j has an roABP-IPS$_{\text{LIN}}$ derivation of width w', then q_i has also, since multiplying the derivation by a single variable does not increase its width. Finally, suppose that $q_i = aq_j + bq_k$ for some $j, k < i$ and $a, b \in \mathbb{F}$. By induction assumption, q_j and q_k have roABP-IPS$_{\text{LIN}}$ derivations of width at most $s_j \cdot w$ and $s_k \cdot w$ respectively. Hence q_i has an roABP-IPS$_{\text{LIN}}$ derivation of width at most $(s_j + s_k) \cdot w$, which is less than $s_i \cdot w$, since the Polynomial Calculus refutation is tree-like. □

5.2 Upper Bounds for Tseitin Formulas and the Functional Pigeonhole Principle

Lastly, we discuss upper bounds for the proof system roABP-IPS$_{\text{LIN}}$. We will observe that Tseitin formulas have polynomial width roABP-IPS$_{\text{LIN}}$ refutations in any variable ordering and show that functional pigeonhole principle has polynomial width roABP-IPS$_{\text{LIN}}$ refutations in a particular ordering of the involved variables.

In algebraic proof complexity, the restriction to Boolean domain is often imposed by including the Boolean axioms $x^2 - x$ for all variables as is done also by . This forces the variables to take values from the set $\{0, 1\}$. This is not however the only possible choice; another common choice is to represent the Boolean values by $\{\pm 1\}$ by including the axioms $x^2 - 1$ for all variables. Although there is a simple linear transformation between these different representation certain complexity measures are sensitive to the choice of representation. For an example the sparsity, or monomial-size, measure can blow-up in this transformation; Tseitin formulas have polynomial sparsity Nullstellensatz refutations in the $\{\pm 1\}$ basis [Gri98], but over $\{0, 1\}$ require exponential sparsity even in the stronger Sum-of-Squares proof system [AH19].

The roABP-IPS$_{\text{LIN}}$ refutations on the other hand are not sensitive for this choice of basis: the linear transformation is applied individually for each variable and thus does not increase the width of the underlying roABPs. This observation shows that the lower bounds presented in this paper in fact give lower bounds over any representation of the Boolean values. Conversely, any upper bounds translate between the different Boolean bases.

Recall the Tseitin formulas $\text{Ts}(G, \nu)$ [Tse68]. A *charge* on an undirected graph G is mapping ν that assigns a value $\nu(u) \in \{0, 1\}$ to each vertex $u \in V(G)$. The charge is odd, if the number of vertices with charge 1 is odd. Tseitin's formulas are defined over set of variables x_e for each edge $e \in E(V)$ and consists of the parity formulas

$$\bigoplus_{\substack{e \in E(V): \\ u \in e}} x_e = \nu(u)$$

for each $u \in V(G)$. The set of formulas is unsatisfiable for any graph with an odd charge. If the graph is of constant degree, the parity formulas can be written as small-width CNFs. Observe,

however, that the formulas can always be written with small-width roABPs in any variable ordering. This observation allows us to state the following proposition.

Proposition 20. *For any graph G with an odd charge ν the Tseitin formula $\mathrm{Ts}(G, \nu)$ has a width $\mathsf{poly}(n)$ roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation in any variable ordering.*

Proof: Grigoriev has shown that Tseitin formulas have polynomial sparsity Nullstellensatz refutation over the $\{\pm 1\}$ basis. The sparse polynomials in the refutation can be written as low-width roABPs in any variable ordering, one path for each monomial. After applying the linear transformation $x \mapsto (1-x)/2$ to each variable, we are left with a low-width roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutation of the Tseitin formula in the $\{0, 1\}$ basis. □

Secondly, we will show that the functional pigeonhole principle can be refuted in small width roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ in a particular variable ordering. Recall that the propositional encoding FPHP_n^{n+1} of the functional pigeonhole principle consists of the following clauses:

- $\bigvee_{k\in[n]} x_{ik}$ for all $i \in [n+1]$; (pigeon axioms)
- $\neg x_{ik} \vee \neg x_{jk}$ for all distinct $i, j \in [n+1]$ and all $k \in [n]$; (pigeonhole axioms)
- $\neg x_{ik} \vee \neg x_{i\ell}$ for all $i \in [n+1]$ and all distinct $k, \ell \in [n]$. (functionality axioms)

We consider the variable ordering where variables are ordered by the pigeonhole: $x_{ik} < x_{j\ell}$, whenever $k < \ell$. The argument given below is similar to the upper bound given for constant-depth IPS in [HLT24], which in turn borrowed from the upper bounds in [GH03] and [RT08a].

Proposition 21. FPHP_n^{n+1} *has width* $\mathsf{poly}(n)$ *roABP-*$\mathrm{IPS}_{\mathrm{LIN}}$ *refutations in the variable ordering by the pigeonhole.*

Proof: First note that the pigeon axioms can be written equivalently in the form $1 - \sum_{k\in[n]} x_{ik}$ and that this equivalence is easily derivable from the standard translation of the pigeon axioms and the functionality axioms as

$$1 - \sum_{k\in[n]} x_{ik} = \prod_{k\in[n]} (1 - x_{ik}) - \sum_{k<\ell} x_{ik}x_{i\ell} \prod_{\ell'>\ell} (1 - x_{i\ell'}).$$

The weights on the functionality axioms above have a constant width read-once oblivious algebraic branching programs. The sum of all these representations of the pigeon axioms equals

$$n + 1 - \sum_{i\in[n+1]} \sum_{k\in[n]} x_{ik}$$

Define the following short-hand: denote by y_k the polynomial $\sum_{i\in[n+1]} x_{ik}$. This polynomial is Boolean valued as $y_k^2 - y_k$ can be easily derivable from the hole axioms and the Boolean axioms. Now write the sum of all pigeon axioms in the form

$$n + 1 - \sum_{k\in[n]} y_k.$$

This is an unsatisfiable subset sum instance in the $\overline{y}$ variables; its unique multilinear refutation is symmetric and can thus be written as a linear combination of elementary symmetric polynomials ([FSTW21] gives an explicit expression). As elementary symmetric polynomials have small width

read-once oblivious algebraic branching programs, this subset sum instance has small width roABP-$\mathrm{IPS}_{\mathrm{LIN}}$ refutations in the $\overline{y}$ variables in the variable ordering $y_1 < \ldots < y_n$.

Finally apply the substitution $y_k \mapsto \sum_{i \in [n+1]} x_{ik}$ to the refutation of the subset sum instance to obtain a refutation of the functional pigeonhole principle. The width of the refutation stays small since the chosen variable ordering interleaves with the ordering on the $\overline{y}$ variables: $x_{ik} < x_{j\ell}$ for any $k < \ell$. □

References

[AF22] Robert Andrews and Michael A. Forbes. Ideals, determinants, and straightening: proving and using lower bounds for polynomial ideals. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 389–402, New York, NY, USA, 2022. Association for Computing Machinery. 2, 8

[AH19] Albert Atserias and Tuomas Hakoniemi. Size-degree trade-offs for sums-of-squares and Positivstellensatz proofs. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 24:1–24:20, 2019. 26

[AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In *CP*, pages 77–91, 2004. 8, 22

[And25] Robert Andrews. Algebraic pseudorandomness in VNC^0. In Srikanth Srinivasan, editor, *40th Computational Complexity Conference, CCC 2025, Toronto, Canada, August 5-8, 2025*, LIPIcs, pages 15:1–15:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. 5

[BDG+99] Maria Luisa Bonet, Carlos Domingo, Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi. Non-automatizability of bounded-depth Frege proofs. In *Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999)*, pages 15–23. IEEE Computer Soc., Los Alamitos, CA, 1999. 9

[BHLS26] Amik Raj Behera, Magnus Rahbek Dalgaard Hansen, Nutan Limaye, and Srikanth Srinivasan. Separation results for constant-depth and multilinear ideal proof systems. *CoRR*, abs/2601.06299, 2026. 8

[BIK+96] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Math. Soc. (3)*, 73(1):1–26, 1996. 2, 3, 12

[BIK+21] Sam Buss, Dmitry Itsykson, Alexander Knop, Artur Riazanov, and Dmitry Sokolov. Lower bounds on OBDD proofs with several orders. *ACM Transactions on Computational Logic*, 22(4):26:1–26:30, 2021. 8, 22

[BLRS25] Amik Raj Behera, Nutan Limaye, Varun Ramanathan, and Srikanth Srinivasan. New bounds for the ideal proof system in positive characteristic. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, LIPIcs, pages 22:1–22:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. 8

[BPR97] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997. 8

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for Frege systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000. 9

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 174–183, New York, 1996. ACM. 3, 12

[CGMS25] Prerona Chatterjee, Utsab Ghosal, Partha Mukhopadhyay, and Amit Sinhababu. IPS lower bounds for formulas and sum of roabps. In C. Aiswarya, Ruta Mehta, and Subhajit Roy, editors, *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2025, BITS Pilani, K K Birla Goa Campus, India, December 17-19, 2025*, LIPIcs, pages 22:1–22:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. 3, 8

[CP14] Siu Man Chan and Aaron Potechin. Tight bounds for monotone switching networks via fourier analysis. *Theory Comput.*, 10:389–419, 2014. 6

[dRGR22] Susanna F. de Rezende, Mika Göös, and Robert Robere. Guest column: Proofs, circuits, and communication. *SIGACT News*, 53(1):59–82, 2022. 7

[EGLT26] Tal Elbaz, Nashlen Govindasamy, Jiaqi Lu, and Iddo Tzameret. Lower bounds against the ideal proof system in finite fields. In *Proceedings of the 58th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2026. 3, 5, 8, 9, 22

[FGGR22] Noah Fleming, Mika Göös, Stefan Grosser, and Robert Robere. On semi-algebraic proofs and algorithms. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, Berkeley, CA, USA, January 31 - February 3, 2022*, LIPIcs, pages 69:1–69:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 13

[For14] Michael A. Forbes. *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. PhD thesis, Massachusetts Institute of Technology, June 2014. 11

[For24] Michael A Forbes. Low-depth algebraic circuit lower bounds over any field. In *39th Computational Complexity Conference (CCC 2024)*, pages 31–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. 5

[FSTW21] Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. Proof complexity lower bounds from algebraic circuit complexity. *Theory Comput.*, 17:1–88, 2021. 2, 3, 5, 7, 8, 9, 12, 21, 27

[GH03] Dima Grigoriev and Edward A. Hirsch. Algebraic proof systems over formulas. *Theoretical Computer Science*, 303(1):83–102, 2003. 3, 27

[GHT22] Nashlen Govindasamy, Tuomas Hakoniemi, and Iddo Tzameret. Simple hard instances for low-depth algebraic proofs. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 188–199. IEEE, 2022. 8, 9, 22

[GP18] Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *J. ACM*, 65(6):37:1–37:59, 2018. 2, 3, 9, 12

[Gri98] Dima Grigoriev. Tseitin's tautologies and lower bounds for Nullstellensatz proofs. In *IEEE Symposium on Foundations of Computer Science*, pages 648–652, 1998. 26

[HLT24] Tuomas Hakoniemi, Nutan Limaye, and Iddo Tzameret. Functional lower bounds in algebraic proofs: Symmetry, lifting, and barriers. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 1396–1404, New York, NY, USA, 2024. Association for Computing Machinery. Full version in ECCC https://eccc.weizmann.ac.il/report/2024/079/. 3, 5, 8, 9, 22, 27

[IR22] Dmitry Itsykson and Artur Riazanov. Automating OBDD proofs is NP-hard. *Electronic Colloquium on Computational Complexity (ECCC)*, 29:TR22–046, 2022. 8

[IRS17] Dmitry Itsykson, Artur Riazanov, and Dmitry Sokolov. On OBDD-based algorithms and proof systems that dynamically change the order of variables. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, pages 43:1–43:14, 2017. 8

[KP89] Jan Krajícek and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *J. Symb. Log.*, 54(3):1063–1079, 1989. 8

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997. 8, 20

[Kra08] Jan Krajícek. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *J. Symb. Log.*, 73(1):227–237, 2008. 8, 22

[KW93] M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 102–111, 1993. 5, 13

[LST26] Jiaqi Lu, Rahul Santhanam, and Iddo Tzameret. $\mathrm{AC}^0[p]$-frege cannot efficiently prove that constant-depth algebraic circuit lower bounds are hard. In *17th Innovations in Theoretical Computer Science Conference, ITCS 2026, Milan, Italy, 2026*, 2026. arXiv preprint arXiv:2509.16824. 2

[LTW18] Fu Li, Iddo Tzameret, and Zhengyu Wang. Characterizing propositional proofs as noncommutative formulas. *SIAM Journal on Computing*, 47(4):1424–1462, 2018. 2

[Pit97] Toniann Pitassi. Algebraic propositional proof systems. In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996, Princeton, New Jersey, USA, January 14-17, 1996*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 215–244, Providence, RI, 1997. American Mathematical Society. 3

[Pit98] Toniann Pitassi. Unsolvable systems of equations and proof complexity. In *Proceedings of the International Congress of Mathematicians, Berlin, 1998*, volume Extra Vol. ICM Berlin 1998, Vol. III, pages 451–460, 1998. 3

[PR18] Toniann Pitassi and Robert Robere. Lifting nullstellensatz to monotone span programs over any field. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 1207–1219, New York, NY, USA, 2018. Association for Computing Machinery. 6, 7, 8, 10, 20, 21

[PS96] Pavel Pudlák and Jiří Sgall. Algebraic models of computation and interpolation for algebraic proof systems. In Paul Beame and Samuel Buss, editors, *Proof Complexity and Feasible Arithmetic*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 279–296. American Mathematical Society, 1996. 3, 8, 13

[PT16] Toniann Pitassi and Iddo Tzameret. Algebraic proof complexity: progress, frontiers and challenges. *ACM SIGLOG News*, 3(3):21–43, aug 2016. 2

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, Sept. 1997. 8

[Raz95a] Alexander A. Razborov. Bounded arithmetic and lower bounds in Boolean complexity. In *Feasible mathematics, II (Ithaca, NY, 1992)*, volume 13 of *Progr. Comput. Sci. Appl. Logic*, pages 344–386. Birkhäuser Boston, Boston, MA, 1995. 8

[Raz95b] Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izv. Ross. Akad. Nauk Ser. Mat.*, 59(1):201–224, 1995. 8

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Comb.*, 19(3):403–435, 1999. 5

[RPRC16] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential Lower Bounds for Monotone Span Programs. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 406–415. IEEE Computer Society, 2016. 6, 7, 8, 10, 20, 21

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005. 4

[RT08a] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. 3, 27

[RT08b] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *Computational Complexity*, 17(3):407–457, 2008. 3

[Seg07] Nathan Segerlind. Nearly-exponential size lower bounds for symbolic quantifier elimination algorithms and obdd-based proofs of unsatisfiability. *CoRR*, abs/cs/0701054, 2007. 8, 22

[Seg08] Nathan Segerlind. On the relative efficiency of resolution-like proofs and ordered binary decision diagram proofs. In *23rd Annual IEEE Conference on Computational Complexity (CCC 2008)*, pages 100–111, 2008. 8

[Sok26] Dmitry Sokolov, 2026. Personal communication. 10

[ST25] Rahul Santhanam and Iddo Tzameret. Iterated lower bound formulas: A diagonalization-based approach to proof complexity. *SIAM Journal on Computing, Special Issue of STOC'21*, 0(0):STOC21–313–STOC21–349, 2025. 2

[Tse68] Grigori Tseitin. *On the complexity of derivations in propositional calculus*. Studies in constructive mathematics and mathematical logic Part II. Consultants Bureau, New-York-London, 1968. 26

[Tza11] Iddo Tzameret. Algebraic proofs over noncommutative formulas. *Inf. Comput.*, 209(10):1269–1292, 2011. 2