Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 64 (2020)

Automating Algebraic Proof Systems is NP-Hard

Susanna F. de Rezende
Lund University

Mika Göös[†]
EPFL

Jakob Nordström
University of Copenhagen & Lund University

Toniann Pitassi
Columbia University & IAS

Robert Robere
McGill University

Dmitry Sokolov
St. Petersburg State University & PDMI RAS

June 10, 2022

Abstract

We show that algebraic proofs are hard to find: Given an unsatisfiable CNF formula F, it is NP-hard to find a refutation of F in the Nullstellensatz or Polynomial Calculus proof systems in time polynomial in the size of the shortest such refutation. Our work extends, and gives a simplified proof of, the recent breakthrough of Atserias and Müller (JACM 2020) that established an analogous result for Resolution.

[†]Part of the work done while at Institute for Advanced Study and Stanford.

ISSN 1433-8092

Contents

1 Introduction

Automatability. A proof system S is (polynomial-time) *automatable* [BPR97] if there is an algorithm that takes as input an unsatisfiable CNF formula F and outputs an S-refutation of F in time polynomial in the size of the shortest S-refutation of F (plus the size of F). Intuitively, automatability addresses the proof search problem: How hard is it to *find* a proof? Automatability (or lack thereof) for well-studied proof systems is a central question for automated theorem proving and SAT solving.

For example, state-of-the-art SAT solvers using conflict-driven clause learning (CDCL) [BS97, MS99, MMZ+01] are based on the most basic propositional proof system, Resolution (R for short). This means that running a CDCL solver (without preprocessing) on an unsatisfiable formula F produces a Resolution refutation of F [BKS04]. Thus, non-automatability of Resolution (studied in a long line of work [Iwa97, ABMP01, AB04, AR08, MPW19, AM20]) implies that any SAT solver based on Resolution will require superpolynomial time even on formulas that are easy, that is, admit a polynomial-size refutation.

Algebraic proof systems. In this paper, we study the automatability of *algebraic* proof systems. We show that it is NP-hard to automate either of the following standard systems:

- (NS) Nullstellensatz [BIK+94],
- (PC) Polynomial Calculus [CEI96, ABRW02].

We still leave open the question of automatability for the *semi-algebraic* proof systems[1]

- (SA) Sherali–Adams [SA94],
- (SoS) Sum-of-Squares [Sho87, Par00, Las01].

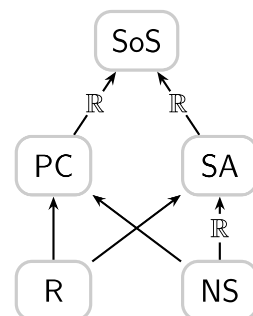

Figure 1: An arrow $\mathsf{A} \longrightarrow \mathsf{B}$ means B efficiently simulates A (only over $\mathbb{R}$ where indicated).

1.1 Our results

For the algebraic proof systems NS and PC, our main result shows that it is NP-hard to approximate the minimum refutation size up to a factor of 2^{n^ϵ} for some constant $\epsilon > 0$. We defer the standard definitions of the algebraic proof systems to Section 8. Our result holds regardless of definitional details such as which underlying field (real numbers, finite fields) we choose, or whether we allow twin variables (separate formal variables for negated literals).

Theorem 1.1 (Main result)**.** *There is a polynomial-time algorithm $\mathcal{A}$ that on input an n-variate 3-CNF formula F outputs a CNF formula $\mathcal{A}(F)$ such that for any system $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$:*

– *If F is* satisfiable*, then $\mathcal{A}(F)$ admits an S-refutation of size at most $n^{O(1)}$.*
– *If F is* unsatisfiable*, then $\mathcal{A}(F)$ requires S-refutations of size at least $2^{n^{\Omega(1)}}$.*

A direct corollary of Theorem 1.1 is that we can rule out automatability in polynomial, quasi-polynomial and subexponential time under corresponding hardness assumptions. To state this more precisely, let QP be the class of problems that can be solved in time $\exp(\log^{O(1)} n)$ and SUBEXP those that can be solved in time $\exp(n^{o(1)})$.

[1] An earlier version of this paper claimed that our non-automatability result for NS and PC also generalises to SA. However, we have since found a flaw in the proposed proof; namely, Lemma 8.3 (Section 8.2) fails for SA.

Corollary 1.2. *For any system* $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$*:*

- S *is not automatable in polynomial time unless* $\mathsf{NP} \subseteq \mathsf{P}$.
- S *is not automatable in quasi-polynomial time unless* $\mathsf{NP} \subseteq \mathsf{QP}$.
- S *is not automatable in subexponential time unless* $\mathsf{NP} \subseteq \mathsf{SUBEXP}$.

We emphasize that our theorem handles all of the proof systems simultaneously. That is, there is one common polynomial-time constructible formula $\mathcal{A}(F)$ that is either easy for all the proof systems, or hard for all of them. This means that proof search is hard for R and NS even if we are allowed to search for proofs in the stronger system PC.

Previously, Galesi and Lauria [GL10a], building on [AR08], proved that NS and PC are not polynomial-time automatable unless the fixed parameter hierarchy collapses. Our result upgrades this to an optimal hardness assumption, namely $\mathsf{P} \neq \mathsf{NP}$. As for upper bounds, the fastest-known search algorithms for PC, SA, and SoS run in exponential time $\exp(\tilde{O}(\sqrt{n \log s}))$, where s is the proof size and the $\tilde{O}$-notation hides $\mathrm{poly}(\log n)$ factors. All these algorithms are based on general size–degree tradeoffs [CEI96, IPS99, PS12, AH19].

1.2 Techniques

Our proof builds on the recent breakthrough of Atserias and Müller [AM20] that showed that automating Resolution is NP-hard. Namely, they proved Theorem 1.1 for $\mathsf{S} = \mathsf{R}$. We give a simpler proof of their theorem that generalizes better, handling more systems simultaneously. The key new ingredient in our approach is a reduction from the *pigeonhole principle* to prove the lower bound in case F is unsatisfiable. As a further simplification, we show how standard size–width tradeoffs can be used to eliminate the "relativization/lifting" step in the Atserias and Müller proof by tweaking their construction of $\mathcal{A}(F)$ slightly. See Section 2 for a detailed overview of our techniques.

1.3 Other related work

Degree-automatability. Many algebraic proof systems possess a (weaker) form of automatability known as *degree*-automatability (as opposed to *size*-automatability), which enables proofs of *low degree* to be found efficiently. More specifically, proofs of degree d can be found in time $n^{O(d)}$ for n-variate formulas: for NS and SA this can be achieved by solving an LP; for PC see [CEI96]; for SoS (under technical assumptions that cover the case of CNF formulas) see [O'D17, RW17].

Degree (or size) automatability yields a meta-approach for search problems. Namely, when the *existence* of a solution can be proven via a low-degree (or small size) proof then degree (or size) automatability can be applied to generate an efficient algorithm for finding a solution. This proofs-as-algorithms approach has led to many beautiful and sometimes surprising new approximation algorithms for a variety of optimization and average-case parameter estimation problems. Examples include dictionary learning [BKS15], tensor decomposition [MSS16], learning mixtures of Gaussians [KSS18], and constraint satisfaction problems [HKP+17, OS19]. What makes these algebraic proof systems special is that they hit a sweet spot, possessing strong power but also being weak enough to admit nontrivial proof search. For example, SA (resp. SoS) gives a standard way of tightening LP (resp. SDP) relaxations of boolean LPs in order to improve performance. Another example of their power is that SA and SoS are able to prove many useful (anti-)concentration inequalities in constant degree [OZ13]. For a comprehensive introduction to the interplay between algebraic proofs and algorithms, see the monograph [FKP19].

Size–degree tradeoffs. Degree-automatability has an interesting consequence for the way non-automatability results are proved: The formula $\mathcal{A}(F)$ we construct admits a short refutation when F is satisfiable, but every such refutation must require large degree (otherwise degree-automatability would allow us to find them quickly). Such formulas—admitting short proofs but none of small degree—were known to exist for R [BG01] and PC [GL10b] (and for NS this is implicit in [BCIP02]). No such CNF formulas have yet been exhibited for SoS, although progress towards this goal was recently made in [Pot20].

Other proof systems. For standard textbook-style proof systems (Frege and Extended Frege) *weak automatability* [AB04]—that is, being polynomially simulated by an automatable proof system—is equivalent to possessing *feasible interpolation.* More specifically, for any proof system that is closed under restrictions, weak automatability implies feasible interpolation [BPR00], and for sufficiently strong proof systems (that admit short proofs of their soundness), the converse holds [Pud03]. Under cryptographic assumptions, Frege, Extended Frege, and bounded-depth Frege systems are known to not have feasible interpolation and therefore are not even weakly automatable [KP98, BPR97, BDG^{+}04].

By contrast, for weak systems that seemingly cannot reason about their own soundness (R, NS, PC, SA, SoS), deciding whether they are automatable has proven more challenging. Until the recent breakthrough by Atserias and Müller [AM20], even the automatability of Resolution was unresolved. In an earlier important paper, Alekhnovich and Razborov [AR08] ruled out automatability of Resolution under the assumption that the fixed parameter hierarchy is proper. However, the best upper bound on the time complexity remained exponential, and it remained open for a long time whether or not this upper bound could be improved until this question was finally resolved in [AM20]. Following in the wake of Atserias and Müller, non-automatability results have also been shown for other weak proof systems such as regular and ordered Resolution [Bel20] (building on a preliminary version of this paper), k-DNF Resolution [Gar20], and cutting planes [GKMP20].

2 Proof overview

Let us now give an overview of how we modify [AM20] to construct our new proof. In this section:

- (§2.1) We recall the the definition of the Resolution proof system.
- (§2.2) We outline a simpler proof of the Atserias–Müller theorem (Theorem 1.1 for Resolution). The details appear in Part I.
- (§2.3) We outline why our simplified proof generalizes, with some additional work, to the setting of algebraic proof systems. The details appear in Part II.

Readers who only care about our simplified proof of Atserias–Müller are in luck: We have organized the paper so that the initial Sections 3–7 present the simplified proof in a self-contained fashion. In particular, no knowledge of algebraic proof systems is required there.

2.1 Resolution basics

Fix an unsatisfiable CNF formula F over variables $x_1, \ldots, x_n$. We call the clauses of F *axioms* and often think of them as sets of literals (x_i or $\bar{x}_i$, where bar denotes negation). A Resolution refutation $\mathcal{P}$ of F, or R-refutation for short, is a sequence of clauses $\mathcal{P} = (C_1, \ldots, C_s)$ ending in the empty clause $C_s = \emptyset$ such that each C_i is either (i) an axiom of F; or (ii) derived from clauses C_j, $C_{j'}$, where $j, j' < i$, using one of the following rules:

- *Resolution rule:* $C_i = (C_j \setminus \{x_k\}) \cup (C_{j'} \setminus \{\bar{x}_k\})$ where $x_k \in C_j$ and $\bar{x}_k \in C_{j'}$.
- *Weakening rule:* $C_i \supseteq C_j$.

The *size* of the refutation is $\|\mathcal{P}\| := s$. The Resolution size complexity of F, denoted $\mathsf{R}(F)$, is the size of a smallest Resolution refutation of F. Another important complexity measure of a refutation $\mathcal{P}$ is its *width* $\mathrm{w}(\mathcal{P})$ defined as the maximum width $|C|$ of any of its clauses $C \in \mathcal{P}$. Define also the *width complexity* $\mathrm{w}_{\mathsf{R}}(F)$ of a formula F as the least width of a Resolution refutation of F.

For visualization purposes, a refutation $\mathcal{P}$ can be thought of as a *directed acyclic graph (dag)*, also called the *refutation dag*: Introduce a node v_i for every clause C_i, and include a directed edge (j, i) if C_j is used to derive C_i. The final clause C_s becomes a *root node* (no parent), while the axioms are *leaves* (no children). A refutation is *tree-like* if this graph is a tree (note that the same clause can label several different nodes), and otherwise it is *dag-like*.

2.2 A simpler proof for the non-automatability of Resolution

Suppose we are given an n-variate 3-CNF formula F as input. The algorithm $\mathcal{A}$ that Atserias and Müller devised computes in two steps: In the first step, the algorithm constructs a "refutation formula" denoted by $\mathrm{Ref}(F)$. In the second step, this formula is "lifted" to produce $\mathrm{Lift}(\mathrm{Ref}(F))$, which is then output by $\mathcal{A}$. We explain these two steps in detail.

Step 1: A block-width lower bound

The refutation formula $\mathrm{Ref}(F)$ (defined precisely in Section 3.1) intuitively states

$$\mathrm{Ref}(F) \quad \equiv \quad \text{“}F \textit{ admits a short dag-like Resolution refutation.”}$$

For now, it suffices to say that the variables of $\mathrm{Ref}(F)$ come partitioned into some number of *blocks*. For a clause C over the variables of $\mathrm{Ref}(F)$, we define its *block-width* $\mathrm{bw}(C)$ as the number of distinct blocks that C *touches*, that is, from which it contains a variable. For a Resolution refutation $\mathcal{P}$ (resp. formula F), we define its *block-width* $\mathrm{bw}(\mathcal{P})$ (resp. $\mathrm{bw}(F)$) as the maximum block-width of its clauses. Finally, for a formula F, we define its *block-width complexity* $\mathrm{bw}_{\mathsf{R}}(F)$ as the minimum block-width of a Resolution refutation of F.

The key property of $\mathrm{Ref}(F)$ is that its block-width complexity depends drastically on F's satisfiability.

Lemma 2.1 (Atserias–Müller)**.** *There is a polynomial-time algorithm that on input an n-variate 3-CNF formula F outputs a block-width-$O(1)$ CNF formula* $\mathrm{Ref}(F)$ *such that*

(i) If F is satisfiable*, then* $\mathrm{Ref}(F)$ *admits a size-$n^{O(1)}$ block-width-$O(1)$ resolution refutation.*
(ii) If F is unsatisfiable*, then* $\mathrm{Ref}(F)$ *requires resolution refutations of block-width $n^{\Omega(1)}$.*

We present the upper bound *(i)* in Section 7 for completeness (and also because our definition of $\mathrm{Ref}(F)$ differs slightly from that of Atserias and Müller). Our main simplification is for the block-width lower bound *(ii)*.

Simplifying part *(ii)* of Lemma 2.1. Atserias and Müller originally proved the lower bound *(ii)* by a direct ad-hoc adversary argument. This was the most involved step in their proof. Our proof is by a mere *reduction* from the usual *pigeonhole principle*. We define in Section 3.3 a convenient, somewhat non-standard encoding of the principle, sometimes called the *retraction weak pigeonhole principle* [Jeř07, PT19]. This encoding, denoted rPHP_m, is an $O(\log m)$-width CNF formula that claims there exists an efficiently invertible injection, encoded in binary, from $2m$ pigeons to m holes. Our reduction in Section 5 translates, with modest loss, width complexity lower bounds for rPHP_{n^2} into block-width complexity lower bounds for $\mathrm{Ref}(F)$.

Lemma 2.2. *For any n-variate unsatisfiable formula F we have* $\mathrm{bw}_{\mathsf{R}}(\mathrm{Ref}(F)) \geq \tilde{\Omega}(\mathrm{w}_{\mathsf{R}}(\mathrm{rPHP}_{n^2})/n)$.

Our simplified proof of *(ii)* is then concluded by invoking known width lower bounds for pigeonhole principles. Indeed, standard techniques [PT19, Proposition 3.4] show that

$$\mathrm{w}_{\mathsf{R}}(\mathrm{rPHP}_m) \;\geq\; \Omega(m).$$

This lower bound and Lemma 2.2 imply that $\mathrm{bw}_{\mathsf{R}}(\mathrm{Ref}(F)) \geq \tilde{\Omega}(n)$, which proves *(ii)*.

Step 2: From block-width to size

The goal of the second step is to transform the block-width gap in Lemma 2.1 into a size gap. There are two alternative approaches to achieve this.

Lifting. This technique was used by Atserias and Müller, although they called it *relativization* after [DR03]; see also [Gar19]. Lifting techniques have produced a plethora of applications in proof complexity; recent examples include [HN12, GP18, dRNV16, GGKS18, GKRS19, dRMN+19, GKMP20]. The general strategy is this: We start with a formula F that is hard in some weak sense (for us, block-width). Then we *compose* (or *lift*) the formula with a carefully chosen *gadget*—usually, each variable of F is replaced with a copy of the gadget—to produce a formula $\mathrm{Lift}(F)$, which we then show is hard in a strong sense (for us, Resolution size).

Tradeoff. The famous size–width tradeoff of Ben-Sasson and Wigderson [BW01] states that any n-variate low-width formula F that has high width complexity, namely $\mathrm{w}_{\mathsf{R}}(F) \gg \sqrt{n}$, also has exponentially large size complexity. Atserias and Müller's original proof did not use the tradeoff result, as their encoding of $\mathrm{Ref}(R)$ did not admit a high enough width complexity. We observe that by defining $\mathrm{Ref}(R)$ in a succinct enough way (technically speaking, using *binary* rather than *unary* encoding to represent numbers), the width complexity $\mathrm{w}_{\mathsf{R}}(\mathrm{Ref}(F))$ ends up in a regime where the tradeoff result applies, which gives us an exponential size lower bound without the need for any gadget composition.

We think both approaches have merits. Lifting is the more robust technique: it is more widely applicable than the tradeoff, as it applies even if the starting formula F has only a small polynomial (block-)width complexity. However, given that we have been able (somewhat unintentionally) to optimize the encoding of $\mathrm{Ref}(R)$, the tradeoff approach can give us a shorter proof.

We will opt to focus on the lifting approach in this paper. We do, however, outline briefly how the alternative tradeoff approach can be carried out in Section 6.4.

Block lifting. We prove in Section 6 a lifting lemma whose notable feature is that it is *block-aware*: the gadgets corresponding to a single block will *share* some input variables. This allows us to lift block-width (rather than width) to Resolution size. The lemma is simple to prove via random restrictions: a proof is implicit in Atserias–Müller, and an even stronger version (lifting to Cutting Planes size) was proved in [GKMP20]. We formulate the lemma here for completeness, and also in order to generalize it to algebraic systems later (see Section 2.3).

Lemma 2.3 (Block lifting)**.** *There is a polynomial-time algorithm that on input a block-width-$O(1)$ CNF formula F outputs a CNF formula* $\mathrm{Lift}(F)$ *such that*

$$2^{\Omega(\mathrm{bw}_{\mathsf{R}}(F))} \;\leq\; \mathsf{R}(\mathrm{Lift}(F)) \;\leq\; 2^{O(\mathrm{bw}(\mathcal{P}))} \cdot \|\mathcal{P}\|,$$

where $\mathcal{P}$ is any Resolution refutation of F.

The main theorem for Resolution follows immediately by combining Lemma 2.1 and Lemma 2.3. Namely, the algorithm that computes $\mathcal{A}(F) \coloneqq \mathrm{Lift}(\mathrm{Ref}(F))$ satisfies Theorem 1.1 for Resolution. This completes the overview of our simplified proof of the non-automatability of Resolution.

2.3 Generalization to algebraic systems

Generalizing the proof from the previous subsection (using lifting) to algebraic systems $\mathsf{S} = \mathsf{NS}, \mathsf{PC}$ is now a matter of generalizing the block-width-based Lemma 2.1 and 2.3.

Terminology. The algebraic proof systems are defined formally in Section 8. For the purpose of this overview, we only sketch some notation. The analogue of width in an algebraic system S is *degree*. The degree of a monomial r is denoted $\deg(r)$; the maximum degree of a monomial in an S-refutation $\mathcal{P}$ is denoted $\deg(\mathcal{P})$; the minimum degree of an S-refutation of a formula F is denoted $\deg_{\mathsf{S}}(F)$. Moreover, we define the *block-degree* $\mathrm{bdeg}(r)$ of a monomial r as the number of blocks that r touches; we extend this definition to refutations and formulas as before. For convenience, when talking about Resolution, we use (block-)degree to mean (block-)width. Finally, we use $\mathsf{S}(F)$ to denote the least *size* $\|\mathcal{P}\|$ of an S-refutation $\mathcal{P}$ of F, measured as the number of monomials in $\mathcal{P}$.

Improved lemmas. We now formulate the improved versions of Lemma 2.1 and 2.3. The statements are as expected, except we replace the formula $\mathrm{Ref}(F)$ with a tree-like variant $\mathrm{TreeRef}(F)$, discussed shortly. Our main result (Theorem 1.1) follows by considering $\mathcal{A}(F) \coloneqq \mathrm{Lift}(\mathrm{TreeRef}(F))$ and applying the improved lemmas. The remainder of this section discusses how to prove these lemmas.

Lemma 2.4 (Improved Lemma 2.1)**.** *There is a polynomial-time algorithm that on input an n-variate 3-CNF formula F outputs a block-width-$O(1)$ CNF formula* $\mathrm{TreeRef}(F)$ *such that for proof systems* $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$ *the following holds:*

(i) If F is satisfiable*, then* $\mathrm{TreeRef}(F)$ *admits a size-$n^{O(1)}$ block-degree-$O(1)$ S-refutation.*
(ii) If F is unsatisfiable*, then* $\mathrm{TreeRef}(F)$ *requires S-refutations of block-degree $n^{\Omega(1)}$.*

Lemma 2.5 (Improved Lemma 2.3)**.** *There is a polynomial-time algorithm that on input a block-width-$O(1)$ CNF formula F outputs a CNF formula* $\mathrm{Lift}(F)$ *such that for proof systems* $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$ *it holds that*

$$2^{\Omega(\mathrm{bdeg}_{\mathsf{S}}(F))} \;\leq\; \mathsf{S}(\mathrm{Lift}(F)) \;\leq\; 2^{O(\mathrm{bdeg}(\mathcal{P}))} \cdot \|\mathcal{P}\|,$$

where $\mathcal{P}$ is any S-refutation of F.

Upper bound *(i)*. The first challenge in generalizing the proof for Resolution is that we do not know whether $\mathrm{Ref}(F)$ for a satisfiable F admits a small Nullstellensatz refutation (we suspect not). This is why we introduce in Section 3.2 a new tree-like variant of the formula that intuitively says

$$\mathrm{TreeRef}(F) \;\equiv\; \text{“}F\text{ admits a short tree-like Resolution refutation where the weakening rule is only applied on axiom clauses.”}$$

This formula is a *strengthening* of $\mathrm{Ref}(F)$, meaning that it is obtained from $\mathrm{Ref}(F)$ by adding new variables and axioms. The addition of the tree structure allows us to show the upper bound for Nullstellensatz. The upper bound for Resolution is inherited from $\mathrm{Ref}(F)$, and for other systems they follow by simulations. See Section 11 for the proof of Lemma 2.4*(i)*.

Lower bound *(ii)*. Our simplified proof established the block-width lower bound for $\mathrm{Ref}(F)$ by a reduction from rPHP_{n^2}. In fact, the same reduction works even for $\mathrm{TreeRef}(F)$ without modification. Moreover, it is known that pigeonhole formulas require large degree for PC [Raz98]. We show, via low-degree reductions, that this degree lower bound applies also to our rPHP_{n^2} encoding, and hence to $\mathrm{TreeRef}(F)$. See Section 9 for the proof of Lemma 2.4*(ii)*.

Lifting block-degree. Algebraic proofs are equally amenable to analysis via random restrictions (the key technique behind the proof of Lemma 2.3) as is Resolution. Hence it is straightforward to strengthen Lemma 2.3 to Lemma 2.5. See Section 10 for the proof.

3 Formulas

In this section we define formulas that will be used throughout the paper. In (§3.1) we introduce a variant of the refutation formula $\mathrm{Ref}(F)$ of Atserias and Müller [AM20]; in (§3.2) we modify $\mathrm{Ref}(F)$ to obtain our tree-like variant, $\mathrm{TreeRef}(F)$; and finally in (§3.3) we define a convenient version of the usual pigeonhole principle.

3.1 $\mathrm{Ref}(F)$ formula

Fix a CNF formula F with variables $x_1, \dots, x_n$ and $m = \mathrm{poly}(n)$ clauses. We define another CNF formula $\mathrm{Ref}(F)$ that informally states that *"F admits a short dag-like Resolution refutation."* Our definition differs slightly from that of Atserias and Müller [AM20]; the differences are discussed below.

Variables. The variables of $\mathrm{Ref}(F)$ come partitioned into n^3 blocks $B_1, \dots, B_{n^3}$. The intention is for a block of variables to *encode* or *represent* a single clause in a purported Resolution refutation of F of length at most n^3. More precisely, each block B_i contains the following variables.

- *Literal set.* There are $2n$ many indicator variables y_ℓ for the literals $\ell \in \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$ of F. A boolean assignment to the y_ℓ is intended to define the set of literals for the clause represented by B_i. As a minor detail (relevant in Section 11), we interpret $y_\ell = 0$ to mean that literal ℓ is included in the block.
- *Block type.* There are two boolean variables $\tau = (\tau_1, \tau_2) \in \{0,1\}^2$ encoding the block's *type* as one of three options: *axiom* ($\tau = 00$), *derived* ($\tau = 01$), or *disabled* ($\tau_1 = 1$). We say a block is *enabled* if its type is *axiom* or *derived.* Accordingly, one of the following groups of variables becomes relevant.
  1. *Axiom.* There are $\log m$ many variables that encode an *axiom index* $j \in [m]$. The intention is for an axiom block B_i to be a weakening of the j-th axiom of F.
  2. *Derived.* There are $O(\log n)$ many variables that encode a pair of *child pointers* $(j, j') \in [n^3] \times [n^3]$ and a *resolved-variable index* $k \in [n]$. The intention is for a derived block B_i to be obtained from B_j and $B_{j'}$ by first resolving on variable x_k and then weakening.
  3. *Disabled.* In this case there are no additional relevant variables.

Axioms. It is now straightforward to write down a list of axioms expressing that a truth assignment to the above variables encodes a valid dag-like Resolution refutation of F. Namely, consider a list of constraints defined as follows, where each constraint involves $O(\log n)$ variables.

- *Root.* We require that the last block B_{n^3} (root of the dag) is enabled and that it represents the empty clause, that is, all literal indicator variables are set to 1. (This defines a list of $2n+1$ constraints, each involving at most two variables.)
- *Derived.* For every derived block B_i with an associated triple $(j, j', k) \in [n^3] \times [n^3] \times [n]$ we require that $j, j' < i$; and that B_j (resp. $B_{j'}$) is enabled and contains literal x_k (resp. $\bar{x}_k$); and that every other literal in B_j (except x_k) or $B_{j'}$ (except $\bar{x}_k$) also appears in B_i.
- *Axiom.* For every axiom block B_i with an associated axiom index $j \in [m]$ we require that every literal appearing in the j-th axiom of F also appears in B_i.
- *Disabled.* We impose no constraints on disabled blocks.

Each of these constraints can be writen as a CNF formula over $O(\log n)$ variables. While there are many ways of writing a given constraint in CNF, any choice of encoding will do. (In fact, any two encodings of a $O(\log n)$-variate constraint can be proved equivalent by Resolution (or any other of the proof systems we are interested in) in size $\exp(O(\log n)) = \mathrm{poly}(n)$.) We define $\mathrm{Ref}(F)$ as the conjunction over all these constraints. In conclusion, $\mathrm{Ref}(F)$ is an $O(\log n)$-CNF formula with $\mathrm{poly}(n)$ clauses of block-width ≤ 3 (the worst case is a constraint for a derived block that involves its two children).

Comparison with Atserias–Müller. Our definition of $\mathrm{Ref}(F)$ differs from that of Atserias and Müller in two ways. Firstly, we encode all pointers (and indices) in *binary* instead of *unary*. For the lifting-based proof, this difference is inconsequential and done for convenience as it yields a formula of low width $\mathrm{w}(\mathrm{Ref}(F)) \leq O(\log n)$, which is nice to work with. For the tradeoff-based proof, in contrast, binary encoding is crucial in order for the lower bound on $\mathrm{w}_{\mathsf{R}}(\mathrm{Ref}(F))$ to be large enough (in case F is unsatisfiable), so that the lower bound on size in terms of width can be applied.

Secondly, we allow a block to be *disabled*, whereas Atserias and Müller only introduced this option in the *relativized* version of $\mathrm{Ref}(F)$. In our simplified proof for Resolution, this difference is inconsequantial: even if we cannot explicitly disable a block by setting its type to *disabled*, we can "manually" achieve the same effect by making the block represent an isolated axiom clause in the refutation dag. More interestingly, the option to disable blocks will be needed in extending our proof to the algebraic setting.

3.2 TreeRef(F) formula

Next, we define a tree-like version of $\mathrm{Ref}(F)$ that informally states *"F admits a short* tree-like *Resolution refutation where the weakening rule is only applied on axiom clauses."* Indeed, $\mathrm{TreeRef}(F)$ is obtained by starting from $\mathrm{Ref}(F)$ and adding some new variables and axioms. Here they are:

- *New variables.* We add to each block $O(\log n)$ many new variables that encode a *parent pointer* $p \in [n^3]$. The intention is for p to point to the unique parent in a tree-like refutation.
- *New axioms (tree-likeness).* For a derived block B_i, we require that both of its children have their parent pointers set to i. In the other direction, for a non-root enabled block B_i, we require that its parent B_p is an enabled derived block having B_i as one of its children.
- *New axioms (no weakening).* For a derived block B_i, we require that every literal in B_i appears in *both* of its children. This new axiom implies (together with the old axioms) that if a derived block B_i (obtained by resolving on x_k) has literal set C, then its children have sets $\{x_k\} \cup C$ and $\{\bar{x}_k\} \cup C$. (Note that we still allow an axiom block to be a weakening of an axiom of F.)

3.3 rPHP formula

Finally, we formulate the *retraction weak pigeonhole principle* rPHP_n [Jeř07, PT19]. This variant features $2n$ pigeons and n holes. It uses a *binary encoding* of the pigeon-mapping, and provides an efficient way to *invert* the mapping. Specifically, the variables of rPHP_n describe two functions, $f\colon [2n] \to [n]$ and $g\colon [n] \to [2n]$, encoded as follows.

- *Pigeon map.* For every pigeon $i \in [2n]$ there are variables f_{ik}, $k \in [\log n]$. These variables encode in binary a hole $f(i) \in [n]$ that is expected to house pigeon i.
- *Hole map.* For every hole $j \in [n]$ there are variables $g_{j\ell}$, $\ell \in [\log 2n]$. These variables encode in binary a pigeon $g(j) \in [2n]$ that is expected to occupy hole j.

The axioms of rPHP_n state that for every $i \in [2n]$ and $j \in [n]$,

$$f(i) = j \implies g(j) = i. \tag{1}$$

In other words, g is a left-inverse of f (meaning $g(f(i)) = i$). Note that we do not require g to be a right-inverse (meaning $f(g(j)) = j$), that is, the mapping f need not be surjective. In conclusion, rPHP_n can be written as a $O(\log n)$-width CNF formula in the variables $(f, g) = (f_{ik}, g_{j\ell})$.

Part I
Non-automatability of Resolution

4 Decision tree reductions

In this section, we define *decision tree reductions*, which will be used in Section 5 to prove a lower bound on the block-width $\mathrm{bw}_{\mathsf{R}}(\mathrm{Ref}(F))$ of refuting the formula $\mathrm{Ref}(F)$ in Resolution. We assume the reader is familiar with the standard notion of a decision tree computing a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ (see, e.g., the textbook [Juk12, §14]). In particular, a depth-d decision tree $\mathcal{T}$ computing f naturally gives rise to both a d-DNF and a d-CNF representation for f. Namely, the associated d-DNF is given by $\bigvee_\ell C_\ell$ where ℓ ranges over the leaves of $\mathcal{T}$ that output 1, and C_ℓ is the conjunction of literals (query outcomes) on the path from root to leaf ℓ. The d-CNF is obtained by negating the d-DNF associated with the negated decision tree $\neg\mathcal{T}$ (that is, $\mathcal{T}$ but with its output values flipped) computing $\neg f$.

4.1 What is a reduction?

A decision tree reduction between formulas F and G is a reduction relating the variables of G to the variables of F via shallow decision trees, and moreover, showing that the axioms of F imply those of G. We formalize this as described next.

Definition 4.1 (Reduction)**.** Let $F(x)$ and $G(y)$ be CNF formulas over variables $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$. A *depth-d reduction*, denoted $F \leq_d^{\mathrm{dt}} G$, consists of the following.

- **Variables.** The reduction is defined by a function $f\colon \{0,1\}^n \to \{0,1\}^m$ such that each output bit $f_i\colon \{0,1\}^n \to \{0,1\}$ (thought of as the value given to y_i) for $i \in [m]$ is computed by a depth-d decision tree.

- **Axioms.** Let $C(y)$ be a clause and view it as a function $C\colon \{0,1\}^m \to \{0,1\}$. Consider the composed function $C \circ f$. It can be computed by a depth-$(d \cdot |C|)$ decision tree, and hence we may naturally write it as a $(d \cdot |C|)$-CNF formula. We require that for every axiom $C \in G$, every clause of $C \circ f$ is a weakening of an axiom of F.

The key property of a reduction is that it translates width complexity bounds.

Lemma 4.2. *If $F \leq_d^{\mathrm{dt}} G$, then $\mathrm{w}_{\mathsf{R}}(F) \leq d \cdot \mathrm{w}_{\mathsf{R}}(G)$.*

This lemma is most elegantly proven using the standard *game semantics* (or *top-down*) characterization of $\mathrm{w}_{\mathsf{R}}(F)$ [Pud00, AD08]. Let us briefly recall the details of this characterization.

Prover–Adversary games. The game associated with an n-variate formula F is played between two competing players, Prover and Adversary. The game proceeds in rounds. In each round the state of the game is recorded by a partial assignment $\rho \in \{0,1,*\}^n$ to the variables of F. The game starts with the empty assignment $\rho = *^n$. In each round:

1. *Query a variable.* Prover chooses an $i \in [n]$ with $\rho_i = *$, after which Adversary chooses $b \in \{0,1\}$. The state is updated by $\rho_i \leftarrow b$.
2. *Forget variables.* Prover chooses a (possibly empty) subset $I \subseteq [n]$. The state is updated by $\rho_i \leftarrow *$ for all $i \in I$.

An important detail is that if Prover queries the i-th variable, forgets it, and then queries it again, Adversary is free to respond with any value regardless of the answer given previously. The game ends when ρ falsifies an axiom of F. The width complexity $\mathrm{w}_{\mathsf{R}}(F)$ of F is characterized by the least w such that there is a Prover strategy of *width* w (maximum number of non-$*$ coordinates in the game state at the end of a round) to end the game no matter how Adversary plays.

Proof of Lemma 4.2. Suppose the reduction $F \leq_d^{\mathrm{dt}} G$ is computed by f. Let $\mathcal{G}$ be a width-w Prover strategy for G. We construct a width-dw Prover strategy $\mathcal{F}$ for F by simulating $\mathcal{G}$ round-by-round. We maintain the invariant that if the game state for $\mathcal{G}$ (partial assignment to y) records a value $y_i = b$ for some $b \in \{0,1\}$, then the game state ρ for $\mathcal{F}$ (partial assignment to x) satisfies $f_i(\rho) = b$ by having enough (but at most d) values of the x_j being recorded in ρ.

The simulation proceeds as follows. In each round:

1. *$\mathcal{G}$ queries y_i.* Here we let $\mathcal{F}$ run the decision tree for $f_i(x)$, which queries $\leq d$ variables of F. This returns a value $f_i(x) = b$ for some $b \in \{0,1\}$ depending on the choices of the Adversary. We then simulate $\mathcal{G}$ by responding $y_i = b$ (that is, we play the role of Adversary for $\mathcal{G}$).
2. *$\mathcal{G}$ forgets y_i for $i \in I$.* Here we let $\mathcal{F}$ forget all x_j's which are not required in knowing the values $f_{i'}(x)$ for those i' for which the value of $y_{i'}$ remains in $\mathcal{G}$'s game state.

These actions keep the width of $\mathcal{F}$ at most dw. When the game ends for $\mathcal{G}$, we claim it does so for $\mathcal{F}$: If the state for $\mathcal{G}$ falsifies an axiom of G, then the state for $\mathcal{F}$ falsifies an axiom of F; this is the contrapositive of the weakening property in Definition 4.1. □

4.2 Block-aware reductions

We also introduce a more fine-grained type of reduction, suitable for studying block-width.

Definition 4.3 (Block-aware reduction). Let $F(x) \leq_d^{\mathrm{dt}} G(y)$ via $f\colon \{0,1\}^n \to \{0,1\}^m$ as in Definition 4.1. Suppose further that the variables $y = (y_1, \dots, y_m)$ of G are partitioned into blocks. We say that the reduction $F \leq_d^{\mathrm{dt}} G$ is *block-aware* if for each block $B \subseteq [m]$ there is a depth-d decision tree that computes all the values $f_B(x) \coloneqq (f_i(x) : i \in B) \in \{0,1\}^B$ simultaneously.

Lemma 4.4. *If $F \leq_d^{\mathrm{dt}} G$ via a block-aware reduction, then $\mathrm{w}_{\mathsf{R}}(F) \leq d \cdot \mathrm{bw}_{\mathsf{R}}(G)$.*

Proof. Prover–Adversary games can equally well characterize block-width (defined naturally for a game state as the number of blocks that the state records values from). Hence we can simply run the proof of Lemma 4.2, but now assuming a new invariant: For each block B such that $\mathcal{G}$ records the value of some y_i where $i \in B$, our simulation $\mathcal{F}$ knows $f_B(x)$ by recording at most d values of the x_j variables. By inspection of the previous proof, it follows that if $\mathcal{G}$ has block-width w, then $\mathcal{F}$ has width at most dw. □

5 Block-width lower bound for $\mathrm{Ref}(F)$

We have now collected the tools we need to prove Lemma 2.2 stating that $\mathrm{bw}_{\mathsf{R}}(\mathrm{Ref}(F)) \geq \tilde{\Omega}(\mathrm{w}_{\mathsf{R}}(\mathrm{rPHP}_{n^2})/n)$ holds, where F is any unsatisfiable n-variate CNF formula, and $\mathrm{Ref}(F)$ and rPHP_m are as defined in Sections 3.1 and 3.3, respectively. Our goal is to describe a block-aware reduction

$$\mathrm{rPHP}_{n^2} \;\leq_{\tilde{O}(n)}^{\mathrm{dt}}\; \mathrm{Ref}(F). \tag{2}$$

This reduction, together with Lemma 4.4, would complete the proof of Lemma 2.2.

5.1 Overview of reduction

As in the original proof of Atserias and Müller [AM20], our reduction is guided by the *full tree-like* Resolution refutation $\mathcal{T}$ of the unsatisfiable formula F. More specifically, $\mathcal{T}$ viewed as a refutation dag is a binary tree of height n, it has the empty clause at its root, and at depth $i \in [n]$ the i-th variable is resolved. Thus, $\mathcal{T}$ has 2^n leaves corresponding to all possible width-n clauses; each such leaf clause is a weakening of some axiom of F.

For any truth assignment to rPHP_{n^2}, our reduction is going to produce an assignment to $\mathrm{Ref}(F)$ that represents a purported refutation of F isomorphic to a *subtree* $\mathcal{T}'$ of the full tree $\mathcal{T}$. We refer to the set of nodes of $\mathcal{T}'$ that have smaller degree in $\mathcal{T}'$ than in $\mathcal{T}$ as the *boundary* of the embedding $\mathcal{T}' \subseteq \mathcal{T}$. We note that $\mathcal{T}'$ will not be a valid refutation of F, because the nodes on the boundary are missing at least one child. However, the interior "local neighborhoods" of $\mathcal{T}'$ will be indistinguishable from the corresponding neighborhoods of $\mathcal{T}$, and those parts do not violate any axioms of $\mathrm{Ref}(F)$. The only axiom violations of $\mathrm{Ref}(F)$ result from the boundary nodes.

We now describe the reduction in detail, relying heavily on the illustration in Figure 2.

5.2 Construction

We start by defining how the variables of $\mathrm{Ref}(F)$ depend on the variables of rPHP_{n^2}. We think of the blocks of $\mathrm{Ref}(F)$ as being arranged in $n+1$ layers with layer $\ell \in \{0,1,\dots,n\}$ containing $\min\{2^\ell, n^2\}$ many blocks; see Figure 2. The top-most layer $\ell = 0$ contains just the root block B_{n^3}. The remaining layers host blocks in an arbitrary but fixed way that respects the block ordering: If block B_i is on a lower layer than block B_j, then $i < j$. A small detail is that so far we have not quite used up all the available n^3 blocks. Indeed, any such leftover blocks we define as *disabled*. From now on, we ignore them and do not draw them in Figure 2.

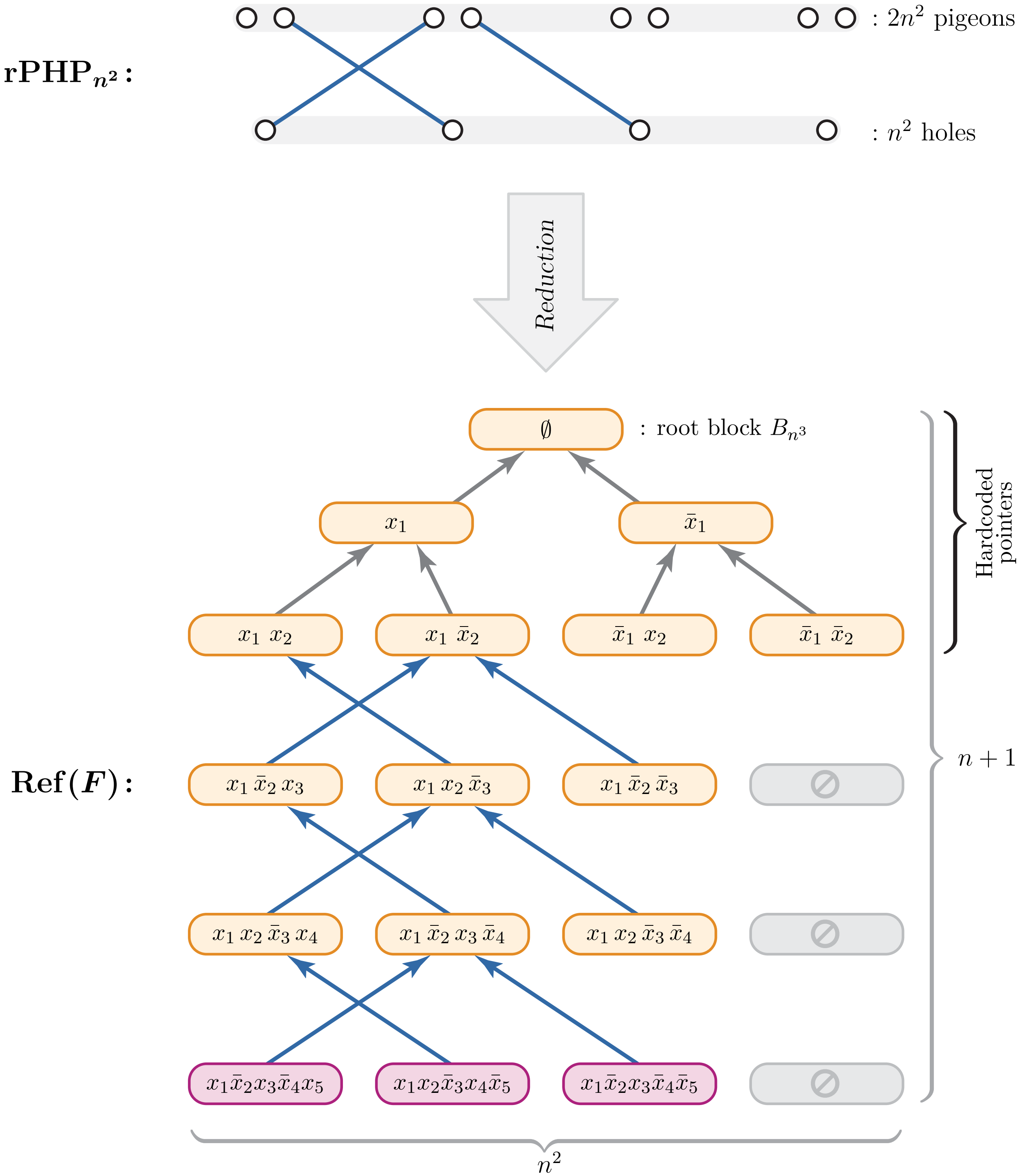

Figure 2: Reduction from rPHP$_{n^2}$ to Ref(F). An assignment to the variables of rPHP$_{n^2}$ defines a *partial matching* $h\colon [2n^2] \to [n^2]$ (drawn in **blue**). Using query access to h we construct an assignment to the variables of Ref(F) that describes a purported refutation of F. The refutation consists of some n^3 blocks arranged in $n+1$ layers. Each block has a type: either *derived* (**yellow**), *axiom* (**purple**), or *disabled* (**gray**). In the refutation dag (as defined in Section 2.1), we draw directed edges *from* children *to* parent (this is the *reverse* direction of the child pointers). The top-most $2\log n$ layers are hardcoded with a tree topology, and between any two remaining layers we insert the partial matching h. The literal set (and other local structure) for each block is computed by locating its natural embedding in the full tree-like refutation $\mathcal{T}$.

We proceed to define the child pointers—which determine the *topology* of the purported refutation—and then the literal sets (and other local structure).

Pointers. The pointers for the top-most $2\log n$ layers we assign so as to build a full binary tree (which in particular matches the topology of $\mathcal{T}$ on these top-most layers). We say this part of the pointer assignment is *hardcoded*, as it does not depend on the variables of rPHP_{n^2}.

Defining the topology for the remaining non-hardcoded layers is the crux of our reduction. Intuitively, we will *copy-and-paste* the pigeon-mapping described by the variables f_{ik} and $g_{j\ell}$ of rPHP_{n^2} (encoding the functions $f\colon [2n] \to [n]$ and $g\colon [n] \to [2n]$) between any two consecutive non-hardcoded layers. This results in several copies of the pigeon-mapping being used in defining the topology.

We first define a partial matching (partial injection) $h\colon [2n^2] \to [n^2] \cup \{*\}$ by

$$h(i) \;\coloneqq\; \begin{cases} f(i) & \text{if } g(f(i)) = i, \\ * & \text{otherwise.} \end{cases} \tag{3}$$

Given a pigeon $i \in [2n^2]$, we can evaluate $h(i)$ by making $O(\log n)$ queries to the boolean variables defining f and g. Moreover, h is easy to invert with query access to f and g. Note that if $h(i) = *$, meaning $f(i) = j$ but $g(j) \neq i$, then this witnesses an axiom violation for rPHP_{n^2} associated with the pair (i, j) as per Equation (1). At the top of Figure 2, we illustrate one possible partial matching resulting from a particular assignment to rPHP_{n^2}.

Consider a layer $\ell \in \{2\log n, \ldots, n-1\}$ that contains n^2 blocks. We think of the child pointers originating from layer ℓ as the $2n^2$ pigeons (each of the n^2 blocks names two children), and the blocks on the next layer $\ell + 1$ as the n^2 holes. More precisely, we define the left (resp. right) child of the i-th block on layer ℓ as the $h(2i-1)$-th (resp. $h(2i)$-th) block on layer $\ell + 1$. If ever $h(i)$ is undefined (meaning an axiom of rPHP_{n^2} associated with i is violated), we define the corresponding pointer as *null* (say, by pointing to the root B_{n^3}, which results in an axiom violation for $\mathrm{Ref}(F)$).

This completes the definition of the topology of the purported refutation described by the variables of $\mathrm{Ref}(F)$. Note that the resulting topology (where we ignore null pointers) is a forest of binary trees: it is constructed by stitching together a binary tree at the top with a layered sequence of partial matchings where we have identified pairs of pigeons (each block couples two pigeons). The lower part of Figure 2 shows how the rPHP_{n^2} assignment at the top defines the structure of the refutation claimed to exist by $\mathrm{Ref}(F)$.

Literal sets. Recall that our goal is to make the purported proof isomorphic to a subtree $\mathcal{T}' \subseteq \mathcal{T}$ (plus some disabled blocks). But now that we have already defined the topology of our purported proof, the definitions of the literal sets (and other local structure) are already determined. To see this, consider the following algorithm (implementable by a moderate-depth decision tree) for computing the literal set for a block B: Starting from B, walk up to its unique parent in the binary forest (this can be done with $O(\log n)$ queries by computing the inverse of h) and continue taking such upward steps until we reach a block without a parent. We have two cases depending on whether the walk terminates at the root block B_{n^3}.

(1) *Root is reached.* Consider the (reverse) path p (sequence of left/right turns) from B_{n^3} to B. This identifies a node v in the full tree $\mathcal{T}$, namely, the node obtained by following the path p starting at the root of $\mathcal{T}$. We simply copy all the local structure at v into B: We make the literal set of B equal that of v. If v is derived in $\mathcal{T}$ by resolving the k-th variable, we make B a derived block and set its resolved-variable index to k. If v is a leaf of $\mathcal{T}$, that is, a weakening of some, say j-th, axiom of F, then we make B an axiom block and set its axiom index to j.

(2) *Root is not reached.* In this case we make B a *disabled* block.

This completes the definition of how the variables of $\mathrm{Ref}(F)$ depend on the variables of rPHP_{n^2}. We finally note that the whole contents of a particular block can be computed by a single decision tree of depth $\tilde{O}(n)$. Indeed, the most expensive part is to perform the walk up the binary forest, which involves at most n (the depth of the purported proof) evaluations of the inverse of h.

5.3 Correctness

It remains to show that every axiom in (the composed version of) $\mathrm{Ref}(F)$ is implied by some axiom of rPHP_{n^2}. We argue the contrapositive: any axiom violation for $\mathrm{Ref}(F)$ implies an axiom violation for rPHP_{n^2}. Since our reduction, by construction, always produces a purported refutation isomorphic to a subtree $\mathcal{T}' \subseteq \mathcal{T}$ (plus some disabled blocks which do not violate axioms of $\mathrm{Ref}(F)$), the only possible axiom violations are caused by a block on layer $\ell \in \{2\log n, \ldots, n-1\}$ containing a null pointer. Any null pointer is caused by the decision tree querying a pigeon i with $h(i) = *$. But this means the decision tree has witnessed a violation of (1), that is, an axiom violation for rPHP_{n^2}, by the discussion following (3). This completes the reduction (2).

5.4 Tree-like extension

To conclude this section, we observe for later use (in Section 9) that the reduction described above can be easily extended to a block-aware reduction

$$\mathrm{rPHP}_{n^2} \ \leq^{\mathrm{dt}}_{\tilde{O}(n)} \ \mathrm{TreeRef}(F). \tag{4}$$

In order to do so, we simply define the *parent pointers* (which are the "new" variables) as the inverses (given by g outside the hardcoded region) of the child pointers defined by the original reduction. To see that the axioms of rPHP_{n^2} imply those of $\mathrm{TreeRef}(T)$, we argue similarly as in Section 5.3: Since $\mathcal{T}$ is a tree-like refutation that uses no weakening (except for the axioms), the output of our reduction (subtree $\mathcal{T}'$ of $\mathcal{T}$) still has its axiom violations only at the boundary of the embedding $\mathcal{T}' \subseteq \mathcal{T}$.

6 Lifting block-width to size

In this section, we prove Lemma 2.3, saying that for the lifted version $\mathrm{Lift}(F)$ of a CNF formula F it holds that $2^{\Omega(\mathrm{bw}_{\mathsf{R}}(F))} \leq \mathsf{R}(\mathrm{Lift}(F)) \leq 2^{O(\mathrm{bw}(\mathcal{P}))}\|\mathcal{P}\|$, where $\mathcal{P}$ is any Resolution refutation of F. We start by describing how the formula $\mathrm{Lift}(F)$ is constructed.

6.1 Lift($\boldsymbol{F}$) formula

Fix a CNF formula F whose variables $x_1, \ldots, x_n$ are partitioned into m blocks. To construct the block-lifted formula $\mathrm{Lift}(F)$, we replace each variable by a copy of a carefully chosen gadget, where gadgets corresponding to the same block partially share variables. Namely, we consider the 3-bit gadget $g\colon \{0,1\}^3 \to \{0,1\}$ defined by $g(x^0, x^1, s) := x^s$. Note that g is computed by a depth-2 decision tree. We now define $\mathrm{Lift}(F)$ formally:

- *Variables.* For every variable x_i of F, the lifted formula will have two variables x_i^0 and x_i^1. Moreover, for every block B of F, we introduce a *selector* variable s_B. Thus, altogether, $\mathrm{Lift}(F)$ has $2n + m$ variables, called *lifted variables.*

$$
\begin{aligned}
&\left(s_{B_1} \vee x^0 \vee \bar{y}^0 \vee s_{B_2} \vee \bar{z}^0 \vee w^0\right) \\
\wedge\ &\left(s_{B_1} \vee x^0 \vee \bar{y}^0 \vee \bar{s}_{B_2} \vee \bar{z}^1 \vee w^1\right) \\
\wedge\ &\left(\bar{s}_{B_1} \vee x^1 \vee \bar{y}^1 \vee s_{B_2} \vee \bar{z}^0 \vee w^0\right) \\
\wedge\ &\left(\bar{s}_{B_1} \vee x^1 \vee \bar{y}^1 \vee \bar{s}_{B_2} \vee \bar{z}^1 \vee w^1\right)
\end{aligned}
$$

Figure 3: The CNF formula for $\mathrm{Lift}(x \vee \bar{y} \vee \bar{z} \vee w)$, where x, y belong to block B_1 and z, w belong to block B_2.

- *Axioms.* Let $C \in F$ be a clause and view it as a function $C\colon \{0,1\}^n \to \{0,1\}$. We define a lifted constraint $\mathrm{Lift}(C)\colon \{0,1\}^{2n+m} \to \{0,1\}$ over the lifted variables as the composition

  $$\mathrm{Lift}(C) \;\coloneqq\; C(g(x_1^0, x_1^1, s_{B(x_1)}), \ldots, g(x_n^0, x_n^1, s_{B(x_n)})),$$

  where $B(x_i)$ denotes the unique block containing x_i. Note that $\mathrm{Lift}(C)$ can be computed by composing a depth-$|C|$ decision tree for C with depth-2 decision trees for the gadgets. This results in a decision tree whose depth is only $d \coloneqq |C| + \mathrm{bw}(C)$ as the gadgets share selector variables. Hence we may write $\mathrm{Lift}(C)$ naturally as a d-CNF formula (as discussed in Section 4). Finally, we define $\mathrm{Lift}(F) \coloneqq \bigwedge_{C \in F} \mathrm{Lift}(C)$.

For concreteness, let us be more explicit about what the CNF expressing $\mathrm{Lift}(C)$ is by inspecting the construction. First, for a variable x and its negation $\neg x$, understood as singleton clauses, we have:

$$
\begin{aligned}
\mathrm{Lift}(x) \;&=\; g(x^0, x^1, s_{B(x)}) \;=\; (s_{B(x)} \vee x^0) \wedge (\bar{s}_{B(x)} \vee x^1), \quad \text{and} \\
\mathrm{Lift}(\neg x) \;&=\; g(\neg x^0, \neg x^1, s_{B(x)}) \;=\; (s_{B(x)} \vee \neg x^0) \wedge (\bar{s}_{B(x)} \vee \neg x^1).
\end{aligned}
$$

Then for an axiom $C = \ell_1 \vee \cdots \vee \ell_w$ in F, we have $\mathrm{Lift}(C) = \bigvee_{i \in [w]} \mathrm{Lift}(\ell_i)$ which can be written in CNF form using the rule $\bigvee_{i \in [w]} F_i = \{C_1 \vee \cdots \vee C_w : C_i \in F_i\}$ for CNF formulas F_i. From this we see that $\mathrm{Lift}(C)$ has $2^{\mathrm{bw}(C)}$ clauses of width $|C| + \mathrm{bw}(C)$ (we refer the reader to Figure 3 for an example). In particular, if F has block-width $O(1)$, then $\mathrm{Lift}(F)$ can be constructed in polynomial time.

6.2 Upper bound for Lift(F)

Let us prove the upper bound $\mathsf{R}(\mathrm{Lift}(F)) \le 2^{O(\mathrm{bw}(\mathcal{P}))} \|\mathcal{P}\|$. We again use the language of Prover–Adversary games from Section 4.1. Besides width, such games can also capture the refutation *size* [Pud00]. Namely, size is characterized by *strategy size*: the total number of states that can ever arise in play (over any number of runs of the game). Thus, let $\mathcal{P}$ be a Prover strategy for F of size $\|\mathcal{P}\|$ and block-width $\mathrm{bw}(\mathcal{P})$. Our goal is to find a small-size strategy $\mathcal{L}$ for $\mathrm{Lift}(F)$.

We start by observing that $\mathrm{Lift}(F) \le_2^{\mathrm{dt}} F$ via $f = (f_1, \ldots, f_n)$ given by $f_i \coloneqq g(x_i^0, x_i^1, s_{B(x_i)})$. The strategy $\mathcal{L}$ is then constructed by simulating $\mathcal{P}$ as in the proof of Lemma 4.2. We proceed to bound $\|\mathcal{L}\|$ by analyzing the simulation carefully. At the start of a simulation round, if $\mathcal{P}$ is in state ρ, then $\mathcal{L}$ is in one of $2^{\mathrm{bw}(\rho)}$ many corresponding states; here the blow-up $2^{\mathrm{bw}(\rho)}$ comes from having to record the values of $\mathrm{bw}(\rho)$ many selector variables. During a simulation step, $\mathcal{L}$ might have to evaluate an f_i, which gives rise to $O(1)$ intermediate states before the start of the next round. We conclude that there is a factor $O(2^{\mathrm{bw}(\rho)})$ overhead in a single round of the simulation. Altogether, we get $\|\mathcal{L}\| \le O(2^{\mathrm{bw}(\mathcal{P})}) \|\mathcal{P}\|$, which proves the upper bound.

6.3 Lower bound for $\mathrm{Lift}(F)$

Finally, we establish the lower bound $2^{\Omega(\mathrm{bw}_{\mathsf{R}}(F))} \leq \mathsf{R}(\mathrm{Lift}(F))$. We show an equivalent claim, namely that $\mathrm{bw}_{\mathsf{R}}(F) \leq O(\log \|\mathcal{P}\|)$ holds for any refutation $\mathcal{P}$ of $\mathrm{Lift}(F)$. Fix such a $\mathcal{P}$ henceforth. We will proceed by a standard argument using random restrictions.

Recall that for a partial truth value assignment ρ and a clause C, the restricted clause $C\restriction_\rho$ is defined to be the trivially true clause 1 if ρ satisfies some literal in C and otherwise the clause C with all literals falsified by ρ removed. This definition extends to sets/sequences of clauses $\mathcal{A}$ in the natural way by restricting all clauses in $\mathcal{A}$, removing those which are satisfied. Given a Resolution refutation $\mathcal{F}$ of a CNF formula F, it is a well-known fact that for any partial assignment ρ it holds that $\mathcal{F}\restriction_\rho$ is a resolution refutation of the restricted formula $F\restriction_\rho$ in at most the same size and width.

We start by defining a random restriction ρ to a subset of the variables of $\mathrm{Lift}(F)$ in two steps:

(1) Let ρ_1 be a random restriction setting each selector variable s_B to a uniform random bit.

(2) Define X_{ρ_1} as the set of variables that contains, for every variable x_i of F, the variable x_i^{1-s} where $s := s_{B(x_i)}$ is determined by ρ_1. Let ρ_2 be a random restriction setting each variable in X_{ρ_1} to a uniform random bit. Let ρ be the concatenation of ρ_1 and ρ_2.

Note that variables from different blocks are assigned independently. Moreover, each literal evaluates to true with probability at least $1/4$. Thus, the probability that a clause of block-width at least w is not satisfied by ρ is at most $(3/4)^w$. Consider the restricted refutation $\mathcal{P}\restriction_\rho$. By a union bound, we see that

$$\mathbf{Pr}[\,\mathcal{P}\restriction_\rho \text{ has a clause of block-width} \geq w\,] \;\leq\; \|\mathcal{P}\| \cdot (3/4)^w.$$

For $w := 3\log\|\mathcal{P}\| > \log\|\mathcal{P}\|/\log(4/3)$ this probability is < 1, and hence there exists some fixed ρ such that $\mathrm{bw}(\mathcal{P}\restriction_\rho) \leq w$. But $\mathcal{P}\restriction_\rho$ is a refutation of the formula $\mathrm{Lift}(F)\restriction_\rho$, which is easily seen to be the same as F after renaming variables. Hence, $\mathrm{bw}_{\mathsf{R}}(F) \leq w = O(\log\|\mathcal{P}\|)$, which completes the proof of Lemma 2.3.

6.4 Alternative proof via tradeoffs

We also wish to discuss an alternative way of proving a Resolution size lower bound for $\mathrm{Ref}(F)$ when F is unsatisfiable via the size–width tradeoff in [BW01] (without the need for gadget composition). We recall that the precise statement of this result is as follows.

Theorem 6.1 ([BW01]). *For any unsatisfiable CNF formula F over N variables it holds that $\mathsf{R}(F) \geq 2^{\Omega((\mathrm{w}_{\mathsf{R}}(F)-\mathrm{w}(F))^2/N)}$.*

As mentioned previously, what makes this tradeoff approach possible is that we have defined $\mathrm{Ref}(F)$ in a more succinct way than Atserias and Müller originally did. However, in order to make this work we need to be a little bit more careful with our choice of parameters as described next. Suppose we had defined $\mathrm{Ref}(F)$ by having a larger number n^6 of blocks (rather than n^3). Then $\mathrm{Ref}(F)$ has $N := O(n^7)$ variables. Now we can modify the reduction in Section 5 to have n^5 (rather than n^2) blocks per layer. This gives us a quantitatively improved reduction $\mathrm{rPHP}_{n^5} \leq^{\mathrm{dt}}_{\tilde{O}(n)} \mathrm{Ref}(F)$. Using this reduction, we conclude that

$$\mathrm{w}_{\mathsf{R}}(\mathrm{Ref}(F)) \;\geq\; \mathrm{w}_{\mathsf{R}}(\mathrm{rPHP}_{n^5})/\tilde{O}(n) \;\geq\; \tilde{\Omega}(n^4) \;=\; \tilde{\Omega}(N^{4/7}) \gg \sqrt{N}.$$

An application of Theorem 6.1 now yields $\mathsf{R}(\mathrm{Ref}(F)) \geq 2^{N^{\Omega(1)}}$ as desired.

Algebraic systems. The tradeoff approach also works for Polynomial Calculus (PC) by using the size–degree tradeoff from [IPS99]. We note that no such tradeoff holds for Nullstellensatz (NS), since there are formulas F with $\deg_{\mathsf{NS}}(F) = \tilde{\Omega}(n)$ and $\mathsf{NS}(F) = n^{O(1)}$ [BCIP02], but the lower bound can be proven using a tradeoff for PC and noting that PC efficiently simulates NS. In the rest of the paper, we will focus on the lifting approach.

7 Resolution upper bound for $\mathrm{Ref}(F)$

Let us now establish Lemma 2.1*(i)*, that is, that if F is satisfiable, then $\mathrm{Ref}(F)$ admits a polynomial-size Resolution refutation. This was already shown by Atserias and Müller [AM20], but we include a proof for completeness and because our encoding differs slightly from theirs. The upper bound holds even for *regular* Resolution, a subsystem of Resolution in which every variable is resolved at most once on any source–sink path in the refutation dag. We present a top-down proof via a *read-once branching program* that makes regularity obvious.

7.1 Read-once branching programs

A *branching program* Π over the variables $x_1, \ldots, x_n$ is a dag with a unique source and where every non-sink node is labelled by a variable x_i and has two outgoing edges labelled 0 and 1. If a node is labelled x_i, we say that x_i is queried at this node. On input $x \in \{0,1\}^n$, the branching program determines a source–sink path, denoted $\mathrm{path}(x)$, by following the corresponding edge labels. The size of a branching program is its number of nodes. A branching program is *read-once* if along any source–sink path each variable is queried at most once.

Let F be an unsatisfiable CNF over variables $x_1, \ldots, x_n$. We say that a branching program Π solves the *falsified clause search problem* for F if the sinks of Π are labelled by clauses of F and for every assignment $x \in \{0,1\}^n$ we have that $\mathrm{path}(x)$ ends at a sink labelled by a clause falsified by x. If we take such a branching program Π that is read-once and reverse the edges, then we get (the refutation dag representation of) a regular resolution refutation, and vice versa, from which we can deduce the following.

Fact 7.1 (Folklore, e.g., [Juk12, §18.2])**.** *The minimum size of a regular Resolution refutation of F is equal to the minimum size of a read-once branching program solving the falsified clause search problem for F.*

7.2 Proof of Lemma 2.1*(i)*

Intuition. To prove Lemma 2.1*(i)* we exhibit a $\mathrm{poly}(n)$-size read-once branching program that solves the falsified clause search problem for $\mathrm{Ref}(F)$. The idea is to fix a satisfying assignment x^* of F and query variables of $\mathrm{Ref}(F)$ down a path starting at the root block B_{n^3} and ending at an axiom block, all the while maintaining the following invariant:

(Invariant:) *The current block represents a clause that is falsified by x^*.*

At each step, since the axioms of $\mathrm{Ref}(F)$ guarantee soundness of the purported refutation, either there is a violation of one of these axioms (and thus we have a clause of $\mathrm{Ref}(F)$ that is falsified), or we can move to a child block that is also falsified by x^*. Once an axiom block is reached, we can easily detect a falsified axiom of $\mathrm{Ref}(F)$: the axiom block cannot be a weakening of any axiom of F since x^* satisfies all axioms of F. Since the branching program only needs to "remember" two blocks at a time, the size of this program is approximately the square of the number of blocks in $\mathrm{Ref}(F)$.

Formal proof. Let x^* be a satisfying assignment of F and let $\ell_1, \ldots, \ell_n$ be the literals satisfied by x^*. Denote by $N \coloneqq n^3$ the number of blocks of $\mathrm{Ref}(F)$. In order to describe the read-once branching program Π, we first define a total of $2N$ special nodes of Π, then explain how these are interconnected by decision trees of size $O(nN + m)$ to form the final branching program.

A path from the source to a node v in Π defines a partial assignment. We say node v *remembers* a partial assignment ρ, if for every source-to-v path the corresponding partial assignment contains ρ. For $i \in [N]$ we have the following special nodes in Π:

- a node v_i that remembers (the partial assignment that encodes) that the block B_i is enabled and that it does not contain any literal ℓ_t for $t \in [n]$; and
- a node v_i^{axiom} that remembers all that v_i remembers and also that B_i is an axiom block.

Let us see how these nodes are interconnected in Π.

***Connecting the source of* Π.** We start by constructing a tree that connects the source of Π to the node v_N. At the source, Π queries the type of B_N. In the case where B_N is disabled, the axiom of $\mathrm{Ref}(F)$ that states that the root should be enabled is falsified, and hence our branching program has solved the search problem. Otherwise Π checks that all literal-indicators y_{ℓ_t}, $t \in [n]$, of B_N are set to 1 (meaning no literal ℓ_t appears in B_N). If any literal ℓ_t is present, then one of the axioms of $\mathrm{Ref}(F)$ stating that the root does not contain any literal is falsified. Finally, if B_N is enabled and does not contain any of the variables y_{ℓ_t}, the path reaches v_N.

***Connecting* v_i.** Now let us construct a tree connecting v_i to v_i^{axiom} and to other nodes v_j, for $j < i$. At node v_i, Π queries whether B_i is an axiom block or a derived block. If B_i is an axiom block, then the path has reached node v_i^{axiom}. Otherwise, Π queries the index $k \in [n]$ corresponding to the variable resolved to obtain B_i. It then queries the pointer $j \in [N]$ naming the child of B_i that should not contain the literal ℓ_k. If $j \geq i$, then there is an axiom of $\mathrm{Ref}(F)$ that is falsified. Otherwise, Π checks that all literal-indicators y_{ℓ_t}, $t \in [n]$, of B_j are set to 1 (if any of the literals ℓ_t is present, then some axiom of $\mathrm{Ref}(F)$ is falsified). Finally, Π checks that B_j is enabled (if not, an axiom of $\mathrm{Ref}(F)$ is falsified) and reaches the node v_j.

***Connecting* v_i^{axiom}.** At node v_i^{axiom}, Π queries the index $j \in [m]$ naming the axiom of F that implies the clause of B_i. Since x^* satisfies every clause of F, the jth clause contains some literal ℓ_t for $t \in [n]$. But this literal is not in B_i, so an axiom of $\mathrm{Ref}(F)$ is falsified.

It is straightforward to check that Π is indeed *read-once*. Indeed, note from the source of Π until the node v_i, Π only queries variables from blocks B_j for $j \geq i$. Moreover, from v_i^{axiom} all paths only query variables (indices to axioms of F) that were not queried before. Finally, since the trees described above are of size $O(nN + m)$, we conclude that Π is of size $O(nN^2 + mN)$.

Part II
Non-automatability of Algebraic Proof Systems

8 Algebraic definitions

In this section, we define: (§8.1) the algebraic proof systems Nullstellensatz (NS) and Polynomial Calculus (PC); and (§8.2) algebraic reductions.

8.1 Algebraic proof systems

All the algebraic proof systems are going to share the following basic setup. We work over the polynomial ring $\mathbb{F}[X]$ where $\mathbb{F}$ is a fixed field and $X \coloneqq \{x_1, x_2, \ldots, x_n\}$ is a set of formal variables. We define the *size* $\|p\|$ of a polynomial $p \in \mathbb{F}[X]$ as the number of its non-zero monomials (when expanded out as a linear combination of monomials). If the variables X are partitioned into blocks, we define the *block-degree* $\mathrm{bdeg}(r)$ of a monomial r as the number of distinct blocks that r touches, and the block-degree of a polynomial as the largest block-degree of any of its monomials.

For a CNF formula F over variables X we use the standard translation of F into a set of polynomial equations F^* defined as follows. First, for each x_i we include in F^* the *boolean axiom* $x_i^2 - x_i = 0$ (enforcing $x_i \in \{0, 1\}$). Second, for each clause $\bigvee_{i \in I} x_i \vee \bigvee_{j \in J} \bar{x}_j$ of F we include in F^* the equation

$$\prod_{i \in I}(1 - x_i) \prod_{j \in J} x_j \;=\; 0. \tag{5}$$

This way, F and F^* have the same set of satisfying assignments. Henceforth, we will sometimes identify F and F^*. We are now ready to define our algebraic proof systems.

Nullstellensatz (NS). Nullstellensatz is a static algebraic proof system based on Hilbert's Nullstellensatz. An NS-*proof* of $f = 0$ from a set of polynomial equations $F = \{f_1 = 0, \ldots, f_m = 0\}$ is a set of polynomials $\mathcal{P} = \{p_1, \ldots, p_m\}$ such that, as formal polynomials,

$$\sum_{i \in [m]} p_i f_i \;=\; f.$$

The *size* of the proof is $\|\mathcal{P}\| \coloneqq \sum_{i \in [m]} \|p_i\| \|f_i\|$, its *degree* is $\deg(\mathcal{P}) \coloneqq \max_{i \in [m]}(\deg(p_i) + \deg(f_i))$ and, if the variables X are partitioned into blocks, its *block-degree* is $\mathrm{bdeg}(\mathcal{P}) \coloneqq \max_{i \in [m]}(\mathrm{bdeg}(p_i) + \mathrm{bdeg}(f_i))$. An NS-*refutation* of F is an NS-proof of $1 = 0$ from F.

Polynomial Calculus (PC). Polynomial Calculus is a dynamic extension of Nullstellensatz. A PC-*proof* of $f = 0$ from a set of polynomial equations $F = \{f_1 = 0, \ldots, f_m = 0\}$ is a sequence of polynomials $\mathcal{P} = (p_1, \ldots, p_s)$ such that $p_s = f$ and for each $i \in [s]$ either (i) $p_i \in F$ or (ii) p_i is derived from polynomials earlier in the sequence using one of the following rules:

- *Linear combination:* From p_j and $p_{j'}$ derive $\alpha p_j + \beta p_{j'}$ for any $\alpha, \beta \in \mathbb{F}$.
- *Multiplication:* From p_j derive $x p_j$ for any $x \in X$.

The *size* of the proof is $\|\mathcal{P}\| \coloneqq \sum_{i \in [s]} \|p_i\|$, its *degree* is $\deg(\mathcal{P}) \coloneqq \max_{i \in [s]} \deg(p_i)$ and, if the variables X are partitioned into blocks, its *block-degree* is $\mathrm{bdeg}(\mathcal{P}) \coloneqq \max_{i \in [s]} \mathrm{bdeg}(p_i)$. A PC-*refutation* of F is a PC-proof of $1 = 0$ from F.

Complexity measures. We define complexity measures uniformly across $\mathsf{S} = \mathsf{NS}, \mathsf{PC}$.

- The *size complexity* $\mathsf{S}(F)$ of a formula F is the minimum size of an S-refutation of F.
- The *degree complexity* $\deg_{\mathsf{S}}(F)$ is the minimum degree of an S-refutation of F.
- The *block degree complexity* $\mathrm{bdeg}_{\mathsf{S}}(F)$ is the minimum block-degree of an S-refutation of F.

Twin variables. Every algebraic proof systems can be extended using so-called *twin variables*. This means that for every variable $x \in X$ we add another formal variable $\bar{x}$, and include the *complementary axiom* $x + \bar{x} - 1 = 0$. The translation of CNF formulas to polynomial equations can be made more concise by the use of twin variables. Polynomial Calculus with twin variables is often called *Polynomial Calculus Resolution* (PCR). Using twin variables does not affect the degree complexity in any of the proof systems, but their introduction could potentially reduce size quite drastically. Our main result (Theorem 1.1) holds in the best of all possible worlds: All upper bounds hold *without* twin variables, and the lower bounds hold *with* twin variables.

Relationships. It is well-known and easy to see that PC (and SA if the field is $\mathbb{R}$) can efficiently simulate NS. A surprising result of Berkholz [Ber18] (recorded in Figure 1) is that SoS efficiently simulates PC over $\mathbb{R}$. In this paper, we need only the following easy simulation.

Fact 8.1 (Simulation)**.** *Suppose a polynomial f admits an* NS*-proof from a set of n-variate polynomials F in size s and (block-)degree d. Then there is a* PC*-proof of f from F in size* $\mathrm{poly}(s, n)$ *and (block-)degree d.* □

Multilinear polynomials. The *multilinearization* of a polynomial p is defined as the polynomial obtained by replacing all terms in p of the form x^i, $i \geq 2$, with x; that is, we work modulo the boolean axioms. It will be convenient to assume that all polynomials appearing in our algebraic manipulations are implicitly multilinearized. For example, the product pq of two multilinear polynomials p and q may not itself be multilinear, but pq can be efficiently proven equivalent to its multilinearization by an application of the boolean axioms. It is well known that this implicit multilinearization does not affect the degree complexity of a formula except by a constant factor, and the size complexity can increase at most polynomially. When we work in a multilinear setting we can equate the *syntactic* representation of a polynomial as an element of $\mathbb{F}[X]$ with its *semantic* representation as a boolean function $\{0,1\}^n \to \{0,1\}$, since each boolean function has a unique representation as a multilinear polynomial.

8.2 Algebraic reductions

We now develop algebraic analogues of the decision tree reductions introduced in Section 4. Notions similar to the next definition have occurred before in, for instance, [BGIP01, LN17a, LN17b].

Definition 8.2 (Algebraic reduction)**.** Let F and G be two sets of polynomials encoding CNF formulas over a field $\mathbb{F}$, defined on variables $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$, respectively. An *algebraic reduction*, denoted $F \leq^{\mathrm{alg}} G$, of degree d consists of the following.

- **Variables.** The reduction is computed by a function $r\colon \{0,1\}^n \to \{0,1\}^m$ such that each output bit $r_i\colon \{0,1\}^n \to \{0,1\}$ is computed by a degree-d polynomial.
- **Axioms.** For any $g \in G$, the *multilinearization* of the polynomial $g \circ r$ has an NS-proof from F (over any field) of degree $d \cdot \deg(g)$.

This definition allows us to transform algebraic refutations of G into refutations of F.

Lemma 8.3. *If $F \leq^{\mathrm{alg}} G$ with degree d, then* $\deg_{\mathsf{S}}(F) \leq d \cdot \deg_{\mathsf{S}}(G)$ *for all* $\mathsf{S} = \mathsf{NS}, \mathsf{PC}$.

Proof. We first prove the lemma for NS. Suppose the reduction is computed by r and let $b = |G|$. Write $G = \{g_1, \ldots, g_b\}$, and let $\mathcal{P} = (p_1, \ldots, p_b)$ be an NS-refutation of G. Consider the expression

$$\sum_{i \in [b]} (p_i g_i) \circ r \;=\; \sum_{i \in [b]} (p_i \circ r)(g_i \circ r) \;=\; 1. \tag{6}$$

This expression is syntactically equal to 1, since $\mathcal{P}$ is a refutation of G. By the definition of reduction, each polynomial $g_i \circ r$ can be deduced from the axioms of F in degree $d \cdot \deg(g_i)$. Therefore, (6) can be written as an NS-refutation of F of degree at most

$$\max_{i\in[b]} \left(\deg(p_i \circ r) + d \cdot \deg(g_i)\right) \leq d \cdot \max_{i\in[b]} \left(\deg(p_i) + \deg(g_i)\right) = d \cdot \deg(\mathcal{P}).$$

We next prove the lemma for PC. Let $\mathcal{P}$ be a PC-refutation of G. We construct a PC-refutation $\mathcal{P}'$ of F. We argue by structural induction over $\mathcal{P}$: whenever $\mathcal{P}$ derives p, in $\mathcal{P}'$ we will derive $p \circ r$.

- *Axioms.* For any axiom $g \in G$ used by $\mathcal{P}$, by the definition of reduction we can derive the polynomial $g \circ r$ in NS—and therefore, by Fact 8.1, also in PC—in degree $d \cdot \deg(g)$.
- *Linear Combination.* If the polynomial p_3 is derived from p_1 and p_2 using a linear combination, then we derive $p_3 \circ r$ from $p_1 \circ r$ and $p_2 \circ r$ in $\mathcal{P}'$ using the same linear combination.
- *Multiplication.* If $y_i p$ is derived from p by the multiplication rule, then we can derive $(y_i p) \circ r = r_i(p \circ r)$ from $p \circ r$ by repeated use of multiplication and linear combinations.

Note that we can always derive $p \circ r$ in degree at most $d \cdot \deg(p)$ and therefore $\deg(\mathcal{P}') \leq d \cdot \deg(\mathcal{P})$. $\square$

Next, we define the algebraic analogue of a *block-aware* reduction. Note that when the algebraic reduction is a applied to a monomial $\prod_{i\in I} y_i$ this will produce the polynomial $\prod_{i\in I} r_i$ on the variables $x = (x_1, \dots, x_n)$, and so we will need to control the degree and block-degree of such polynomials.

Definition 8.4 (Algebraic block-aware reduction)**.** Let F and G be two sets of polynomials encoding CNF formulas over a field $\mathbb{F}$, and suppose that $F \leq^{\mathrm{alg}} G$ by a degree-d reduction $r : \{0,1\}^n \to \{0,1\}^m$ as in the previous definition. Suppose further that the variables of G are partitioned into blocks. The reduction r is a degree-d, size-s *block-aware reduction* if the following two conditions hold:

- **Blocks.** For each block B and each $T \subseteq B$ the polynomial

$$r_T := \text{multilinearization of } \prod_{i\in T} r_i$$

  has degree at most d and size at most s.
- **Axioms.** For any $g \in G$, the *multilinearization* of the polynomial $g \circ r$ has an NS-proof from F (over any field) of degree $d \cdot \mathrm{bdeg}(g)$ and size s.

Suppose in addition that the variables of F are also partitioned into blocks. Then the reduction r is *block-preserving* if $\mathrm{bdeg}(r_T) = O(1)$ for every T contained in a block of G and if for every $g \in G$ the NS-proof of $g \circ r$ from F that satisfies the *Axioms* property above has block-degree $O(1)$.

We note that although the definition specifies both the degree and the size of the reduction, often only one of these measures will be relevant and hence mentioned. For example, the following lemma only takes the degree of the reduction into account.

Lemma 8.5. *If $F \leq^{\mathrm{alg}} G$ via a degree-d block-aware reduction, then $\deg_{\mathsf{S}}(F) \leq d \cdot \mathrm{bdeg}_{\mathsf{S}}(G)$ for all $\mathsf{S} = \mathsf{NS}, \mathsf{PC}$.*

Proof. The case for Nullstellensatz identically follows the proof of Lemma 8.3 except in each monomial of the proof we substitute the corresponding polynomial r_T for each block of variables y^T when $T \subseteq B$ is contained within a block.

For Polynomial Calculus we also closely follow the proof of Lemma 8.3. Every line of a PC-proof is multilinear, so, by the definition of a block-aware reduction and following the same accounting in the proof of Lemma 8.3 we see that the degree of the new proof is at most $d \cdot \mathrm{bdeg}(\mathcal{P})$. $\square$

We end this section by relating algebraic block-aware degree-d reductions to decision tree block-aware depth-d reductions.

Lemma 8.6. *If $F \leq_d^{\text{dt}} G$ via a block-aware reduction then $F \leq^{\text{alg}} G$ via a degree-d block-aware reduction.*

Proof. The main element in this proof is the observation that any function that can be computed by a depth-d decision tree can also be computed by a degree-d polynomial. For a clause C we denote by C^* the translation of C into a polynomial as in Equation 5.

Suppose that a block-aware reduction $F(x) \leq_d^{\text{dt}} G(y)$ as in Definition 4.3 is computed by $r\colon \{0,1\}^n \to \{0,1\}^m$. We need to show that r satisfies the two conditions of Definition 8.4.

By definition of the block-aware decision tree reduction, for each block $B \subseteq [m]$ of G there is a depth-d decision tree that computes all the values $r_B(x) \coloneqq (r_i(x) : i \in B) \in \{0,1\}^B$ simultaneously. Thus, for every $T \subseteq B$, the function $r_T(x)$ can be computed by a depth-d decision tree and hence by a degree-d polynomial. (In more detail, for every leaf v of the decision tree, let q_v be the conical junta that evaluates to 1 precisely when the assignment ends up in leaf v. Then $r_T(x) = \sum_{\text{1-leaves } v} q_v$ is a polynomial of degree at most d.) This proves the first item of Definition 8.4.

More generally, for any clause $C(y)$ there is a depth-$(d \cdot \text{bw}(C))$ decision tree that computes the value of all the variables of $C \circ r$. Therefore, $C \circ r$ can be written as a $(d \cdot \text{bw}(C))$-CNF formula, and furthermore the canonical translation we use from decision trees to CNF formulas will make sure that for any truth value assignment either all clauses in this formula evaluate to true or exactly one clause in the formula evaluates to false (namely the unique clause corresponding to the path leading to the leaf providing the answer 0—all other clauses evaluate to true since the assignment fails to agree with the paths they correspond to).

Now consider any axiom $C \in G$. Recall that the decision tree reduction guarantees that every clause in $C \circ r$ is a weakening of an axiom of F. Thus, we may write $C \circ r = \bigwedge_{i \in I}(A_i \vee B_i)$ where $A_i \in F$ for all $i \in I$. As noted above, we also know for this formula that at most one clause $A_i \vee B_i$ evaluates to false. This in turn means that the multilinearization of $\sum_{i \in I} A_i^* \cdot B_i^*$ computes $C \circ r$ correctly over any field, since at most one polynomial $A_i^* \cdot B_i^*$ evaluates to 1 and all others evaluate to 0. It then follows, by the uniqueness of multilinear polynomial representation of boolean functions, that the multilinearization of $C^* \circ r$ must in fact be syntactically equal to the multilinearization of $\sum_{i \in I} A_i^* \cdot B_i^*$. But this latter expression is a multilinear NS-proof from F (over any field) of degree $d \cdot \text{bw}(C) = d \cdot \text{bdeg}(C^*)$. This establishes the second item of Definition 8.4, and so concludes the proof. ∎

9 Block-degree lower bound for TreeRef(F)

This section is dedicated to proving Lemma 2.4*(ii)*, which says that $\text{bdeg}_{\mathsf{S}}(\text{TreeRef}(F)) \geq n^{\Omega(1)}$, where F is unsatisfiable and $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$. We already know that $\text{bdeg}_{\mathsf{S}}(\text{TreeRef}(F))$ is at least $\tilde{\Omega}(\text{deg}_{\mathsf{S}}(\text{rPHP}_{n^2})/n)$ by the reduction of Section 5.4 and Lemmas 8.5 and 8.6. Hence it suffices to prove

$$\text{deg}_{\mathsf{S}}(\text{rPHP}_n) \;\geq\; \tilde{\Omega}(n). \tag{7}$$

We show this follows from a known degree lower bound for PC (over any field) due to Razborov [Raz98]. This paper uses a different algebraic encoding of the pigeonhole principle, which we recall below. In the rest of this section we show that our encoding reduces to Razborov's encoding by a low-degree reduction. This will prove (7).

Algebraic PHP. Define aPHP_n as the following system of polynomial equations over variables x_{ij} where $i \in [2n]$ and $j \in [n]$.

$$\begin{array}{llcll} \forall i: & Q_i & \coloneqq & \sum_j x_{ij} - 1 \;=\; 0 & \text{"each pigeon occupies a hole,"} \\ \forall i; j \neq j': & Q_{i;j,j'} & \coloneqq & x_{ij}x_{ij'} \;=\; 0 & \text{"no pigeon occupies two holes,"} \\ \forall j; i \neq i': & Q_{i,i';j} & \coloneqq & x_{ij}x_{i'j} \;=\; 0 & \text{"no hole houses two pigeons,"} \\ \forall i,j: & Q_{i,j} & \coloneqq & x_{ij}^2 - x_{ij} \;=\; 0 & \text{"boolean axioms."} \end{array} \qquad (\mathrm{aPHP}_n)$$

Theorem 9.1 ([Raz98]). $\deg_{\mathsf{PC}}(\mathrm{aPHP}_n) \geq \Omega(n)$.

To prove (7), we translate the degree lower bound in Theorem 9.1 to our rPHP encoding via an algebraic reduction. Namely, our goal is to show an algebraic degree-$\tilde{O}(1)$ reduction

$$\mathrm{aPHP}_n \;\leq^{\mathrm{alg}}\; \mathrm{rPHP}_n.$$

Variables. We start by defining how the variables $(f,g) = (f_{ik}, g_{j\ell})$ of rPHP_n (where $i \in [2n]$, $k \in [\log n]$, $j \in [n]$, and $\ell \in [\log 2n]$) depend on the variables x_{ij} of aPHP_n (where $i \in [2n]$ and $j \in [n]$). For convenience, we think of $[n] \coloneqq \{0,1,\ldots,n-1\}$ so that each hole $i \in [n]$ (resp. pigeon $j \in [2n]$) can naturally be thought of as a bit-string $i \in \{0,1\}^{\log n}$ (resp. $j \in \{0,1\}^{\log 2n}$).

- Define $f_{ik} \coloneqq \sum_{j \in J_k} x_{ij}$ where $J_k \coloneqq \{j \in [n] : j_k = 1\}$ are the holes with k-th bit equal to 1.
- Define $g_{j\ell} \coloneqq \sum_{i \in I_\ell} x_{ij}$ where $I_\ell \coloneqq \{i \in [2n] : i_\ell = 1\}$ are the pigeons with ℓ-th bit equal to 1.

Axioms. We need to show that every axiom of rPHP_n (that is, an axiom encoding $g(f(i)) = i$ or a boolean axiom), when substituted with the above linear forms, admit a low-degree NS-proof over any field from the axioms of aPHP_n. With a slight abuse of notation, we write $p(x) \cong q(x)$ to mean that $p(x) - q(x) = 0$ can be derived from aPHP_n in degree $\tilde{O}(1)$. The boolean axioms of rPHP_n are easy to verify. Here they are for f_{ik} (the case of $g_{j\ell}$ is analogous):

$$f_{ik}^2 \;=\; \Big(\sum_{j \in J_k} x_{ij}\Big)^2 \;=\; \sum_{j \in J_k} x_{ij}^2 + \sum_{\substack{j,j' \in J_k \\ j \neq j'}} x_{ij}x_{ij'} \;=\; \sum_{j \in J_k} (x_{ij} + Q_{i,j}) + \sum_{\substack{j,j' \in J_k \\ j \neq j'}} Q_{i;j,j'} \;\cong\; \sum_{j \in J_k} x_{ij} \;=\; f_{ik}.$$

The crux of the reduction is to derive the main rPHP_n axioms encoding $f(i) = j \Rightarrow g(j) = i$ for all $i \in [2n]$ and $j \in [n]$. By the standard translation from clauses, we express these axioms as polynomials; we write $f^1 \coloneqq f$ and $f^0 \coloneqq 1 - f$ for short:

$$\begin{aligned} [f(i) = j \Rightarrow g(j) = i] \;&\equiv\; [g(j) = i \vee f(i) \neq j] \\ &\equiv\; \textstyle\bigwedge_\ell \, [g_{j\ell} = i_\ell \vee \bigvee_k f_{ik} \neq j_k] \\ &\equiv\; \{\, g_{j\ell}^{1-i_\ell} \textstyle\prod_k f_{ik}^{j_k} = 0 \;:\; \ell \in [\log 2n] \,\}. \end{aligned} \qquad (*)$$

Before deriving these polynomial equations, we prove two helper claims.

Claim 9.2. $f_{ik}^0 \cong \sum_{j \in [n] \setminus J_k} x_{ij}$.

Proof. We have $f_{ik}^0 = 1 - f_{ik} = \big(\sum_{j \in [n]} x_{ij} - Q_i\big) - f_{ik} = \sum_{j \in [n] \setminus J_k} x_{ij} - Q_i \cong \sum_{j \in [n] \setminus J_k} x_{ij}$. □

Claim 9.3. $\prod_k f_{ik}^{j_k} \cong x_{ij}$.

Proof. Expand each $f_{ik}^{j_k}$ according to its definition ($j_k = 1$) or by Claim 9.2 ($j_k = 0$):

$$\begin{aligned}
\textstyle\prod_k f_{ik}^{j_k} &\cong x_{ij}^{\log n} + \textstyle\sum_{j' \neq j''} r_{j'j''}(x) \cdot x_{ij'} x_{ij''} && \text{(where } \deg(r_{j'j''}) \leq \log n\text{)} \\
&\cong x_{ij}^{\log n} && (x_{ij'} x_{ij''} = Q_{i;j',j''}) \\
&\cong x_{ij}. && \text{(boolean axioms)} \qquad \square
\end{aligned}$$

We now derive ($*$). By Claim 9.3, we have $(*) = g_{j\ell}^{1-i_\ell} \prod_k f_{ik}^{j_k} \cong g_{j\ell}^{1-i_\ell} \cdot x_{ij}$. Two cases:

$$\begin{aligned}
&\textit{Case } i_\ell = 0 \textit{ (where } i \notin I_\ell\textit{):} & (*) &= \left(\textstyle\sum_{i' \in I_\ell} x_{i'j}\right) \cdot x_{ij} \\
& & &= \textstyle\sum_{i' \in I} Q_{i,i';j} \\
& & &\cong 0; \\
&\textit{Case } i_\ell = 1 \textit{ (where } i \in I_\ell\textit{):} & (*) &= \left(1 - \textstyle\sum_{i' \in I_\ell} x_{i'j}\right) \cdot x_{ij} \\
& & &= x_{ij} - x_{ij}^2 - \textstyle\sum_{i' \in I_\ell \setminus \{i\}} x_{i'j} x_{ij} \\
& & &= -Q_{i,j} - \textstyle\sum_{i' \in I_\ell \setminus \{i\}} Q_{i,i';j} \\
& & &\cong 0.
\end{aligned}$$

Since all derivations have degree $O(\log n)$ we have $\mathrm{aPHP}_n \leq^{\mathrm{alg}} \mathrm{rPHP}_n$ via a degree-$\tilde{O}(1)$ reduction.

10 Lifting block-degree to size

In this section, we prove Lemma 2.5 that states that $2^{\Omega(\mathrm{bdeg}_\mathsf{S}(F))} \leq \mathsf{S}(\mathrm{Lift}(F)) \leq 2^{O(\mathrm{bdeg}(\mathcal{P}))} \|\mathcal{P}\|$ where $\mathcal{P}$ is any S-refutation of F and $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$. We use the same definition of the formula $\mathrm{Lift}(F)$ as in Section 6. For Resolution this is exactly Lemma 2.3.

10.1 Upper bound for Lift(F)

To prove the upper bound $\mathsf{S}(\mathrm{Lift}(F)) \leq 2^{O(\mathrm{bdeg}(\mathcal{P}))} \|\mathcal{P}\|$ for the algebraic proof systems, we start by observing that $\mathrm{Lift}(F) \leq^{\mathrm{alg}} F$ via the degree-2 reduction $r = (r_1, \ldots, r_n)$ given by $r_i :=$ $g(x_i^0, x_i^1, s_{B(x_i)}) = x_i^0(1 - s_{B(x_i)}) + x_i^1 s_{B(x_i)}$. Note that for any polynomial p over the variables of F,

$$\|p \circ r\| \;\leq\; 3^{\mathrm{bdeg}(p)} \cdot \|p\|.$$

We first prove the upper bound for Nullstellensatz by analyzing this reduction. Let $F = \{f_1, \ldots, f_m\}$ and let $\mathcal{P} = \{p_1, \ldots, p_m\}$ be an NS-refutation of F. Recall that $\|\mathcal{P}\| = \sum_{i \in [m]} \|p_i\| \|f_i\|$. Consider the expression

$$\sum_{i \in [m]} (p_i f_i) \circ r \;=\; \sum_{i \in [m]} (p_i \circ r)(f_i \circ r) \;=\; 1,$$

which, as argued in the proof of Lemma 8.3, is a refutation of $\mathrm{Lift}(F)$. Note that the polynomial $p_i \circ r$ has at most $3^{\mathrm{bdeg}(p_i)} \cdot \|p_i\| \leq 3^{\mathrm{bdeg}(\mathcal{P})} \cdot \|p_i\|$ monomials and that $f_i \circ r$ is equal to the sum of the $2^{\mathrm{bdeg}(f_i)} = O(1)$ axioms of $\mathrm{Lift}(f_i)$, each of which has $3^{\mathrm{bdeg}(f_i)} \|f_i\| = O(\|f_i\|)$ monomials. Therefore, we can conclude there is a NS-refutation of size $\sum_{i \in [m]} 3^{\mathrm{bdeg}(\mathcal{P})} \cdot \|p_i\| \cdot O(\|f_i\|) \leq O(3^{\mathrm{bdeg}(\mathcal{P})} \|\mathcal{P}\|)$.

We now prove the upper bound for PC. Let $\mathcal{P}$ be a PC-refutation of F. We construct a PC-refutation $\mathcal{P}'$ of $\mathrm{Lift}(F)$ in the same way as done in the proof of Lemma 8.3: whenever $\mathcal{P}$ derives p, in $\mathcal{P}'$ we will derive the polynomial $p \circ r$ (which has at most $3^{\mathrm{bdeg}(p)} \|p\|$ monomials).

- *Axioms.* For any axiom $f \in F$, we noted already that the polynomial $f \circ r$ is equal to the sum of the $2^{\mathrm{bdeg}(f)} = O(1)$ axioms of $\mathrm{Lift}(f)$, each of which has $3^{\mathrm{bdeg}(f)}\|f\| = O(\|f\|)$ monomials. Thus, $f \circ r$ can be derived in PC in size $O(\|f\|)$.
- *Linear Combination.* If the polynomial p_3 is derived from p_1 and p_2 using a linear combination, then we derive $p_3 \circ r$ from $p_1 \circ r$ and $p_2 \circ r$ using the same linear combination in $\mathcal{P}'$.
- *Multiplication.* If $y_i p$ is derived from p by the multiplication rule, then we can to derive $(y_i p) \circ r = r_i(p \circ r)$ from $p \circ r$ in size $O(\|p \circ r\|)$.

Therefore, the PC-refutation has size $O(3^{\mathrm{bdeg}(\mathcal{P})}\|\mathcal{P}\|)$.

10.2 Lower bound for Lift(F)

Finally, we prove the lower bound $2^{\Omega(\mathrm{bdeg}_{\mathsf{S}}(F))} \leq \mathsf{S}(\mathrm{Lift}(F))$ for $\mathsf{S} = \mathsf{NS}, \mathsf{PC}$. This follows the random restriction argument used for Resolution exactly (Section 6), so, we merely sketch the argument. Namely, we show that $\mathrm{bdeg}_{\mathsf{S}}(F) = O(\log \|\mathcal{P}\|)$, where $\mathcal{P}$ is an algebraic proof in S. The main claim that we need (which is obvious) is that if $\mathcal{P}$ is an S-refutation of any formula G, and ρ is a partial restriction to the variables of G, then $\mathcal{P} \restriction \rho$ is an S-refutation of $G \restriction \rho$.

Letting ρ denote the same random restriction as used in the previous lower bound proof, we observe that each (twin) variable evaluates to 0 with probability at least $1/4$ under ρ. Thus, the probability that any *monomial* of block-degree $\geq d$ in $\mathcal{P}$ remains nonzero after restriction is at most $(3/4)^d$. The same union bound implies that $\mathcal{P}\restriction_\rho$ has a monomial of block-degree $\geq d$ with probability at most $\|\mathcal{P}\|(3/4)^d$, which is < 1 if $d > \log_{4/3}\|\mathcal{P}\|$. Since $\mathcal{P}\restriction_\rho$ is an S-refutation of F by the claim made above, we have that $\mathrm{bdeg}_{\mathsf{S}}(F) \leq \mathrm{bdeg}(\mathcal{P}\restriction_\rho) \leq d = O(\log\|\mathcal{P}\|)$. This completes the proof of Lemma 2.5.

11 Algebraic upper bound for TreeRef(F)

In this section, we prove Lemma 2.4*(i)* that states that $\mathrm{TreeRef}(F)$, where F is satisfiable, admits a size-$n^{O(1)}$ block-degree-$O(1)$ S-refutation for $\mathsf{S} = \mathsf{R}, \mathsf{NS}, \mathsf{PC}$. We prove this for NS, which implies the result for PC by the simulation of Fact 8.1. The result holds for R by the upper bound for $\mathrm{Ref}(F)$ (in Section 7) and the fact that $\mathrm{TreeRef}(F)$ was defined as a strengthening of $\mathrm{Ref}(F)$. Therefore, the goal of this section is to prove the following lemma.

Lemma 11.1 (Algebraic upper bound)**.** *Let F be a satisfiable n-variate formula. There is a size-$n^{O(1)}$ block-degree-$O(1)$ NS-refutation of* $\mathrm{TreeRef}(F)$ *(over any field, without twin variables).*

The proof of this lemma essentially implements the algorithm refuting $\mathrm{Ref}(F)$ (cf. Section 7.2) as an *algebraic reduction* to the *end-of-line* formula EoL_n. We proceed in three steps:

(§11.1) First we define EoL_n, which is a size-$n^{O(1)}$ block-degree-$O(1)$ CNF formula.

(§11.2) Then we reduce $\mathrm{TreeRef}(F)$ to EoL_{n^3}.

(§11.3) Finally, we recall from prior work [GKRS19] that EoL_n admits a small NS-refutation.

The last two steps are formalized in the following two claims. As we want our result to be as general as possible, our algebraic proofs will be implemented over the integers $\mathbb{Z}$ (hence the computations are valid over any field), and assume no twin variables.

Claim 11.2 (Reduction to EoL)**.** *Fix an n-variate satisfiable F. There is a block-aware, block-preserving algebraic reduction* $\mathrm{TreeRef}(F) \leq^{\mathrm{alg}} \mathrm{EoL}_{n^3}$ *of size $n^{O(1)}$.*

Claim 11.3 (Upper bound for EoL)**.** EoL_n *has a size-*$n^{O(1)}$ *block-degree-*$O(1)$ NS*-refutation over* $\mathbb{Z}$*.*

The algebraic upper bound (Lemma 11.1) follows by combining these two lemmas.

Proof of Lemma 11.1. Let r be the reduction in Claim 11.2 and let $\sum_i p_i g_i = 1$ be the NS-refutation of EoL_n given by Claim 11.3. We verify that the composed refutation $\sum_i (p_i g_i) \circ r = \sum_i (p_i \circ r)(g_i \circ r) = 1$ (discussed in Section 8.2) is a refutation of $\mathrm{TreeRef}(F)$ satisfying the lemma.

Since r is a block-preserving algebraic reduction each polynomial $g_i \circ r$ has an NS-proof from $\mathrm{TreeRef}(F)$ in size $n^{O(1)}$ and block-degree $O(1)$. So, let p_i be as above and let t be any monomial in p_i. We have $\mathrm{bdeg}(t) \le O(1)$, so when t is replaced by a product of $\mathrm{bdeg}(t)$ many r_T's (for various T's, each contained in a block of EoL_{n^3}), where each r_T has size $n^{O(1)}$ and block-degree $O(1)$, this results in a polynomial $t \circ r$ of size $(n^{O(1)})^{\mathrm{bdeg}(t)} \le n^{O(1)}$ and block-degree $\mathrm{bdeg}(t) \cdot O(1) = O(1)$. Since $\|p_i\| = n^{O(1)}$, it follows that $(p_i g_i) \circ r = (p_i \circ r)(g_i \circ r)$ has an NS proof from $\mathrm{TreeRef}(F)$ in size $n^{O(1)}$ and block-degree $O(1)$, and hence $\mathrm{TreeRef}(F)$ has an NS refutation in polynomial size and constant block-degree. $\square$

11.1 EoL formula

The end-of-line formula EoL_n states that *"there is an n-node digraph where every node has in/out-degree* 1*, except for one distinguished node that has in-degree* 0 *and out-degree* 1*."* The combinatorial principle underlying EoL_n is central in the theory of total NP search problems [Pap94, BCE+98].

The variables of EoL_n are intended to describe a digraph on vertices $[n]$ where $n \in [n]$ is thought of as a *distinguished* node. Namely, for each $i \in [n]$, there is a *block* of $2 \log n$ boolean variables $z_i \coloneqq (\overleftarrow{z}_i, \vec{z}_i)$ that encode, in binary, a *predecessor* pointer $\overleftarrow{z}_i \in [n]$ and a *successor* pointer $\vec{z}_i \in [n]$. An assignment to the variables $z = (z_1, \dots, z_n)$ defines a digraph $G_z \coloneqq ([n], E_z)$ where

$$(i,j) \in E_z \qquad \text{iff} \qquad \vec{z}_i = j \ \text{ and } \ \overleftarrow{z}_j = i.$$

A small detail is that we allow any node to be a *self-loop*, achieved by setting $\overleftarrow{z}_i = \vec{z}_i = i$.

The axioms of EoL_n are:

- *Distinguished.* The node $n \in [n]$ has $\mathrm{indeg}_{G_z}(n) = 0$ and $\mathrm{outdeg}_{G_z}(n) = 1$.
- *Non-distinguished.* Every node $i \in [n-1]$ has $\mathrm{indeg}_{G_z}(i) = \mathrm{outdeg}_{G_z}(i) = 1$.

In particular, EoL_n can be written as an $O(\log n)$-CNF formula of block-width 2. The reader familiar with pigeonhole principles can observe that our definition is equivalent to a variant of the bijective pigeonhole principle: EoL_n claims the edges of G_z define a bijection $[n] \to [n-1]$.

11.2 Reduction to EoL

Next we prove Claim 11.2: For an n-variate satisfiable F, we give a size-$n^{O(1)}$ block-aware, block-preserving algebraic reduction

$$\mathrm{TreeRef}(F) \ \le^{\mathrm{alg}} \ \mathrm{EoL}_{n^3}.$$

Intuition. The construction of the reduction closely follows the definition of the read-once branching program solving the falsified clause search problem for $\mathrm{Ref}(F)$ described in Section 7.2. Recall that the branching program implemented the following simple algorithm. First, fix a satisfying assignment x^* of F. Then, starting at the root, walk down the purported proof of F given by the assignment to $\mathrm{TreeRef}(F)$, maintaining the invariant that every clause we visit is falsified by x^*.

Since x^* is a satisfying assignment to F we are guaranteed to find a violation of some clause of $\mathrm{TreeRef}(F)$ at some point during this walk.

Our reduction is inspired by this algorithm. The i-th node in EoL_{n^3} will correspond to the block B_i in $\mathrm{TreeRef}(F)$. In particular, the distinguished node $n^3 \in [n^3]$ will correspond to the root block B_{n^3}. Given an assignment y to $\mathrm{TreeRef}(F)$, our reduction outputs an assignment to EoL_{n^3} that encodes the above walk in the purported proof encoded by y. A crucial point (one that lead to our use of $\mathrm{TreeRef}(F)$ instead of $\mathrm{Ref}(F)$) is that each node in EoL_{n^3} has both forward pointers *and* backward pointers in its definition; in our reduction these pointers correspond to the child/parent pointers of each block in $\mathrm{TreeRef}(F)$.

$\wedge$-decision trees. For ease of understanding, we describe the reduction as an $\wedge$-*decision tree*, that is, a decision tree that is allowed to query, in a single step, the logical-and $\bigwedge_{x\in A} x$ of any subset A of variables. Note that ordinary "singleton" queries are still supported by choosing A to contain a single variable. Such trees can be converted into polynomials by a standard method.

Fact 11.4. *If r is computed by a depth-d $\wedge$-decision tree, then r is computed by size-$2^{O(d)}$ polynomial.*

Proof. For each leaf ℓ in the tree, let $r_\ell(x)$ denote the indicator function that is 1 iff the leaf ℓ is reached on input x. Every query $\bigwedge_{x\in A} x$ can be simulated by the monomial $x^A := \prod_{x\in A} x$. Hence we can compute r_ℓ by taking the product along the unique path from root to ℓ of either x^A or $1-x^A$ (depending on the query outcome on the path). Hence, as a multilinear polynomial, r_ℓ satisfies $\|r_\ell\| \le 2^d$. Moreover, r can be written as $r = \sum_\ell r_\ell$ where the sum is over leaves ℓ that output 1. There are at most 2^d leaves, and thus $\|r\| \le 2^{2d}$. $\square$

Reduction. We describe a family of $\wedge$-decision trees $\mathcal{T} = (\mathcal{T}_1, \dots, \mathcal{T}_{n^3})$ where each $\mathcal{T}_i$ has depth $O(\log n)$, queries at most 4 blocks of $\mathrm{TreeRef}(F)$ and outputs values for the variables $z_i = (\overleftarrow{z}_i, \vec{z}_i)$. Our goal is to satisfy the following condition, which will imply (as we will argue below) the *Axiom* property of a block-aware reduction.

(†) *For each assignment y to* $\mathrm{TreeRef}(F)$*, if the output $\mathcal{T}(y)$ violates an axiom of* EoL_{n^3} *involving node-blocks j and j', then the execution of $\mathcal{T}_j(y)$ or $\mathcal{T}_{j'}(y)$ has witnessed* ***by its singleton queries*** *an axiom violation for* $\mathrm{TreeRef}(F)$.

Henceforth, fix a satisfying assignment x^* of F. Given an assignment y to $\mathrm{TreeRef}(F)$, we say a block B is *feasible* iff the clause encoded by B is falsified by x^*. Note that the feasibility of a given block can be decided by a single $\wedge$-query (involving n indicator variables; here we use our convention that literal indicators are set to 1 iff the literal is *not* included in the block). The tree $\mathcal{T}_i$ computes $z_i = (\overleftarrow{z}_i, \vec{z}_i)$ as follows. We start by checking whether B_i is feasible:

B_i is not feasible: Two cases depending on whether B_i is root (that is, $i = n^3$).

- *Non-root.* We make node i into a self-loop by outputting $\overleftarrow{z}_i = \vec{z}_i := i$.
- *Root.* We know that B_{n^3} contains some literal consistent with x^*. By binary search (using $O(\log n)$ many $\wedge$-queries) we can discover a specific literal indicator of B_{n^3} that is set to 0. This violates an axiom of $\mathrm{TreeRef}(F)$. Hence by (†), it is safe to output anything for $(\overleftarrow{z}_i, \vec{z}_i)$.

B_i is feasible: Query B_i's type.

- *Disabled:* If B_i is non-root, we make node i into a self-loop. If B_i is root, then we have found an axiom violation for $\mathrm{TreeRef}(F)$ (and by (†) we can output anything).

- *Axiom:* Here we can find an axiom violation. Query B_i's axiom index j. Since the j-th axiom of F is satisfied by x^*, it contains some literal ℓ consistent with x^*. But since B_i is feasible, B_i does not contain ℓ. Hence ℓ is a literal in the j-th axiom not in B_i, which is a violation.
- *Derived:* Query B_i's child pointers (j, j'), the resolved-variable index k, and the parent pointer p. Query whether B_j and $B_{j'}$ are feasible, and query their type and parent pointers. If B_i is non-root, query also the type and child pointers of B_p.

  We may assume the variables that are singleton-queried above cause no axiom violations for TreeRef(F) (as otherwise we are free to output anything). We may also assume we are in the case where exactly one of B_j and $B_{j'}$ is feasible, say B_j (otherwise we may use binary search to find a violation related to a literal indicator), and both have their parent pointers set to i. We also assume that, if B_i is non-root, then it is a child of B_p. We output $(\overleftarrow{z}_i, \vec{z}_i) := (p, j)$.

  We claim the condition (†) is satisfied: If the decision trees $\mathcal{T}_{i'}$ for $i' = j, j', p$ do not find a violation either, then they will not produce an axiom violation involving node i. Namely, they output $\overleftarrow{z}_j := i$ and $\vec{z}_p := i$ (if B_i is non-root) and the node j' will be made a self-loop.

We need to prove that this reduction is an $n^{O(1)}$-size block-aware, block-preserving algebraic reduction. First, we show that each polynomial r_T generated by the reduction has size $n^{O(1)}$ and block-degree $O(1)$. Since each $\wedge$-decision tree $\mathcal{T}_i$ has depth $O(\log n)$, by Fact 11.4 each output bit (or even the product polynomial r_T for a subset T of output bits) of $\mathcal{T}_i$ can be converted to a polynomial of size $n^{O(1)}$. Furthermore, since $\mathcal{T}_i$ queries variables from at most 4 blocks of TreeRef(F) it follows that $\mathrm{bdeg}(r_T) = O(1)$.

It remains to show that for each polynomial g encoding an axiom of EoL_{n^3}, the polynomial $g \circ r$ has an NS proof from the axioms of F in size $n^{O(1)}$ and block-degree $O(1)$. So, suppose that g is associated to node-blocks j, j' in EoL_{n^3}. There is a unique partial assignment α to the variables of g such that $g(\alpha) = 1$; this assignment falsifies the clause of EoL_{n^3} corresponding to g. Since the block-degree of g is at most two we can write $\alpha = \alpha_{T_j}\alpha_{T_{j'}}$, where $\alpha_{T_j}, \alpha_{T_{j'}}$ assign the variables in the two blocks of g. For $i = j, j'$ let $\mathcal{L}_i$ denote the leaves in the tree $\mathcal{T}_i$ that are consistent with the partial assignment α_{T_i}. We can express

$$g \circ r = \left(\sum_{\ell \in \mathcal{L}_j} r_\ell \right) \left(\sum_{\ell \in \mathcal{L}_{j'}} r_\ell \right)$$

where each polynomial r_ℓ is an indicator function for the corresponding leaf ℓ in each $\wedge$-decision tree. Since $\mathcal{T}_j, \mathcal{T}_{j'}$ are $\wedge$-decision trees, if $(g \circ r)(y) = 1$ it follows that there are leaves $\ell_1 \in \mathcal{L}_j, \ell_2 \in \mathcal{L}_{j'}$ with indicators satisfying $r_{\ell_1}(y) = 1, r_{\ell_2}(y) = 1$, and all other leaf indicators in both trees are 0. By (†), one of these two leaf indicators r_{ℓ_1}, r_{ℓ_2} must witness an axiom violation for TreeRef(F) in its singleton queries, and thus this leaf indicator is a weakening of an axiom of TreeRef(F). Ranging over all y such that $(g \circ r)(y) = 1$, this implies that $g \circ r$ can be written as a sum of weakenings of axioms of TreeRef(F). Since each $\wedge$-decision tree has depth $O(\log n)$ and queries $O(1)$ blocks from TreeRef(F) we can prove $g \circ r$ from TreeRef(F) in NS in size $n^{O(1)}$ and block-degree $O(1)$. This concludes the proof of Claim 11.2.

11.3 Upper bound for EoL

In this subsection we prove Claim 11.3. As mentioned, this was already observed by [GKRS19, Remark 4.2], and so we include the proof only for completeness.

Consider the following functions $q_i(z)$, $i \in [n]$, defined over the boolean variables of EoL_n:

$$q_i(z) \;\coloneqq\; \mathrm{indeg}_{G_z}(i) - \mathrm{outdeg}_{G_z}(i) + \delta_i \qquad \text{where} \qquad \delta_i \coloneqq \begin{cases} 1 & \text{if } i = n, \\ 0 & \text{if } i \neq n. \end{cases}$$

Each q_i can be computed by a decision tree $\mathcal{T}_i$ of depth $O(\log n)$. For example, to evaluate $\mathrm{indeg}_{G_z}(i) \in \{0,1\}$ the tree queries the pointer $\overleftarrow{z}_i$, follows it, and checks whether $\vec{z}_{\overleftarrow{z}_i} = i$. Thus, as in Fact 11.4, q_i can be computed by a degree-$O(\log n)$ polynomial $\sum_\ell r_\ell(z)$ where the sum is over leaves of $\mathcal{T}_i$ that output a non-zero value and

$$r_\ell(z) \;\coloneqq\; \begin{cases} \text{output value of } \ell & \text{if } z \text{ reaches } \ell, \\ 0 & \text{otherwise.} \end{cases}$$

Note that each ℓ that outputs a non-zero value has witnessed (by its queries) an axiom violation of EoL_n, say, an axiom encoded by the polynomial equation $a_\ell = 0$. (That is, $r_\ell(z) \neq 0$ implies $a_\ell(z) \neq 0$, or contrapositively, $a_\ell(z) = 0$ implies $r_\ell(z) = 0$.) This means that r_ℓ can be factored as $r_\ell = t_\ell a_\ell$ where t_ℓ is an arbitrary polynomial. To summarize, we can express $q_i = \sum_\ell r_\ell = \sum_\ell t_\ell a_\ell$ as a sum of polynomial combinations of axioms of EoL_n. Using the fact that, in any graph, the sum of in-degrees equals the sum of out-degrees, we have our NS-refutation:

$$\sum_{i \in [n]} q_i \;=\; \sum_{i \in [n]} \delta_i \;=\; 1.$$

Finally, we note that each q_i has block-degree 3, because any leaf of $\mathcal{T}_i$ queries at most 3 different node-blocks (itself, its potential predecessor and successor). This proves Claim 11.3.

Acknowledgements

We thank Jan Pich, Jan Krajíček, and Pavel Pudlák for comments on an early version of this manuscript. We also thank Shuo Pang, Aaron Potechin, and Madhur Tulsiani for discussions. Finally, we thank the anonymous reviewers for many comments that helped us improve the presentation.

Susanna F. de Rezende was supported by Knut and Alice Wallenberg grant KAW 2018.0371. Jakob Nordström received funding from the Swedish Research Council grant 2016-00782 and the Independent Research Fund Denmark grant 9040-00389B. Toniann Pitassi did this work supported by NSERC and NSF Grant CCF-1900460. Robert Robere was supported by NSERC, the Charles Simonyi Endowment, and indirectly supported by the National Science Foundation Grant No. CCF-1900460. This material is based on work supported by NSERC and NSF Grant CCF-1900460. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[AB04] Albert Atserias and Maria Luisa Bonet. On the automatizability of resolution and related propositional proof systems. *Information and Computation*, 189(2):182–201, 2004. doi:10.1016/j.ic.2003.10.004.

[ABMP01] Michael Alekhnovich, Sam Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *Journal of Symbolic Logic*, 66(1):171–191, 2001. doi:10.2307/2694916.

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. doi:10.1137/S0097539700366735.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008. doi:10.1016/j.jcss.2007.06.025.

[AH19] Albert Atserias and Tuomas Hakoniemi. Size-degree trade-offs for sums-of-squares and positivstellensatz proofs. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, volume 137, pages 24:1–24:20, 2019. doi:10.4230/LIPIcs.CCC.2019.24.

[AM20] Albert Atserias and Moritz Müller. Automating resolution is NP-hard. *Journal of the ACM*, 67(5), 2020. doi:10.1145/3409472.

[AR08] Michael Alekhnovich and Alexander Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, 2008. doi:10.1137/06066850X.

[BCE+98] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.

[BCIP02] Joshua Buresh-Oppenheim, Matthew Clegg, Russell Impagliazzo, and Toniann Pitassi. Homogenization and the polynomial calculus. *Computational Complexity*, 11(3-4):91–108, 2002. Preliminary version in *ICALP '00*. doi:10.1007/s00037-002-0171-6.

[BDG+04] Maria Luisa Bonet, Carlos Domingo, Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi. Non-automatizability of bounded-depth Frege proofs. *Computational Complexity*, 13(1-2):47–68, 2004. doi:10.1007/s00037-004-0183-5.

[Bel20] Zoë Bell. Automating regular or ordered resolution is NP-hard. Technical Report TR20-105, Electronic Colloquium on Computational Complexity (ECCC), 2020. URL: https://eccc.weizmann.ac.il/report/2020/105/.

[Ber18] Christoph Berkholz. The relation between polynomial calculus, Sherali-Adams, and sum-of-squares proofs. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 96, pages 11:1–11:14, 2018. doi:10.4230/LIPIcs.STACS.2018.11.

[BG01] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001. doi:10.1007/s000370100000.

[BGIP01] Sam Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267–289, 2001. doi:0.1006/jcss.2000.1726.

[BIK+94] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. In *Proceedings of the 35th Symposium on Foundations of Computer Science (FOCS)*, pages 794–806, 1994. doi:10.1109/SFCS.1994.365714.

[BKS04] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004. doi:10.1613/jair.1410.

[BKS15] Boaz Barak, Jonathan Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the 47th Symposium on Theory of Computing (STOC)*, pages 143–151, 2015. doi:10.1145/2746539.2746605.

[BPR97] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. No feasible interpolation for TC^0-frege proofs. In *Proceedings of the 38th Symposium on Foundations of Computer Science (FOCS)*, pages 254–263, 1997. doi:10.1109/SFCS.1997.646114.

[BPR00] María Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for Frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000. doi:10.1137/S0097539798353230.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

[CEI96] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Symposium on Theory of Computing (STOC)*, pages 174–183, 1996. doi:10.1145/237814.237860.

[DR03] Stefan Dantchev and Søren Riis. On relativisation and complexity gap for resolution-based proof systems. In *Computer Science Logic*, pages 142–154. Springer, 2003. doi:10.1007/978-3-540-45220-1_14.

[dRMN+19] Susanna de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. Technical Report TR19-186, Electronic Colloquium on Computational Complexity (ECCC), 2019. URL: https://eccc.weizmann.ac.il/report/2019/186/.

[dRNV16] Susanna de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, 2016. doi:10.1109/FOCS.2016.40.

[FKP19] Noah Fleming, Pravesh Kothari, and Toniann Pitassi. Semialgebraic proofs and efficient algorithm design. *Foundations and Trends in Theoretical Computer Science*, 14(1-2):1–221, 2019. doi:10.1561/0400000086.

[Gar19] Michal Garlík. Resolution lower bounds for refutation statements. In *Proceedings of the 44th Mathematical Foundations of Computer Science (MFCS)*, volume 138, pages 37:1–37:13, 2019. doi:10.4230/LIPIcs.MFCS.2019.37.

[Gar20] Michal Garlík. Failure of feasible disjunction property for k-DNF resolution and NP-hardness of automating it. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2020. URL: https://eccc.weizmann.ac.il/report/2020/037/.

[GGKS18] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Symposium on Theory of Computing (STOC)*, pages 902–911. ACM, 2018. doi:10.1145/3188745.3188838.

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating cutting planes is NP-hard. In *Proceedings of the 52nd Symposium on Theory of Computing (STOC)*, 2020. To appear.

[GKRS19] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 38:1–38:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

[GL10a] Nicola Galesi and Massimo Lauria. On the automatizability of polynomial calculus. *Theory of Computing Systems*, 47(2):491–506, 2010. doi:10.1007/s00224-009-9195-5.

[GL10b] Nicola Galesi and Massimo Lauria. Optimality of size-degree tradeoffs for polynomial calculus. *ACM Transactions on Computational Logic*, 12(1), 2010. doi:10.1145/1838552.1838556.

[GP18] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

[HKP+17] Samuel Hopkins, Pravesh Kothari, Aaron Potechin, Prasad Raghavendra, Tselil Schramm, and David Steurer. The power of sum-of-squares for detecting hidden structures. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 720–731, 2017. doi:10.1109/FOCS.2017.72.

[HN12] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time–space trade-offs in proof complexity. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 233–248. ACM, 2012. doi:10.1145/2213977.2214000.

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999. doi:10.1007/s000370050024.

[Iwa97] Kazuo Iwama. Complexity of finding short resolution proofs. In *Mathematical Foundations of Computer Science (MFCS)*, pages 309–318. Springer, 1997. doi:10.1007/BFb0029974.

[Jeř07] Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *Journal of Logic and Computation*, 17(3):587–604, 2007. doi:10.1093/logcom/exm017.

[Juk12] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

[KP98] Jan Krajícek and Pavel Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998. doi:10.1006/inco.1997.2674.

[KSS18] Pravesh Kothari, Jacob Steinhardt, and David Steurer. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Symposium on Theory of Computing (STOC)*, pages 1035–1046, 2018. doi:10.1145/3188745.3188970.

[Las01] Jean Lasserre. An explicit exact SDP relaxation for nonlinear 0–1 programs. In *Proceedings of the 8th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 293–303, 2001. doi:10.1007/3-540-45535-3_23.

[LN17a] Massimo Lauria and Jakob Nordström. Graph colouring is hard for algorithms based on Hilbert's nullstellensatz and Gröbner bases. In *Proceedings of the 32nd Computational Complexity Conference (CCC)*, pages 2:1–2:20, 2017. doi:10.4230/LIPIcs.CCC.2017.2.

[LN17b] Massimo Lauria and Jakob Nordström. Tight size-degree bounds for sums-of-squares proofs. *Computational Complexity*, 26(4):911–948, 2017. doi:10.1007/s00037-017-0152-4.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MPW19] Ian Mertz, Toniann Pitassi, and Yuanhao Wei. Short proofs are hard to find. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 84:1–84:16. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.ICALP.2019.84.

[MS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

[MSS16] Tengyu Ma, Jonathan Shi, and David Steurer. Polynomial-time tensor decompositions with sum-of-squares. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 438–446, 2016. doi:10.1109/FOCS.2016.54.

[O'D17] Ryan O'Donnell. SOS is not obviously automatizable, even approximately. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 67, pages 59:1–59:10. Schloss Dagstuhl, 2017. doi:10.4230/LIPIcs.ITCS.2017.59.

[OS19] Ryan O'Donnell and Tselil Schramm. Sherali–Adams strikes back. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, pages 8:1–8:30, 2019. doi:10.4230/LIPIcs.CCC.2019.8.

[OZ13] Ryan O'Donnell and Yuan Zhou. Approximability and proof complexity. In *Proceedings of the 24th Symposium on Discrete Algorithms (SODA)*, pages 1537–1556, 2013. doi:10.1137/1.9781611973105.111.

[Pap94] Christos Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

[Par00] Pablo Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000.

[Pot20] Aaron Potechin. Sum of squares bounds for the ordering principle. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, volume 169, pages 38:1–38:37, 2020. doi:10.4230/LIPIcs.CCC.2020.38.

[PS12] Toniann Pitassi and Nathan Segerlind. Exponential lower bounds and integrality gaps for tree-like Lovász–Schrijver procedures. *SIAM Journal on Computing*, 41(1):128–159, 2012. doi:10.1137/100816833.

[PT19] Pavel Pudlák and Neil Thapen. Random resolution refutations. *Computational Complexity*, 28(2):185–239, 2019. doi:10.1007/s00037-019-00182-7.

[Pud00] Pavel Pudlák. Proofs as games. *The American Mathematical Monthly*, 107(6):541–550, 2000. doi:10.2307/2589349.

[Pud03] Pavel Pudlák. On reducibility and symmetry of disjoint NP pairs. *Theoretical Computer Science*, 295:323–339, 2003. doi:10.1016/S0304-3975(02)00411-5.

[Raz98] Alexander Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7:291–324, 1998. doi:10.1007/s000370050013.

[RW17] Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 80:1–80:13, 2017. doi:10.4230/LIPIcs.ICALP.2017.80.

[SA94] Hanif Sherali and Warren Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero–one programming problems. *Discrete Applied Mathematics*, 52(1):83–106, 1994. doi:10.1016/0166-218X(92)00190-W.

[Sho87] Naum Shor. An approach to obtaining global extremums in polynomial mathematical programming problems. *Cybernetics*, 23(5):695–700, 1987. doi:10.1007/BF01074929.

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