Automating Algebraic Proof Systems is NP-Hard

Susanna F. de Rezende
Institute of Mathematics of the Czech Academy of Sciences

Mika Gōōs†
Stanford University

Jakob Nordström
University of Copenhagen & Lund University

Toniann Pitassi
University of Toronto & IAS

Robert Robere
IAS

Dmitry Sokolov
St. Petersburg State University & PDMI RAS

May 1, 2020

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, Polynomial Calculus, or Sherali–Adams 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 (FOCS 2019) that established an analogous result for Resolution.

†Part of the work done while at Institute for Advanced Study.
1 Introduction

Automatability. A proof system $S$ is 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) are based on the most basic propositional proof system, Resolution ($Res$ 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 $[Iwa97, ABMP01, AB04, AR08, MPW19, AM19]$) 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 any of the following standard systems:

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

An important proof system that is missing above, and for which we still leave open the question of its automatability, is

- (SoS) Sum-of-Squares $[Sho87, Par00, Las01]$.

1.1 Our result

For the aforementioned proof systems (excluding SoS), our main result shows that it is NP-hard to approximate the minimum refutation size up to a factor of $2^{n^\epsilon}$ for some constant $\epsilon > 0$. In particular, these proof systems are not automatable unless $P = NP$. We defer the standard definitions of the algebraic proof systems to Section 7. 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 $A$ that on input an $n$-variate 3-CNF formula $F$ outputs a CNF formula $A(F)$ such that for any system $S = Res, NS, PC, SA$:

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

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

Previously, Galesi and Lauria $[GL10a]$, building on $[AR08]$, proved that $NS$ and $PC$ are not automatable unless the fixed parameter hierarchy collapses. Our Theorem 1.1 upgrades this to an optimal hardness assumption, namely $P \neq NP$. For $SA$, no previous non-automatability results
were known. 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 $\text{poly}(\log n)$ factors. All these algorithms are based on general size–degree tradeoffs [CEI96, PS12, AH19].

**Techniques.** Our proof builds on the recent breakthrough of Atserias and Müller [AM19] that showed that automating Resolution is NP-hard. Namely, they proved Theorem 1.1 for $S = \text{Res}$. 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. See Section 2 for a detailed overview of our techniques.

### 1.2 Related work

**Degree-automatability.** Algebraic proof systems are central in an exciting body of research that exploits their degree-automatability (as opposed to size-automatability), which is the ability to find proofs of low degree efficiently. For our four systems, 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-automatability yields a meta-approach for discovering new algorithms for search problems. Namely, one starts by certifying the existence of a solution by a low-degree proof, and then applies degree-automatability 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 they way non-automatability results are proved: The formula $A(F)$ 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 Res [BG01]; for NS it is implicit in [BCIP02]; and for PC [GL10b]. None are known for SoS so far.

**Other proof systems.** For standard textbook-style proof systems (Frege and Extended Frege) automatability is equivalent to possessing feasible interpolation. More specifically, for any proof system, automatability implies feasible interpolation, and for sufficiently strong proof systems (that admit short proofs of their soundness), the converse holds. Under cryptographic assumptions, Frege, Extended Frege, and bounded-depth Frege systems are known to not have feasible interpolation and therefore are not automatable [KP98, BPR97, BDG+04].

By contrast, for weak systems that cannot reason about their own soundness (Res, NS, PC, SA, SoS), deciding whether they are automatable has proven more challenging. Until the recent breakthrough by Atserias and Müller [AM19], even the automatability of Resolution was unresolved. In an 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 had been a longstanding question (until [AM19]) whether or not this upper bound could be improved. Following in the wake of Atserias and Müller, other weak systems were shown NP-hard to automate: [GKMP20] proved it for Cutting Planes, and [Gar20] for $k$-DNF Resolution.

2 Proof Overview

Our proof builds directly on the breakthrough of Atserias and Müller [AM19]. In this section:

(§2.1) We recall 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 Sections 3–6.

(§2.3) We outline why our simplified proof generalizes, with some additional work, to the setting of algebraic proof systems. The details appear in Sections 7–10.

Readers who only care about our simplified proof of Atserias–Müller are in luck: We have organized the paper so that the initial Sections 3–6 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 $P$ of $F$ is a sequence of clauses $P = (C_1, \ldots, C_s)$ 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 $\|P\| := s$. The Resolution size complexity of $F$, denoted $\text{Res}(F)$, is the least size of a Resolution refutation of $F$. Another important complexity measure of $P$ is its width $w(P)$ defined as the maximum width $|C|$ of any of its clauses $C \in P$. Define also the width complexity $w(F \vdash \bot)$ of a formula $F$ as the least width of a Resolution refutation of $F$.

For visualization purposes, a refutation $P$ 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 Simpler Atserias–Müller

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 $\text{Ref}(F)$. In the second step, this formula is “lifted” to produce $\text{Lift}(\text{Ref}(F))$, which is then output by $\mathcal{A}$. We explain these two steps in detail.
Step 1: Block-width

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

$$\text{Ref}(F) \equiv \text{"F admits a short dag-like Resolution refutation."}$$

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

The key property of $\text{Ref}(F)$ is that its block-width 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 $\text{Ref}(F)$ such that

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

Simplification. We simplify the proof of the block-width lower bound in case (ii) of Lemma 2.1. (We do not simplify the upper bound (i), although we do improve it in other ways in Section 2.3.) 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 of (ii) is by a mere reduction from the usual pigeonhole principle. We define (Section 3.3) a convenient, somewhat non-standard encoding of the principle, sometimes called the retraction weak pigeonhole principle [Jer07, PT19]. This encoding, denoted $\text{rPHP}_m$, is an $O(\log m)$-width CNF that claims there exists an efficiently invertible injection, encoded in binary, from $2^m$ pigeons to $m$ holes. Our reduction (Section 5) translates, with modest loss, width lower bounds for $\text{rPHP}_{n^2}$ into block-width lower bounds for $\text{Ref}(F)$.

Lemma 2.2. $bw(\text{Ref}(F \vdash \bot)) \geq \tilde{\Omega}(w(\text{rPHP}_{n^2} \vdash \bot)/n)$ for any $n$-variate unsatisfiable formula $F$.

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

$$w(\text{rPHP}_m \vdash \bot) \geq \Omega(m).$$

This lower bound and Lemma 2.2 imply that $bw(\text{Ref}(F \vdash \bot)) \geq \tilde{\Omega}(n)$, which proves (ii).

Step 2: Lifting

The goal of the second step is to transform the block-width gap in Lemma 2.1 into a size gap. A popular way to achieve this is via lifting, although Atserias and Müller used a related relativization technique; 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 in lifting 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 $\text{Lift}(F)$, which we then show is hard in a strong sense (for us, Resolution size).
**Block lifting.** We prove (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 (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 $\text{Lift}(F)$ such that
\[2^{\Omega(bw(F) - 1)} \leq \text{Res}(\text{Lift}(F)) \leq 2^{O(bw(F))} \cdot ||P||,\]

where $P$ 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 $A(F) := \text{Lift}(\text{Ref}(F))$ satisfies Theorem 1.1 for Resolution. This completes our simplified proof of the non-automatability of Resolution.

### 2.3 Generalization

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

**Terminology.** The algebraic proof systems are defined carefully in Section 7. 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 a $S$-refutation $P$ is denoted $\deg(P)$; the minimum degree of a $S$-refutation of a formula $F$ is denoted $\deg_S(F \vdash \bot)$. Moreover, we define the block-degree $b\deg(r)$ 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 $S(F)$ to denote the least size $||P||$ (number of monomials in $P$) of an $S$-refutation $P$ of $F$.

**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 $\text{Ref}(F)$ with a tree-like variant $\text{TreeRef}(F)$, discussed shortly. Our main result (Theorem 1.1) follows by considering $A(F) := \text{Lift}(\text{TreeRef}(F))$ and applying the improved lemmas. The remainder of this section discusses how to prove them.

**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 $\text{TreeRef}(F)$ such that for systems $S = \text{Res}, \text{NS}, \text{PC}, \text{SA}$:

(i) If $F$ is satisfiable, then $\text{TreeRef}(F)$ admits a size-$n^{O(1)}$ block-degree-$O(1)$ $S$-refutation.

(ii) If $F$ is unsatisfiable, then $\text{TreeRef}(F)$ 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 $\text{Lift}(F)$ such that for systems $S = \text{Res}, \text{NS}, \text{PC}, \text{SA}$:
\[2^{\Omega(b\deg_S(F) - 1)} \leq S(\text{Lift}(F)) \leq 2^{O(b\deg(P))} \cdot ||P||,\]

where $P$ is any $S$-refutation of $F$. 5
Upper bound (i). The first challenge in generalizing the proof for Resolution is that we do not know whether \( \text{Ref}(F) \) for a satisfiable \( F \) admits a small Nullstellensatz refutation (we suspect not). This is why we introduce (Section 3.2) a new tree-like variant of the formula that intuitively says

\[
\text{TreeRef}(F) \equiv \text{"F admits a short tree-like Resolution refutation, whose non-leaves do not use weakening."}
\]

This formula is a \textit{weakening} of \( \text{Ref}(F) \) meaning that it is obtained from \( \text{Ref}(F) \) 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 \( \text{Ref}(F) \), and for other systems they follow by simulations. See Section 9 for the proof of Lemma 2.4 (i).

Lower bound (ii). Our simplified proof established the block-width lower bound for \( \text{Ref}(F) \) by a reduction from \( r\text{PHP}_{n^2} \). In fact, the same reduction works even for \( \text{TreeRef}(F) \) without modification. Moreover, it is known that pigeonhole formulas require large degree for PC [Raz98] and SA [GM08]. We show, via low-degree reductions, that these degree lower bounds apply also to our \( r\text{PHP}_m \) encoding, and hence to \( \text{TreeRef}(F) \). See Section 10 for the proof of Lemma 2.4 (ii).

Lifting block-degree. Algebraic proofs are equally amenable to analysis via random restrictions (key technique behind the proof of Lemma 2.3) as Resolution. Hence it is straightforward to strengthen Lemma 2.3 to Lemma 2.5. See Section 8 for the proof.

3 Formulas

In this section we define formulas that will be relevant throughout the paper. In (§3.1) we recall the Atserias–Müller [AM19] construction of the formula \( \text{Ref}(F) \); in (§3.2) we modify \( \text{Ref}(F) \) to obtain our tree-like variant, \( \text{TreeRef}(F) \); and finally in (§3.3) we define a convenient version of the usual pigeonhole principle.

3.1 \text{Ref}(F) formula

Fix a CNF formula \( F \) with variables \( x_1, \ldots, x_n \) and \( m = \text{poly}(n) \) clauses. We define \( \text{Ref}(F) \) [AM19] that informally states \textit{"F admits a short dag-like Resolution refutation."} In preparation for our improved upper bound in Section 9, our definition of \( \text{Ref}(F) \) differs slightly from the original.

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

- \textit{Literal set.} There are \( 2n \) many indicator variables \( y_\ell \) for the literals \( \ell \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\} \) of \( F \). A boolean assignment to the \( y_\ell \) is intended to define the set of literals for the clause represented by \( B_i \). As a minor detail (relevant in Section 9), we interpret \( y_\ell = 0 \) to mean that literal \( \ell \) is included in the block.

- \textit{Block type.} There are two boolean variables encoding the block’s \textit{type}: either \textit{axiom}, \textit{derived}, or \textit{disabled}. Accordingly, one of the following groups of variables become relevant.

  (1) \textit{Axiom.} There are \( \log m \) many variables that encode an \textit{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 triple $(j, j', k) \in [n^3] \times [n^3] \times [n]$. 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 $F$. A formal treatment was given by Atserias and Müller [AM19]. Here we recall the axioms informally:

- **Root.** We require that the last block $B_{n^3}$ (root of the dag) is not disabled and that it represents the empty clause. That is, all literal indicator variables are set to 1.
- **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 not disabled 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.

In conclusion, $\text{Ref}(F)$ can be written as an $O(\log n)$-CNF formula with poly($n$) clauses of block-width $\leq 3$ (the worst case is an axiom for a derived block that involves its two children).

### 3.2 TreeRef($F$) formula

Next, we define a tree-like version of Ref($F$) that informally states “$F$ admits a short tree-like Resolution refutation, whose non-leaves do not use weakening.” Indeed, TreeRef($F$) is obtained by starting from 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 non-disabled block $B_i$, we require that its parent $B_p$ is a 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$_n$ [Jer07, 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, $f: [2n] \rightarrow [n]$ and $g: [n] \rightarrow [2n]$, 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 $j$.  

7
The axioms of rPHP$_n$ state that for every $i \in [2n]$ and $j \in [n],$

$$f(i) = j \implies g(j) = i.$$  \hspace{1cm} (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 in the variables $(f, g) = (f_{ik}, g_{j\ell}).$

## 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 $bw(\text{Ref}(F) \vdash \bot)$. We assume the reader is familiar with the standard notion of a decision tree computing a boolean function $f : \{0, 1\}^n \to \{0, 1\}$ (see, e.g., the textbook [Juk12, §14]). In particular, a depth-$d$ decision tree $T$ 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 $\ell$ ranges over the leaves of $T$ that output 1, and $C_{\ell}$ is the conjunction of literals (query outcomes) on the path from root to leaf $\ell$. The $d$-CNF is obtained by negating the $d$-DNF associated with the negated decision tree $\neg T$ (that is, $T$ but with its output values flipped) computing $\neg f$.

### 4.1 What is a reduction?

A decision tree reduction between formulas $F$ and $G$ consists of 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 in the following.

**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 $F \leq_{dt} G$, consists of the following.

- **Variables.** The reduction is defined by a function $f : \{0, 1\}^n \to \{0, 1\}^m$ such that each output bit $f_i : \{0, 1\}^n \to \{0, 1\}$ (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 $C : \{0, 1\}^m \to \{0, 1\}$. Consider the composed function $C \circ f$. It can be computed by a depth-$d|C|$ decision tree, and hence we may naturally write it as a $d|C|$-CNF. 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 $F \leq_{dt} G$, then $w(F \vdash \bot) \leq d \cdot w(G \vdash \bot)$.

This lemma is most elegantly proven using the standard game semantics (or top-down) characterization of $w(F \vdash \bot)$ [Pud00, AD08]. We recall this game briefly.

**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 $\rho \in \{0, 1, \ast\}^n$ to the variables of $F$. The game starts with the empty assignment $\rho = \ast^n$. In each round:

1. **Query a variable.** Prover chooses an $i \in [n]$ with $\rho_i = \ast$, after which Adversary chooses $b \in \{0, 1\}$. The state is updated by $\rho_i \leftarrow b$. 


2. Forget variables. Prover chooses a 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 $\rho$ falsifies an axiom of $F$. The width complexity $w(F \vdash \bot)$ 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 $F \leq_{dt}^{d} \ F$ is computed by $f$. Let $G$ be a width-$w$ Prover strategy for $F$. We construct a width-$dw$ Prover strategy $\mathcal{F}$ for $F$ by simulating $G$ round-by-round. We maintain the invariant that if the game state for $G$ (partial assignment to $y$) records a value $y_i = b$ for some $b \in \{0,1\}$, then the game state $\rho$ 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 $\rho$. The simulation proceeds as follows. In each round:

1. $G$ 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 $G$ by responding $y_i = b$ (that is, we play the role of Adversary for $G$).

2. $G$ 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 $G$’s game state.

These actions keep the width of $\mathcal{F}$ at most $dw$. When the game ends for $G$, we claim it does so for $\mathcal{F}$: If the state for $G$ 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 $F(x) \leq_{dt}^{d} G(y)$ via $f : \{0,1\}^n \rightarrow \{0,1\}^m$ as in Definition 4.1. Suppose further that the variables $y = (y_1, \ldots, y_m)$ of $G$ are partitioned into blocks. We say that the reduction $F \leq_{dt}^{d} G$ is block-aware if for each block $B \subseteq [m]$ there is a depth-$d$ decision tree that computes all the values $f_B(x) := (f_i(x) : i \in B) \in \{0,1\}^B$ simultaneously.

**Lemma 4.4.** If $F \leq_{dt}^{d} G$ via a block-aware reduction, then $w(F \vdash \bot) \leq d \cdot bw(G \vdash \bot)$.

**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 $G$ 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 $G$ has block-width $w$, then $\mathcal{F}$ has width at most $dw$.

**5 The Reduction**

In this section, we prove Lemma 2.2 that states that $bw(Ref(F) \vdash \bot) \geq \Omega(w(rPHP_{n^2} \vdash \bot)/n)$, where $F$ is any unsatisfiable $n$-variate CNF formula, and $Ref(F)$ and $rPHP_{n^2}$ are as defined in Sections 3.1 and 3.3, respectively. Our goal is to describe a block-aware reduction

$$rPHP_{n^2} \leq_{dt}^{d} \ Ref(F).$$

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

As in the original proof of Atserias and Müller [AM19], our reduction is guided by the full tree-like Resolution refutation $T$ of the unsatisfiable formula $F$. More specifically, $T$ 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 $T$ has $2^n$ leaves corresponding to all possible width-$n$ clauses; each such leaf clause is a weakening of an axiom of $F$.

For any truth assignment to $\text{rPHP}_n^2$ our reduction is going to produce an assignment to $\text{Ref}(F)$ that represents a purported refutation of $F$ isomorphic to a subtree $T'$ of the full tree $T$. We note that $T'$ will not be a valid refutation of $F$, because some nodes on the “boundary” of the embedding $T' \subseteq T$ are missing a child. However, the interior “local neighborhoods” of $T'$ will be indistinguishable from the corresponding neighborhoods of $T$, and those parts do not violate any axioms of $\text{Ref}(F)$. The only axiom violations of $\text{Ref}(F)$ result from the “boundary” nodes.

We now describe the reduction in detail with heavy reference to Figure 2.

5.2 Variables

We start by defining how the variables of $\text{Ref}(F)$ depend on the variables of $\text{rPHP}_n^2$. We think of the blocks of $\text{Ref}(F)$ as being arranged in $n + 1$ layers with layer $\ell \in \{0, 1, \ldots, n\}$ 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.

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 $T$ on these top-most layers). We say this part of the pointer assignment is hardcoded, as it does not depend on the variables of $\text{rPHP}_n^2$.

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 $(f, g)$ of $\text{rPHP}_n^2$ 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) $h: [2n^2] \to [n^2] \cup \{\ast\}$ by

$$h(i) := \begin{cases} f(i) & \text{if } g(f(i)) = i, \\ \ast & \text{otherwise.} \end{cases} \quad (3)$$

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

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 $\ell$ 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 $\ell$ as the $h(2i - 1)$-th (resp. $h(2i)$-th) block on layer $\ell + 1$. If ever $h(i)$ is
Figure 2: Reduction from rPHP$_{n^2}$ to Ref($F$). An assignment to the variables of rPHP$_{n^2}$ defines a partial matching $h$: $[2n^2] \rightarrow [n^2]$ (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 $T$. 
undefined (meaning an axiom of $\text{rPHP}_{n^2}$ 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 $\text{Ref}(F)$).

This completes the definition of the topology of the purported refutation described by the variables of $\text{Ref}(F)$. 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).

### Literal sets.

Recall that our overarching 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) become forced. Indeed, we describe an 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 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 $\text{Ref}(F)$ depend on the variables of $\text{rPHP}_{n^2}$. 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 Axioms

It remains to show that the axioms of $\text{rPHP}_{n^2}$ imply those of $\text{Ref}(F)$. We argue the contrapositive: any axiom violation for $\text{Ref}(F)$ implies an axiom violation for $\text{rPHP}_{n^2}$. 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 $\text{Ref}(F)$), 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 $h(i) = \ast$. But this means the decision tree has witnessed a violation of (1), that is, an axiom violation for $\text{rPHP}_{n^2}$, by the discussion following (3). This completes the reduction (2).

### 5.4 Tree-like extension

To conclude this section, we observe for later use (namely, in Section 10) that the reduction described above can be easily extended to a block-aware reduction

$$\text{rPHP}_{n^2} \preceq_{\tilde{O}(n)}^{dt} \text{TreeRef}(F).$$

Indeed, 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 $\text{rPHP}_{n^2}$ imply those of $\text{TreeRef}(T)$, we argue similarly as in Section 5.3: Since $\mathcal{T}$ is a tree-like refutation that uses no weakening (except at the axioms), the output of our reduction (subtree $\mathcal{T}'$ of $\mathcal{T}$) still has its axiom violations only at the “boundaries” of the embedding $\mathcal{T}' \subseteq \mathcal{T}$.
We start by describing how the formula \( \text{Lift}(F) \) is constructed. First, for a literal \( \ell \), we have \( \text{Lift}(\ell) := g(x^0, x^1, s) := x^s \). Note that \( g \) is computed by a depth-2 decision tree. We now define \( \text{Lift}(F) \) 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, \( \text{Lift}(F) \) has \( 2n + m \) variables, called lifted variables.

- **Axioms.** Let \( C \in F \) be a clause and view it as a function \( C : \{0,1\}^n \rightarrow \{0,1\} \). We define a lifted constraint \( \text{Lift}(C) : \{0,1\}^{2n+m} \rightarrow \{0,1\} \) over the lifted variables as the composition
  \[
  \text{Lift}(C) := C(g(x_1^0, x_1^1, s_B(x_1)), \ldots, g(x_n^0, x_n^1, s_B(x_n))),
  \]
  where \( B(x_i) \) denotes the unique block containing \( x_i \). Note that \( \text{Lift}(C) \) 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 \( d := |C| + \text{bw}(C) \) as the gadgets share selector variables. Hence we may write \( \text{Lift}(C) \) naturally as a \( d \)-CNF (as discussed in Section 4). Finally, we define \( \text{Lift}(F) := \bigwedge_{C \in F} \text{Lift}(C) \).

For concreteness, let us be more explicit about what the CNF expressing \( \text{Lift}(C) \) is by inspecting the construction. First, for a literal \( \ell \) (that is, \( x_i \) or \( \overline{x_i} \)), understood as a singleton clause, we have (using \( -g(x^0, x^1, s) = g(-x^0, -x^1, s) \) in case \( \ell \) is a negated literal):

\[
\text{Lift}(\ell) = g(\ell^0, \ell^1, s_B(\ell)) = (s_{B(\ell)} \lor \ell_0) \land (\overline{s}_{B(\ell)} \lor \ell_1).
\]

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

Let us prove the upper bound $\text{Res}(\text{Lift}(F)) \leq 2^{O(\text{bw}(P))}\|P\|$. 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 several runs of the game). Thus let $P$ be a Prover strategy for $F$ of size $\|P\|$ and block-width $\text{bw}(P)$. Our goal is to find a small-size strategy $L$ for Lift($F$).

We start by observing that Lift($F$) $\leq_2^{\text{dt}} F$ via $f = (f_1, \ldots, f_n)$ given by $f_i := g(x_i^0, x_i^1, s_B(x_i))$. The strategy $L$ is then constructed by simulating $P$ as in the proof of Lemma 4.2. We proceed to bound $\|L\|$ by analyzing the simulation carefully. At the start of a simulation round, if $P$ is in state $\rho$, then $L$ is in one of $2^{\text{bw}(\rho)}$ many corresponding states; here the blow-up $2^{\text{bw}(\rho)}$ comes from having to record the values of $\text{bw}(\rho)$ many selector variables. During a simulation step, $L$ 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 $O(2^{\text{bw}(\rho)})$ overhead in a single round of the simulation. Altogether we get $\|L\| \leq O(2^{\text{bw}(P)})\|P\|$, which proves the upper bound.

6.3 Lower bound for Lift(F)

Finally, we prove the lower bound, namely that $2^{O(\text{bw}(F \upharpoonright \bot))} \leq \text{Res}(\text{Lift}(F))$. We show an equivalent claim: $\text{bw}(F \upharpoonright \bot) \leq O(\log \|P\|)$ for any refutation $P$ of Lift($F$). Fix such a $P$ henceforth.

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

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

1. Let $\rho_1$ be a random restriction setting each selector variable $s_B$ to a uniform random bit.
2. Define $X_{\rho_1}$ 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 $\rho_1$. Let $\rho_2$ be a random restriction setting each variable in $X_{\rho_1}$ to a uniform random bit. Let $\rho$ be the concatenation of $\rho_1$ and $\rho_2$.

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 $\geq w$ is not satisfied by $\rho$ is at most $(3/4)^w$. Consider the restricted refutation $P \upharpoonright \rho$. By a union bound,

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

For $w := 3 \log \|P\|$ this probability is $< 1$, and hence there exists some fixed $\rho$ such that $\text{bw}(P \upharpoonright \rho) \leq w$. But $P \upharpoonright \rho$ is a refutation of the formula Lift($F$) $\upharpoonright \rho$, which is the same as $F$ after renaming variables. Hence, $\text{bw}(F \upharpoonright \bot) \leq w = O(\log \|P\|)$, which completes the proof of Lemma 2.3.

7 Algebraic Proof Systems

In this section, we define: ($\S$7.1) algebraic systems NS, PC, SA; and ($\S$7.2) algebraic reductions.
7.1 Definitions

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 \( X := \{x_1, x_2, \ldots, x_n\} \) 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.

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 \( \bigvee_{i \in I} x_i \lor \bigwedge_{j \in J} \bar{x}_j \) of \( F \) we include in \( F^* \) the equation

\[
\prod_{i \in I} (1 - x_i) \prod_{j \in J} x_j = 0.
\]

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 \( P = \{p_1, \ldots, p_m\} \) such that, as formal polynomials,

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

The size of the proof is \( \|P\| := \sum_{i \in [m]} \|p_i\| \|f_i\| \) and its degree is \( \deg(P) := \max_{i \in [m]} \deg(p_i f_i) \). 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 \( P = (p_1, \ldots, p_s) \) 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 \( xp_j \) for any \( x \in X \).

The size of the proof is \( \|P\| := \sum_{i \in [s]} \|p_i\| \) and its degree is \( \deg(P) := \max_{i \in [s]} \deg(p_i) \). A PC-refutation of \( F \) is a PC-proof of \( 1 = 0 \) from \( F \).

**Sherali–Adams (SA).** Sherali–Adams is a static, (semi-)algebraic proof system that is based on the Sherali–Adams hierarchy of LP relaxations. The system is only defined over real numbers, \( \mathbb{F} = \mathbb{R} \). An SA-proof of \( f \geq 0 \) from a set of polynomial equations \( F = \{f_1 = 0, \ldots, f_m = 0\} \) is a set of polynomials \( P = \{p_1, \ldots, p_m, q\} \) such that

\[
\sum_{i \in [m]} p_i f_i + q = f,
\]

and where \( q \) is a conical junta, that is, of the form

\[
q = \sum_{I,J} \alpha_{I,J} \prod_{i \in I} x_i \prod_{j \in J} (1 - x_j)
\]

where \( \alpha_{I,J} \geq 0 \) are non-negative reals. The size of the proof is \( \|P\| := \|q\| + \sum_{i \in [m]} \|p_i\| \|f_i\| \) and its degree is \( \deg(P) := \max\{\deg(q), \deg(p_i f_i) : i \in [m]\} \). An SA-refutation of \( F \) is an SA-proof of \( -1 \geq 0 \) from \( F \).
Complexity measures. We define complexity measures uniformly across $S = \text{NS, PC, SA}$.

- The size complexity $S(F)$ of a formula $F$ is the minimum size of a $S$-refutation of $F$.
- The degree complexity $\deg_S(F \vdash \bot)$ is the minimum degree of a $S$-refutation of $F$.
- Suppose that the variables $X$ are partitioned into blocks. The block-degree $bdeg(r)$ of a monomial $r$ is the number of distinct blocks that $r$ touches. Moreover, we let $bdeg(P)$ denote the maximum block-degree of a monomial in $P$, and we define $bdeg_S(F \vdash \bot)$ as the minimum block-degree of any $S$-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 $\overline{x}$, and include the complementary axiom $x + \overline{x} - 1 = 0$. 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 can drastically reduce size [ABRW02]. 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 easy simulations.

Fact 7.1 (Simulations). Suppose a polynomial $f$ admits an NS-proof from a set of polynomials $F$ in size $s$ and (block-)degree $d$. Then there is a PC-proof (and an SA-proof if the field is $\mathbb{R}$) of $f$ from $F$ in size $\text{poly}(s,n)$ and (block-)degree $d$. \hfill $\square$

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 is changed at most by a polynomial factor. Our assumption allows to equate a polynomial’s syntactic representation as an element of $F[X]$ with its semantic representation as a boolean function $\{0,1\}^n \to \{0,1\}$: each boolean function has a unique representation as a multilinear polynomial.

7.2 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 the literature [BGIP01, LN17b, LN17a].

Definition 7.2 (Algebraic reduction). Let $F$ and $G$ be two sets of polynomials encoding CNF formulas, defined on variables $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$. An algebraic reduction, denoted $F \leq_{\text{alg}} G$, of degree $d$ and size $s$ consists of the following.

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

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

**Lemma 7.3.** If $F \leq_{\text{alg}} G$ with degree $d$, then $\deg_S(F \vdash \bot) \leq d \cdot \deg_S(G \vdash \bot)$ for all $S = \text{NS}, \text{PC}, \text{SA}$.

**Proof.** We first prove the lemma for \textbf{NS} (the proof for \textbf{SA} is similar, so we omit it). 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 \textbf{NS}-refutation of $G$. Consider the expression

$$\sum_{i \in [b]} (p_i g_i) \circ r = \sum_{i \in [b]} (p_i \circ r)(g_i \circ r) = 1. \quad (5)$$

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, (5) can be written as an \textbf{NS}-refutation of $F$ of degree at most

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

We now prove the lemma for \textbf{PC}. Let $\mathcal{P}$ be a \textbf{PC}-refutation of $G$. We construct a \textbf{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 \textbf{NS}—and therefore, by Theorem 7.1, also in \textbf{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 $y \cdot p$ is derived from $p$ by the multiplication rule, then we can derive $(y \cdot p) \circ r = r_i(p \circ r)$ from $p \circ r$.

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 \textbf{block-aware} reduction.

**Definition 7.4** (Algebraic block-aware reduction). Let $F$ and $G$ be two sets of (polynomials encoding) CNF formulas over a field $F$, and suppose that $F \leq_{\text{alg}} G$ by a degree-$d$ reduction $r : \{0,1\}^n \rightarrow \{0,1\}^m$ as in the previous definition. Suppose further that the variables of $G$ are partitioned into blocks. The reduction $r$ is \textbf{block-aware} if for each block $B$ and each $T \subseteq B$ the following polynomial has degree $\leq d$:

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

**Lemma 7.5.** If $F \leq_{\text{alg}} G$ via a degree-$d$ block-aware reduction, then $\deg_S(F \vdash \bot) \leq d \cdot \deg_S(G \vdash \bot)$ for all $S = \text{NS}, \text{PC}, \text{SA}$.

**Proof.** The case for Nullstellensatz and Sherali–Adams identically follows the proof of Lemma 7.3 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 follow the proof of Lemma 7.3. Every line of a \textbf{PC}-proof is multilinear, so, by the definition of a block-aware reduction and following the same accounting in the proof of Lemma 7.3 we see that the degree of the new proof is at most $d \cdot \deg(\mathcal{P})$. $\square$
8 Algebraic Block Lifting

In this section, we prove Lemma 2.5 that states that \( 2^{\Omega(b\text{deg}(\mathcal{F} \upharpoonright \bot)))} \leq S(\text{Lift}(\mathcal{F})) \leq 2^{O(b\text{deg}(\mathcal{P}))} \|\mathcal{P}\| \)
where \( \mathcal{P} \) is any \( S \)-refutation of \( \mathcal{F} \) and \( S = \text{Res, NS, PC, SA} \). We use the same definition of the formula \( \text{Lift}(\mathcal{F}) \) as in Section 6. For Resolution this is exactly Lemma 2.3.

8.1 Upper bound for Lift(\( \mathcal{F} \))

To prove the upper bound \( S(\text{Lift}(\mathcal{F})) \leq 2^{O(b\text{deg}(\mathcal{P}))} \|\mathcal{P}\| \) for the algebraic proof systems, we start by observing that \( \text{Lift}(\mathcal{F}) \leq^{\text{alg}} \mathcal{F} \) 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 \( \mathcal{F} \),
\[
\|p \circ r\| \leq 3^{b\text{deg}(p)} \cdot \|p\|.
\]

We first prove the upper bound for Nullstellensatz by analyzing this reduction (the proof for Sherali–Adams is analogous). Let \( \mathcal{F} = \{f_1, \ldots, f_m\} \) and let \( \mathcal{P} = \{p_1, \ldots, p_m\} \) be a \( \text{NS} \)-refutation of \( \mathcal{F} \). 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 7.3, is a refutation of \( \text{Lift}(\mathcal{F}) \). Note that the polynomial \( p_i \circ r \) has at most \( 3^{b\text{deg}(p_i)} \cdot \|p_i\| \leq 3^{b\text{deg}(\mathcal{P})} \cdot \|p_i\| \) monomials and that \( f_i \circ r \) is equal to the sum of the \( 2^{b\text{deg}(f_i)} = O(1) \) axioms of \( \text{Lift}(f_i) \), each of which has \( 3^{b\text{deg}(f_i)} \|f_i\| = O(\|f_i\|) \) monomials. Therefore, we can conclude there is a \( \text{NS} \)-refutation of size \( \sum_{i \in [m]} 3^{b\text{deg}(\mathcal{P})} \cdot \|p_i\| \cdot O(\|f_i\|) \leq O(3^{b\text{deg}(\mathcal{P})} \|\mathcal{P}\|) \).

We now prove the upper bound for \( \text{PC} \). Let \( \mathcal{P} \) be a \( \text{PC} \)-refutation of \( \mathcal{F} \). We construct a \( \text{PC} \)-refutation \( \mathcal{P}' \) of \( \text{Lift}(\mathcal{F}) \) in the same way as done in the proof of Lemma 7.3: whenever \( \mathcal{P} \) derives \( p_i \) in \( \mathcal{P}' \) we will derive the polynomial \( p \circ r \) (which has at most \( 3^{b\text{deg}(p)} \|p\| \) monomials).

- **Axioms.** For any axiom \( f \in \mathcal{F} \), we noted already that the polynomial \( f \circ r \) is equal to the sum of the \( 2^{b\text{deg}(f)} = O(1) \) axioms of \( \text{Lift}(f) \), each of which has \( 3^{b\text{deg}(f)} \|f\| = O(\|f\|) \) monomials. Thus, \( f \circ r \) can be derived in \( \text{PC} \) 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 \( y \cdot p \) is derived from \( p \) by the multiplication rule, then we can to derive \( (y \cdot p) \circ r = r_i (p \circ r) \) from \( p \circ r \) in size \( O(\|p \circ r\|) \).

Therefore, the \( \text{PC} \)-refutation has size \( O(3^{b\text{deg}(\mathcal{P})} \|\mathcal{P}\|) \).

8.2 Lower bound for Lift(\( \mathcal{F} \))

Finally, we prove the lower bound \( 2^{\Omega(b\text{deg}(\mathcal{F} \upharpoonright \bot)))} \leq S(\text{Lift}(\mathcal{F})) \) for \( S = \text{NS, SA, PC} \). This follows the random restriction argument used for Resolution exactly (Section 6), so, we merely sketch the argument. Namely, we show that \( b\text{deg}(\mathcal{F} \upharpoonright \bot) = O(\log \|\mathcal{P}\|) \), 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 \( \rho \) is a partial restriction to the variables of \( G \), then \( \mathcal{P} \upharpoonright \rho \) is an \( S \)-refutation of \( G \upharpoonright \rho \).

Letting \( \rho \) 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 \( \rho \). 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 \(P \vdash \rho\) has a monomial of block-degree \(\geq d\) with probability at most \(\|P\|(3/4)^d\), which is \(< 1\) if \(d > \log_{4/3}\|P\|\). Since \(P \vdash \rho\) is an \(S\)-refutation of \(F\) by the claim made above, we have that \(\text{bdeg}_S(F \vdash \bot) \leq \text{bdeg}(P \vdash \rho) \leq d = O(\log \|P\|)\). This completes the proof of Lemma 2.5.

9 Algebraic Upper Bound

In this section, we prove Lemma 2.4(i) that states that \(\text{TreeRef}(F)\), where \(F\) is satisfiable, admits a size-\(n^{O(1)}\) block-degree-\(O(1)\) \(S\)-refutation for \(S = \text{Res}, \text{NS}, \text{PC}, \text{SA}\). We prove this for \(\text{NS}\), which implies the result for \(\text{PC}\) and \(\text{SA}\) by simulations (Fact 7.1). The result holds for \(\text{Res}\) by the original upper bound for \(\text{Ref}(F)\) due to Atserias–Müller and the fact that \(\text{TreeRef}(F)\) was defined as a weakening of \(\text{Ref}(F)\). Therefore, the goal of this section is to prove the following lemma.

**Lemma 9.1** (Algebraic upper bound). Let \(F\) be a satisfiable \(n\)-variate formula. There is a size-\(n^{O(1)}\) block-degree-\(O(1)\) \(\text{NS}\)-refutation of \(\text{TreeRef}(F)\) (over any field, without twin variables).

Our proof has three steps: (§9.1) We first define the so-called end-of-line formula \(\text{EoL}_n\), which is a size-\(n^{O(1)}\) block-degree-\(O(1)\) CNF formula. (§9.2) Then we reduce \(\text{TreeRef}(F)\) to \(\text{EoL}_n\). (§9.3) Finally, we recall from prior work [GKRS19] that \(\text{EoL}_n\) admits a small \(\text{NS}\)-refutation. The last two steps are formalized in the following two claims. As we want our result to be as general as possible, in this section, we work over the integers \(\mathbb{Z}\) (hence the computations are valid over any field), and assume no twin variables.

**Claim 9.2** (Reduction to \(\text{EoL}\)). Fix an \(n\)-variate satisfiable \(F\). There is a block-aware reduction \(\text{TreeRef}(F) \leq_{\text{alg}} \text{EoL}_{n^3}\) of size \(n^{O(1)}\). Furthermore, for each subset \(T\) contained in a block of \(\text{EoL}_{n^3}\), the polynomial \(r_T\) defined by the reduction has block-degree \(O(1)\) (relative to \(\text{TreeRef}(F)\)).

**Claim 9.3** (Upper bound for \(\text{EoL}\)). \(\text{EoL}_n\) admits a block-degree-\(O(1)\) \(\text{NS}\)-refutation (over \(\mathbb{Z}\)).

The algebraic upper bound (Lemma 9.1) follows by combining these two lemmas.

*Proof of Lemma 9.1.* Let \(r\) be the reduction in Claim 9.2 and let \(\sum_i p_i f_i = 1\) be the \(\text{NS}\)-refutation in Claim 9.3. We verify that the composed refutation \(\sum_i (p_i f_i) \circ r = 1\) (discussed in Section 7.2) satisfies the lemma. Consider any monomial \(t\) of \(p_i f_i\). We have \(\text{bdeg}(t) \leq O(1)\), so when \(t\) is replaced by a product of \(\text{bdeg}(t)\) many \(r_T\)’s (for various \(T\)’s, each contained in a block of \(\text{EoL}_{n^3}\), where each \(r_T\) has size \(n^{O(1)}\) and block-degree \(O(1)\)), this results in a polynomial of size \((n^{O(1)})^{\text{bdeg}(t)} \leq n^{O(1)}\) and block-degree \(\text{bdeg}(t) \cdot O(1) = O(1)\). We conclude that \((p_i f_i) \circ r\) (and hence the whole refutation \(\sum_i (p_i f_i) \circ r = 1\) has size \(\|p_i f_i\| : n^{O(1)} \leq n^{O(1)}\) and block-degree \(O(1)\). \(\square\)

9.1 EoL formula

The end-of-line formula \(\text{EoL}_n\) states that “there is an \(n\)-vertex digraph where every vertex has in/out-degree 1, except for one distinguished vertex that has in-degree 0 and out-degree 1.” The combinatorial principle underlying \(\text{EoL}_n\) is central in the theory of total \(\text{NP}\) search problems [Pap94, BCE+98].

The variables of \(\text{EoL}_n\) are intended to describe a digraph on vertices \([n]\) where \(n \in [n]\) is thought of as a distinguished vertex. Namely, for each \(i \in [n]\), there is a block of \(2 \log n\) boolean variables \(z_i := (\bar{z}_i, \bar{z}_i)\) that encode, in binary, a predecessor pointer \(\bar{z}_i \in [n]\) and a successor pointer \(\bar{z}_i \in [n]\). An assignment to the variables \(z = (z_1, \ldots , z_n)\) defines a digraph \(G_z := ([n], E_z)\) where

\[(i, j) \in E_z \quad \text{iff} \quad \bar{z}_i = j \quad \text{and} \quad \bar{z}_j = i.\]
A small detail is that we allow any vertex to be a self-loop, achieved by setting $z_i = z_i = i$.

The axioms of $\text{EoL}_n$ are:

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

In particular, $\text{EoL}_n$ 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: $\text{EoL}_n$ claims the edges of $G_z$ define a bijection $[n] \rightarrow [n-1]$.

## 9.2 Reduction to EoL

Next we prove Claim 9.2: For an $n$-variate satisfiable $F$, we give a size-$n^{O(1)}$ block-aware reduction

$$\text{TreeRef}(F) \leq_{\text{alg}} \text{EoL}_{n^3}.$$  

**Intuition.** Before launching into the proof, we sketch a strategy for refuting refutation formulas in Resolution (building on Pudlák [Pud03]). It will guide us in defining our reduction.

Consider the Prover–Adversary game (Section 4.1) for TreeRef($F$). Our goal, as Prover, is to find a falsified axiom of TreeRef($F$). Henceforth, fix some satisfying assignment $x^*$ of $F$. In short, our Prover strategy is to **walk down the purported proof** maintaining the invariant that every clause we visit is falsified by $x^*$. Namely, we start at the root block $B_{n^3}$ and check that it is falsified by $x^*$. If not, we find that $B_{n^3}$ contains some literal, which falsifies an axiom of TreeRef($F$) (that says $B_{n^3}$ contains no literals) and hence the game ends. So suppose the root is indeed falsified by $x^*$. If the root block was obtained via a Resolution rule from blocks $B_j$, $B_{j'}$ we know by the soundness of the rule and assuming the axioms of TreeRef($F$) hold near the root block that a (unique) child of the root, say $B_j$, is falsified by $x^*$. Our walk then steps into $B_j$, which maintains our invariant. From $B_j$, we continue the walk iteratively: we always find the (unique) child falsified by $x^*$. As long as no false axioms of TreeRef($F$) are encountered in this walk, will eventually reach an axiom block $B$. But since $x^*$ satisfies all axioms of $F$, and $x^*$ falsifies $B$ (by the invariant), it must be the case that $B$ contains a mistake in the purported proof. This ends the game.

Our reduction is inspired by this walk. The $i$-th vertex in $\text{EoL}_{n^3}$ will correspond to the block $B_i$ in TreeRef($F$). In particular, the distinguished vertex $n^3 \in [n^3]$ will correspond to the root block $B_{n^3}$. On input an assignment $y$ to TreeRef($F$), our reduction outputs an assignment to $\text{EoL}_{n^3}$ that encodes the above walk in the purported proof encoded by $y$.

**$\land$-decision trees.** For ease of understanding, we describe the reduction as an $\land$-decision tree, that is, a decision tree that is allowed to query, in a single step, the logical-and $\land_{x \in A} x$ 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 9.4.** If $r$ is computed by a depth-$d$ $\land$-decision tree, then $r$ is computed by size-$2^O(d)$ polynomial.

**Proof.** For each leaf $\ell$ in the tree, let $r_\ell(x)$ denote the indicator function that is 1 iff the leaf $\ell$ is reached on input $x$. Every query $\land_{x \in A} x$ can be simulated by the monomial $x^A := \prod_{x \in A} x$. Hence we can compute $r_\ell$ by taking the product along the unique path from root to $\ell$ of either $x^A$ or $1 - x^A$ (depending on the query outcome on the path). Hence, as a multilinear polynomial, $r_\ell$ satisfies $\|r_\ell\| \leq 2^d$. Moreover, $r$ can be written as $r = \sum_{\ell \in \text{leaves}} r_\ell$ where the sum is over leaves $\ell$ that output 1. There are at most $2^d$ leaves, and thus $\|r\| \leq 2^{2d}$.\qed
**Reduction.** We describe a family of \(\land\)-decision trees \(T = (T_1, \ldots, T_n)\) where \(T_i\) outputs values for the variables \(z_i = (\vec{z}_i, \vec{z}_i)\). Our goal is to satisfy the following condition, which implies the **Axiom** property of a reduction (even **weakening** in Definition 4.1, a special case of an **NS-proof**).

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

Henceforth, fix a satisfying assignment \(x^*\) of \(F\). Given an assignment \(y\) to 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 \(\land\)-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 \(T_i\) computes \(z_i = (\vec{z}_i, \vec{z}_i)\) 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 vertex \(i\) into a self-loop by outputting \(\vec{z}_i = \vec{z}_i := i\).
- **Root.** We know that \(B_n^3\) contains some literal consistent with \(x^*\). By binary search (using \(O(\log n)\) many \(\land\)-queries) we can discover a specific literal indicator of \(B_n^3\) that is set to 0. This violates an axiom of TreeRef\((F)\). Hence by (†), it is safe to output anything for \((\vec{z}_i, \vec{z}_i)\).

\(B_i\) is feasible: Query \(B_i\)'s type.

- **Disabled:** If \(B_i\) is non-root, we make vertex \(i\) into a self-loop. If \(B_i\) is root, then we have found an axiom violation for 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 \(\ell\) consistent with \(x^*\). But since \(B_i\) is feasible, \(B_i\) does not contain \(\ell\). Hence \(\ell\) 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 \((\vec{z}_i, \vec{z}_i) := (p, j)\).

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

Our reduction is block-aware as each \(T_i\) outputs the whole contents of a block. It is also clear that \(T_i\) has depth \(O(\log n)\). Hence by Fact 9.4, each output bit (or even the product polynomial \(r_T\) for a subset \(T\) of output bits) of \(T_i\) can be converted to a polynomial of size \(n^{O(1)}\). To see the “furthermore” part of Claim 9.2, we note that any \(r_T\) is a sum of leaf indicators \(r_i\) of \(T_i\) (using terminology from the proof of Fact 9.4), each of which has queried variables from at most 4 blocks (the block \(B_i\) itself, its two children, and parent). This concludes the proof of Claim 9.2.

### 9.3 Upper bound for EoL

In this subsection we prove Claim 9.3. 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$_n$:

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

Each $q_i$ can be computed by a decision tree $T_i$ of depth $O(\log n)$. For example, to evaluate $\text{indeg}_{G_z}(i) \in \{0, 1\}$ the tree queries the pointer $\tilde{z}_i$, follows it, and checks whether $\tilde{z}_{\bar{i}} = i$. Thus, as in Fact 9.4, $q_i$ can be computed by a degree-$O(\log n)$ polynomial $\sum_{\ell} r_{\ell}(z)$ where the sum is over leaves of $T_i$ that output a non-zero value and

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

Note that each $\ell$ 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 $r_{\ell}$ can be factored as $r_{\ell} = t_\ell a_\ell$ where $t_\ell$ 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 $T_i$ queries at most 3 different vertex-blocks (itself, its potential predecessor and successor). This proves Claim 9.3.

## 10 Algebraic Lower Bound

In this section, we prove Lemma 2.4(ii) that states that $\text{bdeg}_S(\text{TreeRef}(F) \vdash \bot) \geq n^{\Omega(1)}$, where $F$ is unsatisfiable and $S = \text{Res}, \text{NS}, \text{PC}, \text{SA}$. We already know that $\text{bdeg}_S(\text{TreeRef}(F) \vdash \bot) \geq \tilde{\Omega}(\text{deg}_S(\text{rPHP}_n \vdash \bot)/n)$ by the reduction of Section 5.4 and Lemma 7.5. Hence it suffices to prove

$$\text{deg}_S(\text{rPHP}_n \vdash \bot) \geq \tilde{\Omega}(n).$$

We show this follows from known degree lower bounds due to Razborov (for PC, any field) [Raz98] and Georgiou and Magen (for SA) [GM08]. They used a different algebraic encoding of the pigeonhole principle, which we recall below. In the rest of this section (Section 10.1), we show that our encoding reduces to their algebraic encoding by a low-degree reduction. This will prove (6).

### Algebraic PHP.

Define aPHP$_n$ as the following system of polynomial equations over variables $x_{ij}$ where $i \in [2n]$ and $j \in [n]$. (Strictly speaking, [GM08] did not consider the axioms $Q_{i,j,j'} = 0$, but their result holds even if they are included.)

$$\forall i: \quad Q_i := \sum_j x_{ij} - 1 = 0 \quad \text{“each pigeon occupies a hole,”}$$

$$\forall i, j \neq j': \quad Q_{i,j,j'} := x_{ij}x_{ij'} = 0 \quad \text{“no pigeon occupies two holes,”} \quad \text{(aPHP$_n$)}$$

$$\forall j; i \neq i': \quad Q_{i,i',j} := x_{ij}x_{i'j} = 0 \quad \text{“no hole houses two pigeons,”}$$

$$\forall i, j: \quad Q_{i,j} := x_{ij}^2 - x_{ij} = 0 \quad \text{“boolean axioms.”}$$

### Theorem 10.1 ([Raz98, GM08]).

Refuting aPHP$_n$ requires degree $\Omega(n)$ in both PC and SA.
10.1 Reduction from aPHP

To prove (6), we translate the degree lower bound in Theorem 10.1 to our rPHP encoding via an algebraic reduction. Namely, our goal is to show an algebraic degree-$\tilde{O}(1)$ reduction

$$\text{aPHP}_n \leq_{\text{alg}} \text{rPHP}_n.$$  

**Variables.** We start by defining how the variables $(f, g) = (f_{ik}, g_{j\ell})$ of rPHP$_n$ (where $i \in [2n]$, $k \in [\log n]$, $j \in [n]$, $\ell \in [\log 2n]$) depend on the variables $x_{ij}$ of aPHP$_n$ (where $i \in [2n]$, $j \in [n]$). For convenience, we think of $[n] := \{0, 1, \ldots, n - 1\}$ 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 $f_{ik} := \sum_{j \in J_k} x_{ij}$ where $J_k := \{j \in [n] : j_k = 1\}$ are the holes with $k$-th bit equal to 1.
- Define $g_{j\ell} := \sum_{i \in I_\ell} x_{ij}$ where $I_\ell := \{i \in [2n] : i_\ell = 1\}$ are the pigeons with $\ell$-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 $p(x) \equiv q(x)$ 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):

$$f_{ik}^2 = \left(\sum_{j \in J_k} x_{ij}\right)^2 = \sum_{j \in J_k} x_{ij}^2 + \sum_{j,j' \in J_k} x_{ij}x_{ij'} = \sum_{j \in J_k} (x_{ij} + Q_{i,j}) + \sum_{j,j' \in J_k} Q_{i,j,j'} \equiv \sum_{j \in J_k} x_{ij} = f_{ik}.$$

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 $f^1 := f$ and $f^0 := 1 - f$ for short:

$$[f(i) = j \Rightarrow g(j) = i] \equiv [g(j) = i \lor f(i) \neq j]$$
$$\equiv \land_{\ell} [g_{j\ell} = i_\ell \lor \lor_{k} f_{ik} \neq j_k]$$
$$\equiv \{ g_{j\ell}^{1-i_\ell} \prod_{k} f_{ik}^j = 0 : \ell \in [\log 2n]\}. \quad (*)$$

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

**Claim 10.2.** $f_{ik}^0 \equiv \sum_{j \in [n] \setminus J_k} x_{ij}$.

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

**Claim 10.3.** $\prod_{k} f_{ik}^j \equiv x_{ij}$.

**Proof.** Expand each $f_{ik}^j$ according to its definition ($j_k = 1$) or by Claim 10.2 ($j_k = 0$):

$$\prod_{k} f_{ik}^j \equiv x_{ij}^{\log n} + \sum_{j' \neq j''} r_{j'j''}(x) \cdot x_{ij'}x_{ij''}$$

where $\deg(r_{j'j''}) \leq \log n$ (boolean axioms) \qed

$$x_{ij}^{\log n} \equiv x_{ij} \equiv x_{ij}.$$
We now derive (*). By Claim 10.3, we have (*) = \( g_{j^i}^{1-i^j} \prod_k j^i_k \approx g_{j^i}^{1-i^j} \cdot x_{ij} \). Two cases:

Case \( i^j = 0 \) (where \( i^j \notin I^j \)): 
\[
(*) = \left( \sum_{i^j \in I^j} x_{i^j} \right) \cdot x_{ij} \\
= \sum_{i^j \in I} Q_{i,i^j} \\
\approx 0;
\]

Case \( i^j = 1 \) (where \( i^j \in I^j \)): 
\[
(*) = (1 - \sum_{i^j \in I^j} x_{i^j}) \cdot x_{ij} \\
= x_{ij} - x_{ij}^2 - \sum_{i^j \in I^j \setminus \{i\}} x_{i^j} x_{ij} \\
= -Q_{i,j} - \sum_{i^j \in I^j \setminus \{i\}} Q_{i,i^j} \\
\approx 0.
\]

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

Acknowledgements

We thank Shuo Pang, Aaron Potechin, and Madhur Tulsiani for discussions. R.R. was supported by NSERC, the Charles Simonyi Endowment, and indirectly supported by the National Science Foundation Grant No. 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


