Short Proofs are Hard to Find

Ian Mertz
University of Toronto, Canada
mertz@cs.toronto.edu

Toniann Pitassi
University of Toronto, Canada
Institute of Advanced Studies, USA
toni@cs.toronto.edu

Yuanhao Wei
Carnegie Mellon University, USA
yuanhao1@cs.cmu.edu

Abstract

We obtain a streamlined proof of an important result by Alekhnovich and Razborov [2], showing that it is hard to automatize both tree-like and general Resolution. Under a different assumption than [2], our simplified proof gives improved bounds: we show under ETH that these proof systems are not automatizable in time $n^{f(n)}$, whenever $f(n) = o(\log^{1/7-\epsilon} \log n)$ for any $\epsilon > 0$. Previously non-automatizability was only known for $f(n) = O(1)$. Our proof also extends fairly straightforwardly to prove similar hardness results for PCR and Res(r).

2012 ACM Subject Classification Theory of computation → Proof complexity; Hardware → Theorem proving and SAT solving

Keywords and phrases automatizability, Resolution, SAT solvers, proof complexity

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Ian Mertz*: Research supported by NSERC.
Toniann Pitassi: Research supported by NSERC.
Yuanhao Wei: Work done while a student at University of Toronto. Research supported by NSERC.

1 Introduction

Proof complexity first and foremost aims to understand, for a given propositional formula τ, how long of a proof is needed to verify that τ is unsatisfiable. To understand the expressiveness of a proof system, we need to understand what formulas can and cannot be efficiently proven in that system. However, for algorithmic applications where formulas often have fairly short proofs, what is perhaps more important than knowing the worst-case proof length of a given τ is actually *finding* proofs of τ. In particular, even if we're promised that τ has proofs of small size, say polynomial in the size of τ, can we hope to find one that's not too much larger?

This question, of finding optimal proofs in a given system, is known as *automatizability*, introduced by Bonet, Pitassi, and Raz [11]. A proof system $\mathcal{Q}$ is automatizable if there exists an algorithm which, given an unsatisfiable formula τ on n variables, returns a $\mathcal{Q}$-refutation of τ in time poly$(n, |\tau|, S)$ where $S := S_{\mathcal{Q}}(\tau)$ is the size of the shortest $\mathcal{Q}$-refutation of τ. Twenty years later no reasonable proof systems are known to be polynomially automatizable, and little is known even for the more general notion of f-automatizability, where the algorithm can run in time $f(n, |\tau|, S)$.

Understanding the automatizability of various proof systems is a major tool in algorithm design; two well known examples are SAT solvers, where the best algorithms are highly optimized version of the Resolution (Res) proof system (see e.g. [32]) and celebrated algorithmic versions of the Sum-of-Squares (SoS) proof system for approximation [37] and learning (see e.g. [38] for a survey on recent developments in this very active field of research). We especially draw attention to the question of

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access;
Article No. 23; pp. 23:1–23:21

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

automatizing Res. Resolution is a simple and fairly weak proof system, and yet Res proofs are the objects at the heart of the best known SAT solvers, with a long line of research connecting Res size to notions such as conflict driven clause learning and restarts [29, 30, 35, 42]. Automatizing Res is also key to the best known automated theorem provers for propositional and first order logics [16, 17]. Therefore, the tractability of finding short Res proofs lies at the heart of understanding the frontiers and limitations of SAT solving algorithms and automated theorem proving.

Despite the importance of automatizability for Res and other proof systems, our understanding of this question is limited at best. In terms of upper bounds, the best automatizing algorithm for Res runs in slightly subexponential time. In terms of lower bounds, until recently the main hardness result was the landmark paper of Alekhnovich and Razborov [2], who prove that under the assumption FPT $\neq$ W[P],[1] Res (as well as tree-like Res, denoted TreeRes) is not polynomially-automatizable. Using similar ideas, Galesi and Lauria [20] adapted Alekhnovich and Razborov's proof in order to obtain the same result for the Polynomial Calculus (PC) system, an extension of Res which is the proof complexity model for the Groebner basis algorithm [15].[2]

For all other well-studied practical systems almost nothing is known. To give a short list of other well-known proof systems used in algorithm design, we have Cutting Planes (CP), widely used for optimization algorithms (see e.g. [28]); Sherali-Adams (SA), which underlies a general family of linear programming algorithms [41]; and the aforementioned Sum-of-Squares (SoS)-based semi-definite programming algorithms. For these systems we have no extension of the argument of [2], and therefore no notable lower bounds on automatizability.

1.1 Our Contributions

Our motivation for this work is to adapt the techniques of [2], first to move past polynomial automatizability lower bounds for Res (and PC), and second to hopefully shed light on the automatizability of proof systems such as CP, SA, and SoS. The starting point of our contribution is in switching to the exponential time hypothesis (ETH) as opposed to the FPT $\neq$ W[P] assumption in [2, 20]. A central limitation in starting from the assumption that some problem has no FPT algorithm is that FPT algorithms run in time $f(k)n^{O(1)}$, and so the best lower bound one can get from such an assumption, without a careful analysis of f and the range of k, is $n^{\omega(1)}$. In the past decade a line of work by Chen and Lin [14] showed how to obtain fixed parameter lower bounds beyond $f(k)n^{O(1)}$ for gap versions of NP-hard problems, such as dominating set and hitting set, by starting not from an assumption about FPT but from ETH. Analyzing these reductions we can derive a hardness result for a *fixed* f and k, which allows us to go beyond the $n^{\omega(1)}$ barrier in [2, 20], albeit starting from the slightly stronger ETH assumption. We state our main theorem precisely now.

▶ **Theorem 1** (Main Theorem). *Let $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$. Assuming* ETH *holds $\mathcal{Q}$ is not n^f-automatizable for any $f = o(\log^{1/7-\varepsilon} \log n)$ (where $\varepsilon > 0$ is any constant).*

Equally important as extending the results of [2, 20] is our second goal, namely simplifying the presentation of the construction and proofs. Moving to the stronger ETH assumption allows us to change the central formula in a way that, while still using the core machinery of [2], leads to a conceptually simpler formula and proof. The basis of the formula in [2] is the monotone minimum circuit satisfying assignment (MMCSA) problem, which takes as input a poly-size monotone circuit. The natural encoding of their formula as a CNF formula requires extra variables to represent the

[1] The original result of [2] uses FPR, a randomized version of FPT, in place of FPT in the assumption. This was improved to the stated assumption by [19].

[2] While the most well-studied and widely used SAT solvers are based on Res, there have been some implementations that use the Groebner basis algorithm to utilize the more expressive power of PC, see e.g. [12].

internal gates of the monotone circuit, leading to many technicalities involved in proving a Res width lower bound, namely an indirect and highly redundant encoding of the circuit. Our proof starts from the hitting set problem, which is a special case of MMCSA where the circuit is a CNF. Since the formula is already a CNF, the input can be encoded directly as the formula rather than indirectly having variables for each of the gates, and as a result the upper and lower bound proofs in our paper are highly streamlined.

While we do start from a stronger assumption than [2, 20], there are few additional advantages to our new formula beyond presentation. First, going beyond superpolynomial hardness for Res allows us to obtain hardness results on the automatizability of Res(r), a proof system generalizing Res by allowing lines to be disjunctions of size r conjunctions. Prior to our paper nothing was known for Res(r) for any $r \geq 2$, and the formula from [2] would not be able to go past Res(r) for constant r.

▶ **Theorem 2** (Main Theorem for Res(r)). *Let $\mathcal{Q} =$ Res(r). Assuming ETH holds then for any $\varepsilon > 0$, $\mathcal{Q}$ is not $n^{f/\exp(r^2)}$-automatizable for any $f = o(\log^{\frac{1}{7}-\varepsilon} \log n)$ if $r \in O(\sqrt{\log f})$.*

Second, our technique has a direct, and in our view achievable, path to further improvement: if the reduction of [14] were to be improved to allow a lower bound against gap hitting set for larger parameters, it would immediately translate to a stronger non-automatizability result. We discuss this idea in detail in Section 6. Third, our results are also immediately strengthened if, instead of using ETH, one uses a slightly stronger assumption known as the *gap exponential time hypothesis* (GapETH), as introduced in [18, 27]. We formally define GapETH along with ETH in Section 2, but these results require no change in our formula nor our proofs. As with starting from [14] for our ETH results, the work required to use GapETH is analyzing a reduction of Chalermsook et al. [13], so we defer the results and analysis to Appendix B.

1.2 Related Work

Table 1 lists the known results for Res and PC. An early result [1] shows that it is NP-hard to find proofs whose size is a constant factor of optimal, and this holds for *all* standard proof systems.

For stronger proof systems we have more lower bounds, although these bounds still only rule out polynomial automatizabiliy and require cryptographic assumptions. Krajíček and Pudlák showed non-automatizability of the Extended Frege system under the hardness of discrete log [25], with subsequent works proving the same lower bounds for Frege and AC^0-Frege under similar assumptions [10, 11]. Conceptually these more expressive classes should be harder to automatize because there exist many more short proofs than for say Res, but a nice upshot of these results is that they hold for a much weaker notion of automatizability, aptly named *weak automatizability*. Weak automatizability of a proof system $\mathcal{Q}$ only requires that the automatizing algorithm return a proof of τ in *some* proof system, so long as it's close in length to the shortest $\mathcal{Q}$-proof of τ.[3] Clearly hardness of weak automatizability implies hardness of automatizability, and hardness of weak automatizability is closely related to feasible interpolation [36], which was the tool used in the Frege nonautomatizability results listed above.

Turning to upper bounds, there are a class of *width/degree* based automatizability algorithms for Res, PC, SA, and SoS. The width of a Res refutation is the maximal number of literals appearing in any line of the refutation, and the width of a CNF formula τ, denoted $w(\tau)$, is the minimum width of any Res refutation refuting τ. It is not hard to see that exhaustive search allows us to find a Res refutation for τ in time $n^{O(w(\tau))}$ [8]. A non-trivial fact is that the same upper bound

[3] This can be seen as analogous to the two notions of of learning, proper versus nonproper, where the former is required to produce a hypothesis from the original concept class, whereas the latter may produce any hypothesis.

Proof system	Assumption	Result	Reference
all systems	P $\neq$ NP	$\omega(1) \cdot n$	[1]
Res, TreeRes	W[P] $\neq$ FPT	$n^{\omega(1)}$	[2]
Nullsatz, PC, PCR	W[P] $\neq$ FPT	$n^{\omega(1)}$	[20]
Res, TreeRes, Nullsatz, PC, PCR	ETH	$n^{\Omega(\log^{1/7} \log n)}$	this work
Res(r)	ETH	$n^{\Omega(\log^{1/7} \log n / \exp(r^2))}$	this work

Figure 1 Lower bounds on automatizabillity of weak proof systems

holds for PC (due to the Groebner basis paper of Clegg, Impagliazzo, and Edmonds [15]), SA [40], and SoS [26, 34], where the degree of the polynomials appearing in the proofs is used in place of width. These algorithms are known to be tight for width/degree based automatizability, as there exist tautologies τ with proof size $S(\tau) = n^{\Omega(d)}$ for Res, PC, SA, and SoS[4] [6].

A groundbreaking work of Ben-Sasson and Wigderson [9] showed that $w(\tau) \leq \log S(\tau)$ for the special case of TreeRes and $w(\tau) \leq \sqrt{n \log S(\tau)}$ for general Res. Combined with the $n^{O(w(\tau))}$ upper bound for both systems gives automatizability for TreeRes and Res in time $n^{O(\log S(\tau))}$ and $n^{O(\sqrt{n \log S(\tau)})}$, respectively. Perhaps even more surprisingly, a result of [15] gives the same degree/size tradeoff for PC as [9] gave for Res; $d(\tau) \leq \sqrt{n \log S(\tau)}$ for the case of PC, and $d(\tau) \leq \log S(\tau)$ for a static version of PC called *Nullstellensatz* (Nullsatz).[5] Combining these degree bounds with the degree based algorithms gives automatizability for Nullsatz and PC in time $n^{O(\log S(\tau))}$ and $n^{O(\sqrt{n \log S(\tau)})}$, respectively. While these upper bounds are very strong for TreeRes and Nullsatz, for Res and PC they are still weakly exponential, and the results of [9, 15] are tight. Thus non-width/degree based techniques are needed to improve these upper bounds, if indeed they can be improved.

1.2.1 Recent Developments

Since the publication of this paper, Atserias and Müller [4] achieved a major breakthrough by all but resolving the question of automatizability for Res. In particular they show that it is NP-hard to distinguish whether τ has Res refutations of size $n^{1+\epsilon}$ or none of size $2^{n^{1/(2+\epsilon)}}$ for any $\epsilon > 0$, which implies that assuming ETH, Res is not $2^{n^{1/2-\epsilon}}$ automatizable for any $\epsilon > 0$. The proof is elegant and uses a meta-tautology which could possibly be adapted for other proof systems such as TreeRes and PC in the future. As of now these results only apply to Res however, and because of the quasipolynomial automatizability of TreeRes the technique will require some notable changes before getting more general results. Thus our results (and technique) are still at the frontier for all other systems discussed.

2 Preliminaries

Let $\tau = \{C_1, C_2, \ldots, C_m\}$ be an unsatisfiable CNF formula over $X = \{x_1 \ldots x_n\}$. We denote by $|\tau|$ the *size* of τ, and likewise for a proof π refuting τ let $|\pi|$ denote the size of π. For a proof system $\mathcal{Q}$ let $S := S_{\mathcal{Q}}(\tau)$ be the size of the shortest $\mathcal{Q}$-proof refuting τ. A proof system $\mathcal{Q}$ is said to be $f(n, |\tau|, S)$-*automatizable* if there exists an algorithm A such that for every unsatisfiable τ A runs

[4] The degree-automatizability of SoS is not established definitely due to the bit-complexity of the underlying polynomials, which can be exponential [33].

[5] This result of [15] actually preceded [9].

in time $f(n, |\tau|, S)$ and outputs a valid $\mathcal{Q}$-proof refuting τ. A proof system $\mathcal{Q}'$ *p-simulates* $\mathcal{Q}$ if for every $\mathcal{Q}$-proof π refuting τ there is a corresponding $\mathcal{Q}'$-proof π' refuting τ such that $|\pi'| = |\pi|^{O(1)}$.

A *Resolution* (Res) refutation of τ is a sequence of clauses $\pi = \{D_1, D_2, \ldots, D_S\}$ such that $D_S = \emptyset$, and each line D_i is either some initial clause $C_j \in \tau$ or is derived from two previous lines using the *resolution rule*: from $(E \vee x)$, $(F \vee \overline{x})$ we derive $(E \vee F)$, where $x \in X$, E and F are clauses, and $E \vee F$ is their disjunction with repeated literals removed. We can view a Res proof π as a directed acyclic graph with a unique clause D_i at every vertex, with initial clauses $C_j \in \tau$ at the leaves, $\emptyset$ at the root, and having an edge from D_i to D_j if D_i was used to derive D_j. With this view, a TreeRes refutation requires that all non-leaf vertices of the underlying graph have outdegree 1 (so the underlying graph of any TreeRes proof is tree-like).

Given a Res or TreeRes refutation $\pi = \{D_1, D_2, \ldots, D_S\}$, the size of π is the number of lines in π, in this case S. The *width* of a clause D_i is the number of literals in it, and the width of π is the maximum width of a clause in the proof. We denote the width of a clause D_i or proof π by $w(D_i)$ and $w(\pi)$, respectively. Clearly Res can p-simulate TreeRes with respect to size and width, as every TreeRes-proof is also a Res-proof.

An *r-Resolution* (Res(r)) refutation[6] is similar to a Res refutation, but each line D_i is an r-DNF instead of a clause, and the resolution rule is adapted as follows: from $(E \vee (\vee_{j \in J} x_j))$, $(F \vee (\wedge_{j \in J} \overline{x_j}))$ we derive $(E \vee F)$, where $J \subseteq [n]$ such that $|J| \leq r$, E and F are r-DNFs, and $E \vee F$ is their disjunction with repeated conjunctions removed (note that $\vee_{j \in J} x_j$ is a DNF with $|J|$ terms while $\wedge_{j \in J} \overline{x_j}$ is a single term). Note that Res(1) = Res. The size of a Res(r) proof is the number of r-disjunctions in it. (See [39] for more details.)

An *algebraic proof system* for refuting CNF $\tau = \{C_1 \ldots C_{m'}\}$ over variable set X is a proof system where each of the clauses C_i is converted into a polynomial equality or inequality P_i over X, such that any assignment of all x_j to $\{0,1\}^n$ satisfies C_i iff it satisfies P_i. For this paper the conversion is done is by sending every positive literal x_j to $(1 - x_j)$ and every negative literal $\overline{x_j}$ to x_j, and P_i is satisfied if the product of all converted literals in C_i is 0. For example, the clause $C_i = x_1 \vee \overline{x_2} \vee x_3$ is converted to $P_i = (1 - x_1)(x_2)(1 - x_3) = 0$. In addition, we add the equations $x_j^2 - x_j = 0$ for all $j \leq n$. Let the resulting $m = m' + n$ equations corresponding to τ be denoted by $\mathcal{P} = \{P_1, \ldots, P_m\}$. Since every P_i is of the form $p_i = 0$ we use P_i to refer to p_i.

The *Nullstellensatz* (Nullsatz) refutation system [7] is an algebraic proof system that uses Hilbert's Nullstellensatz as a certificate of unsatisfiablility. A Nullsatz proof (over a field $\mathbb{F}$) of τ is a set of polynomials $Q_1, \ldots, Q_m$ such that $\sum_i P_i Q_i$ is the formal polynomial "1". Note that this contradicts the statement that there exists an assignment such that $P_i = 0$ for all i. The size of a Nullsatz refutation π is the sum over all $i \in [m]$ of the number of monomials in the expansion of the term $P_i Q_i$, while the *degree* of the refutation is the maximum degree $\deg(P_i Q_i)$ over all $i \in [m]$. It is known that Nullsatz p-simulates TreeRes.

The *Polynomial Calculus* (PC) system is a dynamic version of Nullsatz [15], where the lines of a PC proof π are all polynomials $Q_1, Q_2, \ldots, Q_S$. The lines Q_i can be any of the initial polynomial equations $\mathcal{P}$ or can be derived from previous lines by the following rules: (1) from Q_i we can derive $x_j Q_i$ or $(1 - x_j) Q_i$ for any variable x_j; (2) from Q_i, Q_j we can derive $aQ_i + bQ_j$ for any $a, b \in \mathbb{R}$. As with Nullsatz the final line Q_S is the formal polynomial "1". Similarly to Nullsatz the degree of a PC proof π is the maximal degree of any line Q_i, and the size of π is the total number of monomials in the refutation, where multiple occurrences of the same monomial are counted for each occurrence. PC trivially p-simulates Nullsatz and the simulation is degree-preserving.

The PCR system is a simple modification to the PC proof system so that it can p-simulate Res

[6] This class is more commonly called k-Resolution, or Res(k), in proof complexity literature, but the parameter k already plays a central role in our paper.

proofs with respect to size. For PCR, polynomials are allowed to use additional variables $\overline{x}_1, \ldots, \overline{x}_n$ and axioms of the form $1 - \overline{x}_j - x_j = 0$ for all $j \in [n]$. Furthermore all terms $(1 - x_j)$ in the input polynomials in $\mathcal{P}$ are replaced by the variables $\overline{x_j}$. Intuitively although the variables x_j and $\overline{x_j}$ are distinct they stand for the negations of one another, which is enforced by the new axiom corresponding to x_j. It is not hard to see that PCR can now p-simulate Res with respect to size.

Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a collection of non-empty sets S_j over $[n]$. A *hitting set* $H \subseteq [n]$ is a set of elements such that $H \cap S_j \neq \emptyset$ for all $j \in [n]$. Let $\gamma(\mathcal{S})$ be the size of the smallest hitting set for $\mathcal{S}$. The *gap hitting set problem* is the task of distinguishing, on input $(\mathcal{S}, k, hk)$, the following two cases: (1) $\gamma(\mathcal{S}) \leq k$; (2) $\gamma(\mathcal{S}) > hk$.

▶ **Definition 3.** *The Exponential Time Hypothesis (*ETH*) states [23] that for sufficiently large m and n, no algorithm running in time $2^{o(n)}$ can decide, for given CNF τ with m clauses and n variables, whether all m clauses of τ are satisfiable or not. The Gap Exponential Time Hypothesis (*GapETH*) states [18, 27] that for sufficiently large m and n, no algorithm running in time $2^{o(n)}$ can decide, for given CNF τ with m clauses and n variables and any constant $\epsilon \in (0, 1)$, whether all m clauses of τ are satisfiable or if at most $(1 - \epsilon)m$ of the clauses are satisfiable.*

We state the following hardness results for the hitting set problem under ETH, which can be deduced from a construction by Chen and Lin [14]. The actual lemma we prove is slightly more technical but actually slightly stronger than the one we state here. A full discussion and proof of the lemma is included in Appendix A[7].

▶ **Lemma 4** (ETH-Hardness of Hitting Set)**.** *Assuming* ETH*, for sufficiently large n and $k = O(\log^{1/7-\epsilon} \log n)$ no algorithm can solve the gap hitting set problem $(\mathcal{S}, k, k^2)$ in time $n^{o(k)}$.*

Consider a set $A \subseteq \{0, 1\}^m$ of m-bit strings such that $|A| = m$. We say that A is (m, k)-*universal* if for every subset $J \subseteq [m]$ of up to k distinct positions in $[m]$, the projection $A|_J$ (restricting the strings in A to these positions) contains all possible $2^{|J|}$ binary strings of length $|J|$. Observe that we can take the dual of the set A in the following sense: if $A = \{a_1, \ldots, a_m\}$, and let $B \subseteq \{0, 1\}^m$ be the set of all strings b_j for $j \in [m]$ such that the ith bit of b_j is the jth bit of a_i. Another way to think about B is taking the strings of A to be the columns of an $m \times m$ matrix and letting B be the columns of that matrix's transpose. We say A is (m, k)-*dual-universal* if B is (m, k)-universal. Equivalently A is (m, k)-dual-universal if for every ordered subset $I \subseteq A$ of up to k distinct strings in A and for every string $s \in \{0, 1\}^{|I|}$, there exists some position $j \in [m]$ such that s is the string formed by concatenating the jth bit of all strings in I in order. The existence of efficiently constructible $(m, \log m/4)$-universal sets is known. It is also known that there exist efficiently constructible sets that are both $(m, \log m/4)$-universal *and* $(m, \log m/4)$-dual-universal. For a concrete example, [2] uses the *Paley graph* G_m on m vertices [8] For the rest of the paper we will fix an arbitrary A that is efficiently computable and is both $(m, \log m/4)$-universal and $(m, \log m/4)$-dual-universal.

3 Main reduction

We first state our main lemma from which Theorem 1 is easily proven.

[7] A similar result holds for GapETH, which can be deduced from recent work of Chalermsook et al [13]. See Appendix B for more details.

[8] Many examples of universal sets (including the Paley graph construction) are discussed in [24], as well as [3, 31]. Alternate constructions use properties such as k-wise independent sample spaces and linear codes, and counting arguments for different parameter regimes exist. Notably the Paley construction fulfills our four essential properties of being small (of size m), polytime constructable, $(m, \log m/4)$-universal, and $(m, \log m/4)$-dual-universal.

▶ **Lemma 5.** *Let $\mathcal{Q} \in \{\text{Res}, \text{TreeRes}, \text{Nullsatz}, \text{PC}, \text{PCR}\}$. For sufficiently large n and $k = O(\log^{1/3} n)$, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$. Then there exists an unsatisfiable CNF $\tau_{\mathcal{S}}$ which can be computed in time $n^{O(1)}$ such that the following two properties hold*

(i) *if $\gamma(\mathcal{S}) \leq k$ then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$;*

(ii) *if $\gamma(\mathcal{S}) > k^2$ then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$.*

Proof of Theorem 1. Assuming that $\mathcal{Q}$ is n^f automatizable for some $f := f(n) = o(\log^{1/7-\epsilon} \log n)$ for $\epsilon > 0$, we describe an efficient algorithm for the gap hitting set problem. Given an instance $(\mathcal{S}, k, k^2)$ of the gap hitting set problem over $[n]$, with n sufficiently large and $k = O(\log^{1/7-\epsilon} \log n)$, we generate the CNF $\tau_{\mathcal{S}}$, and simulate the automatizing algorithm on $\tau_{\mathcal{S}}$ for $n^{O(f)}$ timesteps. If the automatizing algorithm outputs a legal $\mathcal{Q}$ refutation of $\tau_{\mathcal{S}}$ within the allotted time, then we output "$\gamma(\mathcal{S}) \leq k$" and otherwise output "$\gamma(\mathcal{S}) > k^2$". Because $f = o(k)$ the correctness is guaranteed by Lemma 5. Thus we can decide the gap hitting set problem in time $n^{O(f)} = n^{o(k)}$, which by Lemma 4, contradicts ETH. ◀

The rest of the paper is devoted to the proof of Lemma 5. In this section we give the reduction $\tau_{\mathcal{S}}$, and prove the upper and lower bounds needed for the case of Res in Sections 4 and 5. This also gives the upper bound for PC and Res(r); the lower bounds are deferred to Appendices C and D. We briefly note that the strength of the result in Theorem 1 relies solely on the largest value we can set k to. We choose $k = O(\log^{1/7-\epsilon} \log n)$ because this is the largest value we can use and still get a contradiction with Lemma 4, but for Lemma 5 to hold we can tolerate up to $k = O(\log^{1/3} \log n)$, meaning that if the reduction in [14] were improved, a stronger version of Theorem 1 would immediately follow. Likewise, starting from the GapETH assumption we could use Lemma 17 (see Appendix B) in place of Lemma 4 and immediately get the stronger result claimed in Section 1.

Hereafter, fix $k = O(\log^{1/7-\epsilon} \log n)$ and define $m := n^{1/k}$. Observe that $k \log m = \log n$ and $k^2 < \log m$ for large enough n. In what follows we will abuse notation and x_i, y_j will denote a tuple of Boolean variables (rather than a single Boolean variable). The tuple size of x_i, y_j will be clear from context, but generally x_i will be a $O(\log m)$-tuple and y_j will be a $O(\log n)$-tuple. Additionally $\vec{x} = x_1, \ldots, x_n$, $\vec{y} = y_1, \ldots, y_m$ will denote vectors of the tuples x_i and y_j. α_i and β_j will denote a 0/1 assignment to the tuples x_i and y_j respectively, and $\vec{\alpha}, \vec{\beta}$ will each denote a 0/1 assignment to the vector of tuples $\vec{x}, \vec{y}$ respectively.

3.1 The Formula $\psi_{\mathcal{S}}$

Given a hitting set instance $\mathcal{S}$ we will define an unsatisfiable formula $\psi_{\mathcal{S}}$. Recall that A is a set of m-bit strings such that $|A| = m$ and A is both $(m, (\log m)/4)$-universal and $(m, (\log m)/4)$-dual-universal. We also define the *characteristic vector* of a set $S \subseteq [n]$ to be the binary vector $s \in \{0,1\}^n$ such that $s_i = 0$ for all $i \notin S$ and $s_i = 1$ for all $i \in S$.

The formula $\psi_{\mathcal{S}}$ will have variables $\vec{x}$ and $\vec{y}$ that will respectively encode n-by-m matricies M and N. The variables of $\vec{x}$ will define M such that each of the n rows of M is some vector in A, and the variables $\vec{y}$ will define N such that each of the m columns of N is the characteristic vector for some set S from the hitting set instance $\mathcal{S}$. In particular, x_i will indicate a vector in A to serve as the ith row of M, while y_j will indicate a set in $\mathcal{S}$ whose characteristic vector will serve as the jth column of N, with each x_i and y_j being chosen separately. For the remainder of the section, we restrict our attention to matricies M and N defined this way. We say that M and N *intersect* if $M[i,j] = N[i,j] = 1$ for some pair (i,j). $\psi_{\mathcal{S}}$ will be defined so that it is falsified whenever M and N intersect and satisfied otherwise.

Notice that when some column of M is the characteristic vector of a hitting set, $\psi_{\mathcal{S}}$ is falsified because there is no way to pick the corresponding column in N so that the two columns do not

intersect. Conversely, if none of the columns in M represent a hitting set, then there is always a way to pick N so that $\psi_{\mathcal{S}}$ is satisfied (for each column we simply pick the set that was not hit). Therefore proving that $\psi_{\mathcal{S}}$ is unsatisfiable boils down to proving that for any choice of M, some column of M represents a hitting set.

▷ Claim 6. $\psi_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof sketch. Let H be any hitting set of size at most $\frac{\log m}{4}$, which we interpret of as a set of row indicies into M. By the $(m, (\log m)/4)$-dual-universality of A, any set I of at most $(\log m)/4$ strings from A has a location such that all the strings in I contain a 1 at that location. Since rows of M are strings in A, taking $I = H$ there must exist a column j^* such that $M[i, j^*] = 1$ for every $i \in H$. Because H is a hitting set and the jth column of N is the indicator vector of a set $S \in \mathcal{S}$, there must be some $i^* \in H$ such that $N[i^*, j^*] = 1$, and so M and N intersect at (i^*, j^*). ◀

Next, we define the formula more formally. The variables of $\psi_{\mathcal{S}}$ are $\vec{x} = \{x_i \mid i \in [n]\}$ where x_i is a tuple of $\log m$ boolean variables, and $\vec{y} = \{y_j \mid j \in [m]\}$ where y_j is a tuple of $\log n$ boolean variables. Given an assignment $\vec{\alpha} = \{\alpha_i \mid i \in [n]\}$ to the $\vec{x}$-variables, $\vec{\alpha}$ encodes an n-by-m matrix $M_{\vec{\alpha}}$ where the i-th row of $M_{\vec{\alpha}}$ equals $a_{\alpha_i} \in A$ (interpreting α_i as an index in $[m]$). Similarly given an assignment $\vec{\beta} = \{\beta_j \mid j \in [m]\}$ to the $\vec{y}$-variables, $\vec{\beta}$ encodes an n-by-m matrix $N_{\vec{\beta}}$, where column j is the characteristic vector of the set $S_{\beta_j} \in \mathcal{S}$ (interpreting β_j as an index in $[n]$). We will sometimes write $M_{\vec{\alpha}}[i, j]$ as $M_{\alpha_i}[i, j]$ to stress that the ith row of $M_{\vec{\alpha}}$ is determined by α_i. Similarly, we will sometimes write $N_{\vec{\beta}}[i, j]$ as $N_{\beta_j}[i, j]$.

Lastly, we formally define the clauses in $\psi_{\mathcal{S}}$ so that it is falsified whenever $M_{\vec{\alpha}}$ and $N_{\vec{\beta}}$ intersect and satisfied otherwise.

▶ **Definition 7.** *For every $i \in [n]$ and $j \in [m]$, and for every pair of values $\alpha_i \in \{0,1\}^{\log m}$, $\beta_j \in \{0,1\}^{\log n}$ such that $M_{\alpha_i}[i, j] = 1$ and $N_{\beta_j}[i, j] = 1$, we have the clause $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ where $x_i^{\alpha_i} = \wedge_{t \in [n]} (x_i)_t^{(\alpha_i)_t}$ is the conjunction of all variables in x_i, each of which occurs positively when the corresponding bit of α_i is 1 and negatively when the corresponding bit of α_i is 0 (we define $y_j^{\beta_j}$ in the same way). This axiom is falsified iff x_i is assigned value α_i and y_j is assigned value β_j.*

This formula has the property we want because if $M_{\vec{\alpha}}$ and $N_{\vec{\beta}}$ intersect at some location i, j, then the axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ exists in $\psi_{\mathcal{S}}$ and would be falsified. Conversely, if $\psi_{\mathcal{S}}$ is falsified, then some axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ is falsified, which means $M_{\vec{\alpha}}[i, j] = N_{\vec{\beta}}[i, j] = 1$.

It is easy to check that the number of variables in $\psi_{\mathcal{S}}$ is $n \log m + m \log n$. The number of clauses is at most $n^2 m^2$, since there are $nm(mn)$ quadruplets of the form $(i, \alpha_i, j, \beta_j)$.

3.2 Redundantly Encoding $\psi_{\mathcal{S}}$

In order to prove our result we will need a way of proving both upper and lower bounds on $S_{\mathcal{Q}}(\psi_{\mathcal{S}})$, but it turns out that the lower bounds are difficult to prove if we use $\psi_{\mathcal{S}}$ as is. Thus, we will employ a standard trick in proof complexity, which is to redundantly encode the variables in the formula; more specifically we follow [2] and redundantly code blocks of variables, namely each row and column, using error-correcting codes. It is interesting to note that for our formulas, we are unable to prove even width lower bounds without the redundant encoding. In contrast, most proof complexity applications use this trick solely for the purpose of reducing size lower bounds to width lower bounds.

▶ **Definition 8.** *For $q, r, s \in \mathbb{N}$, a (q, r, s)-code is a total function f from $\{0,1\}^q$ to $\{0,1\}^r$ with the property that for any $\rho \in \{0, 1, *\}^q$ such that ρ fixes at most s values to $\{0,1\}$, $f|_\rho$ is surjective on $\{0,1\}^r$. Efficiently computable constructions using linear codes are known for any $r, q = 6r, s = 2r$ (see e.g. [2]). We say that f is r-surjective.*

Let $f_x : \{0,1\}^{6\log m} \to [m]$ be a $(6\log m, \log m, 2\log m)$-code and let $f_y : \{0,1\}^{6\log n} \to [n]$ be a $(6\log n, \log n, 2\log n)$-code. We will have a vector $x_i \in \{0,1\}^{6\log m}$ for each $i \in [n]$ and a vector $y_j \in \{0,1\}^{6\log n}$ for each $j \in [m]$. Given an assignment $\vec{\alpha}$ to all of the $\vec{x}$-variables, we will associate with $\vec{\alpha}$ an n-by-m matrix $M_{\vec{\alpha}}$, where the ith row of $M_{\vec{\alpha}}$ will be the vector $a_{f_x(\alpha_i)} \in A$. Similarly given an assignment $\vec{\beta}$ to all of the $\vec{y}$-variables, we will associate with $\vec{\beta}$ an n-by-m matrix $N_{\vec{\beta}}$, where column j is the characteristic vector corresponding to the set $S_{f_y(\beta_j)} \in \mathcal{S}$ In other words, $N_{\vec{\beta}}[i,j]$ is 1 if and only if set $S_{f_y(\beta_j)}$ contains element i.

We now define our unsatisfiable CNF $\tau_{\mathcal{S}}$ in the same way as $\psi_{\mathcal{S}}$ using these redundant encodings. Note that it is unsatisfiable for exactly the same reason as stated before.

▶ **Definition 9.** *For each $j \in [m]$, the clauses of $\tau_{\mathcal{S}}$ are defined as follows. For every $i \in [n], j \in [m]$ and for every pair of assignments (α_i, β_j) to (x_i, y_j) such that $M_{\alpha_i}[i,j] = 1$ and $N_{\beta_j}[i,j] = 1$, we have the clause $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$.*

In the redundant encoding we have $n \cdot 6\log m$ x-variables and $m \cdot 6\log n$ y-variables, for a total of $O(n \log m)$ variables when $m = n^{1/k} \ll n$. For each $j \in [m]$ the number of clauses in $\tau_{\mathcal{S}}$ is at most $n^7 m^7$ since the total number of pairs (α, β) is at most $n^6 m^6$.

The following two lemmas, which will be the focus of the rest of the paper, give tight upper and lower bounds on $S_{\mathcal{Q}}(\tau_{\mathcal{S}})$ as a function of $\gamma(\mathcal{S})$. Since we can clearly construct $\tau_{\mathcal{S}}$ in time polynomial in n, proving these two lemmas is all we need to finish Lemma 5.

▶ **Lemma 10.** *For sufficiently large n and $k = O(\log^{1/3} n)$, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$ such that $\gamma(\mathcal{S}) \leq k$. Then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$.*

▶ **Lemma 11.** *For sufficiently large n and $k = O(\log^{1/3} n)$, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$ such that $\gamma(\mathcal{S}) > k^2$. Then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for any $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$.*

It may be instructive to note that both the upper and lower bounds are exactly $n^{\Theta(\gamma(\mathcal{S})/k)} = m^{\Theta(\gamma(\mathcal{S}))}$, which is polynomial in the number of distinct assignments to $\alpha_1 \ldots \alpha_{\gamma(\mathcal{S})}$, assuming without loss of generality that the minimum hitting set of $\mathcal{S}$ is the first $\gamma(\mathcal{S})$ elements $\{1 \ldots \gamma(\mathcal{S})\} \subseteq [n]$. In Sections 4 and 5 we show how these assignments exactly characterize the shortest proof of τ.

4 Upper bound in TreeRes

In this section we prove Lemma 10. Note that it suffices to give an upper bound in TreeRes since all of the other proof systems can p-simulate TreeRes.

Proof of Lemma 10. The proof is just a formalization of the argument given in the proof of Claim 6. Using the well-known equivalence between TreeRes proofs and decision trees, it suffices to give a decision tree solving the search problem for $\tau_{\mathcal{S}}$; that is, a decision tree (over the underlying variables of $\tau_{\mathcal{S}}$), where every leaf l is labeled with a clause of $\tau_{\mathcal{S}}$ that is falsified by the partial assignment that labels the path to l.

We will first show that if $\gamma(\mathcal{S}) \leq k$, then there is a height $2\log n$ decision tree (and therefore size n^2) for the unencoded formula $\psi_{\mathcal{S}}$. Since $\gamma(\mathcal{S}) \leq k$, assume without loss of generality that $H = \{1, \ldots, k\}$ is a valid hitting set for $\mathcal{S}$. The decision tree for $\psi_{\mathcal{S}}$ consists of two phases. First, the decision tree will branch on all of the Boolean variables in $x_1, \ldots, x_k$. This will result in a full binary tree, call it T, of depth $k \log m$. In the second phase, at each leaf vertex of T we will query all of the variables of some y_j variable, where the choice of y_j will be a function of the path taken in T.

Consider some path in T leading to leaf $l_{\vec{\alpha}}$, corresponding to the assignment $\vec{\alpha} = \alpha_1, \ldots \alpha_k$ for $x_1, \ldots, x_k$. The assignment $\vec{\alpha}$ corresponds to an ordered set of strings $I \subseteq A$, where $|I| \leq k$. Since $k \in O(log^{1/3} n)$ and $m = n^{1/k}$, $k \leq \frac{\log m}{4}$ for large n. By the $(m, \log m/4)$-dual-universal property of A there is some $j \in [m]$ such that I restricted to position j is all 1's, and thus $M_{\vec{\alpha}}[i, j] = 1$ for all $i \in [k]$. In the second phase, at this leaf vertex $l_{\vec{\alpha}}$ of T we will then query all of the Boolean variables in y_j. Let β_j be one partial assignment to these variables and consider the path labeled by $\vec{\alpha}\beta_j$ leading to the leaf vertex $l_{\vec{\alpha}\beta_j}$. Since $\{1, \ldots, k\}$ is a hitting set for $\mathcal{S}$ we are guaranteed that $N_{\vec{\beta}_j}[i, j] = 1$ for at least one $i \in [k]$, and since $M_{\vec{\alpha}}[i, j] = 1$ for all $i \in [k]$, one of the clauses in $\tau_{\mathcal{S}}$ must be violated by the partial assignment $\vec{\alpha}, \beta_j$, so we label $l_{\vec{\alpha}\beta_j}$ with any such clause. The resulting decision tree thus solves the search problem associated with $\psi_{\mathcal{S}}$ and has height $k \log m + \log n = 2 \log n$.

The decision tree for the redundant version $\tau_{\mathcal{S}}$ is essentially the same but instead we query the redundant encodings of the variables. First, we query $x_1, \ldots, x_k$, resulting in a full binary tree of height $k \cdot 6 \log m$, and then, we query a particular y_j (depending on the path taken in T), which is $6 \log n$ variables, and thus the height is $k \cdot 6 \log m + 6 \log n = 12 \log n$. ◀

5 Nonautomatizability of Res and TreeRes

In this section we prove Lemma 11 for the case of $\mathcal{Q} =$ Res, which implies the result for TreeRes as well. We begin by proving a *wide clause lemma* for $\tau_{\mathcal{S}}$, which alone is enough to prove lower bounds for TreeRes (using the size-width relationship for TreeRes due to Ben-Sasson and Wigderson [9]); for general Res, we apply a standard application of random restrictions to reduce to width.

Our notion of "wide" will be a bit richer than the usual definition. For a clause D, let $I_0(D)$ be the set of all $i \in [n]$ for which there are at least $\log m$ literals in D that correspond to variables from x_i. Likewise let $J_0(D)$ be the set of all $j \in [m]$ for which there are at least $\log n$ literals in D that correspond to variables from y_j.

▶ **Lemma 12** (Wide Clause Lemma). *For sufficiently large n, if $\gamma(\mathcal{S}) > k^2$ and f_x (f_y) is $\log m$-surjective ($\log n$-surjective, respectively), then for any* Res *refutation π refuting $\tau_{\mathcal{S}}$ there exists a clause $D \in \pi$ such that $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$.*

Proof. We follow the *prover-delayer game* of [5, 36] in the style of [6]. The width-w game on an unsatisfiable formula τ is played between a Delayer, who is asserting that she has a satisfying assignment for τ, and a Prover, who is trying to force the Delayer into a contradiction by asking her values of the underlying variables. However, the Prover has limited memory and can only remember the values of up to w of the variables at a time.

Both players know τ and the contents of the Prover's memory, which is initially empty. At the start of each round there are at most $w - 1$ values in memory. The Prover asks the Delayer the value of some variable whose value is not currently in memory. The Delayer responds with an answer (either 0 or 1), and upon receiving the answer, the Prover adds this assignment to his memory (increasing the number of stored values by 1). He can then erase (forget) any existing values from memory, possibly decreasing the number of stored values. The Prover declares victory if at some point, the partial assignment written in his memory falsifies one of the clauses of τ. The Delayer has a winning strategy for the width-w game on τ if no matter how the Prover plays the game, he cannot win. It was shown [5, 36] that the Delayer has a winning strategy for the width-w game if and only if the Res width of τ is at least $w - 1$.

For our formula $\tau_{\mathcal{S}}$, the game proceeds as above, but now let D be the set of literals in the Prover's memory, and we demand instead of only holding w variables total in memory that $|I_0(D)| \leq k^2$ and $|J_0(D)| \leq k$. By the transformation from [36], the Prover has a winning strategy for this game if there is a Res refutation such that $|I_0(D)| \leq k^2 - 1$ and $|J_0(D)| \leq k - 1$ for every clause D.

Therefore the Delayer has a winning strategy for this game if and only if the lemma holds. The Delayer's winning strategy is as follows.

- If the Prover asks about a variable in x_i:
  - If $i \notin I_0(D)$ and after adding this bit there are still less than $\log m$ variables from x_i in memory, the Delayer can answer with either 0 or 1 arbitrarily.
  - If $i \notin I_0(D)$ but after adding this bit to memory there are now $\log m$ variables from x_i in memory, the Delayer uses the fact that $|J_0(D)| \leq k \leq \log m/4$ and the $(m, \log m/4)$-universal property of A to find a string $a_0 \in A$ such that $a_0|_{J_0(D)}$ is the all-zeros string, and uses the surjective property of f_x to find an assignment α_i consistent with the assignment to the x_i variables in memory such that $f_x(\alpha_i) = a_0$. The Delayer will remember the assignment α_i for x_i from now on, and note that $I_0(D)$ now contains i.
  - Finally if $i \in I_0(D)$ then the Delayer is maintaining an assignment α_i for x_i, so she answers according to α_i.
- If the Prover asks about a variable in y_j:
  - If $j \notin J_0(D)$ and after adding this bit there are still less than $\log n$ variables from y_j in memory, the Delayer can answer with either 0 or 1 arbitrarily.
  - If $j \notin J_0(D)$ but there are now $\log n$ variables from y_j in memory, the Delayer uses the fact that $|I_0(D)| \leq k^2 < \gamma(\mathcal{S})$ and finds a set S_0 that doesn't contain any element $i \in I_0(D)$, and uses the surjective property of f_y to find an assignment β_j consistent with the assignment to the y_j variables in memory such that $f_y(\beta_j) = S_0$. The Delayer will remember the assignment β_j for x_j, and note that $J_0(D)$ now contains j.
  - Finally if $j \in J_0(D)$ then the Delayer is already maintaining an assignment β_j for y_j, so she answers according to β_j.
- Whenever the Prover erases a variable from x_i from his memory, if $i \in I_0$ and now there are less than $\log m$ variables from x_i in memory, the Delayer forgets α_i. (note that i is no longer in I_0) Similarly, whenever the Prover erases a variable from y_j from his memory, if $j \in J_0$ and now there are less than $\log n$ variables from y_j in memory, the Delayer removes β_j from J_0. (note that j is no longer in J_0)

Assume for contradiction the game ends with the Prover winning. Consider when the game ends, and say the Prover claims the axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ was falsified, and thus that $M_{\vec{\alpha}}[i,j] = N_{\vec{\beta}}[i,j] = 1$. First, consider the case when either $i \notin I_0$ or $j \notin J_0$. In either case there are is at least one variable in the axiom that is not in memory, which means that it has not been falsified, which is a contradiction. So assume that $i \in I_0$ and $j \in J_0$, and consider the last time that i was added to I_0 and the last time that j was added to J_0. Assume that i was added after j. Since j was in J_0 at the time we defined α_i, $M_{\alpha_i}[i,j] = 0$ by our choice of α_i, which is a contradiction. Finally assume that j was added after i. Then since i was in I_0 at the time we defined β_j, $f_y(\beta_j)$ does not contain i, and so $N_{\beta_j}[i,j] = 0$, which is also a contradiction. ◀

Before proceding on to the proof of Lemma 11, we need to change Lemma 12 slightly, in order to be able to apply a restriction argument to turn width lower bounds into size lower bounds for $\tau_{\mathcal{S}}$. We use the notation $f|_\rho$ to denote the *restriction* of the function f over $x_1 \dots x_s$ by $\rho \in \{0,1,*\}^s$, which is the function f over the variables x_i for all $i \in \rho^{-1}(*)$ obtained by setting all other variables x_j to $\rho(j)$. Likewise we use the notation $\tau|_\rho$ to denote the restriction of the tautology τ by ρ.

▶ **Definition 13.** *Let $\rho_{x_i} \in \{0,1,*\}^{x_i}$ and let $\rho_{y_j} \in \{0,1,*\}^{y_j}$. Furthermore, let $\mathcal{R}$ be the set of all $\vec{\rho} = \{\rho_{x_1} \dots \rho_{x_n}, \rho_{y_1} \dots \rho_{y_m}\}$, such that for all $i \in [n]$ and $j \in [m]$, $|\rho_{x_i}^{-1}(*)| = 5 \log m$ and $|\rho_{y_j}^{-1}(*)| = 5 \log n$. Let f_x^i be the function f_x on the variables $\rho_{x_i}^{-1}(*)$ after restricting all other inputs to ρ_{x_i}, and likewise for f_y^j.*

▶ **Lemma 14** (Wide Clause Lemma under restrictions). *For sufficiently large n and $\rho \in \mathcal{R}$, if $\gamma(\mathcal{S}) > k^2$ then for any* Res *refutation π refuting $\tau_{\mathcal{S}}|_{\vec{\rho}}$ there exists a clause $D \in \pi$ such that $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$.*

We omit the proof of Lemma 14, as it is essentially identical to Lemma 12. The only difference is that in each row i the Delayer chooses α_i based on f_x^i instead of f_x, and likewise for the columns. Note that f_x was $2 \log m$ surjective before the restriction, and since only $\log m$ variables are fixed in every row f_x^i is still $\log m$ surjective (and similarly for f_y^j).

Proof of Lemma 11. Let π be a Res refutation of $\tau_{\mathcal{S}}$ and assume for contradiction that $|\pi| < n^{k/16}$. First, consider a clause $D \in \pi$ such that $|I_0(D)| \geq k^2$. For each $i \in I_0(D)$, the chance that a randomly chosen $\vec{\rho} \in \mathcal{R}$ doesn't set one of the x_i literals in D to 1 is less than $(1 - (\frac{3}{4} \cdot \frac{1}{2}))^{\log m}$. Thus the probability that no $i \in I_0(D)$ sets D to 1 is at most $(\frac{5}{8})^{k^2 \log m} = (\frac{5}{8})^{k \log n} < \frac{1}{n^{k/8}}$. By a union bound the probability that some clause D in π satisfying $|I_0(D)| \geq k^2$ survives a random restriction is less than $\frac{n^{k/16}}{n^{k/8}} = \frac{1}{n^{k/16}}$, using the fact that $|\pi| < n^{k/16}$.

Similarly the probability that some clause $D \in \pi$ satisfying $|J_0(D)| \geq k$ survives a random restriction is at most $\frac{1}{n^{k/16}}$. Thus with probability at least $1 - \frac{2}{n^{k/16}}$, all clauses D satisfying $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$ are set to 1 by a random restriction, and thus there exists a restriction $\vec{\rho} = \{\rho_{x_1} \dots \rho_{x_n}, \rho_{y_1} \dots \rho_{y_m}\}$ setting all such clauses to 1. However, this contradicts Lemma 14, as $\tau_{\mathcal{S}}|_{\vec{\rho}}$ must still have at least one such clause. Thus $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{c_l k}$ for $c_l = \frac{1}{16}$. ◀

6 Conclusions

In terms of optimality of our results, the constructions in [13, 14] are not known to be optimal, and any hardness results against approximating the gap hitting set problem in time $n^{o(k)}$ for a larger value of k immediately gives a lower bound of $n^{o(k)}$ against automatizability. While their results are "optimal" in terms of fixed-parameter tractability guarantees, there is nothing limiting a different reduction from getting the same (or even a weaker) result that works for larger values of the fixed parameter. In fact classically the hitting set problem has no $o(\log n)$ approximations; the obstacle to using this classical hardness is that it only rules out algorithms that get $o(\log n)$ approximation for *all* hitting set sizes, whereas [14] rules out algorithms for any *fixed* hitting set size. Nevertheless it's believed that $\Omega(\log n)$ hardness holds even for fixed hitting set sizes, and getting a reduction that achieves this result would strengthen our argument.

On the flip side, all of our hardness results also work for TreeRes and Nullsatz, and therefore this reduction is limited to quasipolynomial hardness. This is in line with the details of the reduction; by the crucial fact that $k^2 \leq \frac{\log m}{4} = \frac{\log n}{4k}$, this technique can't be strengthened past the $k = o(\log^{1/3} n)$ threshold.[9] Thus, the upper limit of improving the reductions of [13, 14] coincides almost exactly with the upper limit of our argument, and by extension any argument using the machinery of [2].

A central motivation of this work was to make the techniques clear and simple in hopes that they can be made to work for stronger systems such as SA and SoS, where no lower bounds are known. A degree lower bound matching our results for Res and PC would shed light on the limitations of our current approximation algorithms. Similarly it's possible that this proof can be made to work for the case of TreeCP or CP, where instead of arguing lower bounds directly we can hope to leverage the power of lifting theorems [21, 22]; in particular a constant-sized lifting gadget would immediately give results for TreeCP matching our other results.

[9] If we allow the formula to be satisfiable in the case where $\gamma(\mathcal{S}) > k^2$ we only need $k \leq \frac{\log m}{4}$ since we only ever allow the proof to query k columns. This can also be made to work in the base setting where the formula must always be unsatisfiable by standard tricks. However this still yields a barrier of $k = o(\log^{1/2} n)$.

Acknowledgements

The authors are grateful to Noah Fleming, Pravesh Kothari, Denis Pankratov, Robert Robere and Avi Wigderson for helpful conversations. We also thank Yijia Chen for providing more details on the construction in [14], and Pasin Manurangsi for giving feedback on the parameters in [13].

References

1. Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is np-hard to linearly approximate. *J. Symb. Log.*, 66(1):171–191, 2001.
2. Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM J. Comput.*, 38(4):1347–1363, 2008.
3. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.
4. Albart Atserias and Moritz Müller. Automating resolution is np-hard. 2019.
5. Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008.
6. Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. *ACM Trans. Comput. Log.*, 17(3):19:1–19:30, 2016.
7. Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bound on hilbert's nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806, 1994.
8. Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 274–282, 1996.
9. Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.
10. 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.
11. Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for frege systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000.
12. Michael Brickenstein and Alexander Dreyer. Polybori: A framework for gröbner-basis computations with boolean polynomials. *J. Symb. Comput.*, 44(9):1326–1345, 2009.
13. Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-eth to fpt-inapproximability: Clique, dominating set, and more. *CoRR*, abs/1708.04218, 2017.
14. Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 505–514, 2016.
15. Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 174–183, 1996.
16. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *J. ACM*, 5(7):394–397, 1961.
17. Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.
18. Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:128, 2016.
19. K. Eickmeyer, M. Grohe, and M. Gruber. Approximation of natural w[p]-complete minimisation problems is hard. In *23rd Annual IEEE Conference on Computational Complexity, CCC*, pages 8–18, 2008.
20. Nicola Galesi and Massimo Lauria. On the automatizability of polynomial calculus. *Theory Comput. Syst.*, 47(2):491–506, 2010.

21 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 902–911, 2018.

22 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA*, pages 132–143. IEEE Computer Society, 2017.

23 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

24 Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Springer Publishing Company, Incorporated, 1st edition, 2010.

25 Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S^1_2 and EF. *Inf. Comput.*, 140(1):82–94, 1998.

26 J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optimization*, 11(3):796–817, 2001.

27 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 78:1–78:15, 2017.

28 Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1):397 – 446, 2002.

29 Joao Marques-Silva and Karem A. Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48:506–521, 1999.

30 Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *DAC*, 2001.

31 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

32 Jakob Nordström. On the interplay between proof complexity and sat solving. *ACM SIGLOG News*, 2(3):19–44, 2015.

33 Ryan O'Donnell. SOS is not obviously automatizable, even approximately. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:141, 2016.

34 Pablo Parrilo. Structured semidefinite programs and semialgebraic geometry. *Methods in Robustness and Optimization*, 2000.

35 K. Pipatsrisawat and A. Darwiche. On the power of clause-learning sat solvers as resolution engines. *Artificial Intelligence*, 175(2):512 – 525, 2011.

36 Pavel Pudlák. Proofs as games. *The American Mathematical Monthly*, 107(6):541–550, 2000.

37 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 245–254, 2008.

38 Prasad Raghavendra, Tselil Schramm, and David Steurer. High-dimensional estimation via sum-of-squares proofs. *CoRR*, 2018.

39 Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A switching lemma for small restrictions and lower bounds for k-dnf resolution. *SIAM J. Comput.*, 33(5):1171–1200, 2004.

40 H Sherali and W Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Disc. Math*, 3(3):411–430, 1990.

41 Johan Thapper and Stanislav Zivny. The power of sherali-adams relaxations for general-valued csps. *CoRR*, 2016.

42 M. Vinyals, J. Elffers, J. Giráldez-Cru, S. Gocht, and J Nordström. On the power of clause-learning sat solvers as resolution engines. *Artificial Intelligence*, 175(2):512 – 525, 2011.

Appendix A: Hardness of Hitting Set under ETH

In this section we give an overview of how to prove our results under the ETH assumption. We state and prove an analogue of Lemma 4 for ETH using the result of [14], and then point out the (superficial) changes needed to prove nonautomatizability results.

▶ **Lemma 15** (Lemma 4 under ETH). *Assuming* ETH*, for sufficiently large* n*,* $k = O(\sqrt{\log n})$*, and* $h = O(\log^{1/7-\epsilon} \log n)$ *for any* $\epsilon > 0$*, no algorithm can solve the gap hitting set problem* $(\mathcal{S}, k, hk)$ *in time* $n^{o(h)}$*.*

Proof. Let $K := K(n)$ be a function such that $K^{O(K^{3.5})} = \frac{1}{4}\sqrt{\frac{n}{\sqrt{K}}}$, and notice that for any constant $\epsilon > 0$, $K \in \omega(\log^{1/3.5-2\epsilon} n)$. Additionally let $h := h(n) = \sqrt{K}$. Note that while the theorems in [14] are phrased for the *dominating set* problem, there exists a trivial transformation from dominating set to hitting set where the sets are indexed by vertices v and S_v contains exactly the vertices adjacent to v.

We start with the 3-color problem, and the fact that under ETH there exist no $2^{o(n)}$ algorithms to solve 3-coloring. First we consider the standard reduction from 3-coloring to K-clique as follows: given a graph G on n vertices, partition the vertices into K parts $V_1 \dots V_K$, each with exactly n/K vertices. We transform G into a new graph G' whose vertices are split into K groups $V'_1 \dots V'_K$ each of size $3^{n/K}$. In each group V'_i we put a vertex for every possible valid 3-coloring of the vertices in V_i, and we connect two vertices in different groups V'_i, V'_j iff their colorings are consistent in induced graph of G restricted to the vertices $V_i \cup V_j$. We don't connect any two vertices in the same group V'_i. Therefore there exists a K-clique in G' iff there exist ways to 3-color all the different groups of vertices in G consistent with one another, which exists iff G has a 3-coloring. Note that we can assume $|G'| = 3^{n/K}K = 2^{\Theta(n/K)}$, because if it had size $2^{o(n/K)}$ then brute force over all K-cliques would let us find a K-clique in G' (and thus 3-coloring in G) in time $2^{o(n)}$.

Let $d = (30h^2(K+1)^2)^{4K^3+3h} = K^{O(K^3)}$. Following the proof of Theorem 4.1 of [14] we can transform the graph G' into a hitting set instance $\mathcal{S}$ over a universe of size $|G'|^{O(h)} = (2^{O(n/K)})^{O(\sqrt{K})} = 2^{O(n/\sqrt{K})}$ such that (1) if G' has a K-clique then $\gamma(\mathcal{S}) < 1.1 \cdot d^h$; (2) if G' has no K-clique then $\gamma(\mathcal{S}) > h \cdot d^h/3$. Setting $N := 2^{O(n/\sqrt{K})}$ and $k := 1.1d^h = K^{O(K^{3.5})} \leq \sqrt{\log N}$ we get a hitting set instance $(\mathcal{S}, k, hk)$ over $[N]$. Note that we have fulfilled the guarantee that $hk \leq K^{O(K^{3.5})} \leq \frac{1}{4}\sqrt{\log N} \leq \frac{1}{4}\frac{\log N}{\sqrt{\log N}} \leq \frac{1}{4}\log N^{1/k}$ and thus we can set $m := N^{1/K}$ and use $(m, \log m/4)$-universality and dual universality as usual. Furthermore we note that the reduction of [14] requires that $N^{6/(K+6)}$ be greater than $(K+6)!$ and d, both of which are guaranteed because $(K+6)! < K^K \ll N^{1/K}$.

This construction runs in time $f(h, K) \cdot N$, where f is some function that is subsumed in magnitude by N for our choice of h and K. Thus since $N = 2^{O(n/\sqrt{K})}$ and our reduction runs in time $\mathrm{poly}(N)$, under ETH no algorithm can decide the gap hitting set problem $(\mathcal{S}, k, hk)$ in time $N^{o(\sqrt{K})} = N^{o(h)}$. Since $N \ll 2^{O(n)}$ we find that $h = \sqrt{\log^{1/3.5-2\epsilon} n} > O(\log^{1/7-\epsilon} \log N)$ for any $\epsilon > 0$. ◀

We note that by our construction of $\tau_{\mathcal{S}}$, $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) = n^{\Theta(\frac{\gamma(\mathcal{S})}{k})}$. As before if $\gamma(\mathcal{S}) \leq k$ then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) = n^{O(1)}$, but now if $\gamma(\mathcal{S}) > k$ we have $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) = n^{\Omega(h)}$, where before $h = k$ because our instances were $(\mathcal{S}, k, k^2)$. Since $h < k$ all our proofs can be applied with hk in place of k^2. Thus we can now repeat the proof of Theorem 1 to prove nonautomatizability in time n^f for any $f = o(h)$, using Lemma 15 and the fact that the gap between the exponents in the two proof sizes is $\Omega(h)$.

Appendix B: Hardness of Hitting Set under GapETH

In this section we improve Theorems 1 and 2 under the stronger GapETH assumption.

▶ **Theorem 16** (Main Theorem for GapETH). *Let* $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$*. Assuming* GapETH *holds* $\mathcal{Q}$ *is not* n^f*-automatizable for any* $f = \tilde{o}(\log \log n)$*. Furthermore for* $\mathcal{Q} =$ Res(r)*,* $\mathcal{Q}$ *is not* $n^{f/\exp(r^2)}$*-automatizable for any* $f = \tilde{o}(\log \log n)$ *(where* $\varepsilon > 0$ *is any constant) if* $r = O(\sqrt{\log f})$*.*

We do this in a black-box way by improving Lemma 4 in the case of GapETH. The proof follows the reduction in [13] from the *label cover* problem to the gap hitting set problem, but the definitions of the problem and details of the reduction are omitted as we only need to focus on how the parameters change at each step.

▶ **Lemma 17** (Lemma 4 under GapETH)**.** *Assuming* GapETH*, for sufficiently large n and $k = \tilde{O}(\log\log n)$ no algorithm can solve the gap hitting set problem $(\mathcal{S}, k, k^2)$ in time $n^{o(k)}$.*

Proof. Let $K(n)$ be a function such that $K^{K^{O(K)}} = 2^{O(n/K)}$, and note that $K \in \omega(\frac{\log n}{\log\log n})$. Following the proof of Theorems 4.3 and 4.4 of [13] consider an arbitrary label cover instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$, where:

- $|U| = n$
- $|V| = O(n)$
- $|\Sigma_U| = O(1)$
- $|\Sigma_V| = O(1)$
- $|\Pi| = O(|\Sigma_U||\Sigma_V|) = O(1)$

Assuming GapETH it is known that no $2^{o(|U|)}$-time algorithm distinguishes between a *max covering* of size $|U|$ and a max covering of size less than $(1-\epsilon)|U|$ for any sufficiently large (constant) $\epsilon > 0$. We can transform this into a new label cover instance $\Gamma' = (G' = (U', V', E'), \Sigma_{U'}, \Sigma_{V'}, \Pi')$ where

- $|U'| = \binom{|U|}{(K\ln K)/\epsilon} = n^{O(K\ln K)}$
- $|V'| = K$
- $|\Sigma_{U'}| = |\Sigma_U|^{(K\ln K)/\epsilon} = K^{O(K)}$
- $|\Sigma_{V'}| = |\Sigma_V|^{n/K} = 2^{O(n/K)}$
- $|\Pi'| = O(|\Sigma_{U'}||\Sigma_{V'}|) = 2^{O(n/K)}$
- Γ has a max covering of size $|U|$ iff Γ' has a max covering of size $|U'|$
- Γ has no max covering of size $(1-\epsilon)|U|$ iff Γ' has no max covering of size $\frac{1}{K^K}|U'|$

For our choice of K, it holds that $|\Gamma'|$ is dominated by $|\Sigma_{V'}| = 2^{O(n/K)}$. So under GapETH it is impossible to distinguish between Γ' having a max covering of size $|U'|$ and not having a max covering of size $\frac{1}{K^K}|U'|$ in time $2^{o(n)} = |\Gamma'|^{o(K)}$. Theorem 4.4 of [13] shows that distinguishing these two cases on Γ' implies distinguishing *min-right coverings* of size at most $|V'| = K$ and those of size greater than K^2 for the same label cover instance $|\Gamma'|$. Using this fact and following Theorem 5.4 of the same paper, we transform Γ' in time $\mathrm{poly}(|\Gamma'|)$ into a hitting set instance $\mathcal{H} = (\mathcal{U}, \mathcal{S})$, where

- $|\mathcal{U}| = |U'||V'|^{|\Sigma_{U'}|} = n^{O(K\ln K)}K^{K^{O(K)}} = K^{K^{O(K)}}$
- $|\mathcal{S}| = |V||\Sigma_V| = K2^{O(n/K)} = 2^{O(n/K)}$
- $\gamma(\mathcal{S})$ is equivalent to the min-right covering number of $|\Gamma'|$

Define $N = |\mathcal{H}|$, and because $K^{K^{O(K)}} = 2^{O(n/K)}$ we get $N = |\mathcal{U}||\mathcal{S}| = 2^{O(n/K)}$. We now define $k(n)$ to be such that $k(N) = K(n)$, which can be shown to be $\tilde{O}(\log\log n)$ (suppressing $\log\log\log n$ factors). Therefore under GapETH there doesn't exist any algorithm that can distinguish between $\gamma(\mathcal{H}) \le k(N)$ and $\gamma(\mathcal{H}) > k^2(N)$ in time $N^{o(k(N))}$ for hitting set instances $\mathcal{H}$ of size N. ◀

Appendix C: Nonautomatizability of Res(r)

In this section we prove the following Lemma, which proves a lower bound for Res(r) refutations of $\tau_{\mathcal{S}}$. Part (2) of Theorem 1 (the nonautomatizability of Res(r)) follows as a corollary.

▶ **Lemma 18.** *For sufficiently large n, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$ such that $\gamma(\mathcal{S}) > k^2$ and $k^2 < \log m$. Then for $\mathcal{Q} =$ Res(r), $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{k/\exp(r^2)}$.*

Proof. Suppose that π is a small Res(r) refutation of $\tau_{\mathcal{S}}$; thus each line of the proof is a disjunction of size-r conjunctions (where $r \ll \log n$). At a high level, we will show that there exists a random restriction $\rho \in \mathcal{R}$ (defined in Definition 13) such that $\pi|_\rho$ is a small-width Res proof refuting $\tau_{\mathcal{S}}|_\rho$, which contradicts the Wide Clause Lemma (Lemma 12).

In order to prove the existence of such a restriction, we will apply the switching lemma proven in [39] which is specifically designed to work for Res(r). More specifically, the restriction will leave a constant fraction of the variables unset. In contrast, standard switching lemmas such as those used to prove bounded-depth circuit lower bounds leave at most $1/\log n$ variables unset. The fact that the restriction is small (sets only a constant fraction of variables) will allow us to maintain that $\tau_{\mathcal{S}}|_\rho$ is still an encoded version of $\tau_{\mathcal{S}}$, but where the encoding length is somewhat smaller. Whereas standard switching lemmas typically convert disjunctions of r-conjunctions to decision trees of height r (in order to apply it repeatedly), in our case, we only need to apply the switching lemma once, and therefore we are content with converting disjunctions of r-conjunctions to decision trees of height w, where r is much smaller than w. This setting of parameters is what makes it possible to obtain a switching lemma that sets only a constant fraction of the inputs. After applying the restriction, the proof is a sequence of sound inferences, where each line is a height-w decision tree. [39] show how to convert such a refutation into a width w' refutation, where w' is not much larger than w, and thus we can apply our Wide Clause Lemma in order to obtain a contradiction.

The switching lemma (showing the existence of the restriction ρ) is argued in stages; in stage i we show that for any i-DNF D either there exist many restrictions in $\mathcal{R}$ that set D to 1 or we can create a small height decision tree with each leaf labeled by D restricted by the path to the leaf leaf, and such that the resulting DNF at every leaf is a $(i-1)$-DNF. To do this we take the i-DNF from the previous round consider its *covering number*, where the covering number $c(D)$ is the size of the smallest set of variables which intersects every term in D. If the covering number is large, then many terms are independent and are thus set to 1 by a random restriction with high probability. If the covering number is small, then we can query all variables in the cover to turn D into a $(i-1)$-DNF. Continuing until $i = r$ gives us a small height decision tree for all $D \in \pi$ with small $c(D)$, while taking a union bound over all $D \in \pi$ with large $c(D)$ ensures that there exists a restriction $\rho \in \mathcal{R}$ that kills off all such DNFs. The resulting proof π can then be shown to have a small Res proof given these two facts, which completes the proof.

Let s be a parameter to be set later, and assume for contradiction that there exists a Res(r) proof π such that $|\pi| < n^s$. Define sequences $s_0 \dots s_r, p_1 \dots p_r$ as follows:

$$s_0 = (\prod_{i=1}^{r} \frac{2(6/5)^{i+1}}{i}) s \log n$$

$$s_i = (\frac{i}{2(6/5)^{i+1}}) s_{i-1}$$

$$p_i = 2^{-2s_i}$$

Observe that that $s_k = s \log n$, and that $s_i \gg \frac{s_{i+1}}{4}$.

Consider any i-DNF D such that $c(D) > s_i$. By the pigeonhole principle there exist s_i/i terms $T_1 \dots T_{s_i/i}$ in D which are mutually disjoint. Let $\rho \sim \mathcal{R}$ be defined as in Lemma 11. Then the

probability that D is not set to 1 by ρ is at most the probability that no term T_j is set to 1, and since they are disjoint this happens with probability $(1-(\frac{5}{6})^i)^{s_i/i} = e^{-(6/5)^i s_i/i} < 2^{-2s_{i+1}} = p_{i+1}$.

Now consider an r-DNF D. We claim that

$$\Pr_{\rho\sim\mathcal{R}}[\mathsf{DT}(F|_\rho) > \sum_{i=0}^{r-1} s_i] \leq \sum_{i=1}^{r} 2^{(\sum_{j=i}^{r-1} s_j)} p_i$$

where $\mathsf{DT}(F|_\rho)$ is the height of a minimal decision tree for $F|_\rho$. We prove this claim by induction on r. In the case when $r=1$, either $c(D) \leq s_0$, in which case the claim holds trivially, or $c(D) > s_0$, in which case it is killed off with probability $p_1 = p_1 2^{\sum_{j=1}^{r-1} s_j}$ for $r-1=0$.

Inductively assume the claim holds for all $r-1$-DNFs and consider an r-DNF D. Again we consider two cases, when $c(D) \leq s_{r-1}$ and when $c(D) > s_{r-1}$. In the former case, let H be the set of s_{r-1} variables needed to cover all terms of D, and note that $\mathsf{DT}(D) \leq \mathsf{DT}(D/H) + s_{r-1}$ as we can query all variables in H first. Since D/H is an r-DNF, applying the induction hypothesis along with a union bound over all $2^{s_{r-1}}$ settings of H gives

$$\begin{aligned}
\Pr_{\rho\sim\mathcal{R}}[\mathsf{DT}(D|_\rho) > \sum_{i=0}^{r-1} s_i] &\leq 2^{s_{r-1}} \Pr_{\rho\sim\mathcal{R}}[\mathsf{DT}((D/H)|_\rho) > \sum_{i=0}^{r-2} s_i] \\
&\leq 2^{s_{r-1}} \sum_{i=1}^{r-1} 2^{(\sum_{j=i}^{r-2} s_j)} p_i \\
&\leq \sum_{i=1}^{r} 2^{(\sum_{j=i}^{r-1} s_j)} p_i
\end{aligned}$$

In the latter case, when $c(D) > s_{r-1}$, as shown before

$$\Pr_{\rho\sim\mathcal{R}}[\mathsf{DT}(D|_\rho) > \sum_{i=0}^{r-1} s_i] \leq \Pr_{\rho\sim\mathcal{R}}[\mathsf{DT}(D|_\rho) > 0] \leq p_r \ll \sum_{i=1}^{r} 2^{(\sum_{j=i}^{r-1} s_j)} p_i$$

We now use this claim and take a union bound over all $D \in \pi$ to show that there exists a restriction ρ which makes all $D \in \pi$ have a small decision tree, which we then connect to the width of any Res proof of $\tau_\mathcal{S}|_\rho$ to get a contradiction.

$$\begin{aligned}
\Pr_{\rho\sim\mathcal{R}}[\exists D \in \pi \mid \mathsf{DT}(D|_\rho) > \sum_{i=0}^{r-1} s_i] &\leq n^s \sum_{i=1}^{r} 2^{(\sum_{j=i}^{r-1} s_j)} p_i \\
&\leq \sum_{i=1}^{r} 2^{(\sum_{j=i}^{r-1} s_j) + s\log n} p_i \\
&\leq \sum_{i=1}^{r} 2^{(\sum_{j=i}^{r} s_j)} p_i \\
&\leq \sum_{i=1}^{r} 2^{\frac{4}{3}s_i} 2^{-2s_i} \\
&\leq \sum_{i=1}^{r} 2^{-\frac{2}{3}s_i} \\
&\leq r 2^{-\frac{2}{3}s_r} \\
&\leq 2^{\log r - \frac{2}{3} s \log n} \\
&\ll \tfrac{1}{2}
\end{aligned}$$

Thus there exists a $\rho \in \mathcal{R}$ such that for all $D \in \pi$,

$$\mathsf{DT}(D|_\rho) \leq \sum_{i=0}^{r-1} s_i \leq r \cdot s_0 \leq r((\prod_{i=1}^{r} \frac{2(6/5)^{i+1}}{i}) s \log n) \ll \frac{2^{r^2}}{r} s \log n$$

Set $s = \frac{k}{2^{r^2}}$, and thus $\mathsf{DT}(D|_\rho) \ll \frac{k \log n}{r}$. So $\pi|_\rho$ is a Res(r) proof where every line can be represented by a decision tree of height $\frac{k \log n}{r}$. It was shown (Theorem 5.1 in [39]) that these clauses can be made into a Res proof π' refuting $\tau_{\mathcal{S}}|_\rho$ such that $w(\pi') \ll r\frac{k \log n}{r} = k \log n$. But as usual we can still apply Lemma 12 after restricting $\tau_{\mathcal{S}}$ by ρ, and thus we get a contradiction. ◀

We briefly state two special cases of Theorem 1, part (2): the case where r is constant and where we show no (weakly) quasipolynomial automatizability, and the case of maximal r where we achieve superpolynomial automatizability.

▶ **Corollary 19.** *Assuming* GapETH*, for any constant* c*,* Res(c) *is not automatizable in time* n^k *for any* $k = \tilde{o}(\log \log n)$.

▶ **Corollary 20.** *Assuming* GapETH*,* Res(r) *is not* $n^{O(1)}$*-automatizable for any* $r = \tilde{o}(\sqrt{\log \log \log n})$.

Appendix D: Nonautomatizability of Nullsatz and PC

Galesi and Lauria [20] extended the argument due to Alekhnovich and Razborov [2] to prove that Nullsatz, PC and PCR are also not polynomially automatizable. In this section we similarly extend our proof to apply to these systems, obtaining our improved bounds as well. Namely we prove Lemma 11 for the case of $\mathcal{Q} =$ PCR, and by extension Nullsatz and PC.

The strategy is to prove a degree version of the wide clause lemma for $\tau_{\mathcal{S}}$ in the style of Lemma 12, and then the same random restriction argument as before will prove the size lower bound needed for Lemma 11. Recalling the definitions for I_0, J_0 in Lemma 12, for any monomial t let $I_0(t)$ be the set of all $i \in [n]$ for which at least $\log m$ variables from x_i appear in t, and let $J_0(t)$ be the set of all $j \in [m]$ for which at least $\log n$ variables from y_j appear in t. Recall that for PCR there exist distinct variables z and $\overline{z}$, both of which we consider to be variables from their respective x_i or y_j.

▶ **Lemma 21.** *If* $\gamma(\mathcal{S}) \geq k^2$*, then for any* PCR *refutation* π *refuting* $\tau_{\mathcal{S}}$*, there exists a monomial* $t \in p \in \pi$ *such that* $|I_0(t)| \geq k^2$ *or* $|J_0(t)| \geq k$.

Proof. Given a set $P = \{p_1, \ldots, p_m\}$ of polynomials over $F[x_1, \ldots, x_n]$, we denote by $span(P)$ the ideal generated by P – that is the set $\{\sum_i p_i f_i \mid f_i \in F[x_1, \ldots, x_n]\}$. A set of polynomials $f_1, \ldots, f_n$ semantically implies a polynomial g if any assignment that satisfies $f_i = 0$ for all $i \in [n]$ also satisfies $g = 0$. Note that $p \in span(P)$ if and only if P semantically implies p, which we write as $P \vdash p$.

Recall that $\mathcal{P}$ is our set of input clauses converted to polynomial form, and $\mathbf{A}_j$ is the set of clauses associated with column j. Accordingly let $\mathcal{P}_j$ denote the corresponding set of polynomials plus the equations $\{z^2 - z = 0\}$ for every variable z in τ. For a subset $J \subseteq [m]$ of columns, let $\mathcal{P}_J$ denote $\cup_{j \in J} \mathcal{P}_j$, and thus $span(\mathcal{P}_J)$ is the ideal generated by the polynomials $\mathcal{P}_J$.

We will prove our degree bound for PCR refutations of $\tau_{\mathcal{S}}$ by defining a linear operator K which maps polynomials p where $|I_0(t)| < k^2$ and $|J_0(t)| < k$ for all $t \in p$ to polynomials q, and satisfies the following conditions:

1. For all initial polynomials $p \in \cup_{j \in [m]} \mathcal{P}_j$, $K(p) = 0$.
2. K is linear: $K(ap + bq) = aK(p) + bK(q)$ for all constants a, b and polynomials p, q.
3. $K(xt) = K(xK(t))$ for all x
4. $K(1) \neq 0$

The existence of such an operator implies our degree bound as follows. Given an alleged PCR refutation which contains no monomial t where $I_0(t) \geq k^2$ or $J_0(t) \geq k$, applying K to every line in the proof, we have by the properties of K in conditions 1, 2, and 3 that $K(p) = 0$ for every polynomial in the proof. On the other hand since the final line is 1, by property 4 $K(1) \neq 0$, which is a contradiction.

We fix the grlex (graded lexicographical) ordering on all polynomials over $F[x_1, \ldots, x_n]$. Given a polynomial q and $J \subseteq [m]$, let $R_J(q)$ be the minimal (with respect to $<$) polynomial p such that $q - p \in span(\mathcal{P}_J)$. For every monomial t we set $K(t) = R_{J_0(t)}(t)$, and for $p = \sum_i c_i t_i$ set $K(p) = \sum_i c_i K(t_i)$. Intuitively $K(t)$ is how "close" t is to being in the span of the axioms in all columns with many variables in t. Note that this definition is asymmetric with respect to $\vec{x}$ and $\vec{y}$.

We now show the conditions of the linear operator are fulfilled. Consider any initial polynomial, p. If p is $z^2 - z$, then since $K[z^2] = K[z]$, $K[z^2 - z] = 0$ as required. Otherwise p is of the form $x_i^{\alpha_i} y_j^{\beta_i} = 0$. Note that p is a single monomial with $4 \log n$ variables of the form y_j and no variables of the form $y_{j'}$ for $j \neq j'$. So $J_0(p) = \{j\}$ and thus $R_{J_0(p)}(p) = 0$, which fulfills condition 1. By definition K is a linear operator, which fulfills 2. Because $J_0(1) = \emptyset$, $1 \notin span(\mathcal{P}_{J_0(1)})$, and so condition 4 is satisfied.

To prove condition 3, let us first prove the intuitive direction of the equality, namely that $K(xt) \geq K(xK(t))$. We repeatedly make use of the fact that if $J \subseteq J'$ then $R_J(t) \geq R_{J'}(t)$.

$$\begin{aligned}
K(xt) &= R_{J_0(xt)}(xt) \\
&= R_{J_0(xt)}(xR_{J_0(xt)}(t)) \\
&\geq R_{J_0(xt)}(xR_{J_0(t)}(t)) && (1) \\
&= R_{J_0(xt)}(xK(t)) \\
&\geq R_{J_0(xK(t))}(xK(t)) && (2) \\
&= K(xK(t))
\end{aligned}$$

In order to get equality, it is enough to show that (1) and (2) can be made equalities. For (2) note that if we expand $xK(t)$ as a polynomial and apply the linear operator $R_{J_0(xt)}$ to each term, we get that equality holds iff for all monomials t' in $xK(t)$,

$$R_{J_0(xt)}(t') = R_{J_0(xK(t))}(t').$$

We now observe that $J_0(xt) \subseteq J_0(t)$ and $J_0(t') \subseteq J_0(xt), J_0(xK(t))$. Therefore to finish the proof of condition 3 and thus the lemma, we prove the following claim:

▷ Claim 22. For all t where $|I_0(t)| < k^2$ and all $J \supseteq J_0(t)$ such that $|J| < k$, $R_{J_0(t)}(t) = R_J(t)$.

We need to show that $R_{J_0(t)}(t) \geq R_J(t)$ and $R_{J_0(t)}(t) \leq R_J(t)$. The first inequality holds trivially because $J_0(t) \subseteq J$, meaning that any $p \in span(\mathcal{P}_{J_0(t)})$ is also in $span(\mathcal{P}_J)$ as well. Now we prove the other direction, $R_{J_0(t)}(t) \leq R_J(t)$. If we can show that $t - R_J(t) \in span(\mathcal{P}_{J_0(t)})$, then since $R_{J_0(t)}$ is the smallest polynomial p for which $t - p \in span(\mathcal{P}_{J_0(t)})$, it follows that $R_J(t) \geq R_{J_0(t)}$ as desired. This is equivalent to showing that $\mathcal{P}_{J_0(t)} \vdash t - R_J(t)$ by definition of $span$, which is the statement we will now prove.

Assume for contradiction that there exists an assignment $\vec{\alpha}, \vec{\beta}$ that satisfies all axioms in $\mathcal{P}_{J_0(t)}$ but falsifies $t - R_J(t)$. We then prove that there exists an assignment $\vec{\alpha}', \vec{\beta}'$ that satisfies all axioms in $\mathcal{P}_J$ but doesn't touch any variables in $t - R_J(t)$, This means that $\mathcal{P}_J$ doesn't imply $t - R_J(t)$, which contradicts the fact that by definition of R, $t - R_J(t) \in span(\mathcal{P}_J)$. For this we note that the set of variables in t is a superset of the variables in $t - R_J(t)$, and thus $|I_0(t - R_J(t))| < k^2$ and $|J_0(t - R_J(t))| < k$ as per the claim. For simplicity we will refer to these sets as simply I_0 and J_0.

Consider a row $i \in [n] - I_0$. By the $(m, \log m/4)$-universal property of A there exists a string $a \in A$ which is zero in all positions $j \in J$. Since there are at most $\log m$ x_i variables in $t - R_J(t)$, we can leave α_i untouched on those variables and change the rest to give us α'_i, such that $f_x(\alpha'_i) = a$. We do this for all such i, noting that no variables in $t - R_J(t)$ have been changed.

Now consider a row $j \in J - J_0$. Let S_0 be a set that doesn't contain any $i \in I_0$, given to us by the fact that $I_0 < k^2 < \gamma(S)$. Since there are at most $\log n$ y_j variables in $t - R_J(t)$, we can leave β_j untouched on those variables and change the rest to give us β'_j, such that $f_y(\beta'_j) = S_0$. We do this for all such β'_j, noting again that no variables in $t - R_J(t)$ have been changed.

We now claim that $\vec{\alpha}', \vec{\beta}'$ satisfies all axioms in $\mathcal{P}_J$. Consider a row $j \in J_0$. Assume an axiom for row i and column j was violated. If $i \notin I_0$, we are guaranteed that $f_x(\alpha'_i)$ is 0 in the jth entry, so it must be that $i \in I_0$. But then we haven't changed α_i or β_j, and since the original assignment satisfied all axioms in $\mathcal{P}_{J_0}$ the axiom could not have been violated by $\vec{\alpha}', \vec{\beta}'$. Now consider a row $j \in J - J_0$. Assume an axiom for row i and column j was violated. Again if $i \notin I_0$, we are guaranteed that $f_x(\alpha'_i)$ is 0 in the jth entry, so it must be that $i \in I_0$. But then we changed β_j such that $f_y(\beta'_j)$ is 0 in the ith row, and so the axiom could not have been violated by $\vec{\alpha}', \vec{\beta}'$. ◀

Proof of Lemma 11. Our proof is identical as the restriction argument in Section 5. Assume for contradiction that there exists a PCR proof π refuting $\tau_{\mathcal{S}}$ in size less than $n^{k/16}$. We apply the same restriction from Definition 13 to the positive variables $\vec{x}, \vec{y}$ in every line. Then for the negative variables we set $\overline{z}$ to be $*$ if z is set to $*$ and $1 - z$ otherwise. The same analysis proves that there exists a restriction ρ which sets every monomial t with $|I_0(t)| \geq k^2$ or $|J_0(t)| \geq k$ to 0, and the remaining proof $\pi|_\rho$ is a refutation of $\tau_{\mathcal{S}}|_\rho$. Defining f^i_x and f^j_y as before, each f^i_x is still a $(5 \log m, \log m, \log m)$ code and each f^j_y is still a $(5 \log n, \log n, \log n)$ code, and so $\pi|_\rho$ still requires such a monomial by Lemma 21, which is a contradiction. ◀