Short Proofs Are Hard to Find

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

Toniann Pitassi
University of Toronto, Canada
Institute for Advanced Study, Princeton, NJ, USA
toni@cs.toronto.edu

Yuanhao Wei
Carnegie Mellon University, Pittsburgh, PA, 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 $\mathsf{Res}(r)$.

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.ICALP.2019.84

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://www.cs.toronto.edu/~mertz/papers/mpw19.short_proofs_are_hard_to_find.pdf`.

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.

Acknowledgements The authors thank Noah Fleming, Pravesh Kothari, Denis Pankratov, Robert Robere, Mika Göös, 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].

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 $\mathsf{poly}(n, |\tau|, S)$ 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 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 [17, 16]. 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 $\mathsf{FPT} \neq \mathsf{W[P]}$,[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 $\mathsf{FPT} \neq \mathsf{W[P]}$ 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

[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].

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 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 to the full version of the paper.

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 automatizability and require cryptographic assumptions. Krajíček and Pudlák showed non-automatizability of the Extended Frege system under the hardness of

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.

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 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 holds for PC (due to the Groebner basis paper of Clegg, Impagliazzo, and Edmonds [15]), SA [40], and SoS [34, 26], 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.

[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.

[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].

1.2.1 Recent Developments

In a recent breakthrough paper, Atserias and Müller [4] resolve the automatizability of (general) Resolution. 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^δ} automatizable for any $\delta < \frac{1}{2}$. Because of the quasipolynomial upper bounds on automatizability for the other systems that we study here (tree-like Resolution, and the Polynomial Calculus) our results as well as [2] are incomparable with [4]. 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 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 $P_i = (1 - x_1)(x_2)(1 - x_3) = 0$.

[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.

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 $\mathcal{P} = \{P_1, \dots, P_m\}$. 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 $Q_1, \dots, Q_m$ 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 $Q_1, Q_2, \dots, Q_S$. 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 proofs with respect to size. For PCR, polynomials are allowed to use additional variables $\overline{x}_1, \dots, \overline{x}_n$ 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 $\mathcal{S} = \{S_1, \dots, S_n\}$ 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 the full version of the paper[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 $n^{o(k)}$.*

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

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 $A = \{a_1, \dots, a_m\}$, 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.

▶ **Lemma 5.** *Let $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, 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 $\gamma(\mathcal{S}) \le k$ then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \le n^{O(1)}$;*

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

Proof of Theorem 1. Assuming that $\mathcal{Q}$ is n^f automatizable for some $f(n) = o(\log^{1/7-\epsilon} \log n)$ 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 "$\gamma(\mathcal{S}) \le k$" 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 the full paper. 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

[8] Many examples of universal sets (including the Paley graph construction) are discussed in [24], as well as [31, 3]. 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 constructible, $(m, \log m/4)$-universal, and $(m, \log m/4)$-dual-universal.

improved, a stronger version of Theorem 1 would immediately follow. Likewise, starting from the GapETH assumption we could use a stronger version of Lemma 4 and immediately get the stronger result claimed in Section 1 (see the full version of the paper).

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 $\vec{x} = x_1, \dots, x_n$, $\vec{y} = y_1, \dots, y_m$ 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 matrices 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 matrices 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 $\gamma(\mathcal{S}) \le \frac{\log m}{4}$.

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 indices 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.[9] 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, $\vec{\alpha}$

[9] We do not require that the rows of M are *distinct* rows of A, but because we are only looking for a location with a 1 for every row this does not pose an issue. In fact we only ever use the universal and dual universal properties to search for a location with either all 0 or all 1, where repetition doesn't break the universal properties we need.

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 for each $i \in [n]$ and $j \in [m]$, each of the mn possible assignments to (x_i, y_j) adds at most one clause to $\psi_{\mathcal{S}}$.

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.** *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 $m = n^{1/k} \ll n$. The number of clauses in $\tau_{\mathcal{S}}$ is at most $n^7 m^7$, since for each $i \in [n]$ and $j \in [m]$, each of the $m^6 n^6$ possible assignments to (x_i, y_j) adds at most one clause to $\tau_{\mathcal{S}}$.

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 labelled 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 labelled 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 [36, 5] 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 [36, 5] 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 proceeding 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 $(1 - (\frac{1}{6} \cdot \frac{1}{2}))^{\log m}$. Thus the probability that no $i \in I_0(D)$ sets D to 1 is at most $(\frac{11}{12})^{k^2 \log m} = (\frac{11}{12})^{k \log n} < \frac{1}{n^{k/8}}$. By a union bound the probability that some clause D in π satisfying $|I_0(D)| \geq k^2$ is not set to 1 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 $|J_0(D)| \geq k$ is not set to 1 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 $\vec{\rho} = \{\rho_{x_1} \ldots \rho_{x_n}, \rho_{y_1} \ldots \rho_{y_m}\}$ 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 [14, 13] 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.[10]

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.[11] Thus, the upper limit of improving the reductions of [14, 13] 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

[10] 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.

[11] 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 $k = o(\log^{1/2} n)$.

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 [22, 21]; in particular a constant-sized lifting gadget would immediately give results for TreeCP matching our other results.

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. *CoRR*, abs/1904.02991, 2019. `arXiv:1904.02991`.

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. `arXiv:1708.04218`.

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. `arXiv:1807.11419`.

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. `arXiv:1606.02577`.

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.