Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 16 (2023)

Proving Unsatisfiability with Hitting Formulas

Yuval Filmus[1], Edward A. Hirsch[2], Artur Riazanov[3], Alexander Smal[4], and Marc Vinyals[5]

[1]Technion — Israel Institute of Technology, Israel, yuvalfi@cs.technion.ac.il
[2]Ariel University, Israel, edwardh@ariel.ac.il
[3]EPFL, Switzerland, tunyash@gmail.com
[4]Technion — Israel Institute of Technology, Israel, avsmal@gmail.com
[5]University of Auckland, New Zealand, marc.vinyals@auckland.ac.nz

August 14, 2024

Abstract

A hitting formula is a set of Boolean clauses such that any two of the clauses cannot be simultaneously falsified. Hitting formulas have been studied in many different contexts at least since [Iwa89] and, based on experimental evidence, Peitl and Szeider [PS22] conjectured that unsatisfiable hitting formulas are among the hardest for resolution. Using the fact that hitting formulas are easy to check for satisfiability we make them the foundation of a new static proof system Hitting: a refutation of a CNF in Hitting is an unsatisfiable hitting formula such that each of its clauses is a weakening of a clause of the refuted CNF. Comparing this system to resolution and other proof systems is equivalent to studying the hardness of hitting formulas.

Our first result is that Hitting is quasi-polynomially simulated by tree-like resolution, which means that hitting formulas cannot be exponentially hard for resolution and partially refutes the conjecture of Peitl and Szeider. We show that tree-like resolution and Hitting are quasi-polynomially separated, while for resolution, this question remains open. For a system that is only quasi-polynomially stronger than tree-like resolution, Hitting is surprisingly difficult to *polynomially* simulate in another proof system. Using the ideas of Raz–Shpilka's polynomial identity testing for noncommutative circuits [RS05] we show that Hitting is p-simulated by Extended Frege, but we conjecture that much more efficient simulations exist. As a byproduct, we show that a number of static (semi)algebraic systems are verifiable in deterministic polynomial time.

We consider multiple extensions of Hitting, and in particular a proof system Hitting($\oplus$) related to the Res($\oplus$) proof system for which no superpolynomial-size lower bounds are known. Hitting($\oplus$) p-simulates the tree-like version of Res($\oplus$) and is at least quasi-polynomially stronger. We show that formulas expressing the non-existence of perfect matchings in the graphs $K_{n,n+2}$ are exponentially hard for Hitting($\oplus$) via a reduction to the partition bound for communication complexity.

ISSN 1433-8092

Contents

1 Introduction

Propositional proof complexity is a well-established area with a number of mathematically rich results. A propositional proof system [CR79] is formally a deterministic polynomial-time algorithm that verifies candidate proofs of unsatisfiability of propositional formulas in conjunctive normal form. The existence of a proof system that has such polynomial-size refutations for all unsatisfiable formulas is equivalent to **NP** = **co-NP**, and (dis)proving it is out of reach of the currently available methods. Towards this goal, Cook and Reckhow's paper [CR79] started a program to develop new stronger proof systems that have short proofs for tautologies that are hard for known proof systems and to prove superpolynomial lower bounds for these new systems. The idea is that obtaining new results where our previous techniques fail helps in developing new techniques.

One of the oldest propositional proof systems is the propositional version of resolution (Res) [Bla37, DP60] that operates on Boolean clauses (disjunctions of literals treated as sets) and has only a single rule that allows introducing resolvents $\frac{\ell_1 \vee \cdots \vee \ell_k \vee x \qquad \ell'_1 \vee \cdots \vee \ell'_m \vee \overline{x}}{\ell_1 \vee \cdots \vee \ell_k \vee \ell'_1 \vee \cdots \vee \ell'_m}$. Superpolynomial lower bounds on the size of a particular case of resolution proofs are known since [Tse68], while exponential lower bounds on general Res proof size were proven by Haken [Hak85] and Urquhart [Urq87] for the pigeonhole principle and handshaking lemma, respectively. Furthermore Res encompasses CDCL algorithms for SAT [BKS04, PD11], that are the most successful SAT-solving algorithms to date.

Motivated by the quest of finding hard examples for modern SAT-solvers, Peitl and Szeider [PS22] experimentally investigated the hardness of *hitting formulas* for resolution. A hitting formula as a mathematical object has been studied under a number of names and in various contexts (a polynomial-time solvable SAT subclass, partitions of the Boolean cube viewed combinatorially, etc.) [Iwa89, DD98, Kul04, Kul11, GK13, Gwy14, KZ13, PS22, FHK+23]. A formula $H = \bigwedge_i H_i$ in CNF with clauses H_i is a hitting formula if every pair of clauses cannot be falsified simultaneously (that is, there is a variable that appears in the two clauses with different signs). Equivalently, the sets S_i of truth assignments falsifying clauses H_i are disjoint, thus in an unsatisfiable hitting formula every assignment in $\{0,1\}^n$ is covered exactly once by S_i's. Peitl and Szeider conjectured that hitting formulas might be among the hardest formulas for resolution. Their conjecture was supported by experimental results for formulas with a small number of variables.

One of the reasons why hitting formulas received an abundance of attention is that they are one of the classes of CNFs that are polynomial-time tractable for satisfiability checking (along with e.g. Horn formulas and 2-CNFs) [Iwa89]. First, it is straightforward to check whether a CNF formula is hitting: simply enumerate all pairs of clauses and check that they contain some variable with opposite signs. Then the number of satisfying assignments of a hitting formula is $2^n - \sum_i 2^{n-|H_i|}$, where $|H_i|$ is the number of literals in H_i and n is the number of variables in H.

This nice property allows us to think about hitting not only as a class of formulas but as an algorithm to determine satisfiability. For the algorithm to apply to any kind of formulas we need to introduce nondeterminism, and this is best modelled with a proof system. Thus we define a new static proof system based on unsatisfiable hitting formulas. A refutation of an arbitrary CNF F in the Hitting proof system is an unsatisfiable hitting formula such that each of its clauses is a weakening of a clause in F (i.e. a clause of F with extra literals).

By thinking of hitting as a proof system we reinterpret the conjecture of Peitl and Szeider as the following question: is it possible to efficiently formalize the model counting argument above within the Res proof system? Then the question of the hardness of hitting formulas for Res can be phrased in terms of the relative strength of Res and Hitting: can Hitting be separated from

Res? That is, can we find formulas that are easy to refute in Hitting and hard to refute in Res? More in general, by relating Hitting to other proof systems we can pinpoint both the hardness of hitting formulas and the ability to formalize Iwama's counting argument in those proof systems.

It turns out that Hitting is tightly connected to the tree-like version of Res (tl-Res), which is exponentially weaker than Res [BSIW04]. It encompasses all DPLL algorithms [DP60, DLL62], which form the base of multiple (exponential-time) upper bounds for SAT (see, e.g., [DH21] for a survey). A DPLL algorithm splits the input problem F into subproblems $F|_{x=0}$ and $F|_{x=1}$ for some variable x and applies easy simplification rules.

More precisely, tl-Res quasi-polynomially simulates Hitting (Theorem 4.2), and the simulation cannot be improved to a polynomial one (Theorem 4.14). This partially answers the question "How hard can hitting formulas be for resolution?" raised in [PS22] in the following way. Not only every hitting formula has proofs of quasi-polynomial size, their unsatisfiability can be decided in quasi-polynomial time by a DPLL algorithm. The simulation also entails that every exponential-size lower bound we already have for tl-Res holds for Hitting, which in particular allows for a separation of Res from Hitting.

Even though the very weak proof system tl-Res is enough to quasi-polynomially simulate Hitting, it is surprisingly difficult to push the upper bound all the way to a polynomial: even though we compare Hitting to a number of known proof systems with different strengths with the hope of obtaining a polynomial simulation, the only system where we can polynomially simulate Hitting is the very powerful Extended Frege (Corollary 3.5). As a byproduct of this result, we prove also that various static (semi)algebraic proof systems (Nullstellensatz, Sherali–Adams, Lovász–Schrijver, Sum-of-Squares) are indeed Cook–Reckhow (deterministically polynomial-time verifiable) proof systems even when we measure the proof size in a succinct way, ignoring the part enforcing Boolean variables. Such distinction can be safely ignored in lower bound results, but in principle ought to be accounted for when constructing upper bounds. Efficient deterministic formal proof verification becomes more and more important because of the increasing interest in algorithms based on sum-of-squares [BS14, FKP19].

In more detail, we study the relation between various versions of Hitting and known proof systems such as:

- Res($\oplus$), defined in [IS20] by analogy with the system Res(Lin) of [RT08] in the same vein as Krajíček's R(...) systems [Kra98]. No superpolynomial-size lower bound is known for it, however, [IS20] proves an exponential bound for its tree-like version. Res($\oplus$) extends Res by allowing clauses to contain affine equations modulo two instead of just literals, and this is the weakest known bounded-depth Frege system with parity gates where we do not know a superpolynomial-size lower bound.

  We prove two separations showing that Hitting is incomparable with tl-Res($\oplus$) (Sect. 4.3, the separation is quasi-polynomial in one direction and exponential in the other direction).

- Nullstellensatz (NS), defined in [BIK+96] (where also an exponential-size lower bound was proved), along with its version NSR [dRLNS21] that uses dual variables ($\overline{x} = 1 - x$ introduced in [ABSRW02] for PC [CEI96], which is a "dynamic" version of Hilbert's Nullstellensatz that allows generating elements of the ideal step-by-step). An exponential-size lower bound for NSR follows from [BCIP02] (see Corollary 5.4).

- Cutting Planes (CP), defined in [CCT87], uses linear inequalities as its proof lines and has

two rules: the rule introducing nonnegative linear combinations and the integer rounding rule ($\frac{\sum c_i x_i \geq c}{\sum c_i x_i \geq \lceil c \rceil}$ for integer c_i's).

- Frege, defined in [Rec76, CR79], can be thought of as any implicationally complete "textbook" derivation system for propositional logic. Proving superpolynomial lower bounds for it is a long-standing open problem that seems out of reach at the moment.

- Systems augmented by Tseitin's extension rule and its analogues, such as Extended Frege. This rule allows the introduction of new variables denoting some functions of already introduced variables.

Given that known proof systems do not obviously polynomially simulate Hitting, this leaves us with the following question: does augmenting SAT algorithms with the ability to reason about hitting formulas lead to any improvements? Or its counterpart about proof systems, how powerful are proof systems resulting from combining known proof systems with Hitting?

Recall that a DPLL algorithm splits the input problem F into subproblems $F|_{x=0}$ and $F|_{x=1}$. Algorithms that give upper bounds for SAT use more general splittings; in fact one can split over any tautology, that is, consider subproblems $F \wedge G_1, \ldots, F \wedge G_m$, where $G_1 \vee \cdots \vee G_k$ is a tautology. Put in another way, one can split over an unsatisfiable formula $\overline{G_1} \wedge \cdots \wedge \overline{G_k}$ — including unsatisfiable hitting formulas. We use this idea, although in a DAG-like context, to introduce the following generalization of Hitting.

Hitting Res merges Hitting with Res. It uses the weakening rule and also extends the main resolution rule to

$$\frac{C_1 \vee H_1, \ \ldots, \ C_k \vee H_k}{C_1 \vee \cdots \vee C_k}$$

for a hitting formula $H_1 \wedge \cdots \wedge H_k$. It is also p-simulated by Extended Frege (Corollary 3.6).

Other ways in which we can generalize Hitting while keeping with the spirit of the proof system are to allow some leeway in the requirement for the subcubes to form a partition, or in the type of objects that constitute the partition. While at first these may appear to be a mere mathematical curiosity, the connections to Nullstellensatz in the case of Odd Hitting and to the partition bound in the case of Hitting($\oplus$) show that these are natural proof systems.

Hitting[k] strengthens Hitting by allowing to cover a falsifying assignment with at most k sets. Such proofs can be efficiently verified and p-simulated in Extended Frege using the inclusion-exclusion formula and polynomial identity testing (PIT) (Theorem 3.10).

Odd Hitting strengthens Hitting by allowing to cover a falsifying assignment with an odd number of sets. Such proofs also can be efficiently verified and p-simulated in Extended Frege using PIT (Prop. 3.8). This system is equivalent to a certain version of Nullstellensatz, which we discuss in Sect. 5. We prove a lower bound for Odd Hitting (Corollary 5.4) that allows us to separate it from Res.

Hitting($\oplus$) strengthens Hitting by allowing the complements of affine subspaces instead of clauses, that is, a clause can now contain affine equations instead of just literals, and S_i is thus an affine subspace. Such proofs can be verified similarly to Hitting using the Gaussian elimination. We prove an exponential-size lower bound for Hitting($\oplus$) (Theorem 6.9) which, additionally, separates it from CP.

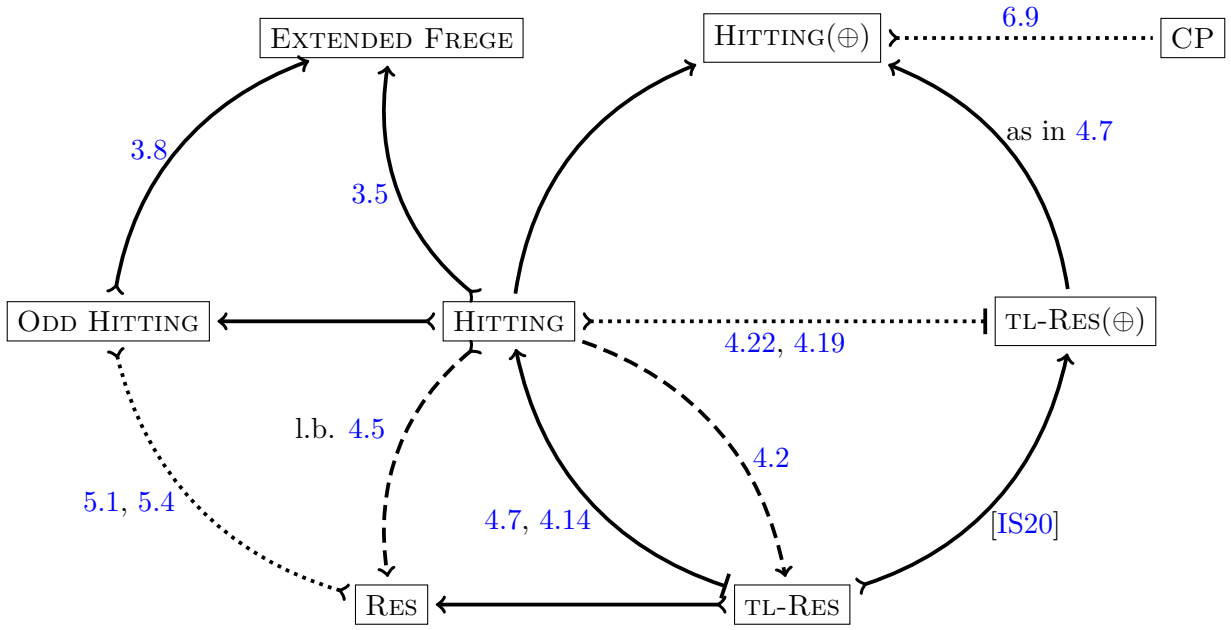

Figure 1: Arrow A ⟶ B means that B p-simulates A, a dashed arrow A ⇢ B means B quasi-polynomially simulates A. A ⟊ B means a quasi-polynomial separation (a lower bound is for the system A). An arrowhead in the tail A ⤍ B means that A is exponentially separated from B. A dotted line A ⋯ B means that we do not know any simulations between A and B. Known simulations involving CP and Extended Frege are not shown.

A summary of our simulations and separations is depicted in Figure 1, and more precise bounds are stated in Table 1. Now we turn to a more detailed discussion.

Statement	Hitting	Hitting(⊕)	Odd Hitting	tl-Res	tl-Res(⊕)	Res	CP
4.14, 4.31	$2^{\tilde{O}(m)}$			$2^{\tilde{\Omega}(m^{2-\varepsilon})}$		$2^{\tilde{O}(m)}$	
4.22	$2^{\tilde{O}(m)}$				$2^{\tilde{\Omega}(m^{2-\varepsilon})}$		
5.4			$2^{\tilde{\Omega}(n)}$			poly	
4.19, [IS20]	$2^{n^{\Omega(1)}}$				poly	$2^{\Omega(n)}$	
6.9		$2^{\Omega(n)}$					poly

Table 1: Precise bounds in our separations. Upper bounds are black and lower bounds are purple.

1.1 Our results and methods

1.1.1 Simulations of Hitting-based systems and proof verification using PIT

Proof verification is not straightforward in static (semi)algebraic proof systems that use either dual variables $\bar{x} = 1 - x$ or do not open the parentheses in $(1 - x)$ for the negation of a variable x (such as static Lovász–Schrijver or Sherali–Adams proofs or NS proofs with dual variables). A similar situation occurs with the verification of Hitting proofs which, contrary to most (or all?)

known proof systems, is based on model counting. Such reasoning is not expressed naturally in propositional logic, and it makes it difficult to simulate Hitting proofs in other proof systems. We observe that Hitting proofs can be expressed similarly to NS proofs with dual variables without explicitly mentioning the side polynomials for $x^2 - x$ and $x + \bar{x} - 1$ (in particular, we notice that over GF(2), such proofs, which we call *succinct* NSR proofs, are equivalent to Odd Hitting proofs, and that over any field they p-simulate Hitting proofs in a straightforward manner). We show that the two problems have the same cure: we provide an efficient polynomial identity testing procedure for multilinear polynomials modulo $x + \bar{x} - 1$ that can also be formalized in Extended Frege.

Our approach uses the main idea of the Raz–Shpilka polynomial identity testing for noncommutative circuits [RS05]. We introduce new variables for quadratic polynomials; crucially it suffices to do so for a basis instead of the potentially exponential number of polynomials. This serves as an inductive step cutting the degrees. Namely, at the first step we consider two variables x_1 and x_2 and quadratic polynomials (potentially, $(1 - x_1)(1 - x_2)$, $(1 - x_1)x_2$, $x_1(1 - x_2)$, x_1x_2, $1 - x_1$, x_1, $1 - x_2$, and x_2) appearing in the monomials m_i as $\bar{x}_1\bar{x}_2$, $\bar{x}_1x_2$, etc., and replace them using linear combinations of new variables $y_i^{1,2}$, thus decreasing the degree by one. At the next step we treat all the variables $y_i^{1,2}$ as a single "layer" (note that they are *not* multiplied by each other). We merge this layer of $y_i^{1,2}$ with x_3, getting a layer of variables $y_j^{1,2,3}$, and so on, until we reach a linear equation, which is easy to verify.

In order to implement this strategy we prove a lemma allowing us to merge two layers of variables (Lemma 3.7) by ensuring that after the merge the equivalence of polynomials still holds.

By using this polynomial identity testing we get not only an efficient algorithm for checking static proofs (including succinct NSR proofs), but also a polynomial simulation in the Extended Polynomial Calculus (Ext-PC) system that has been recently used in [Ale21], where an exponential-size lower bound has been proved. Given that Ext-PC over GF(2) is equivalent to Extended Frege (Prop. 3.3), we obtain p-simulations of Hitting, Hitting Res, Odd Hitting and Hitting$[k]$ in Extended Frege.

A simpler proof that succinct SA is verifiable in polynomial time was developed independently in [dRPR24], where the proof system is named *semantic* SA. Interestingly, after applying algebraic manipulations, their proof implicitly reduces the problem of verifying a SA proof to that of verifying a Hitting proof. This suggests that proofs systems relying on model counting are not that uncommon after all.

1.1.2 Separations of Hitting from classical systems

A polynomial simulation of tl-Res in Hitting (Theorem 4.7) can be easily shown by converting tl-Res to a decision tree, then the assignments in the leaves provide a disjoint partition of the Boolean cube. We show a quasi-polynomial simulation in the other direction through careful analysis of a recursive argument (Theorem 4.2). The main idea is that an unsatisfiable formula containing m clauses must necessarily contain a clause of width $w \leq \log_2 m$, and in a hitting formula this clause must contain a variable that occurs with the opposite sign in at least $(m - 1)/w$ clauses. Making a decision over this variable thus removes a lot of clauses in one of the two branches. We also employ a generalization of this idea to show that Hitting$[k]$ proofs can be quasi-polynomially simulated in Hitting (Prop. 4.4) and hence in tl-Res.

A polynomial simulation in the other direction is impossible because of a superpolynomial separation. To show this result (Theorem 4.14) we use query complexity, and in particular, the

result of [AKK16] separating unambiguous query complexity from randomized query complexity. We lift it using xorification to obtain the desired separation.

We then obtain a two-way separation between Hitting and tl-Res($\oplus$) (Sect. 4.4). On the one hand Tseitin formulas are hard for Res [Urq87] and hence for Hitting. On the other hand, [IS20] shows that they have polynomial-size tl-Res($\oplus$) (and thus also Hitting($\oplus$)) proofs. In the other direction, similarly to the separation between Hitting and tl-Res, we again use the separation of [AKK16] between unambiguous certificate complexity and randomized query complexity as our starting point. However, since for tl-Res($\oplus$) we are unable to use decision trees, we need to go through randomized communication complexity arguments, using a randomized query-to-communication lifting theorem [GPW17].

Eventually, we discuss separations of Hitting from Res and NS. While the relevant lower bounds for Hitting follow directly from known lower bounds for tl-Res, the other direction seems much more difficult, if possible at all. One natural candidate for a separation result could be the formulas that we used to separate Hitting from tl-Res, but this cannot work because they turn out to have Res proofs of polynomial size (Theorem 4.31). We show this fact using dag-like query complexity [GGKS20], the analogue of resolution width in query complexity, which stems from a game characterization of Res [Pud00, AD08]. We need to reprove the result of [AKK16] accordingly, improving it to a separation between unambiguous dag-like query complexity and randomized query complexity. This immediately yields an upper bound on the Res width. Concerning NS, it is a simple observation that Hitting is simulated by NS with respect to width vs degree. Furthermore, as we discussed above, succinct NSR proofs (over any field) simulate Hitting with respect to size, therefore separating Hitting from Res would amount to separating explicit vs succinct NSR size.

1.1.3 A lower bound for Odd Hitting

As mentioned above Odd Hitting is polynomially equivalent to succinct NSR proofs over GF(2), and we explain this in more detail in the beginning of Sect. 5. It is easy to see that Odd Hitting has short proofs of Tseitin formulas and thus it is exponentially separated from Res. The opposite direction (Cor. 5.4) requires slightly more effort. It is known that Res width can be separated from NS degree [BCIP02]. We use this result to get our size separation using xorification and the random restriction technique of Aleknhovich and Razborov (see [BS09]).

1.1.4 A lower bound for Hitting($\oplus$)

Our lower bound for Hitting($\oplus$) (Theorem 6.9) uses a communication complexity argument. Communication complexity reductions have a long history of applications in proof complexity [BPS05, HN12, GP18, IS20, dRNV16]. The first step in these reductions is a simulation theorem, which shows that a refutation of an arbitrary CNF ϕ in the proof system of interest can be used to obtain a low-cost communication protocol solving the communication problem Search(ϕ): given an assignment to the variables of ϕ, find a clause of ϕ falsified by this assignment. The second step is reducing a known hard communication problem (usually set disjointness) to Search(ϕ) for a carefully chosen CNF ϕ.

Until recently the applications of these reductions were limited to either proving a lower bound for a tree-like version of the system or proving a size-space tradeoff, neither of which applies to our result. However, over the last few years, the list of applications of the communication approach in proof complexity has grown significantly. A major breakthrough came in [Sok17, GGKS20] with a

dag-like lifting theorem from resolution to monotone circuits and cutting plane refutations. Another novel idea was introduced in [GHJ+22], where the authors derived a lower bound for Nullstellensatz via a communication-like reduction from the $\Omega(\sqrt{n})$ lower bound on the approximate polynomial degree of AND_n [NS95].

We use yet another twist on this idea: we apply a communication reduction to the *partition bound* [JK10], a generalization of randomized communication protocols which simulates Hitting($\oplus$) (Lemma 6.6). To the best of our knowledge this is the first application of the partition bound in a proof complexity context. We then adapt (Theorem 6.7) a communication reduction from set disjointness in [IR21] so that it works for the partition bound and use the fact that set disjointness is still hard for the partition bound to get our lower bound (Theorem 6.9). The choice of the reduction of [IR21] is not particularly important, and we believe that reductions from [BPS05, GP18, IS20] should also work. A nice feature of the reduction we use is that we get a lower bound for a natural combinatorial principle: a formula encoding the non-existence of a perfect matching in a complete bipartite graph $K_{n,n+2}$. Because this formula is known to have short CP proofs, we obtain a separation between Hitting($\oplus$) and CP as an immediate corollary.

1.2 Further research

Relation between Hitting and Res. Although we have gained a lot of understanding of the hardness of hitting formulas for resolution, the initial question of Peitl and Szeider is not fully answered. In particular, we do not know whether hitting formulas can be superpolynomially hard for Res. The negative answer implies a simulation of Hitting by Res. To show the positive answer it is sufficient to separate two query complexity measures: dag-like query complexity (w) and unambiguous certificate complexity (UC). The dag-like query complexity of the falsified clause search problem for a formula F corresponds to the resolution width of F. The unambiguous certificate complexity for this problem corresponds to the width of Hitting refutations of F. Note that unambiguous certificate complexity only makes sense for functions, while dag-like query complexity is defined for (total) relations. Unfortunately, separating even regular certificate complexity (C) and w is an open problem for *functions* (without the uniqueness requirement the certificate complexity can only decrease, so it might be easier to separate w from C than from UC). Lemma 4.23 and Lemma 4.29 show that w is resistant to known lower bound techniques in the field of query complexity, so tackling it will likely lead to finding new techniques there. Notice that we know how to separate w and C for *relations*: every lower bound on the resolution width for an $O(1)$-CNF formula constitutes a separation for the corresponding falsified clause search problem. Such a separation (constant vs. polynomial) is unachievable for functions (we cannot hope for better than quadratic separation for functions as $\mathsf{w}(R) \leq \mathsf{C}(R)^2$). Can we use ideas from resolution lower bounds to separate w and UC (or at least C)?

Separate Hitting and Hitting[2]. With xorification like in Lemma 4.9 this problem can be shown to be equivalent to a simple (if only in the statement!) question in query complexity: separate unambiguous certificate complexity and 2-unambiguous certificate complexity (where every input has one or two certificates). It is known how to separate one-sided versions of these query models [GKY22], but similarly to the question of Hitting vs Res it is unclear how to extend this to the two-sided case.

Is it possible to separate Hitting($\oplus$) **and** tl-Res($\oplus$)**?** In Section 6.1 we give evidence that a simulation of Hitting($\oplus$) by tl-Res($\oplus$) along the lines of Theorem 4.2 is not possible. That, however, does not rule out the existence of such a simulation. [She21, Conjecture 5.1.3] conjectures that every affine subspace partition can be refined to one corresponding to a parity decision tree with a quasi-polynomial blow-up. With some caveats[1], the statement of this conjecture is equivalent to the existence of quasi-polynomial simulation of Hitting($\oplus$) by tl-Res($\oplus$). So, is there an exponential separation between these two systems? It seems that communication-based lower bounds for tl-Res($\oplus$) can be transferred to Hitting($\oplus$) as it is done in Section 6. There are several other techniques that yield tl-Res($\oplus$) lower bounds such as prover-delayer games [IS20, Gry19], reduction to polynomial calculus degree [GK18], and the recent lifting from decision tree depth to parity decision tree size directly [CMSS22, BK22]. None of those seem to work for Hitting($\oplus$), so it is reasonable to think that some of the yielded formulas may have an upper bound in Hitting($\oplus$). The most promising technique seems to be the lifting since it yields a wide family of formulas hard for tl-Res($\oplus$) with the source of hardness inherent to the tree-like structure of refutations.

Better upper bound on Hitting**.** One intriguing matter is that although a very weak proof system such as tl-Res is enough to quasi-polynomially simulate Hitting, we need to go all the way to the very strong proof system Extended Frege for the simulation to become polynomial. A natural question is then what is the weakest proof system that is enough to polynomially simulate Hitting.

It is consistent with our findings that a fairly weak proof system such as NSR is already enough to simulate Hitting; indeed this would be the case if NSR and succinct NSR were equivalent. Hence we ask the same question regarding succinct (semi)algebraic proof systems: what is the weakest proof system that polynomially simulates succinct NSR or succinct SA? And in particular, is succinct NSR equivalent to NSR and is succinct SA equivalent to SA? One way to answer all these questions would be to formalize the PIT of Theorem 3.4 in a weaker proof system.

The situation with Hitting($\oplus$) is even worse. We have shown how to p-simulate most of the generalizations of Hitting that we defined, including Odd Hitting and Hitting[k], in Ext-PC, but the argument does not work as is for Hitting($\oplus$) since we are relying on a noncommutative PIT. Therefore we do not know even an Extended Frege simulation of Hitting($\oplus$) (though it is of course quite expected).

Non-automatability of Hitting**.** It follows from Theorem 4.2 and quasi-polynomial automatability of tl-Res [BP96] that Hitting is also quasi-polynomially automatable. Can we show that it is impossible to do better? We think that it is possible to adapt the similar result of de Rezende [dR21] for tl-Res.

[1]The refinement might be non-constructive, but its mere existence does not imply the simulation. The simulation might produce parity decision trees that are not refinements of the initial Hitting($\oplus$) refutation but nevertheless, solve the relation Search(ϕ).

2 Basic definitions

2.1 Basic notation

For a function $f : \mathbb{N} \to \mathbb{R}$, $\tilde{O}(f)$ and $\tilde{\Omega}(f)$ denote O and Ω up to logarithmic factors, that is, $g = \tilde{O}(f)$ and $h = \tilde{\Omega}(f)$ if $g = O(f \log^C f)$ and $h = O(f/\log^C f)$ respectively for a constant C. For example, $2^n n^2 = \tilde{O}(2^n)$ and $n/\log n = \tilde{\Omega}(n)$.

Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. The *deterministic query complexity of* f, denoted by $\mathsf{D}(f)$, is the minimal number of (adaptive) queries to the input variables that is enough to compute $f(x)$ for any input x. The *randomized query complexity of* f, $\mathsf{R}(f)$, is the minimum number of queries needed by a randomized algorithm that outputs $f(x)$ for any input x with probability at least $2/3$. A partial assignment $\alpha \in \{0,1,*\}^n$ is a *certificate* for f if for any two assignments $x, y \in \{0,1\}^n$ agreeing with α, $f(x) = f(y)$. The *size* of a certificate is the number of non-star entries. The *certificate complexity of* f *on an input* x, denoted $\mathsf{C}(f,x)$, is size of the smallest certificate α such that x agrees with α. For $b \in \{0,1\}$, the *(one-sided) b-certificate complexity of* f is defined as $\mathsf{C}_b(f) = \max_{x:f(x)=b} \mathsf{C}(f,x)$. The *(two-sided) certificate complexity of* f is the maximum of 0- and 1-certificate complexities, $\mathsf{C}(f) = \max\{\mathsf{C}_0(f), \mathsf{C}_1(f)\}$. We say that a family of certificates $A \subset \{0,1,*\}^n$ is *unambiguous* if any two distinct certificates $\alpha, \beta \in A$ conflict, i.e., there is no assignment that agrees with both α and β. For $b \in \{0,1\}$, the *(one-sided) unambiguous b-certificate complexity of* f, $\mathsf{UC}_b(f)$, is the minimum number w such that there is an unambiguous family of certificates A such that A contains only certificates of size at most w and every $x \in f^{-1}(b)$ agrees with some certificate in A. The *(two-sided) unambiguous certificate complexity of* f is defined as $\mathsf{UC}(f) = \max\{\mathsf{UC}_0(f), \mathsf{UC}_1(f)\}$.

For the definition of the basic proof complexity notions such as proof system and p-simulation, we refer the reader to [CR79]. We consider also quasi-polynomial simulations: a proof system Π_1 *quasi-polynomially simulates* proof system Π_2 if for certain $k \in \mathbb{N}$, for every formula F, the system Π_1 has a proof of F of size at most $2^{(\log s)^k}$, where s is the size of the shortest proof of F in Π_2. One could define (and name) a constructive (analogous to p-simulation) version of this notion, and in fact our quasi-polynomial simulations (Theorem 4.2, Proposition 4.4) are constructive, that is, we can produce the proofs in the simulating proof system in time polynomial in their length. This is not important for the separation results, so we use the term "quasi-polynomial simulation" without emphasizing the constructiveness. We say that a proof system Π_1 is *quasi-polynomially separated* from Π_2 if there is an infinite sequence of formulas F_n, whose size tends to infinity, and a specific $k \in \mathbb{N}$ such that Π_2 has no proofs of F_n of size $2^{(\log(s_n+|F_n|))^k}$, where s_n is the size of the shortest proof of F_n in Π_2. Somewhat abusing the notation we say that Π_1 is *quasi-polynomially stronger* than Π_2 if it polynomially simulates Π_2 and is quasi-polynomially separated from it.

We use the following notation for widely known proof systems: Res for Resolution, tl-Res for tree-like Resolution, Res($\oplus$) for Resolution over XORs of [IS20], tl-Res($\oplus$) for its tree-like version, CP for Cutting Planes, NS for Nullstellensatz, PC for Polynomial Calculus, Frege for Frege and Extended Frege for Extended Frege.

Deterministic communication complexity of a search problem defined by a ternary relation R is the minimal amount of communication (number of bits) that is enough to solve the following communication problem for two players on any input: Alice is given x, Bob is given y, and their goal is to find some z such that $(x,y,z) \in R$. Alice and Bob can exchange information by sending bit messages to each other. At the end of the game both players must know z. *(Public coin) ε-error randomized communication complexity* of a search problem is the minimal amount of communication

that is enough for players to win the communication game with probability at least $1 - \varepsilon$ if the players have access to a public source of random bits. If ε is not explicitly specified then we assume $\varepsilon = 1/3$. More information on the standard definitions of communication complexity can be found in [KN97].

2.2 Hitting formulas and proof system

Iwama [Iwa89] started to study hitting formulas as a polynomial-time tractable subclass of satisfiability problems (see also [Kul04]).

Definition 2.1 (Hitting formula)**.** *A* hitting formula *is a formula $F = C_1 \wedge \cdots \wedge C_m$ in conjunctive normal form such that every two of its clauses C_i and C_j contain contrary literals, that is, there is some literal ℓ such that $\ell \in C_i$ and $\bar{\ell} \in C_j$; in other words, $C_i \vee C_j$ is a tautology.*

Sometimes the notion is defined for formulas in disjunctive normal form. We call them a different name to avoid misunderstanding.

Definition 2.2 (Unambiguous DNF)**.** *An* unambiguous DNF *is the negation of a hitting formula, that is, every two its terms (conjunctions) contradict each other.*

Definition 2.3 (Hitting proof system)**.** *A refutation of a CNF F in* Hitting *is an unsatisfiable hitting formula H such that every clause C in H has a strengthening $C' \subseteq C$ in F.*

Hitting refutations can be verified in polynomial time: the unsatisfiability of H can be easily checked by counting the number of falsifying assignments, as implicitly noticed by Iwama [Iwa89] (note that the sets of falsifying assignments for any two clauses of H are disjoint), and matching clauses to their strengthening is done simply by considering all pairs $C \in H$, $C' \in F$.

The soundness of Hitting is trivial, the completeness is given by the "complete" hitting formula consisting of all possible clauses containing all the variables of F: the unique assignment falsifying such a clause C must also falsify some clause C' of (unsatisfiable) F, which is then the required strengthening of C.

2.3 Other Hitting-based proof systems

2.3.1 Hitting Res

Hitting is a "static" proof system with no real derivation procedure. We add more power to it by incorporating such steps into a Res refutation. Indeed, a resolution step can be generalized to resolve over any contradiction, not just $x \wedge \bar{x}$. In Hitting Res we resolve by hitting formulas:

Definition 2.4 (Hitting Res)**.** *This proof system embraces both* Hitting *and* Res*. One derivation step uses an unsatisfiable hitting formula $H_1 \wedge \cdots \wedge H_k$:*

$$\frac{C_1 \vee H_1,\ \ldots,\ C_k \vee H_k}{C_1 \vee \cdots \vee C_k}.$$

We also allow weakening steps:

$$\frac{C}{C \vee D}.$$

Proposition 2.5. Hitting Res *p-simulates both* Hitting *and* Res*.*

Proof. HITTING RES generalizes RES: if one uses the hitting formula $x \wedge \bar{x}$ at every step, HITTING RES turns exactly into RES. On the other hand, in HITTING we need to demonstrate that every clause of a hitting formula is a weakening of some input clause, and this can be simulated using the weakening rule. □

2.3.2 ODD HITTING

While a hitting formula covers every falsifying assignment exactly once, that is, it satisfies exactly one clause, an odd hitting formula does this an odd number of times.

Definition 2.6 (Odd hitting formula)**.** *An* odd hitting formula *is a formula $F = C_1 \wedge \cdots \wedge C_m$ in conjunctive normal form such that every falsifying assignment falsifies an odd number of its clauses.*

Definition 2.7 (ODD HITTING proof system)**.** *A refutation of a CNF F in* ODD HITTING *is an unsatisfiable odd hitting formula H such that every clause C in H has a strengthening $C' \subseteq C$ in F.*

It is not straightforward how to verify that a (not necessarily unsatisfiable) formula is an odd hitting formula, and how to verify that a formula is an unsatisfiable odd hitting formula (thus verifying ODD HITTING proofs). We show it in Prop. 3.9 and Prop. 3.8.

2.3.3 HITTING[k]

One can generalize hitting formulas by allowing a falsifying assignment to falsify a limited number of clauses (and not just a single clause) [Kul11].

Definition 2.8 (Hitting-k formula)**.** *A* hitting-k formula *is a formula F in conjunctive normal form such that every assignment falsifying F falsifies at most k clauses of F.*

Definition 2.9 (HITTING[k])**.** *A refutation of a CNF F in* HITTING[k] *is an unsatisfiable hitting-k formula H such that every clause C in H has a strengthening $C' \subseteq C$ in F.*

We show in Theorem 3.10 that HITTING[k] refutations can be verified in polynomial time.

2.3.4 HITTING($\oplus$)

HITTING($\oplus$) stands to HITTING the same way as RES($\oplus$) stands to RES, where RES($\oplus$) is the system defined in [IS20] that allows clauses to contain affine equations modulo two instead of just literals. It resembles the system RES(LIN) of [RT08] and falls under the concept of Krajíček's R(...) systems [Kra98].

Definition 2.10 (Hitting($\oplus$) formula)**.** *A* hitting($\oplus$) formula *decomposes $\{0,1\}^n$ into disjoint affine subspaces over* GF(2)*. Namely, it is a conjunction of $\oplus$-clauses of the form $\bigvee_k \left(c_k \oplus \bigoplus_{i \in I_k} x_i\right)$, where $c_k \in \{0,1\}$ are constants, x_i's are variables, and any two its $\oplus$-clauses do not share a common falsifying assignment.*

Note that we can check that two affine subspaces are disjoint using Gaussian elimination, and this gives an efficient way of checking whether a given formula is hitting($\oplus$).

$\oplus$-clauses can be thought of as sets of linear (affine) equations similarly to clauses that we usually think of as sets of literals.

Definition 2.11 (HITTING($\oplus$) proof system)**.** *A refutation of a CNF F in* HITTING($\oplus$) *is an unsatisfiable hitting($\oplus$) formula H such that every $\oplus$-clause C in H has a strengthening $C' \subseteq C$ in F.*

Note that HITTING($\oplus$) can be thought of also as a *proof system for sets of affine subspaces covering* $\{0,1\}^n$, that is, unsatisfiable systems of disjunctions of linear (affine) equations.

3 PIT helps to simulate HITTING, and more

3.1 EXTENDED FREGE p-simulates HITTING

We prove that HITTING can be p-simulated at least in the most powerful logical propositional proof system, EXTENDED FREGE. The obstacle is that the soundness of HITTING is based on the counting argument that involves the number of assignments falsified by a clause, and it is not easy to express this argument in propositional logic.

Our strategy is to p-simulate HITTING in a strong algebraic system that is p-equivalent to EXTENDED FREGE in the case of GF(2).

There are several proof systems extending the power of PC by allowing to express polynomials in a more compact way than linear combinations of monomials. Grigoriev and Hirsch [GH03] introduced $\mathcal{F}$-PC that allows to express polynomials as algebraic formulas without opening the parentheses. Of course, this needs usual associativity–commutativity–distributivity rules to transform these formulas. The next powerful system is EXT-PC considered by Alekseev [Ale21]. This is simply PC with Tseitin's extension rule generalized so that variables can be introduced for arbitrary polynomials. It can be viewed as a way to express PC proofs where polynomials can be represented as algebraic circuits (but transformations of these circuits must be justified using the definitions of extension variables that denote gates values). Eventually, Grochow and Pitassi [Pit96, GP18] suggested to generalize proof systems to allow the randomized verification of the proofs, and in these proof systems, one can switch for free between different circuit representations of a polynomial.

A FREGE system [CR79, §2] is defined as any implicationally complete inference system that uses sound constant-size rule schemata for Boolean formulas (a schema means that the formulas in the rules are represented by meta-variables, for example, F and G in the modus ponens rule $\frac{F;\ F \supset G}{G}$ can be any formulas). An EXTENDED FREGE system additionally allows us to introduce new variables using the axiom schema $x \Leftrightarrow A$ for *any* formula A, where x is a new variable.

Grigoriev and Hirsch [GH03, Theorem 3] prove that $\mathcal{F}$-PC (over any field), a system that allows us to represent polynomials using arbitrary algebraic formulas and to transform them using the ring rules, p-simulates FREGE (and also a similar statement for constant-depth $\mathcal{F}$-PC over finite fields versus FREGE with modular gates). They also state that FREGE p-simulates $\mathcal{F}$-PC over GF(2) [GH03, Remark 5]. We include a formal proof of this statement for completeness. Namely, we prove that $\mathcal{F}$-PC over GF(2) is a FREGE system itself (and it is known that all sound and implicationally complete FREGE systems over all possible sets of Boolean connectives are equivalent [Rec76, Theorem 5.3.1.4.i]).

Proposition 3.1. *$\mathcal{F}$-PC over GF(2) is a FREGE system.*

Proof. $\mathcal{F}$-PC operates with polynomial equations over GF(2), and these equations can be considered as Boolean formulas that use $\oplus$, $\wedge$ and the negation. All its rules are, of course, sound, the system is complete, and the implicational completeness can be shown as follows: if $A_1, \ldots, A_k \models F$, then

$A_1, \dots, A_k, 1 - F \models 1$; by completeness, there is a derivation $A_1, \dots, A_k, 1 - F \vdash^* 1$, which we can multiply by F [GH03, Remark 2]. □

Definition 3.2 ([Ale21])**.** *An* Ext-PC *refutation over R of a set of polynomials $P \subset R[x_1, \dots, x_n]$ is a* PC *refutation over R of a set $P \cup Q$, where Q consists of polynomials defining new variables y_i:*

$$Q := \{y_1 - q_1(x_1, \dots, x_n), y_2 - q_2(x_1, \dots, x_n, y_1), \dots, y_m - q_m(x_1, \dots, x_n, y_1, \dots, y_{m-1})\}$$

where $q_i \in R[x_1, \dots, x_n, y_1, \dots, y_{i-1}]$ are arbitrary polynomials.

While [Ale21] defines Ext-PC over arbitrary fields and even rings, we use it over finite fields only.

Similarly to Prop. 3.1, we prove that Ext-PC over GF(2) is an Extended Frege system (and it is known that all Extended Frege systems are p-equivalent [Rec76, Theorem 5.3.2.a]).

Proposition 3.3. Ext-PC *over* GF(2) *is an* Extended Frege *system.*

Proof. Extension variables can be introduced for any polynomial, but again these polynomials are Boolean formulas in the basis of $\{\oplus, \wedge, {}^-\}$. So Ext-PC is an Extended Frege system. □

The main theorem of this section is

Theorem 3.4. Ext-PC *over a finite field p-simulates* Hitting*.*

We prove it in the next subsection.

Corollary 3.5. Extended Frege *p-simulates* Hitting*.*

Proof. Follows from Theorem 3.4 and Prop. 3.3. □

Corollary 3.6. Extended Frege *p-simulates* Hitting Res*.*

Proof. We show how Extended Frege simulates a single step of Hitting Res refutation that uses a hitting formula $H_1 \wedge \dots \wedge H_k$:

$$\frac{C_1 \vee H_1, \ \dots, \ C_k \vee H_k}{C_1 \vee \dots \vee C_k}.$$

Since Extended Frege is p-equivalent to Extended Res, one can construct in polynomial time an Extended Res refutation $H_1, \dots, H_k \vdash^* \bot$ by Corollary 3.5. Observe that if we weaken the premises to $C_1 \vee H_1$, …, $C_k \vee H_k$, then this refutation turns into a derivation of a subset of $C_1 \vee \dots \vee C_k$. One can now combine the simulations of all steps into a single Extended Res refutation. □

3.2 Proof of Theorem 3.4: Ext-PC p-simulates Hitting

We have a hitting formula $H = \{C_1, \dots, C_m\}$, translate it into a system of polynomial equations $\{h_i = 0\}$ and want to construct an Ext-PC refutation of this system. In fact, $\sum_i h_i \equiv 1$ as polynomials (in what follows, we use the notation $\equiv$ for the equality of polynomials). This is certainly true pointwise on $\{0,1\}^n$, these polynomials are multilinear, and thus $\sum_i h_i$ is identical to 1. It remains to derive this fact in Ext-PC.

We translate formulas in CNF to systems of polynomial equations using the dual variables as in PCR of [ABSRW02]: for every variable x, we introduce a variable x^R along with the axiom $x + x^R - 1 = 0$. Thus every clause $C_i = \ell_1 \vee \cdots \vee \ell_k$ is represented by a monomial $m_i = \ell_1 \cdots \ell_k$: every negative literal of C_i is translated to its variable, and every positive literal is translated to the dual variable. In the proof below, we ignore these formalities and speak in the terms of x and $1 - x$ instead of x and x^R. We switch between these two representations (x^R and $1 - x$) locally when needed (in particular, we never switch to the $1 - x$ representation for more than two variables in a monomial, and switch back to x^R as soon as we are done with the respective step).

As mentioned in the introduction, our approach is based on the Raz–Shpilka deterministic polynomial identity testing for noncommutative circuits [RS05]. The main idea is to introduce new variables for quadratic polynomials: it suffices to do it for a basis. Namely, at the first step we consider two variables x_1 and x_2 and quadratic polynomials (potentially, $(1-x_1)(1-x_2)$, $(1-x_1)x_2$, $x_1(1-x_2)$, x_1x_2, $1-x_1$, x_1, $1-x_2$, and x_2) appearing in the monomials m_i as $x_1^R x_2^R$, $x_1^R x_2$, etc., and replace them using linear combinations of new variables $y_i^{1,2}$, thus decreasing the degree by one. At the next steps we treat all the variables $y_i^{1,2}$ as a single "layer" (note that they are *not* multiplied by each other). We merge this layer of $y_i^{1,2}$ with x_3, getting a layer of variables $y_j^{1,2,3}$, and so on, until we reach a linear equation, which is easy to verify.

In order to implement this strategy, we prove a lemma allowing to merge two layers of variables. This lemma holds over any field $\mathbb{F}$.

Lemma 3.7. *Let $\vec{x}, \vec{y}, \vec{z}$ be three disjoint vectors of variables. Suppose that $P_i(\vec{x}), Q_i(\vec{y}), R_i(\vec{z})$ are polynomials satisfying*

$$\sum_{i=1}^{t} P_i(\vec{x})Q_i(\vec{y})R_i(\vec{z}) \equiv 0. \tag{3.1}$$

Let $W_1, \ldots, W_k \in \mathbb{F}[\vec{x}, \vec{y}]$, where $k \le t$, be a basis for $\{P_i(\vec{x})Q_i(\vec{y}) \mid i \in [t]\}$. In particular, let $S_i(\vec{w}) = \sum_{j=1}^{k} \sigma_{ij} w_j$ be the expansion of $P_i(\vec{x})Q_i(\vec{y})$ in this basis, that is, $P_i(\vec{x})Q_i(\vec{y}) = S_i(\vec{W}(\vec{x}, \vec{y}))$. Then

$$\sum_{i=1}^{t} S_i(\vec{w})R_i(\vec{z}) \equiv 0. \tag{3.2}$$

Proof. Let $T_j(\vec{z}) = \sum_{i=1}^{t} \sigma_{ij} R_i(\vec{z})$. Then

$$\sum_{j=1}^{k} w_j T_j(\vec{z}) = \sum_{i=1}^{t} \sum_{j=1}^{k} \sigma_{ij} w_j R_i(\vec{z}) = \sum_{i=1}^{t} S_i(\vec{w}) R_i(\vec{z})$$

is the polynomial that we are proving to be identically zero. Assuming the contrary, we conclude that for some $j = j^*$, the polynomial T_{j^*} is not identically zero.

If T_{j^*} would be multilinear (as it is in our applications, where all R_i's are linear), that would already be enough to reach a contradiction: there should be some vector $\vec{\rho}$ of 0/1-values such that $T_{j^*}(\vec{\rho}) \neq 0$. Let us substitute $\vec{\rho}$ for $\vec{z}$ and $\vec{W}(\vec{x}, \vec{y})$ for $\vec{w}$ in (3.2). Under this substitution, (3.2) and (3.1) turn into the same equation, which shows a linear dependency of W_j's contradicting the fact that $\{w_j \mid j \in [k]\}$ is a basis.

In order to prove the statement without the multilinearity condition, choose an extension field that is large enough so that we could choose a vector ζ of values in this field such that $T_{j^*}(\zeta) \neq 0$, and perform the same substitution obtaining a linear dependency of W_j's over the extension field and hence in $\mathbb{F}$. □

With this lemma at hand, we are ready to prove the simulation theorem.

Proof of Theorem 3.4. We consider a hitting formula $H = \{C_1, \dots, C_m\}$ and translate each its clause $C_j \in H$ into a product $\prod_{i=1}^{n} P_j^i(x_i)$, where

$$P_j^i(x_i) := \begin{cases} 1, & \text{if } x_i \text{ does not occur in } C_j, \\ x_i, & \text{if } x_i \text{ occurs in } C_j \text{ negatively}, \\ 1 - x_i, & \text{if } x_i \text{ occurs in } C_j \text{ positively}. \end{cases}$$

We call this product a monomial, because in Ext-PC it can be represented using dual variables $x_i^R = 1 - x_i$. We are going to refute $\sum_{j=1}^{m} \prod_{i=1}^{n} P_j^i(x_i)$ in Ext-PC, namely, we derive the polynomial 1 from it.

Consider the vector space spanned by the set $\{P_j^1(x_1)P_j^2(x_2) \mid j \in [m]\}$. We can find a basis $\{Y_i^{1,2}(x_1, x_2) \mid i \in [r]\}$ and introduce extension variables for its polynomials, $y_i^{1,2} = Y_i^{1,2}(x_1, x_2)$. We then consider the linear functions $P_j^{1,2}(Y^{1,2}(x_1, x_2))$ giving the expansion of $P_j^1(x_1)P_j^2(x_2)$ over this basis, and we can derive in Ext-PC that $P_j^{1,2}(\vec{y}^{1,2}) - P_j^1(x_1)P_j^2(x_2) = 0$.

Recall that

$$\sum_{j=1}^{m} \prod_{i=1}^{n} P_j^i(x_i) \tag{3.3}$$

is identically 1, because this is a multilinear polynomial that equals 1 pointwise on $\{0,1\}^n$ (2^n values uniquely define 2^n coefficients of the multilinear polynomial). Then Lemma 3.7 shows that

$$\sum_{j=1}^{m} P_j^{1,2}(\vec{y}^{1,2}) \prod_{i=3}^{n} P_j^i(x_i) \tag{3.4}$$

is also identically 1.

Since the new variables are not multiplied by each other in our monomials, we can continue this process merging the variables $\vec{y}^{1,2}$ with x_3, then the new variables $\vec{y}^{1,2,3}$ with x_4, and so on, until we merge all variables into $\vec{y}^{[n]}$. That is, we eventually arrive at

$$\sum_{j=1}^{m} P_j^{[n]}(\vec{y}^{[n]}) \tag{3.5}$$

for a *linear* function $P_j^{[n]}$. This linear polynomial is also identically 1.

It is easy to see that Ext-PC proves efficiently that all these polynomials are equivalent (switching between dual variables and their definitions whenever needed within two layers of variables), in particular, it derives efficiently (3.5) from (3.3).

Since (3.5) is a linear polynomial that is identically 1, it has all zero coefficients except for the free term that is equal to 1. □

The proof of Theorem 3.4 can be used for proving in Ext-PC similar statements about multilinear polynomials that use dual variables. In particular, it can be used for simulating Odd Hitting and Hitting[k].

Proposition 3.8. Odd Hitting *proofs can be verified in deterministic polynomial time.* Ext-PC *over* GF(2) *p-simulates* Odd Hitting. *In particular,* Extended Frege *p-simulates* Odd Hitting.

Proof. The proof of Theorem 3.4 works in particular over GF(2). □

This argument allows us to verify *unsatisfiable* odd hitting formulas. However, a similar technique also makes it possible to check arbitrary formulas for being odd hitting.

Proposition 3.9. *Given a formula in CNF, it can be checked in deterministic polynomial time whether F is an odd hitting formula.*

Proof. We need to check that there is no falsifying assignment that falsifies an even number of clauses. For each clause $C \in F$, substitute the negation of C as an assignment into F, and drop the identically false clause resulting from substituting $\bar{C}$ into C; denote this formula F_C. Then check that falsifying assignments falsify an even number of clauses of F_C by verifying the identity $\sum_i M_i \equiv 0$ over GF(2), where M_i's are pseudomonomials (non-negative juntas) corresponding to the clauses of F_C. If for some C the identity is false, then F had a falsifying assignment that satisfied an even number of clauses. □

Theorem 3.10. Hitting$[k]$ *proofs can be verified in deterministic polynomial time.* Ext-PC *over a finite field p-simulates* Hitting$[k]$. *In particular,* Extended Frege *p-simulates* Hitting$[k]$.

Proof. For a hitting-k formula $\bigwedge_{i=1}^{m} T_i$, by the inclusion-exclusion formula

$$\sum_{\emptyset \neq I \in 2^{[m]}} (-1)^{|I|+1} \prod_{i \in I} T_i = 1,$$

where we abuse the notation by identifying a clause and its PCR translation into a monomial that uses dual variables. Note that the terms containing more than k clauses T_i's are zeros.

Now we can proceed by analogy with the proof of Theorem 3.4 and Corollary 3.5. □

3.3 Bonus: succinct proofs and efficient verification of static (semi)algebraic proof systems

The proof of Theorem 3.4 does not just provide an Ext-PC proof, it provides a deterministic polynomial-time verification procedure for polynomial identity testing for multilinear identities modulo $x + \bar{x} = 1$. It can be used in other settings, for example, for verifying proofs in static (semi)algebraic systems.

Historically, deterministic polynomial-time verification of proofs in such systems has not been a major concern, because proving a superpolynomial lower bound for such a system implies a lower bound for a variety of systems that emerge from supplementing the basic static system with additional means of verification, for example, the axioms of the polynomial ring as in [GH03]. However, an increased interest to automated search for sum-of-squares-based proofs reveals the need for an efficient deterministic formal proof verification procedure. A typical static proof constitutes a formal combination of polynomials including non-negative juntas written in a formal way (without opening the parentheses). To verify such a proof one needs to check that this polynomial is identical to a constant. Opening the parentheses would not work as it would produce far too many monomials.

Fortunately, the proof of Theorem 3.4 demonstrates that any multilinear polynomial identity using dual variables over a finite field can be verified efficiently. In this subsection we show how to use this idea for static proof checking.

Verifying NS proofs is easy: it suffices to open parentheses in products of two polynomials, each of them being represented as a sum of monomials with coefficients. Alekhnovich et al [ABSRW02] suggested using *dual* variables in PCR, essentially adding extension axioms for the negations of variables to PC. Such dual variables can be (and have been) also used in other algebraic and semialgebraic proof systems, in particular, NS turns into a more powerful system NSR [dRLNS21]. It is more difficult to verify NSR proofs, however, the proof of Theorem 3.4 already does it over a finite field: opening the parentheses, as before, without expanding the definitions of dual variables turns the proof into a sum of monomials involving the input and dual variables, exactly as studied in the proof of Theorem 3.4. In fact, the identity being verified is a *succinct* NSR proof of the form

$$\sum f_i g_i \equiv 1 \tag{3.6}$$

without explicitly mentioning the Boolean axioms $x^2 - x = 0$ or the dual variables axioms $x + \bar{x} - 1 = 0$, that is, f_i's are only the translations of the original clauses. To verify this identity using the framework of Theorem 3.4, we turn $f_i g_i$'s into multilinear polynomials by dropping monomials containing dual variables x and $\bar{x}$ together and reducing the degrees of variables to one in other monomials.

When we switch to $\mathbb{Q}$ in NSR or turn our attention to semialgebraic proof systems, there is a problem with this approach: the coefficients can grow in our transformations between the bases, and the bit-size of the new proof can become superpolynomial in terms of the bit-size of the original proof (note that according to the Cook–Reckhow definition we are taking into account the bit-size of the proof and not just the degree or the number of monomials as they sometimes do in the context of algebraic proofs). To avoid this obstacle, we can proceed as follows:

- transform the proof from rationals to integers by multiplying it by all the denominators, the free term 1 or -1 then becomes a different positive/negative constant,
- use the Chinese Remainder Theorem to employ verification in finite fields.

Let M be an upper bound on the absolute values of coefficients before monomials in (3.6) or a similar equation in semialgebraic proof systems. Then by the Chinese Remainder Theorem, it suffices to verify that $P \equiv 0$ modulo primes whose product exceeds $2M + 1$. Using a deterministic polynomial-time primality algorithm, we can find all primes up to $K \log(M) \log\log(M)$ in polynomial time. For an appropriate value of K, their product exceeds $2M + 1$. Then it is enough to verify the required identity modulo each of these primes, that is, in the respective finite fields.

Common static systems proofs can be verified as multilinear polynomial identities using dual variables by considering succinct proofs and multilinearization as above.

Static LS, SA, SCS. The static Lovász–Schrijver [GHP02] and Sherali–Adams [DMR09, ALN16], and Subcube Sums proof systems [FMSV23] are defined as follows.

A *pseudomonomial* is a product of input variables and their negations (in the form of $1 - x$). A *conic junta* is a nonnegative linear combination of pseudomonomials. We can generalize proofs using juntas right away by allowing dual variables (and then juntas become simply monomials) and augmenting the system with axioms $x + \bar{x} - 1 = 0$, as is done in [DMR09, dRLNS21]. We can also consider succinct proofs as above by formulating identities modulo $x^2 - x = 0$ and $x + \bar{x} - 1 = 0$ for every variable x.

Static LS is defined for a system of inequalities $s_i \geq 0$ that typically include the (obvious) translations of Boolean clauses as linear inequalities along with inequalities $x^2 - x \geq 0$,

$x - x^2 \geq 0$, $x \geq 0$, $1 - x \geq 0$ for every variable. It is defined as a formal sum-of-products $\sum s_i J_i \equiv -1$, where J_i's are conic juntas. The dual variables version can be defined similarly by augmenting the list of s_i's with $x + \bar{x} - 1 \geq 0$, $1 - x - \bar{x} \geq 0$.

Sherali–Adams proofs are considered for systems of polynomial equations (the translations of clauses into pseudomonomials C_i along with $x^2 - x = 0$ for every variable), the proof is a formal sum-of-products $\sum C_i(P_i - Q_i) + R \equiv -1$. It can be viewed as a static Lovász–Schrijver proof, where $C_i = 0$ is represented by the two inequalities $C_i \geq 0$ and $-C_i \geq 0$. Note that "additive" (linear inequalities) and "multiplicative" (equations for pseudomonomials) representations of clauses are in fact equivalent with respect to size [GHP02, Lemmas 3.1, 3.2], where efficiently representing long clauses is, of course, only possible with dual variables.

A Subcube Sums proof can be viewed a restriction of succinct unary SA where $P_i - Q_i = 1$ and only the size of R counts towards the size of the proof, with copies of the same monomial counted with multiplicity. For instance, a SCS proof of a hitting formula is simply "0". Similarly to Hitting, this is an inherently succinct proof system.

Also, non-propositional versions of these systems are available, where the axioms are not translations of clauses of a Boolean formula, and multiplying such axioms is allowed, which would a priori cause a problem with our strategy. However, for translations of clauses, the identities that we need to verify are still linear combinations of monomials in the input and dual variables, so they fall completely under the same framework.

Static LS_+^∞, SoS. These systems augment the previous systems with squares. A proof in LS_+^∞ is an identity of the sort

$$\sum s_i J_i + \sum_j q_j^2 J'_j \equiv -1$$

where q_j are arbitrary polynomials and J_i's and J'_j's are conic juntas. There are several definitions of SoS in the literature, but modulo the equivalence between equations and pairs of opposite inequalities, they are essentially the same in the propositional case. These systems still fall under our framework, as opening the parentheses in a square of a sum of monomials, $(\sum a_i M_i)^2 = \sum a_i^2 M_i^2 + \sum (2a_i a_j) M_i M_j$, gives only a quadratic number of monomials.

4 Hitting vs tl-Res and other classical systems

In this section we prove that while Hitting p-simulates tl-Res, in the other direction tl-Res simulates Hitting only quasi-polynomially. Moreover, tl-Res is quasi-polynomially weaker than Hitting.

We also relate Hitting to other proof systems: the tree-like version of Res($\oplus$) (they are incomparable), certain versions of NS, and Res.

4.1 tl-Res quasi-polynomially simulates Hitting

We can use a construction of small decision trees from DNF covers of Boolean functions to quasi-polynomially simulate Hitting in tl-Res.

Theorem 4.1 ([EH89])**.** *Let D_0 and D_1 be DNF covers of size r and s that cover the 0s and 1s of a function $f\colon \{0,1\}^n \to \{0,1\}$ respectively. Then there is a decision tree computing f of size at most $2(rsn)^{\log(r+s)+1}$.*

We adapt the argument of Ehrenfeucht–Haussler to prove the following theorem. Intuitively, every hitting formula defines a subcube partition of the Boolean cube $\{0,1\}^n$. The structure of this partition can be used to greedily construct a decision tree (TL-RES refutation) that always queries the most conflicting variable in the narrowest clause.

Theorem 4.2. *If a CNF formula F has a* HITTING *refutation of size m, then F has a* TL-RES *refutation of size at most $O(2^{2\log^3 m})$.*

Proof. Let G be a hitting formula with n variables and m clauses that is a refutation of F and let us recursively build a decision tree that solve the falsified clause search problem of G. As every clause of G is a weakening of some clause of F the resulting tree will also solve the falsified clause search problem of F. Let C be a narrowest clause in G, which has width at most $\log m$ (otherwise a union bound shows that the formula is satisfiable). Since G is a hitting formula, every other clause has at least one contrary literal with respect to C. Let ℓ_i be the literal in C that appears in the maximal number of other clauses with a different sign, which by an averaging argument is at least $(m-1)/\log m$ times. The decision tree queries the corresponding variable. If the answer does not satisfy C then it satisfies all clauses containing $\overline{\ell_i}$, hence the resulting formula has $n-1$ variables and at most $m-(m-1)/\log m$ clauses. Otherwise the formula has $n-1$ variables and at most $m-1$ clauses.

The number of leaves in the decision tree satisfies the recurrence

$$S(n,m) \leq S(n-1, m-(m-1)/\log m) + S(n-1, m-1).$$

We claim that $S(n,m) \leq n^{2\log^2 m}$. Indeed,

$$\begin{aligned}
S(n,m) &\leq S(n-1, m-(m-1)/\log m) + S(n-1, m-1) \\
&\leq (n-1)^{2(\log(m-(m-1)/\log m))^2} + (n-1)^{2\log^2(m-1)} \\
&\leq (n-1)^{2(\log(m(1-1/(2\log m))))^2} + (n-1)^{2\log^2 m} \\
&= (n-1)^{2(\log m + \log(1-1/(2\log m)))^2} + (n-1)^{2\log^2 m} \\
&\leq (n-1)^{2\log m \cdot (\log m + \log(1-1/(2\log m)))} + (n-1)^{2\log^2 m} \\
&\leq (n-1)^{2\log m \cdot (\log m - 1/(2\log m))} + (n-1)^{2\log^2 m} \\
&= (n-1)^{2\log^2 m - 1} + (n-1)^{2\log^2 m} \\
&\leq (n-1+1)^{2\log^2 m}.
\end{aligned}$$

□

Remark 4.3. *An alternative, more combinatorial way to compute the number of leaves of the tree is as follows. At a node of type (n', m') (i.e., with n' variables and m' clauses), we have a left turn to a node of type (n'', m''), where $n'' < n'$ and $m'' \leq (1-1/(2\log m'))m' \leq (1-1/(2\log m))m'$, and a right turn to a node of type (n'', m''), where $n'' < n'$ and $m'' < m'$. Every path from the root to a leaf (a node of type $(n', 1)$) thus contains at most $\log_{(1-1/(2\log m))^{-1}} m = \log m / \log(1-1/(2\log m))^{-1} = O(\log^2 m)$ left turns. We can identify each leaf with a binary string of length at most n with $O(\log^2 m)$ many zeroes. The number of leaves is thus at most*

$$\binom{\leq n}{\leq O(\log^2 m)} \leq n^{O(\log^2 m)}.$$

The argument can be extended to HITTING[k] simulation by HITTING (and hence by TL-RES).

Proposition 4.4. HITTING *quasi-polynomially simulates* HITTING[k] *up to* $k = (\log m)^{O(1)}$.

Proof. Consider a hitting-2 formula with m clauses. If it contains fewer than three clauses then it mentions constantly many variables, and so can be refined to a hitting formula of constant size. Otherwise, it mentions at most m variables, and the narrowest three clauses T_i, T_j, T_k each contain at most $\log_2 m + O(1)$ literals. Consider sets of assignment S_i, S_j, S_k falsified by these clauses. If for some other clauses T_a, T_b their respective sets S_a, S_b are not disjoint, then $S_a \cap S_b \cap S_c = \emptyset$ for some $c \in \{i, j, k\}$, and so one of the literals in T_c appears negated in one of T_a, T_b. Hence we can find a variable assignment such that for $\Omega(1/\log m)$ fraction of non-disjoint pairs S_a, S_b, the variable assignment removes one of the clauses. Construct a decision tree whose root asks about this variable, and recurse. A "left turn" is the answer which hits the $\Omega(1/\log m)$ fraction. After $O(\log^2 m)$ left turns, the remaining sets are disjoint (the clauses form a hitting formula), which is a leaf in our decision tree. Since the depth is at most n, the decision tree contains $\binom{\leq n}{\leq O(\log^2 m)}$ many leaves, and so the original formula can be refined to a hitting formula of size $n^{O(\log^2 m)}$.

A similar argument works for reducing hitting-t formulas to hitting-$(t-1)$ formulas. In the base case, fewer than t terms, the formula mentions fewer than $t-1$ variables, and so there is a refinement of size 2^t. The decision tree contains $n^{O(t \log^2 m)}$ many leaves, and altogether the reduction loses a multiplicative factor of $n^{O(t \log^2 m)}$. We can reduce all the way to a hitting formula of size $n^{O(t^2 \log^2 m)}$.

By analogy, HITTING[$\log^k m$] can be simulated in HITTING with complexity $n^{O(\log^{k+2} m)}$. □

Later in Theorem 4.14 we prove that the simulation of HITTING by TL-RES cannot be polynomial; however, we do not know whether it can be improved to $m^{O(\log m)}$.

Corollary 4.5. *There are formulas that have polynomial-size* RES *proofs but require exponential-size* HITTING *proofs.*

Proof. [BSIW04] proves that there are formulas that have polynomial-size RES proofs but require exponential-size TL-RES proofs. Using Theorem 4.2 the statement follows. □

Remark 4.6. *Similarly to Theorem 4.2,* HITTING RES *can be quasi-polynomially simulated in* RES *(every hitting resolution step can be simulated using Theorem 4.2), and thus an exponential-size lower bound for it also follows from exponential-size lower bounds for* RES *(e.g., [Urq87]).*

4.2 HITTING is quasi-polynomially stronger than TL-RES

Theorem 4.7. HITTING *p-simulates* TL-RES.

Proof. A TL-RES proof can be viewed as a decision tree [BSIW04]: every application of the resolution of clauses $C \vee x$ and $D \vee \bar{x}$ by the variable x is considered as a decision by the variable x, so that the decision $x = 0$ leads to the node $C \vee x$ from the node $C \vee D$, and the decision $x = 1$ leads to the node $D \vee \bar{x}$. Therefore, the assignment of decisions starting at the root, labelled by the final empty clause, and ending at a leaf, labelled by a clause L of the input formula F, falsifies L. The negation of such assignment, viewed as a clause N, is a weakening of L. The conjunction of all such N's is an unsatisfiable hitting formula H that is a proof of F. The number of clauses in H is at most the number of leaves in the decision tree and therefore at most the number of occurrences of F's clauses in the TL-RES refutation. □

We use $\oplus$-lifting to prove our separation result. We need three folklore or easy statements that we include for the sake of completeness.

Definition 4.8 (Composition of multivariate functions)**.** *For Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$, we define $f \circ g^n\colon \{0,1\}^{nm} \to \{0,1\}$ as*

$$(f \circ g^n)(x_1^1, \dots, x_m^1, \dots, x_1^n, \dots, x_m^n) := f(g(x_1^1, \dots, x_m^1), \dots, g(x_1^n, \dots, x_m^n)).$$

Lemma 4.9. *If $f\colon \{0,1\}^n \to \{0,1\}$ is a function requiring decision tree depth d, then $f \circ (\oplus_2)^n$ requires decision tree size 2^d.*

Proof. We consider a size-s decision tree T computing $(f \circ (\oplus_2)^n)(x_1^1, x_2^1, \dots, x_1^n, x_2^n)$ and transform it into a querying strategy for f using at most $\log_2 s$ queries. Starting at the root, we go down the tree both querying $f(\vec{y})$'s inputs y_i and giving answers to the queries made in the original tree T. We will be interested in the number of leaves in the current subtree of T.

Assume that T queries x_j^i. If the other variable x_{2-j}^i has not been queried yet, we do not query inputs of f and simply choose the answer $\tau \in \{0,1\}$ for $x_j^i = \tau$ that directs us to the subtree that has fewer leaves. Otherwise, we choose $\tau = y_i \oplus x_{2-j}^i$. Therefore, we go into the subtree of T containing at least twice fewer leaves at least once per query to y_i's, and we can do it at most $\log_2 s$ times before we come to a leaf. □

Lemma 4.10. *Suppose a function $f\colon \{0,1\}^n \to \{0,1\}$ has an unambiguous DNF of width w and $g\colon \{0,1\}^m \to \{0,1\}$ has a decision tree of depth d. Then $f \circ g^n$ has an unambiguous DNF of width wd.*

Proof. A decision tree of depth d can be represented as an unambiguous DNF of width d itself, so we have a set G of terms (conjunctions) of width at most d contradicting each other, and it is split into two sets G_0 and G_1 equivalent to $g(\vec{x}) = 0$ and $g(\vec{x}) = 1$, respectively.

Consider a term in the unambiguous DNF representation of $f(\vec{y})$ and replace each literal $y_i = \alpha$ with unambiguous DNF representation of $g(\vec{x}^i) = \alpha$ obtained from renaming the variables in G_α, and then expand. □

Lemma 4.11. *Suppose an unambiguous DNF has width w. Then it has at most 2^w terms.*

Proof. Let n be the number of variables in the DNF. Then each term has at least 2^{n-w} satisfying assignments. Then since no assignment satisfies two terms, the number of terms is at most $\frac{2^n}{2^{n-w}} = 2^w$. □

In order to prove our result, we need the following separation of randomized query complexity (deterministic is enough for our purpose) and unambiguous certificate complexity from [AKK16].

Definition 4.12 ([ABDK16, AKK16])**.** *Let $f\colon \{0,1\}^N \to \{0,1\}$ be a function, $c = 10 \log N$ and $m = c \cdot \mathsf{C}(f) \log N = 10\mathsf{C}(f) \log^2 N$. Then the cheat sheet version of f, denoted f_{CS}, is a total function $f_{\mathrm{CS}}\colon (\{0,1\}^N)^c \times (\{0,1\}^m)^{2^c} \to \{0,1\}$.*

Let the input be written as $(x^1, x^2, \dots, x^c, Y_1, Y_2, \dots, Y_{2^c})$, where for all $i \in [c]$, $x^i \in \{0,1\}^N$, and for all $j \in [2^c]$, $Y_j \in \{0,1\}^m$. Let $\ell_i = f(x^i)$ and $\ell \in [2^c]$ be the positive integer corresponding to the binary string $\ell_1, \dots, \ell_c$. Then we define the value of f_{CS} to be 1 if and only if Y_ℓ contains certificates for $f(x^i) = \ell_i$ for all $i \in [c]$.

At first glance, the definition of f_{CS} might look nonconstructive due to the usage of $\mathsf{C}(f)$. However, the theorem of [AKK16] uses an appropriate upper bound on $\mathsf{C}(f)$, which is proved along with the interactive construction of the function.

Theorem 4.13 ([AKK16, Theorem 5.1])**.** *Let $f_0 = \mathrm{AND}_n$ and f_k be defined inductively as $f_k := \mathrm{AND}_n \circ (\mathrm{OR}_n \circ f_{k-1})_{\mathrm{CS}}$, where f_k has $O(n^{25^k})$ inputs. Then $\mathsf{R}(f_k) = \tilde{\Omega}(n^{2k+1})$ and $\mathsf{UC}(f_k) = \tilde{O}(n^{k+1})$.*

Theorem 4.14. *For every $\varepsilon > 0$, there exists a sequence of unsatifiable hitting formulas G_m containing $2^{\tilde{O}(m)}$ clauses of width at most $\tilde{O}(m)$ such that G_m requires* TL-RES *proof size $2^{\tilde{\Omega}(m^{2-\varepsilon})}$.*

Proof. We consider the composition of a function separating R and UC with the parity function. Let f be the function given by Theorem 4.13. Let F_0 and F_1 be the respective unambiguous DNFs, they both have width $\tilde{O}(n^{k+1})$ and thus by Lemma 4.11 have size $2^{\tilde{O}(n^{k+1})}$. Therefore, Lemma 4.10 provides two unambiguous DNFs, $F_0 \circ (\oplus_2)^m$ representing $f \circ (\oplus_2)^m$ and $F_1 \circ (\oplus_2)^m$ representing $\bar{f} \circ (\oplus_2)^m$, both of width $\tilde{O}(n^{k+1})$ and of size $2^{\tilde{O}(n^{k+1})}$.

Consider a CNF $F = \overline{F_0 \circ (\oplus_2)^m \vee F_1 \circ (\oplus_2)^m}$. This is an unsatisfiable hitting formula, so it is a short HITTING proof of itself. It contains $2^{\tilde{O}(n^{k+1})}$ clauses.

Suppose F has a TL-RES refutation of size s. Let us view it as a decision tree solving the falsified clause search problem for F. Now let us change leaf labels in the following way: a leaf labeled with a clause that came from $F_i \circ (\oplus_2)^m$ gets label i. It is easy to see that the resulting tree computes $f \circ (\oplus_2)^m$. Recall that f is constructed by Theorem 4.13 and its query complexity is $\tilde{\Omega}(n^{2k+1})$. By Lemma 4.9 its decision tree must have size $2^{\tilde{\Omega}(n^{2k+1})}$. Now the theorem claim holds for $m = n^{k+1}$. □

4.3 HITTING and TL-RES(⊕) are incomparable

4.3.1 A hard formula for HITTING

In this section we observe that there exist formulas that are easy for TL-RES(⊕) and exponentially hard for HITTING. For this, we recall the separation of TL-RES(⊕) from RES shown in [IS20] for Tseitin formulas.

Definition 4.15 (Tseitin formulas $T_{G,c}$)**.** *For a constant-degree graph $G = (V, E)$ and a 0/1 vector c of "charges" for the vertices, consider the following linear system in the variables x_e for $e \in E$:*

$$\bigwedge_{v \in V} \left(\bigoplus_{e \ni v} x_e = c_v \right),$$

where $\bigoplus_{v \in V} c_v = 1$. In the corresponding Tseitin formula $T_{G,c}$ in CNF each vertex constraint $\bigoplus_{e \ni v} x_e = c_v$ expands into $2^{\deg v - 1}$ clauses of width $\deg v$.

Theorem 4.16 ([Urq87])**.** *There exists a family of constant-degree graphs G_n with n nodes and a family of charge vectors c_n such that Ts_{G_n,c_n} requires* RES *refutation of size $2^{\Omega(n)}$.*

Theorem 4.17 ([IS20])**.** *For any graph G and charges c the Tseitin formula $\mathrm{Ts}_{G,c}$ has a tree-like Res(⊕) refutation of size linear in the size of the CNF.*

Given the quasi-polynomial simulation of Theorem 4.2 and the following generalization of Theorem 4.7, we can separate HITTING from TL-RES($\oplus$) and HITTING($\oplus$).

Proposition 4.18. *If F has a tree-like* RES($\oplus$) *refutation of size s, then it has a* HITTING($\oplus$) *refutation of size s.*

Proof. By analogy with Theorem 4.7, the leaves of a tree-like RES($\oplus$) refutation form a HITTING($\oplus$) refutation. □

Corollary 4.19. *There exists a family of CNF formulas F_n such that F_n requires Resolution refutation of size $2^{\Omega(n)}$, Hitting refutation of size $2^{n^{\Omega(1)}}$ and admits polynomial-size tree-like Res($\oplus$) refutation (and, consequently, polynomial-size* HITTING($\oplus$) *refutation).*

Proof. Take Ts_{G_n,c_n} from Theorem 4.16 and apply Prop. 4.18. □

4.3.2 A hard formula for TL-RES($\oplus$)

In addition to separating HITTING from TL-RES, we can follow the same plan to separate it from a stronger TL-RES($\oplus$) proof system, that is, to lift a separation between unambiguous certificate complexity and query complexity. We cannot use decision tree size to bound TL-RES($\oplus$) size, but rather the stronger randomized communication complexity measure.

Theorem 4.20 ([IS20, Theorem 3.11])**.** *Let F be an unsatisfiable CNF that has tree-like Res($\oplus$) refutation of size t then the randomized communication complexity of the falsified clause search problem for F is $O(\log t)$.*

An analogue of Lemma 4.9 for randomized communication complexity also holds, with the difference that we need to compose the DNFs F_0 and F_1 from Theorem 4.14 with the indexing function instead of $\oplus$. The indexing function $\textsc{Indexing}_m\colon [m] \times \{0,1\}^m \to \{0,1\}$ is defined as $\textsc{Indexing}_m(i,x) = x_i$, i.e. it accepts an index and a vector and returns the element of the vector with the given index. Observe that $\textsc{Indexing}_m$ has a decision tree of depth $\lceil \log_2 m \rceil + 1$: we first query the index and then query a single bit of the vector.

Theorem 4.21 ([GPW17])**.** *If a function $f\colon \{0,1\}^n \to \{0,1\}$ requires a randomized decision tree of depth t, then the function $f \circ (\textsc{Indexing}_m)^n$ where $m = n^{256}$ requires randomized communication cost $\Omega(t \log n)$.*

This is all we need to prove the separation.

Theorem 4.22. *For every $\epsilon > 0$, there is a sequence of unsatisfiable hitting formulas G_m containing $2^{\tilde{O}(m)}$ clauses of width $\tilde{O}(m)$ that requires* TL-RES($\oplus$) *proof size $2^{\tilde{\Omega}(m^{2-\epsilon})}$.*

Proof. We construct F as follows: take $f_k\colon \{0,1\}^M \to \{0,1\}$ from Theorem 4.13, where $M = O(n^{25^k})$ and $m = \mathsf{UC}(f_k) = \tilde{O}(n^{k+1})$, and compose it with $\textsc{Indexing}_{M^{256}}$. By Lemma 4.10, $f_k \circ (\textsc{Indexing}_{M^{256}})^M$ and $\neg(f_k \circ (\textsc{Indexing}_{M^{256}})^M)$ both have unambiguous DNF representations of width $\tilde{O}(n^{k+1})$. Then F is the conjunction of the negated terms in these representations.

On the one hand, by Lemma 4.11 F has size $2^{\tilde{O}(m)}$, and since it is already a hitting formula, it serves as a HITTING proof of itself.

On the other hand, let s be the size of the smallest TL-RES($\oplus$) refutation of F. Then by Theorem 4.20 there exists a randomized communication protocol solving the falsified clause search

problem for F of cost $O(\log s)$. Observe that such protocol can be easily converted into a protocol solving $f \circ (\textsc{Indexing}_{k^{256}})^k$ with the same cost and at most the same probability of error: if the protocol returns a clause corresponding to a term in the representation of $f \circ (\textsc{Indexing}_{k^{256}})^k$ answer 1, otherwise answer 0. Then by Theorem 4.21 we have that $\log s = \tilde{\Omega}(n^{2k+1})$, i.e.

$$s = 2^{\tilde{\Omega}(n^{2k+1})} = 2^{\tilde{\Omega}(m^{2-\epsilon})}.$$ □

4.4 Relation to Res and NS

As we discussed in Section 4.3.1, a corollary of Theorem 4.2, which shows that tl-Res quasi-polynomially simulates Hitting, is that if a proof system $\mathcal{P}$ is exponentially separated from tl-Res then $\mathcal{P}$ is also exponentially separated from Hitting. Since this is the case with Res and NS—which have short proofs of the ordering principle and the bijective pigeonhole principle [BR96] respectively, while tl-Res requires exponentially long proofs of both—we conclude that Res and NS are exponentially separated from Hitting.

In this section we explore whether a simulation or separation in the other direction exists. We show that the formula that we used for the quasi-polynomial separation of Hitting from tl-Res has short Res refutations, and therefore cannot be used for showing a separation from Res. We also show that in a sense NS simulates Hitting.

4.4.1 Dag-like query complexity of functions

Our resolution upper bound is in fact an upper bound on a width, a measure that can be studied through its characterization as a two-player game [Pud00, AD08]. Building on such game, Göös et al. [GGKS20] introduced the following generalization of the query complexity of a function $f\colon \{0,1\}^n \to M$. We view it as a game between two players, the *querier* and the *adversary*. The querier maintains a partial assignment $\rho\colon [n] \to \{0,1,*\}$. At each step the querier can either query a variable $i \in \rho^{-1}(*)$, in which case the adversary picks a value $\alpha \in \{0,1\}$ and assigns $\rho(i) := \alpha$, or forget a variable $i \in \rho^{-1}(\{0,1\})$, assigning $\rho(i) := *$. Note that the adversary may answer differently the next time a forgotten variable is queried. The game ends only if ρ is a certificate for f, that is if there exists $m \in M$ such that for all $x \in \{0,1\}^n$ consistent with ρ, we have $f(x) = m$. The width of a particular game π is $\mathsf{w}(\pi)$, the maximal size (number of assigned variables) of ρ at any step of the game, and the leaf-width is $\mathsf{w}_{\text{out}}(\pi)$, the maximum size of ρ at any terminal step of the game. The dag-like query complexity of f is the width of the game assuming optimal play where querier aims to minimize the width and adversary aims to maximize it. Denote the dag-like query complexity of f by $\mathsf{w}(f)$.

This notion can be similarly defined for relations, and for an unsatisfiable CNF $\phi = \bigwedge_{i\in[m]} C_i$ the dag-like query complexity of the falsified clause search relation $\{(x,i) \mid C_i(x) = 0\}$ is exactly the resolution width of ϕ [AD08]. If ϕ is a hitting formula, this relation is actually a function.

As a preliminary step towards our upper bound we compute the dag-like query complexity of the OR $\circ$ AND function, which is a key part of the construction of Theorem 4.13. We begin by proving that dag-like query complexity is sub-multiplicative with respect to composition, and in fact we can be a bit more precise.

Lemma 4.23. *Let $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$ be two Boolean functions. Let π be the dag-like query complexity game for g. Then*

$$\mathsf{w}(f \circ g^n) \le (\mathsf{w}(f) - 1) \cdot \mathsf{w}_{\text{out}}(\pi) + \mathsf{w}(\pi) \le \mathsf{w}(f) \cdot \mathsf{w}(g).$$

Proof. We simulate the querier's strategy for f in the following way: suppose at some point the querier for f had an assignment $\rho\colon [n] \to \{0,1,*\}$. Then the querier for $f \circ g^n$ has assignments $\rho'_1, \dots, \rho'_n\colon [m] \to \{0,1,*\}$ such that whenever $\rho(i) = \alpha \neq *$, ρ'_i corresponds to an α-certificate of g. If the querier for f forgets a variable i, we assign $\rho'_i := *^m$, if they query a variable i, we run the querying strategy for g within ρ'_i and eventually end up with a certificate for g as an assignment ρ'_i, by definition $|(\rho'_i)^{-1}(\{0,1\})| \leq \mathsf{w}_{\mathrm{out}}$. The maximum number of assigned variables when simulating a query to variable i is then $\mathsf{w}(\pi)$ corresponding to assignment ρ'_i, and $(\mathsf{w}(f)-1)\cdot\mathsf{w}_{\mathrm{out}}(\pi)$ corresponding to the remaining assignments. □

Corollary 4.24. *Let f, g be as in Lemma 4.23. Then*

$$\mathsf{w}(f \circ g^n) \leq (\mathsf{w}(f) - 1) \cdot \mathsf{C}(g) + \min\{\mathsf{C}_0(g)\mathsf{C}_1(g), m\}.$$

Proof. Consider the standard strategy used to bound the query complexity of g in terms of its certificate complexity, which can be described as follows. Let us pick an arbitrary 0-certificate ρ for g and query all its variables. If the values match ρ, we return it, otherwise, we claim that the current partial assignment τ contains a variable from every 1-certificate of g. Thus $g|_\tau$ has 1-certificate complexity at most $\mathsf{C}_1(g) - 1$. Let us apply this procedure (query some 0-certificate that agrees with the current partial assignment) until we find a 0-certificate or the certificate complexity shrinks to zero. In the latter case, there exists a 1-certificate which is a restriction of the current assignment.

Using this strategy for g as the game π in Lemma 4.23 (and forgetting all the variables outside the found certificate just before the game ends) we get $\mathsf{w}(\pi) \leq \mathsf{C}_0(g)\mathsf{C}_1(g)$ and $\mathsf{w}_{\mathrm{out}}(\pi) \leq \max\{\mathsf{C}_0(g), \mathsf{C}_1(g)\}$. Since we never have to query a variable twice we can assume $\mathsf{w}(\pi) \leq m$. □

Corollary 4.25. *For g as in Lemma 4.23 we have*

$$\mathsf{w}(\mathrm{AND}_n \circ g^n) \leq (n-1)\mathsf{C}_1(g) + \mathsf{w}(g),$$
$$\mathsf{w}(\mathrm{OR}_n \circ g^n) \leq (n-1)\mathsf{C}_0(g) + \mathsf{w}(g).$$

Proof. Consider a decision tree for AND_n which queries input bits one by one and returns 0 as soon as it encounters a 0-bit. Since in this tree all the assignments have at most a single 0, the cumulative size of the certificates for g that we store until the end of the game is at most $(n-1)\mathsf{C}_1(g)$. This combined with the argument in Lemma 4.23 yields the upper bound. The upper bound for OR_n is analogous. □

Corollary 4.26. $\mathsf{w}(\mathrm{OR}_n \circ \mathrm{AND}_n) \leq 2n - 1$.

We also need to introduce a notion of unambiguous dag-like query complexity, and its one-sided variants. We define $\mathsf{uw}(f)$ to be the dag-like query complexity when querier is limited to strategies where the certificates at terminal states are unambiguous. We define $\mathsf{uw}_0(f)$ (resp. $\mathsf{uw}_1(f)$) to be $\mathsf{uw}(f)$ when the adversary always answers consistently with a 0-input (resp. a 1-input).

Lemma 4.27. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then*

$$\mathsf{uw}_0(\mathrm{AND}_n \circ g^n) \leq \mathsf{uw}_0(g) + (n-1)\mathsf{uw}_1(g),$$
$$\mathsf{uw}_1(\mathrm{AND}_n \circ g^n) \leq n \cdot \mathsf{uw}_1(g).$$

Proof. We use the same composition strategy from Lemma 4.23, taking the same strategy for AND_n as in Corollary 4.26, and an unambiguous game π for g. At the time of evaluating g_i we have that assignments $\rho'_1, \ldots, \rho'_{i-1}$ are all unambiguous 1-certificates, hence assigning up to $(i-1)\mathsf{uw}_1(g)$ many variables, and the bounds follow. The terminal certificates are unambiguous because they are the composition of unambigous certificates. Suppose that it is not the case, so there is some input x that is covered with two certificates c_1, c_2. Certificates c_1, c_2 are the compositions of some unambiguous AND_n certificates a_1, a_2, respectively, with the unambiguous certificates for g. If $a_1 \neq a_2$ then the composed certificates cannot agree. If $a_1 = a_2$ then c_1 and c_2 differ in the part of certificate that correspond to some copy of g. But that is impossible as certificates for g are unambiguous. □

4.4.2 Upper bound in Res

To construct a Res refutation we first reprove the upper bound part of Theorem 4.13—separating unambiguous certificate complexity from randomized query complexity—strengthening it to an upper bound for unambiguous dag-like query complexity in place of unambiguous certificate complexity. We need to make a few minor changes arising from the fact that $\mathsf{w}(\mathrm{AND}_n) = n$ while $\mathsf{C}_0(\mathrm{AND}_n) = 1$, but using the fact that $\mathsf{w}(\mathrm{OR}_n \circ \mathrm{AND}_n) = O(n)$ and not $\Theta(n^2)$ is enough for our purposes.

Lemma 4.28 ([AKK16])**.** *The following are unambiguous certificates for f_{CS}.*

- 0*-certificates: c unambiguous certificates that $f^c = \ell$ together with the contents of Y_ℓ.*
- 1*-certificates: the contents of Y_ℓ together with the positions described in Y_ℓ.*

Lemma 4.29. *Let $f\colon \{0,1\}^N \to \{0,1\}$ be a Boolean function. Then*

$$\mathsf{w}(f_{\mathrm{CS}}) = O(\mathsf{w}(f)\log^2 n), \tag{4.1}$$

$$\mathsf{uw}_0(f_{\mathrm{CS}}) = O(\mathsf{uw}(f)\log^2 n), \tag{4.2}$$

$$\mathsf{uw}_1(f_{\mathrm{CS}}) = O(\mathsf{w}(f)\log^2 n). \tag{4.3}$$

Proof. To prove (4.1) we use the following strategy. We query the c copies of f in parallel to obtain a pointer $\ell \in [2^c]$, using width $O(\mathsf{w}(f)\cdot c)$. We then query all of Y_ℓ and the positions described by Y_ℓ, checking whether they are indeed a set of certificates for f^c, for a total width of $O(\mathsf{w}(f)\cdot c + m)$. Since we choose $c = 10\log N$ and $m = c\cdot \mathsf{C}(f)\cdot \log N = O(\mathsf{w}(f)\log^2 N)$, the total width is $O(\mathsf{w}(f)\cdot c + m) = O(\mathsf{w}(f)\log^2 n)$.

To get (4.3) we follow the same strategy as for (4.1). To prove (4.2) we use unambiguous strategies to query the c copies of f, which increases the width to $O(\mathsf{uw}(f)\cdot c + m)$. By Lemma 4.28 the certificates we obtain in both cases are unambiguous. □

Lemma 4.30. *Let k be a constant, and let f_k be the function defined in Theorem 4.13. Then $\mathsf{uw}(f_k) = \tilde{O}(n^{k+1})$.*

Proof. Let $g_1 = \mathrm{OR}_n \circ \mathrm{AND}_n$ and $g_k = g_1 \circ (g_{k-1})_{\mathrm{CS}}$, so that $f_k = \mathrm{AND}_n \circ (g_k)_{\mathrm{CS}}$.

Let us first show by induction that $\mathsf{w}(g_k) = \tilde{O}(n^k)$. We already proved the base case $\mathsf{w}(g_1) = O(n)$ in Corollary 4.26. Assuming that $\mathsf{w}(g_k) = \tilde{O}(n^k)$, the dag-like query complexity of its cheatsheet version is $\mathsf{w}((g_k)_{\mathrm{CS}}) = \tilde{O}(n^k)$ by Lemma 4.29. Together with the fact that decision-DAG-width is

sub-multiplicative with respect to composition proven in Lemma 4.23, this implies that $\mathsf{w}(g_{k+1}) = \tilde{O}(n^{k+1})$.

Now we show by induction that $\mathsf{uw}(f_k) = \tilde{O}(n^{k+1})$. We already proved the base case $\mathsf{uw}(f_0) = O(n)$ in Lemma 4.27.

For the induction step we have

$$\begin{aligned}\mathsf{uw}_0(f_{k+1}) &= \mathsf{uw}_0(\mathrm{AND} \circ (g_{k+1})_{\mathrm{CS}}) \leq \mathsf{uw}_0((g_{k+1})_{\mathrm{CS}}) + n \cdot \mathsf{uw}_1((g_{k+1})_{\mathrm{CS}}) \\ &= \tilde{O}(\mathsf{uw}(g_{k+1}) + n \cdot \mathsf{w}(g_{k+1})) = \tilde{O}(n \cdot \mathsf{uw}(f_k) + n \cdot \mathsf{w}(g_{k+1})) = \tilde{O}(n^{k+2} + n^{k+2}), \\ \mathsf{uw}_1(f_{k+1}) &= \mathsf{uw}_1(\mathrm{AND} \circ (g_{k+1})_{\mathrm{CS}}) \leq n \cdot \mathsf{uw}_1((g_{k+1})_{\mathrm{CS}}) = \tilde{O}(n^{k+2})\end{aligned}$$

concluding the proof. □

Finally, we can build the Res refutation.

Theorem 4.31. *The formula G_m of Theorem 4.14 has a* Res *refutation of size $2^{\tilde{O}(m)}$.*

Proof. Let F_0 and F_1 be the unambiguous DNFs that we obtain from the leaves of the strategy for f_k of Lemma 4.30. Since we used the same certificates as in Theorem 4.13, these are the same as in Theorem 4.14, and we only need to modify our strategy to output the certificates themselves rather than 0 and 1 in order to obtain a strategy for S, the falsified clause search problem of $\overline{F_0 \vee F_1}$, of query complexity $\mathsf{w}(S) = \tilde{O}(n^{k+1})$. By sub-multiplicativity of query complexity we have that the dag-like query complexity of S', the falsified clause search problem of G_m, is also $\mathsf{w}(S') = \tilde{O}(n^{k+1}) = \tilde{O}(m)$. From the equivalence between dag-like query complexity and width we have that G_m has a Res refutation of width $\tilde{O}(m)$, which implies a size upper bound of $|\mathrm{vars}(G_m)|^{\tilde{O}(m)} = 2^{\tilde{O}(m)}$. □

4.4.3 Upper bound in NS

Given a clause $C = \bigvee_{i \in P} x_i \vee \bigvee_{i \in N} \overline{x_i}$, let $p_C = \prod_{i \in P}(1 - x_i) \cdot \prod_{i \in N} x_i$ be the polynomial whose roots are the satisfying assignments of C. Recall that a NS certificate that a set of polynomials $\{p_i\}$ has no common root is a set of polynomials $\{q_i\}$ such that $\sum p_i q_i \equiv 1$, and the degree of a certificate is $\max_i \deg(p_i q_i)$. A NS refutation of a CNF F is a NS certificate for $\{p_C \mid C \in F\} \cup \{x_i^2 - x_i\}$.

It turns out that NS simulates Hitting with respect to degree.

Proposition 4.32. NS *degree is at most* Hitting *width.*

Proof. Let F be a CNF and H be a Hitting refutation of F. Observe that the polynomial $\sum_{C \in F} p_C$ is identical to 1 because H is a partition of the hypercube. Fix for each clause $C \in H$ a clause $C' \in F$ such that $C' \subseteq C$. Let $q_D = \sum_{\{C \in H \mid C' = D\}} p_{C \setminus C'}$. We claim that the set of polynomials $\{q_D \mid D \in F\}$ is a NS certificate for F, and indeed we have $\sum_{D \in F} p_D q_D = \sum_{C \in F} p_C = 1$. Furthermore, the degree of $p_D q_D$ is bounded by the degree of p_C, which equals the width of H. □

When measuring the size of a NS refutation it is more appropriate to consider a definition that allows us to introduce dual variables $\bar{x} = 1 - x$ [ABSRW02] resulting in a new system NSR [dRLNS21], since otherwise a formula containing a wide clause with many positive literals would already require exponential size when translated to polynomials. We discuss this system in Sect. 3. Moreover, we discuss *succinct* NSR proofs that contain only side polynomials for the input axioms and not for $x^2 - x = 0$ or $x + \bar{x} - 1 = 0$. In fact, Prop. 4.32 already shows that succinct NSR polynomially simulates Hitting with respect to size.

5 Odd Hitting

As mentioned in Sect. 3.3, Odd Hitting proofs can be verified similarly to NSR proofs over GF(2). The two proof systems are similar: an NSR proof is a Nullstellensatz proof from pseudomonomial equations $m_i = 0$ that are translations of the input clauses C_i, the Boolean equations $x_j^2 - x_j = 0$ for every variable x_j, and the axioms $\bar{x}_j + x_j - 1 = 0$ defining the dual variables. The side polynomials over GF(2) are just sums of some monomials q_{ik}, $r_{j\ell}$, and s_{jt} such that

$$\sum_i m_i \sum_k q_{ik} + \sum_j (x_j^2 - x_j) \sum_\ell r_{j\ell} + \sum_j (\bar{x}_j + x_j - 1) \sum_t s_{jt} \equiv 1.$$

On the other hand, an Odd Hitting proof also can be written using polynomials over GF(2) as a sum of pseudomonomial equations $m_i = 0$ multiplied by sums of monomials (every such product expresses a weakening of C_i)

$$\sum_i m_i \sum_k q_{ik} \equiv 1 \pmod{\langle x_j^2 - x_j, \bar{x}_j + x_j - 1 \rangle_j}.$$

The difference is that in Odd Hitting the equivalence is only modulo the ideal, thus Odd Hitting gives *succinct* NSR proofs, as in Sect. 3.3. In the opposite direction, every NSR proof after cutting the degrees and dropping $r_{j\ell}$'s and s_{jt}'s provides a valid Odd Hitting proof.

Like NS over GF(2), Odd Hitting can efficiently refute Tseitin formulas modulo 2 (see Def. 4.15), which require exponential-size resolution proofs [Urq87].

Proposition 5.1. *For any constant-degree graph $G = (V, E)$ and 0/1-vector c,* Odd Hitting *has a polynomial-size refutation of $T_{G,c}$.*

Proof. Each truth assignment falsifies an odd number of vertex constraints. For each constraint, it falsifies exactly one of the $2^{\deg v - 1}$ clauses. Thus the total number of falsified clauses is odd, and $T_{G,c}$ itself is an unsatisfiable odd-hitting formula. □

A separation between Odd Hitting and NS without dual variables follows immediately from the separation between NSR and NS of de Rezende et al [dRLNS21].

In the opposite direction, there are formulas that require exponentially larger proofs in Odd Hitting than in Res. Dmitry Sokolov [private communication] suggested that the well-known technique of xorification can produce an exponential separation between the size of Res and NSR proofs from the bounds of [BCIP02]:

Theorem 5.2 ([BCIP02])**.** *There exists a family of formulas that have* Res *proofs of constant width and require* NS *degree $\Omega(n/\log n)$.*

We notice that this technique is still viable for succinct NSR proofs, and hence Odd Hitting. In the following lemma we apply xorification and the random restriction technique of Alekhnovich and Razborov (see [BS09]) to get the separation.

Lemma 5.3. *Let F be a CNF formula that requires degree d to refute in* NS *over a field $\mathbb{F}$. Then $F \circ (\oplus_2)^n$ requires size $2^{\Omega(d)}$ to refute in succinct* NSR *over $\mathbb{F}$.*

Proof. Denote by $y_{i,0}$ and $y_{i,1}$ the variables appearing in $F \circ (\oplus_2)^n$ as $y_{i,0} \oplus y_{i,1}$ instead of the variable y_i in F. Let $\sum f_i g_i \equiv 1$ be a succinct NSR refutation of $F \circ (\oplus_2)^n$, and let s be the number of monomials in the refutation, which we assume for the sake of contradiction to be less than $(4/3)^d$. Let $\mathcal{D}$ be the probability distribution over random restrictions where for every pair of variables $y_{i,0}, y_{i,1}$ we sample $j \in \{0,1\}$ and $b \in \{0,1\}$ uniformly and we assign $y_{i,j} = b$ while leaving $y_{i,1-j}$ unassigned. Observe that for $\rho \sim \mathcal{D}$ we have $\{f_i\restriction_\rho \mid f_i \in F \circ (\oplus_2)^n\} = F$ and that ρ respects Boolean axioms, therefore $\sum f_i g_i\restriction_\rho = \sum f_i\restriction_\rho \cdot g_i\restriction_\rho$ is a NSR refutation of F.

Since for any monomial m we have

$$\Pr_{\rho\sim\mathcal{D}}[\deg(m\restriction_\rho) \geq d] \leq \Pr_{\rho\sim\mathcal{D}}[\deg(m) \geq d \text{ and } m\restriction_\rho \neq 0] \leq (3/4)^d,$$

the union bound gives that

$$\Pr_{\rho\sim\mathcal{D}}[\deg(f_i g_i\restriction_\rho) \geq d] \leq s \Pr_{\rho\sim\mathcal{D}}[\deg(m\restriction_\rho) \geq d] \leq s \cdot (3/4)^d < 1.$$

Therefore there exists a restriction such that $\deg(f_i g_i\restriction_\rho) < d$, contradicting our hypothesis that F requires NS degree d. □

Combining xorification with a lower bound on the degree of pebbling formulas we obtain a separation between Odd Hitting and Res.

Corollary 5.4. *There exists a family of formulas that have* Res *proofs of polynomial size and require* Odd Hitting *proofs of size* $2^{\Omega(n/\log n)}$.

Proof. By Theorem 5.2 and Lemma 5.3. □

6 Hitting($\oplus$)

Hitting($\oplus$) extends Hitting to formulas that work with linear equations modulo two. We know from Cor. 4.19 that Tseitin formulas separate Hitting($\oplus$) from Hitting and Res.

In this section, we show that perfect matching formulas (that have polynomial-size CP proofs) require exponential-size Hitting($\oplus$) refutations. In order to do this, we lift them using (binary) xorification and then reduce the question to the known communication complexity lower bound for set disjointness.

6.1 Evidence against quasi-polynomial simulation by tl-Res($\oplus$)

Theorem 4.2 suggests that there might be a quasi-polynomial simulation of Hitting($\oplus$) by tl-Res($\oplus$), which would imply that all exponential-size tl-Res($\oplus$) lower bounds imply exponential-size lower bounds for Hitting($\oplus$). Recall that the crux of the proof of Theorem 4.2 is that we can always find a literal that appears in an $\Omega(1/\log n)$ fraction of the clauses of a hitting formula, and this provides an efficient splitting of the problem. The following proposition shows that the analogue of this does not exist for Hitting($\oplus$), which is a piece of evidence that there is no similar simulation in the case of Hitting($\oplus$). Namely, we prove that, contrary to the case of Hitting, there are formulas that cannot be split so efficiently.

We use the construction that appears in [BFIK23, Section 5.1] for a different purpose.

Proposition 6.1. *There exists an unsatisfiable hitting($\oplus$) formula over $2t-1$ variables consisting of 2^t $(t-1)$-dimensional affine subspaces such that for every set $S \subseteq [2t-1]$ there is at most one affine space in the formula contained in the codimension-1 affine subspace $\left\{x \in \{0,1\}^{2t-1} \mid \bigoplus_{i \in S} x_i = \alpha\right\}$ for $\alpha \in \{0,1\}$.*

Proof. Let $\mathbb{F}_{2^t}$ be the finite field of order 2^t, which we can identify with polynomials of degree smaller than t over $\mathbb{F}_2$. Given such a polynomial $b_0 + b_1 x + \cdots + b_{t-1}x^{t-1}$, we can identify it with the string $b_0 b_1 \dots b_{t-1}$.

For every $\alpha \in \mathbb{F}_{2^t}$, we define

$$V_\alpha := \{(c, \alpha c) \mid c = c_0 + c_1 x + \cdots + c_{t-1}x^{t-1} \in \mathbb{F}_{2^t}, c_0 = 1\}.$$

We identify V_α with a subset of $\{0,1\}^{2t-1}$ by converting both parts to strings, concatenating them, and removing the initial 1.

Claim 6.2. *Sets $\{V_\alpha\}_{\alpha \in \mathbb{F}_{2^t}}$ form an unsatisfiable hitting($\oplus$) formula.*

Proof. Define

$$U_\alpha := \{(d, \alpha d) \mid d \in \mathbb{F}_{2^t}\}.$$

It is easy to see that U_α is a vector subspace of $\mathbb{F}_2^{2t}$ as $\eta_1(d_1, \alpha d_1) + \eta_2(d_2, \alpha d_2) = ((\eta_1 d_1 + \eta_2 d_2), \alpha(\eta_1 d_1 + \eta_2 d_2)) \in U_\alpha$. If $(x, y) \in U_\alpha \cap U_\beta$ for $\alpha \neq \beta$ then $y = \alpha x = \beta x$, and so $x = y = 0$. This shows that U_α, U_β are trivially intersecting subspaces. Also, if $x \neq 0$ then (x, y) is covered by $U_{y/x}$. Together with $U_\infty = \{(0, d) \mid d \in \mathbb{F}_{2^t}\}$, we obtain the object called *standard Desarguesian spread* of $\mathbb{F}_{2^t}^2 \setminus \{(0,0)\}$ (which we think of as a projective plane).

We obtain V_α by restricting U_α to those d satisfying $d_0 = 1$. Thus $V_\alpha \cap V_\beta = \emptyset$ and the V_α's cover all points of the form (x, y) where $x_0 = 1$. Moreover, V_α is an affine subspace since it is an intersection of U_α and an affine space $\{(c, d) \mid c_0 = 1\}$ projected on the last $2t-1$ coordinates. We conclude that the V_α's constitute a hitting($\oplus$) formula. □

Now we proceed to prove the uniqueness of an affine space V_α contained in an arbitrary codimension-1 affine subspace of $\{0,1\}^{2t-1}$. We can think of the entire domain as $V = \{(c, d) : c, d \in \mathbb{F}_{2^t}, c_0 = 1\}$. The subspaces are $V_\alpha = \{(c, d) : d = \alpha c\}$. Any affine form on the encoding of length $2t-1$ (obtained by removing the initial 1) corresponds to a linear form on the "homogenized" encoding of length $2t$. Moreover, we can express each such form as $(\gamma c + \delta d)_0$, where $\gamma, \delta \in \mathbb{F}_{2^t}$, and we take the coefficient of x^0 in the result.

A subspace V_α satisfies the constraint $(\gamma c + \delta d)_0 = 0$ if $((\gamma + \alpha\delta)c)_0 = 0$ for all $c \in \mathbb{F}_{2^t}$ such that $c_0 = 1$. Let $\eta = \gamma + \alpha\delta$. Choosing $c = 1$, we see that $\eta_0 = 0$. If $d \in \mathbb{F}_{2^t}$ satisfies $d_0 = 0$ then $(\eta d)_0 = (\eta(d+1))_0 + \eta_0 = 0$. Therefore $(\eta c)_0 = 0$ for all $c \in \mathbb{F}_{2^t}$. In particular, if $\eta = 0$, since otherwise we obtain a contradiction by choosing $c = \eta^{-1}$.

There are now two cases. If $\delta = 0$ then $\eta = 0$ implies $\gamma = 0$. If $\delta \neq 0$ then $\eta = 0$ implies $\alpha = -\gamma/\delta$. We conclude that an affine equation that involves only the first half holds for no subspace (unless it is the trivial $0 = 0$), and any other affine equation holds for precisely one subspace. □

6.2 Communication simulation of Hitting($\oplus$)

Consider the following complexity measure on relations introduced by Jain and Klauck [JK10, Definition 22]. Let $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$, let $\mathsf{rect}(\mathcal{X}, \mathcal{Y})$ (or just rect) be the set of subrectangles of

$\mathcal{X} \times \mathcal{Y}$ (that is, $\{A \times B \mid A \subseteq \mathcal{X},\ B \subseteq \mathcal{Y}\}$), let $\mathrm{supp}(S) = \{(x,y) \in \mathcal{X} \times \mathcal{Y} \mid \exists o \in \mathcal{O}\colon (x,y,o) \in S\}$. Then ε-partition bound of S is denoted by $\mathsf{prt}_\varepsilon(S)$ and defined by the value of the following linear program:

$$\begin{aligned} & \sum_{o \in \mathcal{O}} \sum_{R \in \mathsf{rect}} w_{o,R} \to \min \\ \forall (x,y) \in \mathrm{supp}(S), & \sum_{o:(x,y,o)\in S} \sum_{(x,y)\in R \in \mathsf{rect}} w_{o,R} \geq 1-\varepsilon \\ \forall o, R \in \mathcal{O} \times \mathsf{rect}, & \ w_{o,R} \geq 0 \\ \forall (x,y) \in \mathcal{X} \times \mathcal{Y}, & \sum_{o \in \mathcal{O}} \sum_{(x,y)\in R \in \mathsf{rect}} w_{o,R} = 1. \end{aligned}$$

That is, we put weights $w_{o,R}$ on every answer o and rectangle R so that for every question (x,y) the total weight is 1 for all rectangles where (x,y) participates and for all answers, and for every question that does have an answer, the weight $1-\varepsilon$ is concentrated on the correct answer(s).

Remark 6.3. *Note that if we aggregate the "answers" $o \in \mathcal{O}$ into larger containers $I(o') \in \mathcal{O}'$ so that $\mathcal{O} = \bigsqcup_{o' \in \mathcal{O}'} I(o')$ and $(x,y,o') \in \mathcal{O}' \Leftrightarrow \exists o \in I(o')\ (x,y,o) \in \mathcal{O}$, we can only decrease the value of prt_ε.*

This measure lower bounds communication complexity in the following way:

Theorem 6.4 ([JK10])**.** *ε-error randomized communication complexity of a relation S is at least $\log \mathsf{prt}_\varepsilon(S)$.*

The classical communication complexity lower bound for set disjointness works for this measure as well. The set disjointness relation $\textsc{Disj}_n\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined as $\textsc{Disj}_n(x,y) = \bigwedge_{i\in[n]} \neg(x_i \wedge y_i)$.

Theorem 6.5 ([Raz92, JK10])**.** $\log \mathsf{prt}_\varepsilon(\textsc{Disj}_n) = \Omega(n)$.

We prove that $\textsc{Hitting}(\oplus)$ size lower bounds can be deduced from lower bounds on prt_ε. Recall that $\textsc{Hitting}(\oplus)$ can be considered not just a proof system for formulas in CNFs, but as a proof system for sets of disjunctions of affine equations (sets of affine subspaces), which we will still call *clauses*. We will use it in our proof (though the final bound will be for a formula in CNF).

Lemma 6.6. *Let $F = \bigwedge_{i\in[m]} C_i$ be a set of affine subspaces over n variables having a $\textsc{Hitting}(\oplus)$ refutation of size s. Let $X \sqcup Y = [n]$ be an arbitrary partition of variables of F and let the clause-search relation associated with F be*

$$S := \{(x,y,j) \in \{0,1\}^X \times \{0,1\}^Y \times [m] \mid C_j(x,y) = 0\}.$$

Then for any constant $\varepsilon \in (0,1)$, $\mathsf{prt}_\varepsilon(S) = O(s^2)$ (the constant in O may depend on ε).

Proof. Let $A_1, \ldots, A_s$ be the affine subspaces forming the partition of $\{0,1\}^n$ corresponding to the $\textsc{Hitting}(\oplus)$ refutation of F, i.e. for every A_j there exists $C_{h(j)} \in F$ such that for every $x \in A_j$, $C_{h(j)}(x) = 0$.

Now let us define the values of $w_{i,R}$ for $i \in [m]$ and a rectangle $R \in \mathsf{rect}(\{0,1\}^X, \{0,1\}^Y)$ corresponding to $\mathsf{prt}_\varepsilon(S)$.

For each $j \in [s]$ we define a part of the weights induced by A_j. Let $A_j = \{(x,y) \in \{0,1\}^X \times \{0,1\}^Y \mid M^X x + M^Y y = a\}$ where $a \in \{0,1\}^k$, $M^X \in \{0,1\}^{[k]\times X}$, $M^Y \in \{0,1\}^{[k]\times Y}$. Consider a uniformly distributed matrix $V \in \{0,1\}^{t\times k}$, then if $(x,y) \in A_j$,

$$\Pr_V\left[VM^X x + VM^Y y = Va\right] = 1,$$

and if $(x,y) \notin A_j$,

$$\Pr_V\left[VM^X x + VM^Y y = Va\right] = \frac{1}{2^t}.$$

For matrix V and $u, v \in \{0,1\}^t$, define the rectangle $R^V_{u,v} := \{(x,y) \mid VM^X x = u, VM^Y y = v\}$. We can now define weights $w^{(j)}_{i,R}$ (recall that $h(j)$ is the index of the clause falsified in A_j):

$$w^{(j)}_{i,R} := \begin{cases} 2^{-kt} & \text{if } i = h(j) \text{ and } R = R^V_{u,u+Va} \text{ for some } u \in \{0,1\}^t \text{ and } V \in \{0,1\}^{k\times t}, \\ 0 & \text{otherwise.} \end{cases}$$

(Note that the points of $R_{u,u+Va}$ are guaranteed to fulfil the condition $VM^X x + VM^Y y = Va$ above.) Then

$$\sum_{\substack{i\in[m]\\ R\ni(x,y)}} w^{(j)}_{i,R} = \sum_{R\ni(x,y)} w^{(j)}_{h(j),R} = \mathbf{E}_V w^{(j)}_{h(j),\,R^V_{VM^X x, VM^Y y}} = \begin{cases} 1 & \text{if } (x,y) \in A_j, \\ 2^{-t} & \text{otherwise.} \end{cases}$$

Now fix $t := \lceil \log_2(2s/\varepsilon) \rceil$, so $2^{-t} \le \varepsilon/2s$, and set $w_{i,R} := \frac{1}{1+(s-1)2^{-t}} \sum_{j\in[s]} w^{(j)}_{i,R}$. Then

$$\sum_{\substack{i\in[m]\\ R\ni(x,y)}} w_{i,R} = \frac{1}{1+(s-1)2^{-t}} \left(\overbrace{\sum_{\substack{i\in[m]\\ R\ni(x,y)}} w^{(j(x,y))}_{i,R}}^{1} + \sum_{j\in[s]\setminus\{j(x,y)\}} \overbrace{\sum_{\substack{i\in[m]\\ R\ni(x,y)}} w^{(j)}_{i,R}}^{2^{-t}} \right) = 1,$$

where $j(x,y) \in [s]$ is such that $A_{j(x,y)} \ni (x,y)$. Let $i(x,y)$ be the index of some fixed clause falsified in $A_{j(x,y)}$. Let us check the second condition on the weights:

$$\sum_{\substack{(x,y,i)\in S\\ R\ni(x,y)}} w_{i,R} \ge \sum_{R\ni(x,y)} w_{i(x,y),R} \ge \frac{1}{1+(s-1)2^{-t}} \sum_{R\ni(x,y)} w^{(j(x,y))}_{i(x,y),R} = \frac{1}{1+(s-1)2^{-t}} \ge \frac{1}{1+\varepsilon/2} \ge 1-\varepsilon.$$

The objective function of the linear program is

$$\mathsf{prt}_\varepsilon(S) = \sum_{\substack{i\in[m]\\ R\in\mathsf{rect}}} w_{i,R} \le \sum_{\substack{i\in[m],\ j\in[s]\\ R\in\mathsf{rect}}} w^{(j)}_{i,R} \;\le\; s2^t \le 4s^2/\varepsilon.$$

The second inequality holds as by definition of $w^{(j)}_{i,R}$, for every $j \in [s]$, it is non-negative in the 2^{t+kt} cases, and equals 2^{-kt} in each of them. □

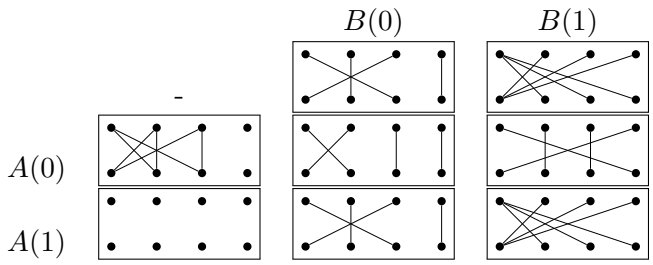

Figure 2: The graphs $A(0), A(1), B(0)$, and $B(1)$ and their pairwise symmetric differences. Only $A(1) \oplus B(1)$ is not a matching.

6.3 Lower bounds on prt_ε

To get a lower bound on the size of HITTING($\oplus$) refutations using Lemma 6.6, we need to show a lower bound on prt_ε for clause-search relations. The idea is to utilize known reductions from set disjointness, $\textsc{Disj}_n$, that have been developed for randomized communication complexity.

The perfect matching principle PM_G is a formula in CNF stating that a given subset of $E(G)$ is a perfect matching. It has a variable z_e for every edge $e \in E(G)$, and for every $v \in V(G)$ it contains

- the clauses $\bar{z}_f \vee \bar{z}_g$ for every pair of edges f, g adjacent to v, and
- the clause $\bigvee_{e \ni v} z_e$.

Define a set of clauses (with affine equations) $\mathrm{PM}_G^{\oplus}$ by replacing every variable z_e in the formula PM_G by $x_e \oplus y_e$ (here x_e and y_e are variables of $\mathrm{PM}_G^{\oplus}$).

For a graph G, the relation $\textsc{Search-PM}_G^{\oplus} \subseteq \{0,1\}^{E(G)} \times \{0,1\}^{E(G)} \times V(G)$ for $\mathrm{PM}_G^{\oplus}$ is defined as $\textsc{Search-PM}_G^{\oplus} := \{(E_1, E_2, v) \mid \text{degree of } v \text{ in } (V(G), E_1 \oplus E_2) \text{ is not } 1\}$. In other words, the $\textsc{Search-PM}_G^{\oplus}$ problem asks to find a witness that the symmetric difference of the input sets is not a perfect matching. Strictly speaking, it does not ask for a specific clause; however, Remark 6.3 explains that aggregating clauses into the set of clauses related to a single vertex is enough.

In the following theorem, we construct a communication-complexity reduction of $\textsc{Disj}_n$ to $\textsc{Search-PM}_G^{\oplus}$ for a specific graph G by providing two functions $f_{\mathcal{A}}$, $f_{\mathcal{B}}$ for Alice and Bob and another function g for recovering the result after $\textsc{Search-PM}_G^{\oplus}$ finds a failing vertex in the graph $(V, f_{\mathcal{A}}(\ldots) \oplus f_{\mathcal{B}}(\ldots))$. We later use this reduction in order to prove bounds on prt.

Theorem 6.7 (variation on [IR21, Theorem 16])**.** *Let $G = K_{40n+1,40n+3}$. There exist a finite set R, functions $f_{\mathcal{A}}, f_{\mathcal{B}} \colon \{0,1\}^n \times R \to \{0,1\}^{E(G)}$ and a function $g \colon V(G) \times R \to \{0,1\}$ such that for every $a, b \in \{0,1\}^n$ and for every family of random variables $\{\mathbf{p}_{x,y}\}_{x,y \in \{0,1\}^{E(G)}}$ over $V(G)$,*

$$\Pr_{r \leftarrow R}\left[g(\mathbf{p}_{f_{\mathcal{A}}(a,r), f_{\mathcal{B}}(b,r)}, r) \neq \textsc{Disj}_n(a,b) \mid (f_{\mathcal{A}}(a,r), f_{\mathcal{B}}(b,r), \mathbf{p}_{f_{\mathcal{A}}(a,r), f_{\mathcal{B}}(b,r)}) \in \textsc{Search-PM}_G^{\oplus}\right] \leq \frac{1}{10},$$

where r is uniformly distributed over R.

Proof. Itsykson and Riazanov [IR21] described four graphs $A(0), A(1), B(0), B(1)$, all of them subgraphs of $K_{4,4}$, with the following two properties (see Fig. 2):

- $A(a) \oplus B(b)$ is a perfect matching consisting of four edges for all $(a,b) \neq (1,1)$.

- $A(1) \oplus B(1)$ is the disjoint sum of $K_{1,3}$ and $K_{3,1}$, where in both cases the vertex numbered 1 on one side is connected to the vertices numbered $2, 3, 4$ on the other side.

Using this gadget, we define $R, f_{\mathcal{A}}, f_{\mathcal{B}}, g$:

- $R = S_{40n+1} \times S_{40n+3} \times \{0,1\}^{E(G)}$, where S_m is the symmetric group on m elements. In what follows, we denote an element of R by (π_1, π_2, H).
- $f_{\mathcal{A}}(a, \pi_1, \pi_2, H)$: start with a graph G_A that is a disjoint union of 10 copies of $A(a_1), \ldots, A(a_n)$ (total of $10n$ subgraphs) and a copy of $K_{1,3}$ with its vertex on the left numbered $40n+1$. Compute the symmetric difference with H, then permute the vertices on the left using π_1 and the vertices on the right using π_2.
- $f_{\mathcal{B}}(b, \pi_1, \pi_2, H)$: start with a graph G_B that is a disjoint union of 10 copies of $B(b_1), \ldots, B(b_n)$ and a copy of the complement of $K_{1,3}$ (that is, the empty graph 1×3). As for $f_{\mathcal{A}}$, we compute the symmetric difference with H and then apply π_1, π_2.
- $g(v, \pi_1, \pi_2, H)$: return 1 if v is vertex $\pi_1(40n+1)$ on the left.

If $\textsc{Disj}_n(a, b) = 1$ then the graph $f_{\mathcal{A}}(a, r) \oplus f_{\mathcal{B}}(b, r)$ consists of a matching of $40n$ edges together with a copy of $K_{1,3}$, after shuffling both sides according to r. The only vertex of degree other than 1 is $\pi_1(40n+1)$, and so if $\mathbf{p}$ returns a vertex in $\textsc{Search-PM}^{\oplus}$, it must return $\pi_1(40n+1)$. Consequently, g always returns 1. In this case, the probability in the statement of the theorem is zero.

Now suppose that $\textsc{Disj}_n(a, b) = 0$, say $a_1 = b_1 = 1$. Let $G_{\oplus} = G_A \oplus G_B$, where G_A, G_B are the graphs in the definitions of $f_{\mathcal{A}}, f_{\mathcal{B}}$, and define the permutation that touches only vertices in the ten copies of $A(1) \oplus B(1)$ and the vertices of the final $K_{1,3}$.

$$\sigma_1 = (1\ 5\ 9\ \cdots\ 37\ 40n+1),$$
$$\sigma_2 = (2\ 6\ 10\ \cdots\ 38\ 40n+1)\,(3\ 7\ 11\ \cdots\ 39\ 40n+2)\,(4\ 8\ 12\ \cdots\ 40\ 40n+3).$$

Thus $G_{\oplus}$ is invariant under permuting vertices on the left using σ_1 and those on the right using σ_2 (simultaneously).

For fixed π_1, π_2, the distribution of $(f_{\mathcal{A}}(a, r), f_{\mathcal{B}}(b, r))$ (whose randomness comes only from H) can be described as follows: sample a pair (H_A, H_B) whose symmetric difference is $G_{\oplus}$, and then permute the result according to π_1, π_2, an operation we denote by superscripting the pair. Therefore

$$\Pr_{r \leftarrow R}\left[g(\mathbf{p}_{f_{\mathcal{A}}(a,r), f_{\mathcal{B}}(b,r)}, r) \neq \textsc{Disj}_n(a,b) \mid (f_{\mathcal{A}}(a,r), f_{\mathcal{B}}(b,r), \mathbf{p}_{f_{\mathcal{A}}(a,r), f_{\mathcal{B}}(b,r)}) \in \textsc{Search-PM}_G^{\oplus}\right] =$$
$$\Pr_{\substack{\pi_1,\pi_2 \\ H_A \oplus H_B = G_{\oplus}}}\left[\mathbf{p}_{H_A^{\pi_1,\pi_2}, H_B^{\pi_1,\pi_2}} = \pi_1(40n+1) \mid (H_A^{\pi_1,\pi_2}, H_B^{\pi_1,\pi_2}, \mathbf{p}_{H_A^{\pi_1,\pi_2}, H_B^{\pi_1,\pi_2}}) \in \textsc{Search-PM}_G^{\oplus}\right].$$

Denote $S(\pi_1, \pi_2) := \left(H_A^{\pi_1,\pi_2}, H_B^{\pi_1,\pi_2}, \mathbf{p}_{H_A^{\pi_1,\pi_2}, H_B^{\pi_1,\pi_2}}\right)$. Let $\mathbf{t}$ be chosen uniformly at random from $\{0, \ldots, 10\}$. Since (π_1, π_2) and $(\pi_1 \sigma_1^{\mathbf{t}}, \pi_2 \sigma_2^{\mathbf{t}})$ have the same distribution, the probability above is

equal to

$$\Pr_{\substack{\pi_1,\pi_2\\ H_A\oplus H_B=G_\oplus}}\left[\mathbf{p}_{H_A^{\pi_1\sigma_1^{\mathbf{t}},\pi_2\sigma_2^{\mathbf{t}}},H_B^{\pi_1\sigma_1^{\mathbf{t}},\pi_2\sigma_2^{\mathbf{t}}}}=\pi_1\big(\sigma_1^{\mathbf{t}}(40n+1)\big)\;\middle|\; S(\pi_1\sigma_1^{\mathbf{t}},\pi_2\sigma_2^{\mathbf{t}})\in \textsc{Search-PM}_G^{\oplus}\right]=$$

$$\Pr_{\substack{\pi_1,\pi_2\\ H_A\oplus H_B=G_\oplus^{\sigma_1^{\mathbf{t}},\sigma_2^{\mathbf{t}}}}}\left[\mathbf{p}_{H_A^{\pi_1,\pi_2},H_B^{\pi_1,\pi_2}}=\pi_1(\sigma_1^{\mathbf{t}}(40n+1))\;\middle|\; S(\pi_1,\pi_2)\in \textsc{Search-PM}_G^{\oplus}\right]=$$

$$\Pr_{\substack{\pi_1,\pi_2\\ H_A\oplus H_B=G_\oplus}}\left[\mathbf{p}_{H_A^{\pi_1,\pi_2},H_B^{\pi_1,\pi_2}}=\pi_1(\sigma_1^{\mathbf{t}}(40n+1))\;\middle|\; S(\pi_1,\pi_2)\in \textsc{Search-PM}_G^{\oplus}\right]\le\frac{1}{11},$$

since $\sigma_1^{\mathbf{t}}(40n+1)$ is uniformly distributed over $\{1,5,9,\dots,37,40n+1\}$. □

In the next theorem we use the reduction from Theorem 6.7 in order to transform prt bounds for $\textsc{Search-PM}_G^{\oplus}$ into bounds for the disjointness.

Theorem 6.8. *For the graph G from Theorem 6.7 and small enough constant ε,*

$$\mathsf{prt}_{1/3}(\textsc{Disj}_n)\le\mathsf{prt}_{\varepsilon}(\textsc{Search-PM}_G^{\oplus}).$$

Proof. Let $w_{v,R}$ for $v\in V(G)$ and $R\in\mathsf{rect}(\{0,1\}^{E(G)},\{0,1\}^{E(G)})$ be optimal weights in the ε-partition bound for $\textsc{Search-PM}_G^{\oplus}$. Let $f_{\mathcal{A}},f_{\mathcal{B}},g$ be the reduction functions from Theorem 6.7. Let $f_{\mathcal{A}}^r(x):=f_{\mathcal{A}}(x,r)$ and $f_{\mathcal{B}}^r(x):=f_{\mathcal{B}}(x,r)$. We claim that the following weights $w'_{\alpha,R}$ for $\alpha\in\{0,1\}$ and $R\in\mathsf{rect}(\{0,1\}^n,\{0,1\}^n)$ yield the upper bound on $\mathsf{prt}_{1/3}(\textsc{Disj}_n)$:

$$w'_{\alpha,X\times Y}:=\mathbf{E}_r\sum_{\substack{A,B:\\ \bigcup(f_{\mathcal{A}}^r)^{-1}(A)=X\\ \bigcup(f_{\mathcal{B}}^r)^{-1}(B)=Y}}\;\sum_{\substack{v:\\ g(v,r)=\alpha}}w_{v,A\times B}.$$

Let us verify the properties of w'. First, we check that the sum of weights of rectangles covering a point is exactly 1:

$$\sum_{\substack{\alpha\in\{0,1\}\\ X\times Y\ni(x,y)}}w'_{\alpha,X\times Y}=\mathbf{E}_r\sum_{v\in V(G)}\;\sum_{\substack{A,B:\\ A\ni f_{\mathcal{A}}^r(x)\\ B\ni f_{\mathcal{B}}^r(y)}}w_{v,A\times B}=1.$$

Now we check that the sum of weights covering a point (x,y) and labeled with the correct answer $\alpha=\textsc{Disj}_n(x,y)$ is at least $2/3$:

$$\sum_{X\times Y\ni(x,y)}w'_{\alpha,X\times Y}=\mathbf{E}_s\sum_{\substack{A\ni f_{\mathcal{A}}^r(x)\\ B\ni f_{\mathcal{B}}^r(y)\\ v\colon g(v,r)=\alpha}}w_{v,A\times B}\ge\mathbf{E}_s\sum_{\substack{A\ni f_{\mathcal{A}}^r(x),\ B\ni f_{\mathcal{B}}^r(y)\\ v\colon g(v,r)=\alpha\\ (f_{\mathcal{A}}^r(x),f_{\mathcal{B}}^r(y),v)\in\textsc{Search-PM}_G^{\oplus}}}w_{v,A\times B}$$

$$=\mathbf{E}_s\sum_{\substack{A\ni f_{\mathcal{A}}^r(x),\ B\ni f_{\mathcal{B}}^r(y)\\ (f_{\mathcal{A}}^r(x),f_{\mathcal{B}}^r(y),v)\in\textsc{Search-PM}_G^{\oplus}}}w_{v,A\times B}-\mathbf{E}_s\sum_{\substack{A\ni f_{\mathcal{A}}^r(x),\ B\ni f_{\mathcal{B}}^r(y)\\ v\colon g(v,r)\neq\alpha\\ (f_{\mathcal{A}}^r(x),f_{\mathcal{B}}^r(y),v)\in\textsc{Search-PM}_G^{\oplus}}}w_{v,A\times B}$$

$$\ge 1-\varepsilon-\underbrace{\mathbf{E}_s\sum_{\substack{v\colon g(v,r)\neq\alpha\\ (f_{\mathcal{A}}^r(x),f_{\mathcal{B}}^r(y),v)\in\textsc{Search-PM}_G^{\oplus}}}\;\sum_{\substack{A\ni f_{\mathcal{A}}^r(x)\\ B\ni f_{\mathcal{B}}^r(y)}}w_{v,A\times B}}_{r}.$$

Now let us define a family of random variables $\mathbf{p}_{a,b}$ for $a, b \in \{0,1\}^{E(G)}$ over $V(G)$ such that $\Pr[\mathbf{p}_{a,b} = v] = \sum_{A\times B \ni (a,b)} w_{v,A\times B}$ and rewrite

$$\mathbf{E}_r \sum_{\substack{v\colon g(v,r)\neq\alpha \\ (f^r_{\mathcal{A}}(x), f^r_{\mathcal{B}}(y), v)\in \textsc{Search-PM}^{\oplus}_G}} \Pr[\mathbf{p}_{f^r_{\mathcal{A}}(x), f^r_{\mathcal{B}}(y)} = v]$$
$$= \Pr_r\left[g(\mathbf{p}_{f^r_{\mathcal{A}}(x), f^r_{\mathcal{B}}(y)}, r) \neq \alpha \wedge (x, y, \mathbf{p}_{f^r_{\mathcal{A}}(x), f^r_{\mathcal{B}}(y)}) \in \textsc{Search-PM}^{\oplus}_G\right] \leq \frac{1}{10}.$$

The last inequality holds by Theorem 6.7, so for $\varepsilon < \frac{9}{10} - \frac{2}{3}$ the condition $\sum_{X\times Y \ni (x,y)} w'_{\alpha,X\times Y} \geq \frac{2}{3}$ holds. Now let us compute the sum of all w':

$$\sum_{\alpha\in\{0,1\}; X,Y\subseteq\{0,1\}^n} w'_{\alpha,X\times Y} = \mathbf{E}_s \sum_{X,Y\subseteq\{0,1\}^n; v\in V(G)} w_{v,f_{\mathcal{A}}(A,r)\times f_{\mathcal{B}}(B,r)} \leq \mathsf{prt}_{\varepsilon}(\textsc{Search-PM}^{\oplus}_G). \qquad \square$$

6.4 A lower bound on the size of Hitting(⊕) refutations

We can now derive the lower bound.

Theorem 6.9. *Any* Hitting(⊕) *refutation of* PM_G *for the graph* G *from Theorem 6.7 contains* $2^{\Omega(n)}$ *many subspaces.*

Proof. Let ε be the constant from Theorem 6.8. If there is a Hitting(⊕) refutation of $\mathrm{PM}_G(\bar{z})$ with s subspaces, then we can convert it to a Hitting(⊕) refutation of $\mathrm{PM}^{\oplus}_G(\bar{x}, \bar{y})$ (viewed as a set of subspaces) with the same number of subspaces simply by replacing every variable z_e with $x_e \oplus y_e$ in the hitting(⊕) formula. Indeed, if a subspace $M\bar{z} = \bar{a}$ implies that a clause related to a vertex v is false, then the subspace $M(\bar{x} + \bar{y}) = \bar{a}$ implies that the degree of v in $(V(G), E_1 \oplus E_2)$ is not 1 (that is, one of the clauses of $\mathrm{PM}^{\oplus}_G$ related to v is false). Lemma 6.6 (note also Remark 6.3) then implies that $\mathsf{prt}_{\varepsilon}(\textsc{Search-PM}^{\oplus}_G) = O(s^2)$, and so $\mathsf{prt}_{1/3}(\textsc{Disj}_n) = O(s^2)$ according to Theorem 6.8. Theorem 6.5 implies that $\log s = \Omega(n)$. $\square$

Remark 6.10. *Note that the* PM_G *formulas for* $K_{i,j}$ *for* $i \neq j$ *have polynomial-size* CP *proofs: it can be easily derived from the 2-clauses that the number of edges around a vertex is at most 1; then take the sum of such inequalities around all vertices in the smaller part, and take the sum of the other input inequalities in the larger part.*

Acknowledgements

We thank Jan Johannsen, Ilario Bonacina, Oliver Kullmann, and Stefan Szeider for introducing us to the topic; Zachary Chase, Susanna de Rezende, Mika Göös, Amir Shpilka, and Dmitry Sokolov for helpful discussions. We also thank anonymous reviewers for their comments that helped us to improve the presentation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC.

References

[ABDK16] Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *STOC'16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 863–876. ACM, New York, 2016. doi:10.1145/2897518.2897644.

[ABSRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. doi:10.1137/S0097539700366735.

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

[AKK16] Andris Ambainis, Martins Kokainis, and Robin Kothari. Nearly optimal separations between communication (or query) complexity and partitions. In *31st Conference on Computational Complexity*, volume 50 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 4, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016. doi:10.4230/LIPIcs.CCC.2016.4.

[Ale21] Yaroslav Alekseev. A lower bound for polynomial calculus with extension rule. In *36th Computational Complexity Conference*, volume 200 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 21, 18. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021. doi:10.4230/LIPIcs.CCC.2021.21.

[ALN16] Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. *ACM Trans. Comput. Log.*, 17(3):19, 2016. doi:10.1145/2898435.

[BCIP02] Josh Buresh-Oppenheim, Matthew Clegg, Russell Impagliazzo, and Toniann Pitassi. Homogenization and the polynomial calculus. *Comput. Complex.*, 11(3-4):91–108, 2002. doi:10.1007/s00037-002-0171-6.

[BFIK23] John Bamberg, Yuval Filmus, Ferdinand Ihringer, and Sascha Kurz. Affine vector space partitions. *Des. Codes Cryptogr.*, 2023. doi:https://doi.org/10.1007/s10623-023-01263-z.

[BIK+96] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Math. Soc.*, 73(3):1–26, 1996. doi:10.1112/plms/s3-73.1.1.

[BK22] Paul Beame and Sajin Koroth. On disperser/lifting properties of the index and inner-product functions, 2022. URL: https://arxiv.org/abs/2211.17211, doi:10.48550/ARXIV.2211.17211.

[BKS04] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.*, 22:319–351, 2004. doi:10.1613/jair.1410.

[Bla37] Archie Blake. *Canonical expressions in Boolean algebra.* PhD thesis, The University of Chicago, 1937.

[BP96] P. Beame and T. Pitassi. Simplified and improved resolution lower bounds. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 274–282, 1996. doi:10.1109/SFCS.1996.548486.

[BPS05] Paul Beame, Toniann Pitassi, and Nathan Segerlind. Lower bounds for Lovász-Schrijver systems and beyond follow from multiparty communication complexity. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, pages 1176–1188, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[BR96] Paul Beame and Søren Riis. More on the relative strength of counting principles. In Paul Beam and Samuel R. Buss, editors, *Proof Complexity and Feasible Arithmetics, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, April 21-24, 1996*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 13–35. DIMACS/AMS, 1996. doi:10.1090/dimacs/039/02.

[BS09] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, 2009. doi:10.1137/080723880.

[BS14] Boaz Barak and David Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. *Electron. Colloquium Comput. Complex.*, TR14-059, 2014. URL: https://eccc.weizmann.ac.il/report/2014/059, arXiv:TR14-059.

[BSIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004. doi:10.1007/s00493-004-0036-5.

[CCT87] W. Cook, C. R. Coullard, and G. Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

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

[CMSS22] Arkadev Chattopadhyay, Nikhil S. Mande, Swagato Sanyal, and Suhail Sherif. Lifting to parity decision trees via stifling, 2022. URL: https://arxiv.org/abs/2211.17214, doi:10.48550/ARXIV.2211.17214.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.

[DD98] G. Davydov and I. Davydova. Dividing formulas and polynomial classes for satisfiability. In *SAT'98, 2nd Workshop on the Satisfiability Problem*, page 12–21, 1998.

[DH21] Evgeny Dantsin and Edward A. Hirsch. Worst-case upper bounds. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 669–692. IOS Press, 2021. doi:10.3233/FAIA200999.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[DMR09] Stefan S. Dantchev, Barnaby Martin, and Mark Nicholas Charles Rhodes. Tight rank lower bounds for the Sherali–Adams proof system. *Theor. Comput. Sci.*, 410(21-23):2054–2063, 2009. doi:10.1016/j.tcs.2009.01.002.

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[dR21] Susanna F. de Rezende. Automating tree-like resolution in time $n^{o(\log n)}$ is ETH-hard. *Procedia Computer Science*, 195:152–162, 2021. Proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium. URL: https://www.sciencedirect.com/science/article/pii/S1877050921021608, doi:https://doi.org/10.1016/j.procs.2021.11.021.

[dRLNS21] Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Dmitry Sokolov. The power of negative reasoning. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 40:1–40:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.40.

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

[dRPR24] Susanna F. de Rezende, Aaron Potechin, and Kilian Risse. Clique is hard on average for sherali-adams with bounded coefficients. *CoRR*, abs/2404.16722, 2024. URL: https://doi.org/10.48550/arXiv.2404.16722, arXiv:2404.16722, doi:10.48550/ARXIV.2404.16722.

[EH89] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Inform. and Comput.*, 82(3):231–246, 1989. doi:10.1016/0890-5401(89)90001-1.

[FHK+23] Yuval Filmus, Edward Hirsch, Sascha Kurz, Ferdinand Ihringer, Artur Riazanov, Alexander Smal, and Marc Vinyals. Irreducible subcube partitions. *Elec J Comb*, 2023. URL: https://arxiv.org/abs/2212.14685, doi:10.48550/ARXIV.2212.14685.

[FKP19] Noah Fleming, Pravesh Kothari, and Toniann Pitassi. Semialgebraic proofs and efficient algorithm design. *Found. Trends Theor. Comput. Sci.*, 14(1-2):1–221, 2019. doi:10.1561/0400000086.

[FMSV23] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. MaxSAT resolution and subcube sums. *ACM Trans. Comput. Logic*, 24(1), 2023. doi:10.1145/3565363.

[GGKS20] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory Comput.*, 16(13):1–30, 2020. doi:10.4086/toc.2020.v016a013.

[GH03] Dima Grigoriev and Edward A. Hirsch. Algebraic proof systems over formulas. *Theoret. Comput. Sci.*, 303(1):83–102, 2003. Logic and complexity in computer science (Créteil, 2001). doi:10.1016/S0304-3975(02)00446-2.

[GHJ+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP, 2022. URL: https://arxiv.org/abs/2205.02168, doi:10.48550/ARXIV.2205.02168.

[GHP02] Dima Grigoriev, Edward A. Hirsch, and Dmitrii V. Pasechnik. Complexity of semialgebraic proofs. *Moscow Mathematical Journal*, 2(4):647–679, 2002. doi:10.17323/1609-4514-2002-2-4-647-679.

[GK13] Matthew Gwynne and Oliver Kullmann. Towards a theory of good SAT representations. *CoRR*, abs/1302.4421, 2013. URL: http://arxiv.org/abs/1302.4421, arXiv:1302.4421.

[GK18] Michal Garlík and Leszek Aleksander Kołodziejczyk. Some subsystems of constant-depth frege with parity. *ACM Trans. Comput. Logic*, 19(4), nov 2018. doi:10.1145/3243126.

[GKY22] Mika Göös, Stefan Kiefer, and Weiqiang Yuan. Lower Bounds for Unambiguous Automata via Communication Complexity. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 126:1–126:13, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16467, doi:10.4230/LIPIcs.ICALP.2022.126.

[GP18] Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *J. ACM*, 65(6):37:1–37:59, 2018. doi:10.1145/3230742.

[GPW17] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for bpp. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–143, 2017. doi:10.1109/FOCS.2017.21.

[Gry19] Svyatoslav Gryaznov. Notes on resolution over linear equations. In René van Bevern and Gregory Kucherov, editors, *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2019. doi:10.1007/978-3-030-19955-5_15.

[Gwy14] Matthew Gwynne. *Hierarchies for efficient clausal entailment checking: With applications to satisfiability and knowledge compilation.* PhD thesis, Swansea University, 2014.

[Hak85] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[HN12] Trinh Huynh and Jakob Nordstrom. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 233–248, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214000.

[IR21] Dmitry Itsykson and Artur Riazanov. Proof Complexity of Natural Formulas via Communication Arguments. In Valentine Kabanets, editor, *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:34, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2021.3.

[IS20] Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Ann. Pure Appl. Logic*, 171(1):102722, 31, 2020. doi:10.1016/j.apal.2019.102722.

[Iwa89] Kazuo Iwama. CNF-satisfiability test by counting and polynomial average time. *SIAM J. Comput.*, 18(2):385–391, 1989. doi:10.1137/0218026.

[JK10] Rahul Jain and Hartmut Klauck. The partition bound for classical communication complexity and query complexity. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 247–258. IEEE Computer Society, 2010. doi:10.1109/CCC.2010.31.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

[Kra98] Jan Krajíček. Discretely ordered modules as a first-order extension of the cutting planes proof system. *Journal of Symbolic Logic*, 63(4):1582–1596, 1998. doi:10.2307/2586668.

[Kul04] Oliver Kullmann. The combinatorics of conflicts between clauses. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing (SAT 2003)*, pages 426–440, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-24605-3_32.

[Kul11] Oliver Kullmann. Constraint satisfaction problems in clausal form II: minimal unsatisfiability and conflict structure. *Fundam. Informaticae*, 109(1):83–119, 2011. doi:10.3233/FI-2011-429.

[KZ13] Oliver Kullmann and Xishun Zhao. On Davis-Putnam reductions for minimally unsatisfiable clause-sets. *Theoret. Comput. Sci.*, 492:70–87, 2013. doi:10.1016/j.tcs.2013.04.020.

[NS95] Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4, 01 1995. doi:10.1145/129712.129757.

[PD11] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

[Pit96] Toniann Pitassi. Algebraic propositional proof systems. In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996, Princeton, New Jersey, USA, January 14-17, 1996*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 215–244. DIMACS/AMS, 1996. doi:10.1090/dimacs/031/07.

[PS22] Tomás Peitl and Stefan Szeider. Are hitting formulas hard for resolution? *CoRR*, abs/2206.15225, 2022. doi:10.48550/arXiv.2206.15225.

[Pud00] Pavel Pudlák. Proofs as games. *Am. Math. Mon.*, 107(6):541–550, 2000. URL: http://www.jstor.org/stable/2589349.

[Raz92] A. A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. doi:10.1016/0304-3975(92)90260-M.

[Rec76] Robert A. Reckhow. *On the Lengths of Proofs in the Propositional Calculus*. PhD thesis, University of Toronto, Department of Computer Science, 1976. Available from https://www.cs.toronto.edu/~sacook/homepage/reckhow_thesis.pdf.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Comput. Complexity*, 14(1):1–19, 2005. doi:10.1007/s00037-005-0188-8.

[RT08] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. doi:10.1016/j.apal.2008.04.001.

[She21] Suhail Sherif. *Communication Complexity and Quantum Optimization Lower Bounds via Query Complexity*. PhD thesis, Tata Institute of Fundamental Research, Mumbai, 2021.

[Sok17] Dmitry Sokolov. Dag-like communication and its applications. In Pascal Weil, editor, *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings*, volume 10304 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2017. doi:10.1007/978-3-319-58747-9_26.

[Tse68] G. S. Tseitin. On the complexity of derivation in the propositional calculus. *Zapiski nauchnykh seminarov LOMI*, 8:234–259, 1968. English translation of this volume: Consultants Bureau, N.Y., 1970, pp. 115–125.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il