

THE EXPONENTIAL TIME HYPOTHESIS IS FALSE

OPENAI

ABSTRACT. We give a deterministic algorithm for exactly counting the satisfying assignments of an arbitrary k -CNF formula. For every fixed $k \geq 2$, on a formula with n variables and m clauses, our algorithm uses $\text{poly}_k(n, m)$ space and runs in time $(n + m)^{O_k(1)} 2^{n/(\log n)^{\frac{1}{2} + o(1)}}$. Our algorithm refutes the Exponential Time Hypothesis (ETH), which postulates that deciding 3-SAT on n variables requires $2^{\Omega(n)}$ time in the worst case.

CONTENTS

1. Introduction	1
2. Technique overview	7
3. A fast counting algorithm via shared linear summaries	21
4. Constructing low-degree replacement forms	26
5. Counting common zeros of sparse polynomial equations	35
6. Reducing $\#k$ -SAT to counting common zeros	39
Appendix A. Partitioning the satisfying assignments	44
Appendix B. Exact arithmetic and space	47
Appendix C. Additional Consequences of Refuting ETH	51
Appendix D. A sharper bound for exact counting	53
References	64

1. INTRODUCTION

For every fixed $k \geq 3$, k -SAT is the canonical NP-complete problem [Coo71], and its exponential-time complexity underlies the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH) [IP01, IPZ01]. In this paper, we refute ETH by giving, for every fixed $k \geq 2$, a deterministic algorithm that counts the satisfying assignments of a k -CNF in subexponential time and polynomial space.

Write the input k -CNF on n declared variables with m input clauses as

$$F(x_1, \dots, x_n) = \bigwedge_{e=1}^m [(x_{i_{e,1}} \oplus b_{e,1}) \vee \dots \vee (x_{i_{e,k}} \oplus b_{e,k})],$$

where $i_{e,j} \in \{1, \dots, n\}$ and $b_{e,j} \in \{0, 1\}$ are the variable index and negation bit of the j th literal in clause e , and $\oplus$ denotes addition modulo two. Repeating a literal permits clauses with fewer than k literals. The indices $i_{e,j}$ and bits $b_{e,j}$ are fixed by F ; only $x_1, \dots, x_n$ vary over assignments. Our aim is to compute the quantity

$$\#F = |\{x \in \{0, 1\}^n : F(x) = 1\}|. \quad (1.1)$$

Testing $\#F > 0$ also decides satisfiability. We keep k fixed as n grows. Throughout, $\log$ denotes the natural logarithm.

Theorem 1.1. *For each fixed $k \geq 2$, there exists a deterministic algorithm that, given a k -CNF F on $n \geq 20$ declared variables with m clauses, computes $\#F$ as defined in (1.1) in*

$$(n + m)^{O_k(1)} 2^{O_k(n \log \log n / \sqrt{\log n})}$$

bit operations and $(n + m)^{O_k(1)}$ bits of space. For each fixed $D > 0$, its running time on inputs with $m \leq Dn$ is

$$(n + m)^{O_{k,D}(1)} 2^{O_{k,D}(n / \sqrt{\log n})}.$$

The rational interpolation argument in Section 6.4 gives the weaker exponent $O_k(\rho(n))$, where

$$\rho(n) = \frac{n(\log \log n)^2}{\sqrt{\log n}} = \frac{n}{(\log n)^{\frac{1}{2} + o(1)}}.$$

Appendix D proves both bounds in Theorem 1.1 by tensor contraction.

When $\#F > 0$, the standard self-reduction finds a satisfying assignment by fixing one variable at a time and checking that the remaining count is positive. Choosing each bit according to the two conditional counts instead gives an exactly uniform random satisfying assignment, with $\text{poly}_k(n, m)$ expected overhead [JVV86, Theorem 3.3(1)].

The theorem also yields arbitrarily accurate subexponential-time approximation algorithms for Max- k -SAT. The Unique Games Theorem [Ope26, Section 8.1] implies NP-hardness bounds for Max- k -SAT that match, up to arbitrarily small error in the approximation ratio, the guarantees of the semidefinite-programming algorithm of [Rag08]. Write $\text{OPT}(F)$ for the maximum number of clauses of F simultaneously satisfied by an assignment.

Corollary 1.2. *For every fixed $k \geq 2$ and $\varepsilon > 0$, there exists a randomized algorithm that, given a k -CNF F on $n \geq 20$ variables with m clauses, finds an assignment satisfying at least $\text{OPT}(F) - \varepsilon m$ clauses with probability $1 - 2^{-\Omega(n)}$, in*

$$(n + m)^{O_{k,\varepsilon}(1)} 2^{O_{k,\varepsilon}(\rho(n))}$$

time and $(n + m)^{O_{k,\varepsilon}(1)}$ space.

Proof. We may assume $m > 0$. Let $f(x)$ be the fraction of clauses satisfied by x , and let $\hat{f}(x)$ be that fraction in a sample of $O(n/\varepsilon^2)$ clauses chosen independently and uniformly with replacement. Hoeffding's inequality and a union bound over all 2^n assignments give, with probability $1 - 2^{-\Omega(n)}$,

$$|\hat{f}(x) - f(x)| \leq \varepsilon/2 \quad \text{for every } x \in \{0, 1\}^n.$$

Thus, if $\hat{x}$ maximizes $\hat{f}$, then $f(\hat{x}) \geq \max_x f(x) - \varepsilon$. Given an integer threshold b , a Boolean circuit with at most two inputs per gate and size $O_{k,\varepsilon}(n)$ tests whether an assignment satisfies at least b sampled clauses. The standard circuit-to-3-CNF reduction [Bar23, Lemmas 15.9–15.10] produces a formula with $O_{k,\varepsilon}(n)$ variables and clauses that is satisfiable exactly when this threshold can be attained. Theorem 1.1, binary search on the threshold, and self-reduction then find an exact sample optimizer within the stated bounds. Truncating rejection sampling after $O(n + \log(n/\varepsilon^2))$ trials per sample gives worst-case $\text{poly}(n, m, \varepsilon^{-1})$ sampling time and adds at most $2^{-\Omega(n)}$ to the failure probability. $\square$

Note that runtime bounds of the form $2^{o(n)}$ do not transfer under general polynomial-time reductions ($2^{n^{o(1)}}$ would be needed for this), so our result does not imply that general NP problems admit subexponential time algorithms. However we do obtain subexponential time algorithms for counting graph colorings and perfect matchings.

Corollary 1.3. *On simple graphs with v vertices and e edges, the number of independent sets and, for each fixed $q \geq 2$, the number of proper q -colorings can be computed deterministically in $2^{o(v)}$ time and $\text{poly}_q(v, e)$ space. The number of perfect matchings can be computed deterministically in $2^{o(v+e)}$ time and $\text{poly}(v, e)$ space.*

Proof. For $G = (V, E)$, introduce a Boolean variable x_u for each $u \in V$. Satisfying assignments of

$$\bigwedge_{\{u,w\} \in E} (\neg x_u \vee \neg x_w)$$

correspond bijectively to the independent sets $\{u \in V : x_u = 1\}$. For proper q -colorings, use one Boolean variable per vertex and color, require exactly one color at each vertex, and forbid equal colors at adjacent vertices. This gives a counting-preserving q -CNF encoding with qv variables and $O_q(v + e)$ clauses.

For perfect matchings, use one Boolean variable per edge and a linear-size circuit checking that exactly one selected edge meets each vertex. The standard circuit-to-3-CNF reduction [Bar23, Lemmas 15.9–15.10] gives $O(v + e)$ variables and clauses. The gate variables have unique values once the edge bits are fixed, so this encoding also preserves the count. Theorem 1.1 gives the stated time and space bounds. $\square$

The fixed-density case of Theorem 1.1 gives a circuit lower bound for E^{NP} , the class of languages decidable in deterministic time $2^{O(n)}$ with an NP oracle.

Corollary 1.4. *Let $C_L(n)$ be the minimum number of noninput gates in a circuit computing L on $\{0, 1\}^n$, allowing arbitrary Boolean gates of fan-in at most two and unrestricted fan-out. There exist a language $L \in E^{NP}$ and a constant $c > 0$ such that*

$$C_L(n) > cn\sqrt{\log n} \quad \text{for infinitely many } n. \quad (1.2)$$

Appendix C.1 proves the corollary. Appendix C.2 gives algorithms for inversion and exact preimage counting of Boolean maps with constant output locality.

1.1. Related work. We first discuss earlier SAT and counting algorithms and the complexity hypotheses contradicted by Theorem 1.1. We then compare our approach with earlier algebraic algorithms and identify the existing constructions used in our proof, whose roles are developed in the roadmap in Section 2.

1.1.1. SAT algorithms and the exponential-time hypotheses. Schöning’s local-search algorithm and PPSZ are landmarks in the development of exponential-time algorithms for k -SAT [Sch99, PPSZ05]. Schöning’s randomized algorithm starts from a random assignment and repeatedly flips a uniformly chosen variable in an unsatisfied clause. Its bounded-error decision time is

$$(n + m)^{O(1)}(2 - 2/k)^n$$

for each fixed $k \geq 3$ [Sch99, Section 2]. In particular, this gives $(n + m)^{O(1)}(4/3)^n$ time for 3-SAT. The PPSZ algorithm [PPSZ05] uses a different mechanism: bounded resolution derives additional clauses, after which variables are assigned in a random order, taking forced values when available and guessing otherwise. For $k = 3, 4$, its original analysis gave stronger bounds for uniquely satisfiable formulas than for general ones. The unique-SAT guarantees were later extended to general 3-SAT and 4-SAT; for 3-SAT this gave a randomized $O(1.308^n)$ bound [Her14]. Hansen, Kaplan, Zamir, and Zwick improved the PPSZ bound for every fixed $k \geq 3$ by biasing the algorithm’s guesses [HKZZ19]. Scheder obtained an improvement for the same range of k by sharpening the analysis of the original PPSZ algorithm [Sch24, Theorem 1.5]. For general 3-SAT, a preprint by Jiang and Cai gives a randomized $O^*(1.307031578^n)$ bound [JC26, Corollary 1.2], where O^* suppresses factors polynomial in the input size. Liu’s deterministic algorithm combines branching with derandomized local search and solves 3-SAT in $O(1.32793^n)$ time [Liu18, Theorem 21].

Randomized local search has also produced effective heuristics. The WalkSAT heuristic chooses an unsatisfied clause and flips one of its variables, combining random choices with greedy moves [SKC94]. Experiments on formulas from circuit synthesis and diagnosis demonstrated the practical value of this approach [SKC94, pp. 340–342, Tables 4–5].

Among uniformly random k -CNFs, instances at the satisfiability threshold were conjectured to be the hardest for deciding satisfiability [VW21, Section 1]; for fixed $k \geq 3$ and lower clause densities $m/n \leq 2^k/k^C$, where C is a universal constant, polynomial-time almost-uniform sampling of satisfying assignments is known with high probability over the formula [CLWYY25, Theorem 1.2]. For a uniformly random formula with polynomially many clauses, conditioned on satisfiability, Vyas and Williams [VW21, Theorem 3] give a PPZ-based algorithm that finds a solution in $2^{n(1-\Omega(\log k/k))}$ time with high probability over both the formula and the algorithm’s random choices.

The Exponential Time Hypothesis (ETH) asserted that 3-SAT requires $2^{\Omega(n)}$ deterministic time in the worst case, and hence so does k -SAT for every fixed $k \geq 3$ [IP01, Theorem 1]. The Strong Exponential Time Hypothesis (SETH) made the sharper prediction that k -SAT requires $2^{n(1-o_k(1))}$ time as k grows. Precisely, for every $\varepsilon \in (0, 1)$, there is a fixed k for which no deterministic algorithm solves k -SAT in $2^{(1-\varepsilon)n}(n+m)^{O(1)}$ time. The running-time bounds for Schöning’s algorithm and PPSZ helped motivate this prediction: their improvements over exhaustive search diminish as k grows [IP01, Open Problems, item 2].

The Gap Exponential Time Hypothesis (Gap-ETH) strengthens ETH by postulating constants $\delta, \gamma > 0$ for which no $O(2^\gamma n)$ -time algorithm can distinguish a satisfiable 3-CNF from one in which every assignment violates at least a δ fraction of the clauses, even when $m = O(n)$ [Din16, MR17]. The randomized version rules out an algorithm with the same running time that distinguishes the two cases with probability at least $2/3$ on every promised input [CCK⁺20, Conjecture 2.5]. It has been used to prove hardness of approximating Label Cover [Din16] and dense constraint satisfaction problems [MR17]. It also rules out nontrivial fixed-parameter approximations for Clique and Dominating Set. For Clique, the excluded ratios are $o(\text{OPT})$, where OPT is the maximum clique size [CCK⁺20, Corollaries 5.3 and 5.6].

The Nondeterministic Strong Exponential Time Hypothesis (NSETH) asserts that, for every $\varepsilon > 0$, some fixed k admits no nondeterministic $2^{(1-\varepsilon)n}(n+m)^{O(1)}$ -time algorithm for deciding whether a k -CNF is *unsatisfiable*, or equivalently whether a k -DNF is a tautology [CGI⁺16]. NSETH has been used to establish barriers to deterministic fine-grained reductions from SAT to problems such as 3SUM and All-Pairs Shortest Paths [CGI⁺16], and to delineate SETH-based lower bounds for approximating directed graph diameter [Li21].

Theorem 1.1 refutes ETH, SETH, Gap-ETH, and NSETH. Indeed, testing whether the exact count is positive or zero decides both satisfiability and unsatisfiability in deterministic $2^{o(n)}(n+m)^{O_k(1)}$ time for every fixed k ; deciding satisfiability also solves the Gap-ETH promise problem. The deterministic algorithm also refutes the bounded-error randomized versions of ETH [DHM⁺14, Section 1] and SETH [CDL⁺16, Definition 2.1], and of Gap-ETH. It does not conflict with unconditional exponential lower bounds for restricted proof systems such as regular resolution [BI13]. In 2016, Ryan Williams refuted the Merlin–Arthur version of SETH (MA-SETH) by giving a protocol that certifies k -CNF unsatisfiability in $2^{n/2} \text{poly}(n, m)$ time [Wil16, Theorem 3.3].

1.1.2. Exact counting and counting complexity. In Valiant’s framework, #SAT and #3-SAT are #P-complete under parsimonious reductions [Val79, Section 3, Facts 1–2]. For fixed $k \geq 3$, previous exact-counting algorithms achieve $\text{poly}_k(n, m)2^{n-\Omega(n/k)}$ time, with a Las Vegas expected-time bound in [IMP12, Lemma 3.5] and a deterministic bound in [CW21, Theorem 1.6]. For #2-SAT, Wahlström’s algorithm counts satisfying assignments in $O^*(1.2377^n)$ time [Wah08]. For #3-SAT, Kutzkov uses complementary branching to count satisfying assignments in $O^*(1.637^n)$ time [Kut26]. Both algorithms are deterministic.

The counting Exponential Time Hypothesis (#ETH) asserted that exactly counting the satisfying assignments of a 3-CNF requires $2^{\Omega(n)}$ deterministic time [DHM⁺14, Section 1]. Under #ETH, Curticapean [Cur18] rules out deterministic $2^{o(v)}$ -time algorithms for counting perfect matchings even in unweighted bipartite graphs with v vertices and $O(v)$ edges. The perfect-matching algorithm

in Corollary 1.3 applies in particular to these graphs. Our algorithm (Theorem 1.1) solves #3-SAT deterministically in $2^{o(n)}$ poly(n, m) time and therefore refutes #ETH.

1.1.3. Implications for fine-grained complexity. ETH and SETH became standard assumptions for parameterized and fine-grained complexity [IPZ01, VW19]. ETH-based reductions yielded lower bounds for the dependence on solution size in Clique [CHKX06], and on treewidth in constraint satisfaction and partitioned subgraph isomorphism [Mar10]. SETH supported sharp exponential bounds for Hitting Set and Set Splitting [CDL⁺16], and for Independent Set and Dominating Set parameterized by treewidth [LMS18]. It also gave bounds on possible speedups of pseudopolynomial algorithms for Subset Sum and Bicriteria Path [ABHS19].

For problems in P, SETH was used to rule out polynomial-factor speedups over quadratic running times for edit distance [BI18], longest common subsequence and dynamic time warping [BK15], Fréchet distance [Bri14], and regular-expression matching [BI16]. Other applications concerned approximating graph diameter [RV13] and maintaining reachability counts or the size of the largest strongly connected component under graph updates [AW14]. Together these results made SETH a common basis for explaining the apparent optimality of algorithms across quite different problems.

Our refutation of ETH and SETH removes one basis for the hardness claims above. Some of the same lower bounds follow from weaker hypotheses that may still hold. For example, the lower bounds for longest common subsequence and regular-expression matching follow from the Orthogonal Vectors Conjecture (OVC) [BK15, BI16]. OVC asserts that for every fixed $\varepsilon > 0$, no $O(N^{2-\varepsilon} \text{poly}(d))$ -time algorithm can decide whether two sets of N Boolean vectors of dimension d contain an orthogonal pair, one vector from each set. The reduction from SAT to Orthogonal Vectors shows that SETH implies OVC [Wil05, VW19]. The lower bounds for Fréchet distance, graph diameter approximation, and the treewidth dependence of Independent Set and Dominating Set also follow from the hypothesis that unrestricted CNF-SAT admits no $2^{(1-\delta)n} \text{poly}(n, m)$ -time algorithm for any constant $\delta > 0$ [Bri14, RV13, LMS18]. Theorem 1.1 gives a running-time guarantee for each fixed clause width k , while this hypothesis allows the width to grow with n .

To obtain a faster algorithm for a target problem from our SAT algorithm, we need a reduction in the other direction, from that problem to fixed-width SAT, that controls the number of variables. Set Splitting with sets of size at most k has such a reduction [CDL⁺16, Observation 3.6]: use one Boolean variable per element to specify its side of the split. For each set, one clause requires at least one of its variables to be true, and a second requires at least one to be false. The resulting k -CNF has one variable per universe element, so for fixed k Theorem 1.1 gives subexponential time in the universe size, up to a polynomial input-size factor.

1.1.4. Prior work on algebraic algorithms for combinatorial problems. Our algorithm is algebraic, based on polynomial interpolation of carefully chosen auxiliary variables. Below we discuss prior work on using algebraic methods to design algorithm for combinatorial problems. For broader background, see the surveys of Koutis and Williams [KW16] on algebraic fingerprints, Williams [Wil14] on the polynomial method in algorithm design, and Husfeldt [Hus11] on algorithmic inclusion–exclusion.

Williams’s split-and-list algorithm [Wil05] partitions the variables of a 2-CSP into three groups, enumerates partial assignments, and combines them using fast matrix multiplication. Koivisto [Koi06] gives a direct sum–product formulation. The algorithm of Sections 3–6 likewise enumerates assignments within groups. Fixing the shared measurements $\vec{y}$ makes their contributions multiply independently. Scott and Sorkin [SS09, Section 6] extend split-and-list to polynomial-valued weights and arbitrary rings. Our summation routine similarly uses truncated polynomial rings to retain several coefficients in one computation.

Björklund [Bjö16, Sections 2–3] uses inclusion–exclusion and polynomial interpolation to reduce permanent and directed Hamiltonian-cycle counting to a weighted sum over smaller instances. He adds the weights of identical instances in a table, then solves each distinct instance with nonzero total weight once. Björklund, Kaski, and Williams [BKW19] use low-degree curves and generalized

Takeya sets to recover polynomial values from evaluations on a small structured set, yielding faster algorithms for fermionants, including the permanent of integer matrices whose entries are bounded in absolute value by a fixed constant. Our polynomial replacements serve a similar purpose: they connect the desired count to structured evaluations from which it can be reconstructed. Here the useful structure is separation into groups after fixing a few shared measurements.

Bhargava, Ghosh, Kumar, and Mohapatra [BGKM23, Sections 3.1–3.2] evaluate multivariate polynomials over finite fields along low-degree curves, using Hasse derivatives and Hermite interpolation when values alone do not suffice. For a univariate restriction g , the order- j Hasse derivative at t is the coefficient of z^j in $g(t + z)$. We use the same recovery principle: compute several Taylor coefficients at each point admitting fast summation, then recover the original count by Hermite interpolation.

Woodruff and Yekhanin [WY07] use values and first derivatives along a line to recover a target value in private information retrieval. Kopparty, Saraf, and Yekhanin [KSY14] use higher-order Hasse derivatives for local correction of multiplicity codes, extending the line-based recovery of Reed–Muller codes. In our algorithm, the matrix construction supplies low-degree replacement forms that let us compute the Taylor coefficients at the interpolation points by fast summation after fixing a few shared measurements.

1.1.5. Antecedents of the matrix construction. Our counting algorithm evaluates sums of products of polynomials over Boolean assignments. The polynomials share variables, so these sums do not factor into independent smaller sums. We introduce linear substitutions, depending polynomially on an interpolation parameter, that make such a factorization possible after fixing a few integer linear measurements. An auxiliary square rational matrix M_0 underlies these substitutions: a bound on $\text{rank}(I - M_0)$ limits the number of measurements, while restrictions on the nonzero entries of M_0 keep the substitution degrees small. Section 2.7 explains this connection.

Related interpolation methods have been used to obtain low-rank approximations to structured matrices. In [AW17, Section 3], a low-degree polynomial matches $(-1)^j$ on an interval of integer inner products j , using the interpolation lemma of [AW15, Lemma 3.1]. This gives low-rank matrices agreeing with Walsh–Hadamard matrices in most entries, a result in the study of matrix rigidity [Val77]. Related interpolation gives rigidity bounds for Kronecker powers in [AGP23, Theorem 6.1]. Dvir and Liu [DL20, Claim 2.20 and Lemmas 3.3–3.4] use a spectral approach: sparse changes to a function invariant under permutations of its coordinates make many Fourier coefficients vanish, reducing the rank of its group-circulant matrix.

Our construction also interpolates $(-1)^j$ on a short interval, now to obtain a polynomial in commuting permutation matrices. The same polynomial gives both the rank bound for $I - M_0$ and the restrictions on the nonzero entries of M_0 . We prove these properties and their quantitative bounds directly in Section 4.

1.1.6. Sparsification and parity tests. We use the counting-preserving version [DHM⁺14, Appendix A] of the sparsification lemma [IPZ01], with quantitative bounds from [CIP06, Section 6] and [Cal09, Lemma 5.14 and equation (5.4)]. This partitions the satisfying assignments among sparse formulas on the same Boolean variables. Appendix A records the disjoint branching argument and the bounds needed when its parameter varies with the input size.

The initial parity tests are the constraints of a Tanner code [Tan81, Section II], applied to the binary vector recording which clauses an assignment violates. We use the edge-based expander-code construction of [SS96, Sections IV and VI]: each graph edge carries one bit, and each vertex requires its incident bits to belong to a fixed binary linear code. The local code’s minimum distance, together with graph expansion, forces every nonzero vector satisfying all these constraints to have large support. This construction and distance argument are standard. Lemma 6.1 supplies the parameter choices needed here, including the bounds on the number and support size of the tests, and constructs the graph and local code by deterministic exhaustive searches. The graph is the bipartite double

cover of a Cayley multigraph. The shifts defining that multigraph nearly balance every nonconstant parity on the base group: this is the small-bias condition [NN93, AR94], stated explicitly in (6.2). Fixing unused edge bits to zero gives a code on exactly m coordinates. Additional parity tests exclude the remaining nonzero violation vectors. We choose these tests by the averaging principle underlying the method of conditional expectations [Rag88], using exact counts to evaluate each candidate. Section 6 gives the constructions and quantitative bounds needed here.

Theorem 1.1 invites a renewed search for algorithms for problems whose known hardness results relied on ETH or SETH. The examples in Section 1.1.3 suggest both immediate consequences and problems for which new algorithmic ideas are needed.

1.2. Notation. For an integer $s \geq 0$, write $[s] = \{1, \dots, s\}$, with $[0] = \emptyset$. For a condition E , the scalar $\mathbb{1}_{\{E\}}$ equals one if E holds and zero otherwise. We use natural logarithms, writing $\log_2$ when the base is two. For complexity bounds, $\text{poly}(t_1, \dots, t_s)$ denotes a bound of the form $(1 + t_1 + \dots + t_s)^{O(1)}$, with absolute implicit constants. Subscripts, as in $\text{poly}_{k,c}(t_1, \dots, t_s)$, allow those constants to depend on the indicated fixed parameters. Integers are stored in signed binary notation, so an integer z uses $O(\log(|z| + 2))$ bits. Rational numbers are stored as an integer numerator and a positive integer denominator; stated bit-length bounds apply to each of them. Space is measured in bits. Arithmetic-operation counts are distinguished from bit-operation counts.

For a finite set S , vectors in $\mathbb{Q}^S$ have coordinates indexed by S and are column vectors unless indicated otherwise. For $U \subseteq S$, the indicator vector $\mathbf{1}_U \in \{0, 1\}^S$ has entries $(\mathbf{1}_U)_v = \mathbb{1}_{\{v \in U\}}$ for $v \in S$. We write $\vec{0}$ for the zero vector. For $v \in S$, the row $\delta_v \in \mathbb{Q}^{1 \times S}$ selects coordinate v , so $\delta_v x = x_v$. A vector indexed by $U \subseteq S$ is extended to S by putting zero in each coordinate of $S \setminus U$.

We write $\vec{X} = (X_1, \dots, X_n)$ for the vector of formal variables and use arrows for polynomial coefficient vectors such as $\vec{P}_{f,v}(T)$. For a coefficient row $\vec{p} = (p_i)_{i \in S}$ and a vector $z = (z_i)_{i \in S}$, we write their scalar pairing as $\langle \vec{p}, z \rangle = \sum_{i \in S} p_i z_i$. In particular, $\langle \vec{P}_{f,v}(T), \vec{X} \rangle$ is a scalar linear form in the variables X_i .

For a polynomial $P(\vec{X}, z)$ in the variables $\vec{X}$ and a variable z , $[z^j]P$ denotes the coefficient of z^j , viewed as a polynomial in $\vec{X}$, and $\deg_{\vec{X}} P$ denotes total degree in $\vec{X}$. The degree of a row of polynomials is the largest degree of any entry. Unless otherwise specified, polynomial coefficients are rational and matrix kernels are taken over $\mathbb{Q}$.

2. TECHNIQUE OVERVIEW

We first count satisfying assignments of a 3-CNF in which each variable occurs in at most three clauses. The product of its clause polynomials has degree at most $3n$, so we can use interpolation to recover the exact count from sums over small groups of variables. Section 2.8 then uses disjoint sparsification and variable copying to count the satisfying assignments of an arbitrary k -CNF in $(n + m)^{O_k(1)} 2^{o(n)}$ time for every fixed $k \geq 2$. Taking $k = 3$ already refutes ETH. Section 2.9 explains how counting on the original variables gives the exponent $O_k(\rho(n))$ of Proposition 6.4.

2.1. The bounded-occurrence setting. Let F have n declared Boolean variables. Each clause contains at most three literals, and each variable appears in at most three clauses, counting positive and negative occurrences together. Our task is to count the satisfying assignments of F exactly.

We remove tautological clauses and repeated literals within a clause. An empty clause gives count zero immediately. These operations preserve the occurrence bound. For each remaining clause e , let $I_e \subseteq [n]$ be its variable set. For each variable that occurs in no clause, add a factor equal to one with scope $\{v\}$. Such a factor leaves the count unchanged and ensures that every variable has an assigned position in the construction. We call these constant-one terms *dummy clauses*. Let $\mathcal{F}$ index

all clauses, including the dummy clauses. Then

$$|I_e| \leq 3, \quad \#\{e \in \mathcal{F} : v \in I_e\} \leq 3, \quad n \leq M := \sum_{e \in \mathcal{F}} |I_e| \leq 3n. \quad (2.1)$$

All sums below still range over the original n Boolean variables.

2.2. A direct polynomial for the count. Write $L_{e,v}(X_v) = X_v$ for a positive literal and $L_{e,v}(X_v) = 1 - X_v$ for a negative literal. The satisfaction polynomial of clause e is

$$\Phi_e((X_v)_{v \in I_e}) = 1 - \prod_{v \in I_e} (1 - L_{e,v}(X_v)).$$

For example, $x_a \vee \neg x_b \vee x_c$ gives

$$\Phi_e(X_a, X_b, X_c) = 1 - (1 - X_a)X_b(1 - X_c).$$

On Boolean inputs this polynomial is one when the clause is satisfied and zero otherwise. For a dummy clause e , put $\Phi_e = 1$. Therefore

$$\#F = \sum_{x \in \{0,1\}^n} \prod_{e \in \mathcal{F}} \Phi_e((x_v)_{v \in I_e}). \quad (2.2)$$

The product has total degree at most $M \leq 3n$. This linear degree bound is the useful consequence of the three-occurrence assumption.

Equation (2.2) is an exact expression for the count, but evaluating its 2^n summands would be exhaustive search. To motivate the algorithm, suppose we were given a partition of the variables into r groups $S_1, \dots, S_r$ satisfying both of the following conditions:

- (1) Each group contains at most $2n/r$ variables.
- (2) For every clause, all of its variables belong to a single group.

This is a hypothetical, very restrictive assumption; we do not assume that the input formula admits such a partition.

Under this assumption, we could count satisfying assignments in each group independently. Let N_g be the number of assignments to the variables in S_g that satisfy all clauses in that group. Any such assignment can be combined with any satisfying assignments in the other groups, because no clause connects different groups. Every global satisfying assignment arises in exactly one such way. Consequently,

$$\#F = \prod_{g=1}^r N_g.$$

Computing each N_g by exhaustive search within its group examines a total of at most

$$\sum_{g=1}^r 2^{|S_g|} \leq r 2^{2n/r}$$

local assignments, with polynomial work per assignment. For large r , this is much faster than enumerating all 2^n global assignments.

We will replace the clause inputs so that different groups interact only through a small collection of shared linear summaries. Once the values of those summaries are fixed, we can again sum over each group separately.

2.3. A polynomial family with easier evaluation points. We introduce a parameter T . At $T = 0$, the replacements must restore the original clause inputs. At selected nonzero points $T = 1, \dots, Q$, they should let us separate the groups by fixing shared linear summaries. Here Q is the number of selected points, and R will bound the degree of the replacement coefficients. We choose both parameters later.

Write $\vec{X} = (X_1, \dots, X_n)$ for the vector of formal variables. For each occurrence $v \in I_e$, use the polynomial coefficient vector $\vec{P}_{e,v}(T) = (p_{e,v,1}(T), \dots, p_{e,v,n}(T))$ to replace X_v in clause e by the scalar inner product

$$\langle \vec{P}_{e,v}(T), \vec{X} \rangle = \sum_{i=1}^n p_{e,v,i}(T) X_i.$$

We call this expression the *replacement linear form* for the occurrence of variable v in clause e . It replaces the input X_v in the clause polynomial Φ_e . We require

$$\langle \vec{P}_{e,v}(0), \vec{X} \rangle = X_v, \quad \deg \vec{P}_{e,v} \leq R.$$

Different clauses may use different replacement linear forms for the same variable. Define

$$H(T) = \sum_{x \in \{0,1\}^n} \prod_{e \in \mathcal{F}} \Phi_e(\langle \vec{P}_{e,v}(T), x \rangle)_{v \in I_e} \in \mathbb{Q}[T]. \quad (2.3)$$

At zero every replacement restores the original variable, while the degree of each clause polynomial grows by at most a factor of R . Hence

$$H(0) = \#F, \quad \deg H \leq MR \leq 3nR. \quad (2.4)$$

We will recover $H(0)$ from Taylor coefficients at $t = 1, \dots, Q$. The degree bound means that $O(1 + nR/Q)$ coefficients per point suffice, as Section 2.5 explains. Our goal is therefore to make R small relative to Q : many points with low-degree replacements require fewer coefficients at each point.

At a nonzero t , the value $\langle \vec{P}_{e,v}(t), x \rangle$ of a replacement linear form need not lie in $\{0, 1\}$, even when x is Boolean. The transformed clause polynomials therefore need not act as satisfaction indicators.

2.4. Separating the sum by fixing measurements. Fix $t \in [Q]$. We are given the coefficient vectors $\vec{P}_{e,v}(t)$, whose entries are now constants. We explain what structure lets us compute $H(t)$ quickly, and how to use it. Throughout this subsection, write $S_g = S_{t,g}$ and $C_g = C_{t,g}$.

Definition 2.1 ((r, τ, D) -design). For the given coefficient vectors $\vec{P}_{e,v}(t)$, an (r, τ, D) -*design* at t consists of groups S_g , clause assignments $\sigma_e(t)$, and measurement matrices C_g satisfying the following four conditions:

- (1) *Small groups.* Partition the variables into disjoint groups $[n] = S_1 \sqcup \dots \sqcup S_r$, with $|S_g| \leq 2n/r$.
- (2) *One assigned group per clause.* Assign each clause e to one group $\sigma_e(t) \in [r]$. Condition 4 must hold with this same group for every replacement linear form of clause e .
- (3) *Few bounded measurements.* Each group g has an integer matrix C_g with columns indexed by S_g and c_g rows. Each row specifies one linear measurement of the variables in S_g . There are at most $2\tau n$ rows in total, and the sum of absolute values of the coefficients in each row is at most D . For an assignment x , write

$$x_g = x|_{S_g}, \quad y_g = C_g x_g, \quad \vec{y} = (y_1, \dots, y_r).$$

- (4) *Dependence on other groups only through measurements.* Suppose clause e is assigned to group $g = \sigma_e(t)$. Each replacement linear form of this clause may depend freely on the variables in S_g . Its dependence on any other group S_h must be a linear combination of

that group's measurements y_h . Precisely, for every $v \in I_e$, there must be coefficient vectors $\vec{\lambda}_{e,v,h} \in \mathbb{Q}^{c_h}$, for $h \neq g$, such that, for every $x \in \mathbb{Q}^n$,

$$\langle \vec{P}_{e,v}(t), x \rangle = \langle \vec{P}_{e,v}(t)|_{S_g}, x_g \rangle + \sum_{h \neq g} \langle \vec{\lambda}_{e,v,h}, y_h \rangle. \quad (2.5)$$

Equivalently, $\vec{P}_{e,v}(t)|_{S_h}$ lies in the row space of C_h for every $h \neq g$.

The matrices C_h are shared by all clauses. A clause's assigned group need not contain its original variables. Once $\vec{y}$ is fixed, however, the entire transformed clause polynomial depends only on the variables in its assigned group.

The algorithm. Given an (r, τ, D) -design at t , first find the coefficient vectors $\vec{\lambda}_{e,v,h}$ by solving the linear systems guaranteed by condition 4. For fixed $\vec{y}$, let $\mathcal{X}_{t,g}(y_g)$ be the assignments to the variables in S_g whose measurements equal y_g :

$$\mathcal{X}_{t,g}(y_g) = \{u \in \{0, 1\}^{S_g} : C_g u = y_g\}.$$

For such an assignment u , compute the contribution of group g as follows. For each clause e assigned to g , first use (2.5) to compute the values of its replacement linear forms from u and the fixed measurements. Next evaluate the clause polynomial Φ_e on these values. Finally, multiply the values obtained from all clauses assigned to g . Denote this product by $A_{t,g}(u; \vec{y})$. No assignments to the other groups are needed for this calculation.

Each measurement of a Boolean assignment is an integer in $[-D, D]$. Initialize $Z = 0$ and perform the following steps:

- (1) Try every measurement tuple $\vec{y} \in \mathcal{Y} := \prod_{g=1}^r \{-D, \dots, D\}^{c_g}$. For each tuple, perform steps 2 and 3.
- (2) For each group g , try every Boolean assignment u to its variables. Keep it only if $C_g u = y_g$. Add the corresponding products $A_{t,g}(u; \vec{y})$ to obtain the group sum

$$B_{t,g}(\vec{y}) = \sum_{u \in \mathcal{X}_{t,g}(y_g)} A_{t,g}(u; \vec{y}).$$

If no assignment satisfies $C_g u = y_g$, set $B_{t,g}(\vec{y}) = 0$.

- (3) Multiply the group sums and add the result $\prod_{g=1}^r B_{t,g}(\vec{y})$ to Z .

After all measurement tuples have been processed, return Z .

To see why this returns $H(t)$, fix $\vec{y}$ and choose one assignment from each set $\mathcal{X}_{t,g}(y_g)$. Since the groups contain disjoint sets of variables, these choices combine into one full assignment. Every full assignment with measurements $\vec{y}$ arises in exactly one such way. Its product of clause polynomials is the product of the group contributions $A_{t,g}$. Multiplying out the group sums and then summing over $\vec{y}$ therefore gives

$$H(t) = \sum_{\vec{y} \in \mathcal{Y}} \prod_{g=1}^r \left(\sum_{u \in \mathcal{X}_{t,g}(y_g)} A_{t,g}(u; \vec{y}) \right). \quad (2.6)$$

Every Boolean assignment has exactly one measurement tuple, so this sum counts each assignment's contribution exactly once.

There are at most $(2D + 1)^{2\tau n}$ measurement tuples. For each tuple, the algorithm tries at most $\sum_g 2^{|S_g|} \leq r 2^{2n/r}$ assignments to individual groups. Consequently, the running time is

$$(2D + 1)^{2\tau n} 2^{2n/r}$$

up to polynomial factors. The first factor bounds the number of measurement tuples; the second bounds the number of assignments within one group. The algorithm scans one group at a time and keeps only the current measurement tuple, the current assignment, and running sums and products; see Lemma 3.2.

For the parameter choices in Section 2.6, we can construct replacement forms of degree at most R that restore the original variables at zero and admit an (r, τ, D) -design at every $t \in [Q]$. Section 2.7 gives the construction. We first explain how these properties let us recover the count efficiently.

2.5. Recovering the count from Taylor coefficients. Our goal is to recover $H(0) = \#F$ from information at the points $t = 1, \dots, Q$. Since $\deg H \leq 3nR$, the Q values $H(t)$ alone may not determine H . We instead compute several Taylor coefficients at each point. Choose

$$q = \left\lfloor \frac{3nR}{Q} \right\rfloor, \quad J = q + 1.$$

Suppose we know the coefficients of $z^0, \dots, z^q$ in $H(t+z)$ for every $t \in [Q]$. These data determine H uniquely: if two polynomials of degree at most $3nR$ have the same data, their difference is divisible by $(T-t)^J$ for every $t \in [Q]$, hence by $\prod_{t=1}^Q (T-t)^J$. This product has degree $QJ > 3nR$, so the difference must be zero. Each Taylor coefficient gives a linear equation in the coefficients of H . Solving this rational linear system recovers H , and its constant coefficient is the desired count $H(0)$.

It remains to compute these Taylor coefficients. Fix $t \in [Q]$; we want $H(t+z)$ through degree q in z . The design controls the replacement forms at t , but need not control their changes from t to $t+z$. We first give those changes a separate input, so that fixing this input preserves the separation of groups.

Keeping the shifted inputs separate. Write

$$\Delta \vec{P}_{e,v}(z) = \vec{P}_{e,v}(t+z) - \vec{P}_{e,v}(t).$$

Keep x as the Boolean assignment being summed over, and introduce a separate formal vector $\vec{X}$ for the changes. Define

$$\mathcal{G}_x(\vec{X}, z) = \prod_{e \in \mathcal{F}} \Phi_e((\langle \vec{P}_{e,v}(t), x \rangle + \langle \Delta \vec{P}_{e,v}(z), \vec{X} \rangle)_{v \in I_e}). \quad (2.7)$$

Here x indexes the polynomial, while $\vec{X}$ and z are its formal variables. Setting $\vec{X} = x$ restores each shifted input $\langle \vec{P}_{e,v}(t+z), x \rangle$, so

$$H(t+z) = \sum_{x \in \{0,1\}^n} \mathcal{G}_x(x, z).$$

On the other hand, fixing $\vec{X} = \mathbf{1}_U$ for some $U \subseteq [n]$ makes every correction $\langle \Delta \vec{P}_{e,v}(z), \mathbf{1}_U \rangle$ independent of x . All dependence on x then remains through $\langle \vec{P}_{e,v}(t), x \rangle$, where the design applies. Our task is therefore to recover the sum with $\vec{X} = x$ from sums with $\vec{X}$ fixed.

The degree bound makes this possible with few fixed substitutions. Every entry of $\Delta \vec{P}_{e,v}(z)$ is divisible by z , so each occurrence of an X_i in an expanded term contributes at least one factor of z . Thus

$$\deg_{\vec{X}}([z^j] \mathcal{G}_x(\vec{X}, z)) \leq j.$$

Thus, after discarding powers of z above q , every monomial involves at most q distinct variables from $\vec{X}$.

Recovering the shifted sum by inclusion–exclusion. Work modulo z^J , so that only the coefficients through z^q are retained. Replace every positive power of X_i in $\mathcal{G}_x$ by X_i . This preserves its values on Boolean inputs. The degree bound above gives, for Boolean $\vec{X}$,

$$\mathcal{G}_x(\vec{X}, z) \equiv \sum_{\substack{E \subseteq [n] \\ |E| \leq q}} a_E(x, z) \prod_{i \in E} X_i \pmod{z^J},$$

where $a_E(x, z)$ is the coefficient of the indicated monomial after this replacement. Evaluating at $\mathbf{1}_U$, the vector with ones precisely in U , sums the coefficients indexed by subsets of U . Inclusion–exclusion extracts each one:

$$a_E(x, z) \equiv \sum_{U \subseteq E} (-1)^{|E|-|U|} \mathcal{G}_x(\mathbf{1}_U, z) \pmod{z^J}.$$

When we set $\vec{X} = x$, the monomial $\prod_{i \in E} x_i$ equals one exactly when all the bits in E are one. Its contribution to the sum over x is therefore $\sum_{x: x_i=1 \ (i \in E)} a_E(x, z)$, with x Boolean. Substituting the coefficient formula above reduces this to sums with the formal input fixed.

Computing the restricted sums. For $U \subseteq E \subseteq [n]$ with $|E| \leq q$, define

$$\mathcal{Z}_{t,E,U}(z) = \sum_{\substack{x \in \{0,1\}^n \\ x_i=1 \ (i \in E)}} \mathcal{G}_x(\mathbf{1}_U, z) \text{ in } \mathbb{Q}[z]/(z^J). \quad (2.8)$$

Here U fixes the formal input $\vec{X} = \mathbf{1}_U$, while E fixes bits of the summation variable x to one. Summing the monomial contributions gives

$$H(t+z) \equiv \sum_{\substack{E \subseteq [n] \\ |E| \leq q}} \sum_{U \subseteq E} (-1)^{|E|-|U|} \mathcal{Z}_{t,E,U}(z) \pmod{z^J}. \quad (2.9)$$

This recovers all $J = q + 1$ Taylor coefficients together. Lemma 3.3 states the general identity used here.

Each sum $\mathcal{Z}_{t,E,U}$ can be evaluated by the algorithm of Section 2.4. With $\vec{X} = \mathbf{1}_U$ fixed, each correction $\langle \Delta \vec{P}_{e,v}(z), \mathbf{1}_U \rangle$ is independent of x . All dependence on x comes through the forms $\langle \vec{P}_{e,v}(t), x \rangle$, so fixing $\vec{y}$ still separates the clause polynomials. The restrictions $x_i = 1$ are local to each group. The separation identity uses only addition, multiplication, and distributivity, so it also applies to these truncated polynomial values. Each clause is evaluated after substituting $\vec{X} = \mathbf{1}_U$, using only univariate polynomials in z of degree less than J and discarding higher powers after each multiplication. The multivariate expansion and its coefficients $a_E(x, z)$ are used only to justify the identity; the algorithm never constructs them.

Repeating this calculation at $t = 1, \dots, Q$ provides all the Taylor coefficients needed for the interpolation above, and therefore recovers $H(0) = \#F$.

2.6. The three costs and the running time. Jumping ahead, Section 2.7 will construct replacement forms that restore the original variables at zero, have degree at most R , and admit an (r, τ, D) -design at every $t \in [Q]$. For sufficiently large n , the construction takes polynomial time and achieves the following parameter bounds:

$$\begin{aligned} r &= \Theta\left(\frac{\sqrt{\log n}}{(\log \log n)^2}\right), & Q &= \Theta\left(\frac{\log n}{\log \log n}\right), \\ R &= \Theta(\sqrt{\log n}), & \tau &= \Theta\left(\frac{1}{r \log n}\right), & D &= r^{2Q} \leq (2n)^{1/128}. \end{aligned} \quad (2.10)$$

We first show how these bounds give a fast counting algorithm; the next subsection will justify that they can all be achieved simultaneously.

Recall that $q = J - 1 = \lfloor 3nR/Q \rfloor$. For each evaluation point t , Section 2.5 computes one restricted sum for each pair $U \subseteq E$ with $|E| \leq q$. There are $\binom{n}{j}$ choices of an E of size j , and then 2^j choices of U . Each restricted sum enumerates measurement tuples and assignments within individual groups.

Thus the three costs multiply to give, up to polynomial factors,

$$\underbrace{\sum_{j=0}^{\min(q,n)} 2^j \binom{n}{j}}_{\text{Taylor data}} \cdot \underbrace{(2D+1)^{2\tau n}}_{\text{measurements}} \cdot \underbrace{2^{2n/r}}_{\text{one group}}. \quad (2.11)$$

To bound the first cost, note that $1 \leq q \leq n/2$ for sufficiently large n , since $R/Q \rightarrow 0$. Using $\binom{n}{j} \leq (en/j)^j$ and the fact that $(2en/j)^j$ increases for $1 \leq j \leq n$, we obtain

$$\sum_{j=0}^q 2^j \binom{n}{j} \leq (q+1) \left(\frac{2en}{q} \right)^q.$$

Since $q = \Theta(nR/Q)$, the logarithm of this cost is $O(n(R/Q) \log(2Q/R))$. Taking logarithms of all three costs in (2.11) therefore gives

$$O\left(n \left[\frac{R}{Q} \log \frac{2Q}{R} + \tau \log(2D+1) + \frac{1}{r} \right]\right).$$

Each term is small under (2.10):

$$\begin{aligned} \frac{R}{Q} \log \frac{2Q}{R} &= O\left(\frac{\log \log n}{\sqrt{\log n}} \cdot \log \log n\right), \\ \tau \log(2D+1) &= O\left(\frac{1}{r \log n} \cdot \log n\right) = O(1/r), \\ \frac{1}{r} &= O\left(\frac{(\log \log n)^2}{\sqrt{\log n}}\right). \end{aligned}$$

Thus the bounded-occurrence instance can be counted in

$$n^{O(1)} 2^{O(n(\log \log n)^2 / \sqrt{\log n})} \quad (2.12)$$

bit operations and polynomial space. The finitely many smaller inputs are handled by exhaustive search.

The number of points Q , the number of groups r , and the truncation order $J = O(1 + nR/Q)$ are polynomial in n . Hence the loops over points and groups, arithmetic on polynomials of degree less than J , and the final interpolation add only polynomially many operations per iteration of the three enumerations above.

For the bit-complexity and space bounds, see Appendix B.4.

2.7. Constructing the low-degree replacement linear forms. Consider one occurrence of X_v in a clause. At $T = 0$ its replacement must equal X_v . At each $t = 1, \dots, Q$, the clause must be able to evaluate that same replacement using only its assigned group and a few shared measurements. Choosing a different linear form at each point and interpolating would generally cost degree Q . To keep Taylor recovery cheap, we need degree R much smaller than Q .

Suppose a term in the replacement uses a variable outside the clause's assigned group only at $t = 2$. Multiplying the term by $1 - T/2$ leaves it unchanged at zero and removes it at $t = 2$. At every other evaluation point the clause can read the variable from its own group. The same trick works for several such points: one factor per point, at a cost of one degree each. The challenge is therefore to write each input as a sum in which the shared measurements determine one part, and each remaining term uses a variable outside the assigned group at only a few points. We first construct such a decomposition on auxiliary coordinates that record the group choices, then substitute the variables of the formula.

Step 1: Set up the auxiliary coordinates.

At each evaluation point, an occurrence (e, v) has two group choices: the group containing variable v , and the group assigned to clause e . Across the Q points, these choices form a sequence of Q pairs. We therefore use the coordinate set

$$\mathcal{B} = ([r] \times [r])^Q, \quad D = |\mathcal{B}| = r^{2Q}.$$

Write an index as

$$u = ((a_u(1), d_u(1)), \dots, (a_u(Q), d_u(Q))).$$

We will construct one replacement row for every such index, so it will be available whatever group choices we later make for the formula.

Let $\vec{\zeta} = (\zeta_w)_{w \in \mathcal{B}}$ be formal placeholders for the original variables that we will insert in Step 4. At point t , coordinate ζ_w belongs to group $a_w(t)$, whereas the replacement for ζ_u is assigned to group $d_u(t)$. Thus the coordinates that this replacement may use freely are

$$\mathcal{S}_{t, d_u(t)}, \quad \mathcal{S}_{t, g} := \{w \in \mathcal{B} : a_w(t) = g\}.$$

In particular, ζ_w belongs to the assigned group exactly when $a_w(t) = d_u(t)$. The sets $\mathcal{S}_{t, g}$ partition $\mathcal{B}$, with D/r coordinates in each group.

Our auxiliary goal is to construct coefficient rows $\vec{P}_u(T) \in \mathbb{Q}[T]^D$ and one matrix

$$C \in \{0, 1, -1\}^{c \times D}, \quad c \leq \frac{D\tau}{3r},$$

such that

$$\langle \vec{P}_u(0), \vec{\zeta} \rangle = \zeta_u, \quad \deg \vec{P}_u \leq R - 1,$$

and, at every $t \in [Q]$, the replacement linear form $\langle \vec{P}_u(t), \vec{\zeta} \rangle$ is a linear combination of the available data

$$\vec{\zeta}|_{\mathcal{S}_{t, d_u(t)}} \quad \text{and} \quad C\vec{\zeta}. \tag{2.13}$$

The same measurements $C\vec{\zeta}$ serve every row and every evaluation point. This is the auxiliary counterpart of the dependence condition in the design from Section 2.4.

Step 2: Construct the constant matrix.

We start from the identity

$$\zeta_u = ((I_D - M_0)\vec{\zeta})_u + \sum_{w \in \mathcal{B}} M_0[u, w]\zeta_w.$$

We want the first term to be determined by $C\vec{\zeta}$. In the sum, each nonzero coefficient should connect incompatible groups at only a few points, so that a low-degree polynomial can remove those dependencies. Define these points by

$$E_{u, w} = \{t \in [Q] : a_w(t) \neq d_u(t)\}.$$

The two requirements on $M_0 \in \mathbb{Q}^{D \times D}$ are

$$\begin{aligned} M_0\vec{\xi} &= \vec{\xi} \quad \text{for every } \vec{\xi} \in \ker C, \\ M_0[u, w] &\neq 0 \implies |E_{u, w}| \leq R - 1. \end{aligned} \tag{2.14}$$

The first says that $I_D - M_0$ vanishes on $\ker C$. Equivalently, each of its rows lies in the row space of C , so $(I_D - M_0)\vec{\zeta}$ is a linear function of the measurements $C\vec{\zeta}$. The second will bound the degrees of the polynomials used in Step 3.

Restricting the support by keeping f low-degree. Let Π_s be the permutation matrix that swaps the two entries of pair s in every index, leaving all other pairs unchanged. These matrices commute and satisfy $\Pi_s^2 = I_D$. Put

$$V_0 = \prod_{s=1}^Q \Pi_s, \quad W_0 = \frac{1}{2} \sum_{s=1}^Q (I_D - \Pi_s), \quad M_0 = V_0 f(W_0),$$

where we will choose a polynomial f of degree at most $R-1$. The complete swap V_0 already respects every group: its nonzero entry in row u is at the column w satisfying

$$(a_w(s), d_w(s)) = (d_u(s), a_u(s)) \quad \text{for every } s.$$

In particular, $E_{u,w}$ is empty.

Multiplying by $f(W_0)$ can undo swaps, but the degree of f limits how many. Expanding $f(W_0)$ gives products of at most $R-1$ swaps. Repeated swaps cancel because $\Pi_s^2 = I_D$. Multiplying by V_0 therefore gives

$$M_0 = \sum_{\substack{S \subseteq [Q] \\ |S| \leq R-1}} \gamma_S \prod_{s \notin S} \Pi_s \quad \text{for some } \gamma_S \in \mathbb{Q}. \quad (2.15)$$

The permutation in the summand indexed by S swaps every pair outside S . Its nonzero entries therefore satisfy $a_w(s) = d_u(s)$ for every $s \notin S$, so $E_{u,w} \subseteq S$. Every summand is zero whenever $|E_{u,w}| > R-1$, and hence so is their sum. This proves the second condition in (2.14) for any polynomial f of degree at most $R-1$.

Choosing f to fix most basis directions. We now choose f so that M_0 fixes most basis directions. First consider one position, with coordinates indexed by $[r] \times [r]$. Let $\vec{e}_{ab}$ be the vector that is one at (a, b) and zero elsewhere. Swapping the pair has the orthogonal eigenbasis

$$\begin{aligned} \vec{e}_{aa} \quad (a \in [r]), \quad \vec{e}_{ab} + \vec{e}_{ba} \quad (a < b), & \quad \text{eigenvalue } +1, \\ \vec{e}_{ab} - \vec{e}_{ba} \quad (a < b), & \quad \text{eigenvalue } -1. \end{aligned}$$

There are $r(r+1)/2$ vectors with eigenvalue $+1$ and $r(r-1)/2$ with eigenvalue -1 . All entries are in $\{0, 1, -1\}$.

For each position $s \in [Q]$, choose one such vector $\vec{\psi}_s$, and define $\vec{\psi} \in \mathbb{Q}^{\mathcal{B}}$ coordinatewise by

$$\psi_u = \prod_{s=1}^Q \psi_s(a_u(s), d_u(s)).$$

These $D = (r^2)^Q$ choices give an orthogonal basis: inner products factor into the inner products at the Q positions. The swap Π_s acts on $\vec{\psi}$ by the eigenvalue of $\vec{\psi}_s$. If exactly j of the choices have eigenvalue -1 , then

$$W_0 \vec{\psi} = j \vec{\psi}, \quad V_0 \vec{\psi} = (-1)^j \vec{\psi}, \quad M_0 \vec{\psi} = (-1)^j f(j) \vec{\psi}.$$

Thus we make M_0 fix these vectors by setting $f(j) = (-1)^j$.

For a uniformly chosen basis vector, the number j has distribution

$$W \sim \text{Bin}(Q, p), \quad p = \frac{r-1}{2r},$$

because at each position a fraction p of the r^2 choices have eigenvalue -1 . Most basis vectors therefore have j near Qp . Choose an interval $\mathcal{I}$ of R consecutive integers near Qp , and interpolate $f(j) = (-1)^j$ for $j \in \mathcal{I}$. This gives $\deg f \leq R-1$ and makes M_0 fix every basis vector with $j \in \mathcal{I}$. Step 5 specifies the endpoints and checks that the interval lies in $\{0, \dots, Q\}$.

Measuring the exceptional directions. Use the transposes of the remaining basis vectors as the rows of C . Their entries are in $\{0, 1, -1\}$, and by orthogonality, $\ker C$ is spanned by the basis vectors fixed above. This proves the first condition in (2.14).

We can now see how to choose R . The standard deviation of W is $\sqrt{Qp(1-p)} = \Theta(\sqrt{Q})$. Taking R much larger than $\sqrt{Q}$ puts most basis vectors inside the interval, so only a small fraction need to be measured. We can still take R much smaller than Q , so that Taylor recovery needs only $1 + \lfloor 3nR/Q \rfloor = o(n)$ coefficients per point. For the choices in Step 5, the binomial tail bound gives

$$c = D \Pr[W \notin \mathcal{I}] \leq 2D \exp\left(-\frac{R^2}{8Q}\right) \leq \frac{D\tau}{3r},$$

as required. Thus low degree gives restricted support, while interpolation makes the correction $I_D - M_0$ depend on only these few measurements.

Step 3: Define the auxiliary replacement rows.

We now turn the decomposition of ζ_u from Step 2 into its replacement. Keep the term determined by measurements unchanged. For each $M_0[u, w] \neq 0$, multiply its coefficient in the other term by

$$\varphi_{u,w}(T) = \prod_{s \in E_{u,w}} (1 - T/s).$$

This polynomial equals one at zero and vanishes at every point where ζ_w lies outside the assigned group. Define the coefficient row $\vec{P}_u(T)$ by the identity

$$\langle \vec{P}_u(T), \vec{\zeta} \rangle = ((I_D - M_0)\vec{\zeta})_u + \sum_{\substack{w \in \mathcal{B} \\ M_0[u,w] \neq 0}} M_0[u, w] \varphi_{u,w}(T) \zeta_w. \quad (2.16)$$

This directly gives all three required properties:

- At $T = 0$, every multiplier is one, so the two terms sum to ζ_u .
- Each multiplier has degree $|E_{u,w}| \leq R - 1$, so $\deg \vec{P}_u \leq R - 1$.
- At $T = t$, the first term is a linear function of $C\vec{\zeta}$. In the sum, a coefficient can survive only if $a_w(t) = d_u(t)$, so only coordinates in $\mathcal{S}_{t,d_u(t)}$ remain.

Thus the replacement uses exactly the kinds of data allowed in (2.13).

Step 4: Substitute the original variables.

The auxiliary rows are now constructed. To use them for the formula, choose schedules

$$a_v \in [r]^Q \quad (v \in [n]), \quad d_e \in [r]^Q \quad (e \in \mathcal{F}).$$

At point t , put variable v in group $a_v(t)$ and assign clause e to group $d_e(t)$:

$$S_{t,g} = \{v : a_v(t) = g\}, \quad \sigma_e(t) = d_e(t).$$

The occurrence (e, v) uses the auxiliary label

$$u(e, v) = ((a_v(s), d_e(s)))_{s \in [Q]} \in \mathcal{B}.$$

Let h be the largest number of occurrences receiving any one label. We need both small variable groups and small h . For our three-occurrence, width-three formula, Lemma 4.5 chooses the schedules deterministically in polynomial time so that

$$|S_{t,g}| \leq 2n/r, \quad h \leq 2M/D,$$

provided

$$12D^2 \leq M, \quad 2Qr^2 \leq n. \quad (2.17)$$

The idea is to derandomize a random choice of schedules. If all a_v and d_e are chosen independently and uniformly from $[r]^Q$, each label has expected multiplicity M/D , and each group $S_{t,g}$ has expected size n/r . Under the assumptions in (2.17), the standard method of conditional expectations derandomizes this choice in polynomial time, ensuring that every label has multiplicity at most $2M/D$ and every group has size at most $2n/r$. Step 5 verifies these assumptions for our parameter choices.

Different occurrences can receive the same label while requiring different original variables at that coordinate. Use h copies of $\mathcal{B}$, placing occurrences with the same label in different copies. Let $j(e, v) \in [h]$ denote the copy used by (e, v) ; put X_v at its coordinate $u(e, v)$ and zero in unused positions. Write $\vec{\zeta}^{(j)}(\vec{X})$ for the resulting filled vector in copy j . Every entry is zero or an original variable. Repeated appearances of X_v use the same value, so these copies introduce no new Boolean variables.

Apply the auxiliary row to the copy containing the occurrence:

$$\langle \vec{P}_{e,v}(T), \vec{X} \rangle = \left\langle \vec{P}_{u(e,v)}(T), \vec{\zeta}^{(j(e,v))}(\vec{X}) \right\rangle. \quad (2.18)$$

At zero the right side reads the coordinate containing X_v , and substituting variables preserves the degree bound $R-1$. At point t , its freely used coordinates contain only variables v' with $a_{v'}(t) = d_e(t)$. These are precisely the variables in the clause's assigned group $S_{t,d_e(t)}$. All occurrences of clause e use this same assigned group.

The remaining dependence is through the hc linear measurements $C\vec{\zeta}^{(j)}(\vec{X})$, over the h copies. These may involve several original groups. Fixing a total such as $x_1 + x_2$ would still couple two groups containing x_1 and x_2 . We therefore split each measurement into its contributions from individual groups. For each measurement $\ell(\vec{X})$, write

$$\ell(\vec{X}) = \sum_{g=1}^r \ell|_{S_{t,g}}(\vec{X}|_{S_{t,g}}),$$

and collect the coefficient rows of these restrictions into $C_{t,g}$. Every auxiliary measurement is now a sum of group measurements. For a clause assigned to group g , the part from $S_{t,g}$ is a local linear form, and every other part is a linear combination of the corresponding group's measurements. This is the condition in (2.5).

There are rhc rows in total. Since $hD \leq 2M \leq 6n$,

$$\sum_{g=1}^r c_{t,g} = rhc \leq rh \frac{D\tau}{3r} \leq 2\tau n.$$

Each row of C has D entries in $\{0, 1, -1\}$. Substituting variables, combining repeated copies, and restricting to a group cannot increase the sum of absolute coefficients. Hence every row of $C_{t,g}$ is integral with absolute coefficient sum at most D . Together with the group-size bound, this proves that the replacement forms admit an (r, τ, D) -design at every point.

Step 5: Verify the promised parameter bounds.

The construction needs the two balancing inequalities (2.17) and the tail bound $2e^{-R^2/(8Q)} \leq \tau/(3r)$. We now choose parameters of the orders stated in (2.10) and verify these requirements.

Put $\lambda = \lceil \log_2 n \rceil$ and $\nu = \lceil \log_2 \lambda \rceil$, and choose

$$\begin{aligned} r &= \left\lfloor \frac{\sqrt{\lambda}}{\nu^2} \right\rfloor, & Q &= \left\lfloor \frac{\lambda}{128\nu} \right\rfloor, & R &= 4\lceil \sqrt{\lambda} \rceil, \\ \tau &= \frac{1}{r\lambda}, & D &= r^{2Q}. \end{aligned} \quad (2.19)$$

These have the orders stated in (2.10). For sufficiently large n , we have $r \geq 2$, $R \geq 32$, and $6R \leq Q$, since $R/Q = O(\nu/\sqrt{\lambda}) \rightarrow 0$. We check the remaining requirements in turn:

- *Interpolation interval.* With $p = (r-1)/(2r)$ as in Step 2, take

$$A_0 = \left\lfloor Qp - \frac{R-1}{2} \right\rfloor, \quad \mathcal{I} = \{A_0, \dots, A_0 + R-1\}.$$

Since $Q/4 \leq Qp \leq Q/2$ and $6R \leq Q$, this interval lies in $\{0, \dots, Q\}$. Moreover, $R \geq 32$ ensures that it contains every integer within $R/4$ of Qp . The binomial tail bound therefore gives the estimate $\Pr[W \notin \mathcal{I}] \leq 2e^{-R^2/(8Q)}$ used in Step 2.

- *Polynomial dimension and balanced schedules.* Since $\log_2 r \leq \nu/2$,

$$\log_2 D = 2Q \log_2 r \leq \lambda/128, \quad D \leq (2n)^{1/128}.$$

For sufficiently large n , this gives

$$12D^2 \leq 12(2n)^{1/64} \leq n \leq M, \quad 2Qr^2 = O(\lambda^2) \leq n,$$

as required for Step 4.

- *Few measurements.* We have $R^2/(8Q) \geq 256\nu$. Since $\lambda \leq 2^\nu$ and $r^2 \leq \lambda$,

$$2e^{-R^2/(8Q)} \leq 2e^{-256\nu} \leq \frac{1}{3\lambda^2} \leq \frac{1}{3r^2\lambda} = \frac{\tau}{3r}.$$

Thus Step 2 supplies the auxiliary measurement bound used in Step 4.

Finally, all construction dimensions are polynomially bounded in n . Lemma 4.6 gives polynomial construction time for the matrices and rows in Steps 2–3, and Lemma 4.5 does so for the schedules in Step 4. The transfer uses only $hD \leq 6n$ coordinate positions. We have therefore constructed, in polynomial time, replacement forms that restore every original input at zero, have degree at most $R - 1 \leq R$, and admit the promised (r, τ, D) -designs at every $t \in [Q]$. This supplies the bounds used in Section 2.6; Lemma 4.8 records the parameter check for the more general setting of the full proof.

2.8. A weaker algorithm for general $\#k$ -SAT. The algorithm above already gives a subexponential algorithm for general $\#k$ -SAT: after disjoint sparsification, we reduce each instance directly to a 3-CNF in which every variable occurs at most three times. Both steps preserve the count, as follows.

Fix $k \geq 2$. We use the counting-preserving version [DHM⁺14, Appendix A] of sparsification [IPZ01], with quantitative bounds from [CIP06, Section 6] and [Cal09, Lemma 5.14 and equation (5.4)]. As recorded in Lemma 6.3, this expresses the input count as the sum of the counts of at most $\text{poly}_k(n, \sigma^{-1})2^{\sigma n}$ formulas G . Each G has the same n declared variables, with at most

$$B = O_k\left((\sigma^{-1} \log(2/\sigma))^{k-1}\right)$$

occurrences per variable, and hence at most Bn nonempty clauses. An empty clause gives count zero immediately.

First turn G into a 3-CNF without changing its count. Keep clauses of size at most three. For each longer clause, compute its disjunction with binary OR gates and require the final output to be one. For a gate $z = u \vee v$, impose

$$(\neg u \vee z) \wedge (\neg v \vee z) \wedge (u \vee v \vee \neg z).$$

Here u, v are input literals or earlier gate outputs, and z is fresh. These clauses enforce equality with the OR, so the gate values are uniquely determined by the original assignment. Consequently, a satisfying assignment of G has exactly one extension to the new formula, and an assignment not satisfying G has none. This uses $O(k)$ gates per clause.

Next reduce the occurrences. If a variable x occurs $s \geq 2$ times in this 3-CNF, replace its occurrences by distinct copies $x_1, \dots, x_s$, preserving their signs, and add Tovey's implication cycle [Tov84, p. 86]:

$$(\neg x_1 \vee x_2) \wedge \dots \wedge (\neg x_{s-1} \vee x_s) \wedge (\neg x_s \vee x_1).$$

The cycle forces all copies to agree. Each copy appears in one clause inherited from the 3-CNF and two cycle clauses, hence in exactly three clauses. Variables with at most one occurrence are kept unchanged, including unused declared variables. This standard variable-copy reduction [Gol08,

Proposition 4.7] therefore also gives a bijection between satisfying assignments. The final formula $\widehat{G}$ has $\#\widehat{G} = \#G$ and

$$n \leq N := \#\{\text{variables of } \widehat{G}\} = O_k(Bn).$$

All its clauses have size at most three, including the two-literal cycle clauses, and every variable occurs at most three times. Thus it is exactly an instance of the setting of this overview.

Apply (2.12) to each $\widehat{G}$ and add the counts. When σ^{-1} is bounded by a fixed power of $\log n$, so is B , and the resulting time is

$$(n + m)^{O_k(1)} 2^{O_k(\sigma n + Bn(\log \log n)^2 / \sqrt{\log n})}.$$

For example, choosing σ within a constant factor of $(\log n)^{-1/(2k)}$ yields

$$(n + m)^{O_k(1)} 2^{O_k(n(\log \log n)^{k+1} / (\log n)^{1/(2k)})} = (n + m)^{O_k(1)} 2^{o(n)}$$

with polynomial space, processing the sparsification tree and the reduced instances one at a time. The parameter σ may be chosen rational.

2.9. A faster algorithm for $\#k$ -SAT via counting solutions to sparse low-degree polynomial equations over $\mathbb{F}_2$. The running-time exponent in Section 2.8 is $O_k(\sigma n + B\rho(n))$, where $\rho(n) = n(\log \log n)^2 / \sqrt{\log n}$. If sparsification gave $B = O_k(1)$ at $\sigma = \Theta(\rho(n)/n)$, we would already have a $2^{O_k(\rho(n))}$ -time algorithm for general $\#k$ -SAT up to polynomial factors: copying would use only $O_k(n)$ variables per leaf, and the number of leaves would fit within the same bound. At this choice of σ , however, the guarantee is only $B = (\log n)^{O_k(1)}$.

We obtain the $2^{O_k(\rho(n))}$ bound in two steps. First, the counting algorithm of this overview extends to systems of $O(n)$ polynomial equations over $\mathbb{F}_2$, each of degree at most k with polylogarithmically many monomials (Theorem 2.2). Second, even if a leaf G has Bn clauses, we can construct such a system on the original n variables whose common zeros are exactly the satisfying assignments of G . We illustrate the reduction first with a self-contained randomized construction, then explain the counting algorithm.

A randomized reduction to sparse polynomial equations. Write $v_e(x) = 1$ if x violates clause e , and 0 otherwise. A *parity test* on a set S of clauses is the polynomial equation $\sum_{e \in S} v_e(x) = 0$ over $\mathbb{F}_2$: it accepts if x violates an even number of clauses in S . Every satisfying assignment passes, so we need only catch every nonsatisfying one. If G has $m_G \leq n$ clauses, one test per clause suffices. Suppose $m_G > n$.

A short test can easily miss a single violated clause, but all assignments that violate only that clause give identical answers. Thus there are only m_G singleton patterns to catch. Use $8n$ independent tests. For each, draw $L = 2B \lceil \log_2(2m_G) \rceil$ clauses uniformly with replacement. For each draw, independently add its violation bit to the parity sum with probability $1/2$; two included copies cancel. Since $m_G \leq Bn$, each clause is drawn at least logarithmically often in expectation overall. For a fixed set of $w \geq 1$ violated clauses, let $u = Lw/m_G$ be the expected number of hits per test. If the sample hits the set, the coin on the first hit makes the parity odd with probability $1/2$, even with all other coins fixed. Thus the rejection probability is

$$p_w = \frac{1}{2} (1 - (1 - w/m_G)^L) \geq \frac{1}{2} (1 - e^{-u}).$$

We count *sets* for few violations and *assignments* for many:

- If $u \leq 1$, then $p_w \geq u/4$. There are at most m_G^w sets of size w , so the chance that at least one escapes all tests is at most $m_G^w e^{-8np_w} \leq m_G^w e^{-2nLw/m_G} \leq 16^{-w}$. Summing over $w \geq 1$ gives failure probability at most $1/15$.
- If $u > 1$, then $p_w > 1/4$. A union bound over the 2^n assignments bounds the chance that any assignment in this case escapes by $2^n (3/4)^{8n} < 1/4$.

Thus with probability greater than $2/3$, the common zeros are exactly the satisfying assignments of G . Each v_e is a product of at most k factors of the form x_j or $1 + x_j$, so each test has degree at most k and at most $2^k L$ monomials, polylogarithmic in n . Section 6 constructs short tests with the same solution set deterministically. We first use parity checks from an expander code to ensure that any nonsatisfying assignment still passing must violate many clauses. A random short test then catches each such assignment with probability at least $1/4$. For each candidate test, we count the assignments that pass both that test and all tests chosen so far. We append a candidate that minimizes this count: every satisfying assignment passes every candidate, so this eliminates at least a quarter of the remaining nonsatisfying assignments.

Counting the common zeros. Recall from Section 2.2 that $\Phi_e(x)$ is 1 if x satisfies clause e and 0 otherwise. What should replace this factor when we test a whole set S_i of clauses at once? First count how many clauses in *this test* are violated, using ordinary integer addition:

$$L_i(x) = \sum_{e \in S_i} v_e(x) = \sum_{e \in S_i} (1 - \Phi_e(x)).$$

The test passes exactly when $L_i(x)$ is even. Choose $\ell = n + 1$, so that a count of at most 2^n is determined by its residue modulo 2^ℓ . The complemented Beigel–Tarui polynomial p_ℓ (Lemma 5.1) is 1 modulo 2^ℓ on even integers and 0 on odd integers. For example, violating two clauses in S_i still passes the test: $p_\ell(2) \equiv 1 \pmod{2^\ell}$. Writing m_{test} for the number of tests, our counting expression becomes

$$\underbrace{\sum_x \prod_{e \in G} \Phi_e(x)}_{\text{one factor per clause}} \quad \longrightarrow \quad \underbrace{\sum_x \prod_{i=1}^{m_{\text{test}}} p_\ell(L_i(x))}_{\text{one factor per test}}.$$

The left side is $\#G$. The least nonnegative residue of the right side modulo 2^ℓ is the number of assignments passing every test. When the tests catch every nonsatisfying assignment, this is exactly $\#G$. The same encoding works for any sparse polynomial equation $P_i(x) = 0$ over $\mathbb{F}_2$: take $L_i(x)$ to be the integer sum of its listed monomials. Again the equation holds exactly when this sum is even.

To reuse the Taylor step, we need a product of input degree $O(n)$. Multiplying the m_{test} test factors directly gives only the bound $O(km_{\text{test}}\ell)$, potentially quadratic in n . The key is to truncate the product *across all tests together*. Section 5.1 writes $p_\ell(y)$ as a sum of terms $e_j(y)h(y)^j$, where e_j has degree at most one and $h(y) = y(y-1)$ is even at every integer. A product term containing ℓ or more copies of h is therefore zero modulo 2^ℓ on Boolean assignments, even if those copies come from different tests. We keep just the terms with fewer than ℓ copies in total, tracking that number as we multiply. Each retained term has at most m_{test} linear factors and $\ell - 1$ quadratic factors in the L_i 's. Since $\deg L_i \leq k$, its input degree is at most $k(m_{\text{test}} + 2(\ell - 1)) = O(n)$.

For the replacement forms, what matters is which variables each factor reads. Where Section 2.3 used one form for each variable in a clause, we now use one for each variable in a test, reusing it throughout that test. Each sparse equation reads only polylogarithmically many variables. Branching on the values of the $O(n/\sqrt{\log n})$ most frequent variables leaves each remaining variable in only polylogarithmically many tests, as required by Section 4. On each branch, fixing the shared measurements again lets us sum one group at a time. Because the truncated product has input degree $O(n)$, the degree- R replacements give $\deg_T H = O(nR)$: the same $O(1 + nR/Q)$ Taylor coefficients per point recover the contributions at $T = 0$ with $0, 1, \dots, \ell - 1$ copies of h . Their sum modulo 2^ℓ , taken as the least nonnegative residue, is the exact branch count. Adding the branch counts gives the number of common zeros.

Theorem 2.2 (Counting common zeros of sparse polynomials). *Fix integers $k \geq 2$ and $c \geq 1$. Let $P_1, \dots, P_{m_{\text{test}}}$ be polynomials over $\mathbb{F}_2$ in $n \geq 20$ Boolean variables, with $m_{\text{test}} \leq cn$. Each P_i is given*

as a list of at most $(\log n)^c$ monomials of degree at most k . The exact number of common zeros in $\{0,1\}^n$ can be computed deterministically in $2^{O_{k,c}(n(\log \log n)^2/\sqrt{\log n})}$ bit operations and $n^{O_{k,c}(1)}$ space.

Sections 3 and 4 supply the counting routine and the replacement forms; Section 5 uses them to prove the theorem directly for polynomial equations.

For the final system constructed for G , the number of common zeros is $\#G$. Section 6 uses only $2^{(\log n)^{O_k(1)}}$ calls to the counter per leaf, which preserves the running-time exponent in Theorem 2.2. Finally, choosing σ of order $(\log \log n)^2/\sqrt{\log n}$ in Section 6.4 keeps B polylogarithmic and the number of leaves within $2^{O_k(n(\log \log n)^2/\sqrt{\log n})}$. Their satisfying assignments are disjoint, so adding their counts gives the bound in Proposition 6.4.

3. A FAST COUNTING ALGORITHM VIA SHARED LINEAR SUMMARIES

Sections 3–5 prove Theorem 2.2. We begin with the sum from Section 2.2: for each Boolean assignment, multiply the factors, then add the products over all assignments. Here the factors can be any rational polynomials. We show how to compute the sum when the replacement forms and measurements are given. Section 4 constructs them using only the variables read by each factor. Section 5 supplies one factor per sparse polynomial equation.

3.1. Polynomial factors and the value to recover. Let $\mathcal{F}$ index the factors, and let $I_f \subseteq [n]$ be the variables read by Φ_f . With rational coefficients our target is

$$Z = \sum_{x \in \{0,1\}^n} \prod_{f \in \mathcal{F}} \Phi_f((x_v)_{v \in I_f}). \quad (3.1)$$

For clause satisfaction indicators, Z is the number of satisfying assignments. The proof also works with coefficients in any commutative $\mathbb{Q}$ -algebra $\mathcal{A}$: sums and products then take place in $\mathcal{A}$, and interpolation uses only rational scalars. Readers may take $\mathcal{A} = \mathbb{Q}$ throughout this section.

Different factors may use different replacements for the same variable. Give each incidence (f, v) its own indeterminate $Y_{f,v}$ and suppose

$$\deg_{\{Y_{f,v}\}} \left(\prod_{f \in \mathcal{F}} \Phi_f((Y_{f,v})_{v \in I_f}) \right) \leq d. \quad (3.2)$$

This is the degree of the whole product computed over $\mathcal{A}$. Choose replacement rows $\vec{P}_{f,v}(T)$ of degree at most R with $\vec{P}_{f,v}(0) = \delta_v$, and form

$$H(T) = \sum_{x \in \{0,1\}^n} \prod_{f \in \mathcal{F}} \Phi_f((\langle \vec{P}_{f,v}(T), x \rangle)_{v \in I_f}) \in \mathcal{A}[T]. \quad (3.3)$$

The replacements restore the inputs at zero and multiply the degree by at most R . Thus $H(0) = Z$ and $\deg_T H \leq d_H := dR$. Our application will have $d = O(n)$, as in the bounded-occurrence example.

3.2. The required rows, partitions, and measurements. We use the same design as in Section 2.4, with a group assignment for each factor in place of each clause. The final condition below records the coefficient bound needed for exact arithmetic. Here $\tilde{\xi}$ denotes extension by zero from a group to all n coordinates.

Definition 3.1. For the factor scopes $(I_f)_{f \in \mathcal{F}}$, let $r \geq 2$, $D \geq 1$, and $1 \leq R < Q$ be integers, and let $0 < \tau \leq 1/2$. We require the following data and properties.

- (1) For each $t \in [Q]$, a partition $[n] = S_{t,1} \sqcup \cdots \sqcup S_{t,r}$, allowing empty parts, with $|S_{t,g}| \leq 2n/r$.
- (2) For every factor $f \in \mathcal{F}$, a sequence $\sigma_f \in [r]^Q$. The value $\sigma_f(t)$ assigns factor f to a group at point t .

- (3) For every $t \in [Q]$ and $g \in [r]$, an integer matrix $C_{t,g} \in \mathbb{Z}^{[c_{t,g}] \times S_{t,g}}$, where $c_{t,g}$ is a nonnegative integer. Each row has sum of absolute values at most D , and

$$\sum_{g=1}^r c_{t,g} \leq 2\tau n.$$

For an assignment x , the entries of $C_{t,g}(x|_{S_{t,g}})$ are the measurements on group g .

- (4) For every $f \in \mathcal{F}$ and $v \in I_f$, a row $\vec{P}_{f,v}(T) \in \mathbb{Q}[T]^{1 \times n}$ with $\vec{P}_{f,v}(0) = \delta_v$ and $\deg \vec{P}_{f,v} \leq R$. For every $t \in [Q]$, $g \in [r] \setminus \{\sigma_f(t)\}$, and $\xi \in \mathbb{Q}^{S_{t,g}}$, require

$$C_{t,g}\xi = \vec{0} \implies \langle \vec{P}_{f,v}(t), \tilde{\xi} \rangle = 0,$$

where $\tilde{\xi} \in \mathbb{Q}^n$ is the extension of ξ by zero.

- (5) With $d_{\text{den}} = (R-1)Q!$, every coefficient of $d_{\text{den}}\vec{P}_{f,v}$ is an integer with $O(Q \log Q + \log D)$ bits, with an absolute implied constant.

To interpret the kernel condition, fix t , f , and $v \in I_f$, and put $g = \sigma_f(t)$. Write $x_h = x|_{S_{t,h}}$. If $x, x' \in \mathbb{Q}^n$ agree outside $S_{t,h}$ for some $h \neq g$ and satisfy $C_{t,h}x_h = C_{t,h}x'_h$, then $x'_h - x_h \in \ker C_{t,h}$. The condition gives

$$\langle \vec{P}_{f,v}(t), x' \rangle = \langle \vec{P}_{f,v}(t), x \rangle.$$

Applying this one group at a time shows that the tuple $(\langle \vec{P}_{f,v}(t), x \rangle)_{v \in I_f}$ is determined by x_g and $(C_{t,h}x_h)_{h \neq g}$. Fixing all the measurement vectors therefore makes the entire factor a function of x_g . This applies to every polynomial factor Φ_f , including factors with coefficients in $\mathcal{A}$.

Each measurement of a Boolean vector lies in $[-D, D] \cap \mathbb{Z}$. We bound the sum of absolute values of the matrix entries because the construction can place the same Boolean coordinate in several auxiliary positions. After those positions are identified, entries of $C_{t,g}$ need not belong to $\{0, 1, -1\}$. A rank bound alone would not control the number of possible measurements: the single row $(1, 2, 4, \dots, 2^{n-1})$ distinguishes all 2^n Boolean vectors.

3.3. Separating the sum by fixing measurements. Fixing the measurements makes every factor depend on only its assigned group. The following lemma states the resulting calculation over a commutative ring, so it applies equally to rational-valued factors and to factors in $\mathcal{A}$. The restrictions $x_i = 1$ will be needed for Taylor coefficients and are local to individual groups.

Lemma 3.2. *Let $[n] = S_1 \sqcup \dots \sqcup S_r$ be a partition, allowing empty parts. For each $g \in [r]$, let C_g be an integer matrix with c_g rows and columns indexed by S_g . Suppose each row has sum of absolute values at most an integer $D \geq 0$, and write $c = \sum_g c_g$. Let $\mathcal{A}$ be a commutative ring with identity, and let $A_g : \mathbb{Q}^n \rightarrow \mathcal{A}$ satisfy*

$$A_g(x + \xi) = A_g(x) \quad (x \in \mathbb{Q}^n, \xi \in \ker C_h, h \neq g),$$

where ξ is extended by zero outside S_h . For any $E \subseteq [n]$, the sum

$$Z_E = \sum_{\substack{x \in \{0,1\}^n \\ x_i = 1 \ (i \in E)}} \prod_{g=1}^r A_g(x)$$

can be evaluated by examining at most

$$2(2D+1)^c \sum_{g=1}^r 2^{|S_g|}$$

Boolean vectors, with at most one A_g evaluation, $\text{poly}(n, r, c, \log(D+2))$ additional bit operations, and $O(1)$ ring operations per vector. Apart from the evaluators' own storage and $O(1)$ ring elements, the procedure uses $\text{poly}(n, r, c, \log(D+2))$ bits of space.

Proof. For $y_g \in \mathbb{Z}^{c_g}$, define

$$\mathcal{X}_g(y_g) = \{u \in \{0, 1\}^{S_g} : C_g u = y_g\}, \quad \mathcal{Y} = \prod_{g=1}^r \{-D, \dots, D\}^{c_g}.$$

Every tuple of measurements of a Boolean assignment belongs to $\mathcal{Y}$, and $|\mathcal{Y}| = (2D + 1)^c$. We first identify the contribution of a fixed measurement tuple $\vec{y} = (y_1, \dots, y_r)$. If every $\mathcal{X}_g(y_g)$ is nonempty, let $\bar{x}_g \in \mathcal{X}_g(y_g)$ be a representative for each g , and put

$$\mathcal{X}_{g,E}(y_g) = \{u \in \mathcal{X}_g(y_g) : u_i = 1 \text{ for all } i \in E \cap S_g\}.$$

If $x_h \in \mathcal{X}_h(y_h)$ for every h , then $x_h - \bar{x}_h \in \ker C_h$. Applying the invariance hypothesis successively for $h \neq g$ gives

$$A_g\left(\sum_{h=1}^r x_h\right) = A_g\left(x_g + \sum_{h \neq g} \bar{x}_h\right).$$

The assignments with measurement tuple $\vec{y}$ and fixed bits indexed by E form $\prod_g \mathcal{X}_{g,E}(y_g)$. Define

$$\Sigma_g(\vec{y}) = \sum_{x_g \in \mathcal{X}_{g,E}(y_g)} A_g\left(x_g + \sum_{h \neq g} \bar{x}_h\right). \quad (3.4)$$

Distributivity shows that these assignments contribute $\prod_g \Sigma_g(\vec{y})$ to Z_E . If $\mathcal{X}_{g,E}(y_g)$ is empty, then $\Sigma_g(\vec{y}) = 0$. The representatives $\bar{x}_g$ themselves need not satisfy the fixed-bit conditions indexed by E . Every Boolean assignment has a unique measurement tuple, so

$$Z_E = \sum_{\vec{y} \in \mathcal{Y}} \prod_{g=1}^r \Sigma_g(\vec{y}),$$

where the product is taken to be zero if any $\mathcal{X}_g(y_g)$ is empty.

The algorithm evaluates this identity one measurement tuple at a time, starting with $Z = 0_{\mathcal{A}}$. For each $\vec{y} \in \mathcal{Y}$, a first scan of each $\{0, 1\}^{S_g}$ finds a representative $\bar{x}_g \in \mathcal{X}_g(y_g)$; if any such set is empty, the algorithm skips this tuple. Otherwise it initializes $p = 1_{\mathcal{A}}$. To compute $\Sigma_g(\vec{y})$, it initializes $s = 0_{\mathcal{A}}$ and scans $u \in \{0, 1\}^{S_g}$ again. Whenever

$$C_g u = y_g, \quad u_i = 1 \quad (i \in E \cap S_g),$$

the algorithm performs the update

$$s \leftarrow s + A_g\left(u + \sum_{h \neq g} \bar{x}_h\right).$$

At the end of this scan, $s = \Sigma_g(\vec{y})$, and the algorithm sets $p \leftarrow ps$. After all groups, it sets $Z \leftarrow Z + p$. After all tuples $\vec{y}$ have been processed, $Z = Z_E$.

For each $\vec{y}$, the scan finding the representatives and the scan computing the $\Sigma_g(\vec{y})$ each examine at most $\sum_g 2^{|S_g|}$ Boolean vectors. This gives the stated bound. Checking a vector's measurements and fixed bits uses $\text{poly}(n, r, c, \log(D + 2))$ bit operations: each matrix row has sum of absolute values at most D , so its partial sums have $O(\log(D + 2))$ bits. There is at most one factor evaluation per vector. The retained state is $\vec{y}$, $(\bar{x}_1, \dots, \bar{x}_r)$, g , u , s , p , and Z . The tuple $\vec{y}$ contains c integers in $[-D, D]$, the representatives contain n bits in total, and $s, p, Z \in \mathcal{A}$. $\square$

To evaluate (3.3) at a fixed t , group the transformed factors by $\sigma_f(t)$ and apply the lemma in $\mathcal{A}$ with $E = \emptyset$. Definition 3.1 gives its invariance hypothesis, so the sum is $H(t)$. As in the overview, the two exponential costs are the measurement choices and the Boolean assignments in one group.

3.4. Taylor coefficients from fixed Boolean substitutions. The Q values $H(1), \dots, H(Q)$ need not determine H . Set

$$q = \left\lfloor \frac{d_H}{Q} \right\rfloor, \quad J = q + 1. \quad (3.5)$$

Since $QJ > d_H$, it suffices to compute

$$\mathcal{H}_t(z) = \sum_{j=0}^{J-1} ([z^j]H(t+z))z^j \quad (t \in [Q]). \quad (3.6)$$

Hermite interpolation recovers H , and hence $Z = H(0)$, from these QJ coefficients in $\mathcal{A}$. Indeed, the rational Hermite matrix for polynomials of degree less than QJ is invertible over $\mathbb{Q}$. Its inverse acts on elements of $\mathcal{A}$ by rational linear combinations.

Fix t . The replacement forms at $t+z$ need not respect the design at t . Write

$$\Delta \vec{P}_{f,v}(z) = \vec{P}_{f,v}(t+z) - \vec{P}_{f,v}(t).$$

Every entry is divisible by z . For fixed $x \in \mathbb{Q}^n$, introduce independent indeterminates $\vec{X}$ and replace each input by

$$\langle \vec{P}_{f,v}(t), x \rangle + \langle \Delta \vec{P}_{f,v}(z), \vec{X} \rangle. \quad (3.7)$$

Let

$$\mathcal{G}_x(\vec{X}, z) = \prod_{f \in \mathcal{F}} \Phi_f \left(\langle \vec{P}_{f,v}(t), x \rangle + \langle \Delta \vec{P}_{f,v}(z), \vec{X} \rangle \right)_{v \in I_f}. \quad (3.8)$$

Setting $\vec{X} = x$ gives $\sum_x \mathcal{G}_x(x, z) = H(t+z)$. Holding $\vec{X}$ fixed instead makes every correction independent of x , so the design at t still separates the factors. Moreover, each occurrence of $\vec{X}$ carries a factor of z , giving

$$\deg_{\vec{X}}([z^j]\mathcal{G}_x(\vec{X}, z)) \leq j. \quad (3.9)$$

The next identity recovers the substitution $\vec{X} = x$ from fixed Boolean substitutions and restrictions on the assignments being summed.

Lemma 3.3. *Let $q \geq 0$ be an integer and let $\mathcal{A}$ be a commutative ring with identity. For each $x \in \{0, 1\}^n$, let $p_x(X_1, \dots, X_n)$ be a polynomial over $\mathcal{A}$ of degree at most q . Then*

$$\sum_{x \in \{0, 1\}^n} p_x(x) = \sum_{\substack{E \subseteq [n] \\ |E| \leq q}} \sum_{U \subseteq E} (-1)^{|E|-|U|} \sum_{\substack{x \in \{0, 1\}^n \\ x_i = 1 \ (i \in E)}} p_x(\mathbf{1}_U). \quad (3.10)$$

Here $\mathbf{1}_U$ is the indicator vector of U .

Proof. For fixed x , we replace every positive power of X_i by X_i . This leaves Boolean evaluations unchanged and does not increase the degree. By Möbius inversion on subsets, the resulting multilinear polynomial has the expansion

$$\tilde{p}_x(\vec{X}) = \sum_{E \subseteq [n]} c_{x,E} \prod_{i \in E} X_i, \quad c_{x,E} = \sum_{U \subseteq E} (-1)^{|E|-|U|} p_x(\mathbf{1}_U).$$

The degree bound gives $c_{x,E} = 0$ whenever $|E| > q$. Setting $\vec{X} = x$ and summing over x now gives the claimed identity: the monomial $\prod_{i \in E} x_i$ selects exactly the assignments with $x_i = 1$ for every $i \in E$. $\square$

Apply Lemma 3.3 to $[z^j]\mathcal{G}_x$ over $\mathcal{A}$ for each $j < J$. All coefficients can use the same pairs $U \subseteq E$, $|E| \leq q$, so compute them together in

$$\mathcal{R} = \mathcal{A}[z]/(z^J).$$

For a fixed pair (E, U) , let $\Phi_{f,U}(x; z)$ be factor f in (3.8) after setting $\vec{X} = \mathbf{1}_U$. Group these factors by their assignments:

$$A_{g,U}(x; z) = \prod_{f: \sigma_f(t)=g} \Phi_{f,U}(x; z). \quad (3.11)$$

Their product is $\mathcal{G}_x(\mathbf{1}_U, z)$. For $h \neq g$, changing x by a vector in $\ker C_{t,h}$ changes neither the original inputs of these factors nor their fixed corrections. Lemma 3.2 therefore computes the restricted sum with $x_i = 1$ for $i \in E$ by examining at most

$$2r(2D + 1)^{2\tau n} 2^{2n/r} \quad (3.12)$$

Boolean vectors. Combining all the pairs gives

$$H(t + z) \equiv \sum_{\substack{E \subseteq [n] \\ |E| \leq q}} \sum_{U \subseteq E} (-1)^{|E| - |U|} \sum_{\substack{x \in \{0,1\}^n \\ x_i = 1 \ (i \in E)}} \mathcal{G}_x(\mathbf{1}_U, z) \pmod{z^J}. \quad (3.13)$$

The group products and sums are formed in $\mathcal{R}$. The result is $\mathcal{H}_t(z)$; repeating at all t supplies the interpolation data. The algorithm never stores a polynomial in all n indeterminates: it sets $\vec{X} = \mathbf{1}_U$ first and stores only the J coefficients in $\mathcal{A}$.

3.5. The three exponential costs. For each $t \in [Q]$, the number of pairs $U \subseteq E \subseteq [n]$ with $|E| \leq q$ is

$$C_q = \sum_{s=0}^{\min(q,n)} 2^s \binom{n}{s}. \quad (3.14)$$

In particular, $C_0 = 1$ and $C_q = 3^n$ for $q \geq n$. For every $0 < \eta \leq 1$,

$$\eta^q C_q \leq \sum_{s=0}^n \binom{n}{s} (2\eta)^s = (1 + 2\eta)^n, \quad (3.15)$$

because $\eta^q \leq \eta^s$ for every retained index $s \leq q$.

Proposition 3.4 (Common counting routine). *Let $\mathcal{A}$ be a commutative $\mathbb{Q}$ -algebra. Suppose the product of the factors in (3.1), computed over $\mathcal{A}$, satisfies the degree bound (3.2), and that the replacement rows, partitions, and measurements satisfy Definition 3.1. With q, J as in (3.5), the routine computes Z by examining at most*

$$2Qr C_q (2D + 1)^{2\tau n} 2^{2n/r}$$

Boolean vectors. For each vector it evaluates at most $|\mathcal{F}|$ factors in $\mathcal{A}[z]/(z^J)$. The additional operations per vector and the final interpolation take polynomially many operations in $\mathcal{A}$ and over $\mathbb{Q}$, in the parameters $n, |\mathcal{F}|, D, Q, R, J, r$ and the total scope length. Apart from the factor evaluators' own storage, the routine retains only polynomially many algebra elements, rational numbers and index bits.

Proof. There are Q points, C_q pairs per point, and at most (3.12) Boolean vectors per pair. Forming each group product uses at most $|\mathcal{F}|$ factor evaluations. Addition and multiplication modulo z^J cost $O(J^2)$ operations on coefficients in $\mathcal{A}$. The preceding Taylor identity and interpolation prove correctness. Process the points, pairs, measurement tuples, and local assignments sequentially, retaining their current indices and accumulators. Together with the QJ interpolation coefficients, this gives the stated storage bound. Bit complexity additionally depends on the cost of evaluating the factors and on coefficient sizes; these are verified for the sparse polynomial application in Section 5.4. $\square$

When $d = O(n)$, we have $d_H = O(nR)$ and $q = O(nR/Q)$. Taking $\eta = R/Q$ in (3.15) gives

$$\log C_q \leq q \log(Q/R) + n \log(1 + 2R/Q) = O\left(n \frac{R}{Q} \log \frac{2Q}{R}\right).$$

Consequently, if factor evaluation and coefficient arithmetic have polynomial bit cost, the running time is a polynomial factor times

$$2^{O\left(n \left[\frac{R}{Q} \log \frac{2Q}{R} + \tau \log(2D+1) + \frac{1}{r} \right] \right)}. \quad (3.16)$$

These are exactly the three costs from the overview. The parameter choice in Lemma 4.8 makes their sum $O((\log \log n)^2 / \sqrt{\log n})$ when factor scopes and their maximum incidence are bounded by a fixed power of $\log n$. The routine also applies when d is larger; then one keeps the actual value $q = \lfloor dR/Q \rfloor$ in the bound of Proposition 3.4.

We have so far assumed that the design is supplied. The next section constructs it from the factor scopes. The remaining two sections use the resulting routine first to count the zeros of a supplied polynomial system and then to choose such a system for a formula.

4. CONSTRUCTING LOW-DEGREE REPLACEMENT FORMS

The counting routine needs one replacement form for each pair (f, v) where factor f reads variable v . For the application in Section 5, f is a polynomial equation and its scope is the union of the variable sets of its monomials. This section uses only those scopes. We construct the rows $\vec{P}_{f,v}(T)$ and measurement matrices $C_{t,g}$ in Definition 3.1, together with the partitions $S_{t,g}$ and factor assignments σ_f . The main difficulty is to keep $\deg \vec{P}_{f,v} \leq R$ with R much smaller than Q . The degree bound for H grows linearly with R , so a small R/Q reduces the number of Taylor coefficients needed at each point. Choosing the rows independently at $T = 0, 1, \dots, Q$ and interpolating would generally give degree Q . We instead construct their values at all these points together.

The construction first gives polynomial rows on a complete array of auxiliary coordinates. We then place each factor–variable pair in a copy of that array. Several positions can contain the same Boolean variable x_v ; their values are always equal, and they do not become independent variables in the counting sum.

4.1. The auxiliary coordinates. We first specify the larger coordinate set $\mathcal{B}$ on which the auxiliary rows $\vec{P}_u(T)$ will act, together with the incidence bounds needed to place the actual pairs (f, v) into copies of it. Let $(I_f)_{f \in \mathcal{F}}$ be the factor scopes, with $I_f \subseteq [n]$, and write

$$\begin{aligned} \mathcal{A} &= \{(f, v) : f \in \mathcal{F}, v \in I_f\}, & M &= |\mathcal{A}|, \\ |I_f| &\leq w \quad (f \in \mathcal{F}), & \#\{f : v \in I_f\} &\leq b \quad (v \in [n]). \end{aligned} \quad (4.1)$$

Thus w bounds the number of variables in a factor's scope, and b bounds the number of factors containing one variable. The scope bound w need not bound the degree of the factor polynomial. A factor with empty scope is constant and needs no auxiliary position; assign such a factor to group one at every point. If necessary, adjoin a distinct constant-one factor with scope $\{v\}$ for every $v \in [n]$. This changes neither the product nor the count, and increases the maximum incidence by at most one. With b denoting the bound after this addition, we assume

$$n \geq 1, \quad b, w \geq 1, \quad n \leq M \leq bn.$$

Throughout this section, $r \geq 2$ and $1 \leq R < Q$ are integers and $0 < \tau \leq 1/2$.

At each point $t \in [Q]$, the variable x_v will belong to one group, while a factor containing v may be assigned to another. We will choose

$$a_v \in [r]^Q \quad (v \in [n]), \quad d_f \in [r]^Q \quad (f \in \mathcal{F}),$$

where $a_v(t)$ specifies the group containing x_v and $d_f(t)$ specifies the group assigned to factor f . The position for (f, v) will have label (a_v, d_f) . In particular, all positions containing x_v will have the same first sequence a_v .

Before choosing a_v and d_f , take the complete set of possible labels:

$$\mathcal{B} = ([r]^Q)^2, \quad u = (a_u, d_u), \quad D = |\mathcal{B}| = r^{2Q}. \quad (4.2)$$

An auxiliary vector $\zeta \in \mathbb{Q}^{\mathcal{B}}$ has one rational coordinate ζ_u for each pair u . The label u stays fixed as ζ varies. For $t \in [Q]$ and $g \in [r]$, define

$$S_{t,g}^{\mathcal{B}} = \{u \in \mathcal{B} : a_u(t) = g\}. \quad (4.3)$$

For fixed t , these sets partition $\mathcal{B}$ into r sets of size D/r .

For each $u \in \mathcal{B}$, we seek a polynomial row $\vec{P}_u(T) \in \mathbb{Q}[T]^{1 \times \mathcal{B}}$ of degree at most $R - 1$ with $\vec{P}_u(0) = \delta_u$. At $T = t$, the value $\langle \vec{P}_u(t), \zeta \rangle$ should be determined by

$$\zeta|_{S_{t,d_u(t)}^{\mathcal{B}}} \quad \text{and} \quad C^{\mathcal{B}}\zeta,$$

where $C^{\mathcal{B}}$ is one integer matrix, independent of u, t , with entries in $\{0, 1, -1\}$ and few rows. Later, using $u_{f,v} = (a_v, d_f)$ will make every input of factor f use the group $d_f(t)$. The choices of a_v, d_f and the placement into copies are deferred to Sections 4.4 and 4.5, respectively.

4.2. The matrices M_0 and $C^{\mathcal{B}}$. To construct the rows $\vec{P}_u(T)$, we seek a matrix $M_0 \in \mathbb{Q}^{\mathcal{B} \times \mathcal{B}}$ that fixes $\ker C^{\mathcal{B}}$ and satisfies

$$M_0[u, y] \neq 0 \implies |\{s \in [Q] : a_y(s) \neq d_u(s)\}| \leq R - 1. \quad (4.4)$$

The identity

$$\zeta_u = ((I - M_0)\zeta)_u + \sum_{y \in \mathcal{B}} M_0[u, y]\zeta_y \quad (4.5)$$

explains both requirements. If M_0 fixes $\ker C^{\mathcal{B}}$, the first term is determined by $C^{\mathcal{B}}\zeta$; we will leave it independent of T . In the sum, multiply each nonzero $M_0[u, y]$ by a polynomial that is one at zero and vanishes at every t with $a_y(t) \neq d_u(t)$. The bound (4.4) permits degree at most $R - 1$. At $T = t$, the sum will then use only the coordinates $y \in S_{t,d_u(t)}^{\mathcal{B}}$.

For $s \in [Q]$, let $\vartheta_s : \mathcal{B} \rightarrow \mathcal{B}$ exchange the entries $a_u(s)$ and $d_u(s)$, leaving every other entry unchanged. Define its permutation matrix Π_s by

$$(\Pi_s \zeta)_u = \zeta_{\vartheta_s(u)}.$$

The exchanged pair $\vartheta_s(u)$ belongs to $\mathcal{B}$ because this set contains every pair of sequences in $[r]^Q$. The matrices Π_s commute and satisfy $\Pi_s^2 = I$. Their product

$$V_0 = \prod_{s=1}^Q \Pi_s$$

exchanges the complete sequences a_u and d_u . Thus $V_0[u, y] \neq 0$ implies $a_y(s) = d_u(s)$ for every s ; it satisfies (4.4) with no mismatches. We will modify V_0 so that the resulting matrix also fixes the kernel of a short list of integer measurements.

Define

$$W_0 = \frac{1}{2} \sum_{s=1}^Q (I - \Pi_s). \quad (4.6)$$

On a common eigenvector ψ of the Π_s , let j be the number of indices s for which $\Pi_s \psi = -\psi$. Then

$$W_0 \psi = j \psi, \quad V_0 \psi = (-1)^j \psi, \quad V_0 p(W_0) \psi = (-1)^j p(j) \psi$$

for every polynomial p . Therefore $V_0 p(W_0)$ fixes ψ whenever $p(j) = (-1)^j$. We will construct an explicit common eigenbasis in which the proportion of vectors with a given j is $\Pr[W = j]$ for $W \sim \text{Bin}(Q, (r-1)/(2r))$.

For an interval $\mathcal{I} = \{A_0, \dots, A_0 + R - 1\} \subseteq \{0, \dots, Q\}$, we can enforce the equalities $f_{\mathcal{I}}(j) = (-1)^j$ for $j \in \mathcal{I}$ with a polynomial of degree at most $R - 1$. We use the explicit interpolation formula

$$f_{\mathcal{I}}(z) = (-1)^{A_0} \sum_{j=0}^{R-1} (-2)^j \binom{z - A_0}{j}, \quad M_0 = V_0 f_{\mathcal{I}}(W_0), \quad (4.7)$$

where

$$\binom{z - A_0}{j} = \frac{1}{j!} \prod_{a=0}^{j-1} (z - A_0 - a)$$

and the empty product is one. For $0 \leq q < R$, the binomial theorem gives $f_{\mathcal{I}}(A_0 + q) = (-1)^{A_0} (1 - 2)^q = (-1)^{A_0 + q}$.

Lemma 4.1. *For $\mathcal{B}$, $\mathcal{I}$ and M_0 as above, there is a matrix $C^{\mathcal{B}} \in \{0, 1, -1\}^{[c_{\mathcal{B}}] \times \mathcal{B}}$ such that*

$$c_{\mathcal{B}} \leq D \Pr[W \notin \mathcal{I}], \quad W \sim \text{Bin}\left(Q, \frac{r-1}{2r}\right),$$

and $M_0 \xi = \xi$ for every $\xi \in \ker C^{\mathcal{B}}$.

Proof. For $(\alpha, \delta) \in [r]^2$, let $e_{\alpha\delta}$ be the indicator function of that ordered pair. Exchanging its two entries has the orthogonal eigenbasis

$$e_{\alpha\alpha} \quad (\alpha \in [r]), \quad e_{\alpha\delta} + e_{\delta\alpha}, \quad e_{\alpha\delta} - e_{\delta\alpha} \quad (1 \leq \alpha < \delta \leq r).$$

The first two types have eigenvalue $+1$, and the third has eigenvalue -1 . There are $r(r+1)/2$ vectors with eigenvalue $+1$ and $r(r-1)/2$ with eigenvalue -1 .

For each $s \in [Q]$, choose one of these functions $\psi_s : [r]^2 \rightarrow \{0, 1, -1\}$, and define

$$\psi(u) = \prod_{s=1}^Q \psi_s(a_u(s), d_u(s)) \quad (u \in \mathcal{B}).$$

The $D = r^{2Q}$ choices give nonzero orthogonal vectors: their inner products factor into the inner products of the functions ψ_s . They form a basis of $\mathbb{Q}^{\mathcal{B}}$ with entries in $\{0, 1, -1\}$. If exactly j of the chosen ψ_s have eigenvalue -1 , then $W_0 \psi = j\psi$ and $V_0 \psi = (-1)^j \psi$. Hence M_0 fixes ψ whenever $j \in \mathcal{I}$.

To make $\ker C^{\mathcal{B}}$ a space fixed by M_0 , we define the rows of $C^{\mathcal{B}}$ to be the transposes of the basis vectors with $j \notin \mathcal{I}$. A fraction $(r-1)/(2r)$ of the choices at each position have eigenvalue -1 , independently across positions. The fraction of the D basis vectors retained as rows is therefore $\Pr[W \notin \mathcal{I}]$, proving the row bound. If $C^{\mathcal{B}} \xi = \vec{0}$, then ξ is orthogonal to every retained basis vector. It is a linear combination of the basis vectors with $j \in \mathcal{I}$, all of which M_0 fixes. $\square$

For a bound $\epsilon > 0$, the condition

$$\Pr\left[\text{Bin}\left(Q, \frac{r-1}{2r}\right) \notin \mathcal{I}\right] \leq \epsilon \quad (4.8)$$

thus gives $c_{\mathcal{B}} \leq \epsilon D$. After placing the factor inputs, we will use $\epsilon = \tau/(rb)$. We next prove the other requirement on M_0 .

Lemma 4.2. *For all $u, y \in \mathcal{B}$, if $M_0[u, y] \neq 0$, then*

$$|\{s \in [Q] : a_y(s) \neq d_u(s)\}| \leq R - 1.$$

Proof. Expand $f_{\mathcal{I}}(W_0)$ in the commuting matrices Π_s . Since $\deg f_{\mathcal{I}} \leq R - 1$ and $\Pi_s^2 = I$, each resulting monomial is a scalar multiple of $\prod_{s \in J} \Pi_s$ for some $J \subseteq [Q]$ with $|J| \leq R - 1$. Multiplication by V_0 gives

$$V_0 \prod_{s \in J} \Pi_s = \prod_{s \notin J} \Pi_s.$$

A nonzero (u, y) entry of this permutation matrix requires $a_y(s) = d_u(s)$ for every $s \notin J$. There are at most $R - 1$ other positions. If more than $R - 1$ positions disagree, every summand has zero (u, y) entry, so $M_0[u, y] = 0$ as well. $\square$

4.3. The polynomial rows. We now use M_0 to construct the auxiliary rows $\vec{P}_u(T)$ sought in Section 4.1. Lemma 4.1 makes $(I - M_0)\zeta$ a function of $C^{\mathcal{B}}\zeta$. Lemma 4.2 bounds the number of points at which a nonzero term $M_0[u, y]\zeta_y$ uses a coordinate outside $S_{t, d_u(t)}^{\mathcal{B}}$. We now multiply that term by a polynomial vanishing at those points and equal to one at zero.

For $M_0[u, y] \neq 0$, define

$$E_{u, y} = \{s \in [Q] : a_y(s) \neq d_u(s)\}, \quad \varphi_{u, y}(T) = \prod_{s \in E_{u, y}} \left(1 - \frac{T}{s}\right). \quad (4.9)$$

The empty product is one. Then $\deg \varphi_{u, y} \leq R - 1$, $\varphi_{u, y}(0) = 1$, and $\varphi_{u, y}(t) = 0$ whenever $a_y(t) \neq d_u(t)$. Define

$$\vec{P}_u(T) = \delta_u(I - M_0) + \sum_{\substack{y \in \mathcal{B} \\ M_0[u, y] \neq 0}} M_0[u, y]\varphi_{u, y}(T)\delta_y. \quad (4.10)$$

This row acts as a linear form in ζ , with coefficients that are polynomials in the single parameter T .

Lemma 4.3. *The rows in (4.10) have degree at most $R - 1$ and satisfy $\vec{P}_u(0) = \delta_u$ for every $u \in \mathcal{B}$. For $t \in [Q]$ and $g \in [r]$, let $C_{t, g}^{\mathcal{B}}$ be the restriction of $C^{\mathcal{B}}$ to the columns $S_{t, g}^{\mathcal{B}}$. For every u, t, g with $g \neq d_u(t)$ and every $\xi \in \mathbb{Q}^{S_{t, g}^{\mathcal{B}}}$, its extension $\tilde{\xi}$ by zero to $\mathcal{B}$ satisfies*

$$C_{t, g}^{\mathcal{B}}\xi = \vec{0} \implies \langle \vec{P}_u(t), \tilde{\xi} \rangle = 0.$$

More generally, $\langle \vec{P}_u(t), \zeta \rangle$ is determined by $\zeta|_{S_{t, d_u(t)}^{\mathcal{B}}}$ and $C^{\mathcal{B}}\zeta$.

Proof. The degree bound follows from that of $\varphi_{u, y}$. At zero,

$$\vec{P}_u(0) = \delta_u(I - M_0) + \delta_u M_0 = \delta_u.$$

Applying (4.10) to ζ gives

$$\langle \vec{P}_u(T), \zeta \rangle = ((I - M_0)\zeta)_u + \sum_{\substack{y \in \mathcal{B} \\ M_0[u, y] \neq 0}} M_0[u, y]\varphi_{u, y}(T)\zeta_y. \quad (4.11)$$

At $T = t$, all summands with $a_y(t) \neq d_u(t)$ vanish. If $\tilde{\xi}$ is supported on $S_{t, g}^{\mathcal{B}}$ with $g \neq d_u(t)$, each remaining summand is zero on $\tilde{\xi}$. The hypothesis $C_{t, g}^{\mathcal{B}}\xi = \vec{0}$ also gives $C^{\mathcal{B}}\tilde{\xi} = \vec{0}$, so $(I - M_0)\tilde{\xi} = \vec{0}$ by Lemma 4.1. This proves the implication.

For the final assertion, two vectors with the same measurements under $C^{\mathcal{B}}$ differ by an element of its kernel. Thus $(I - M_0)\zeta$ is determined by $C^{\mathcal{B}}\zeta$. The remaining terms in (4.11) at $T = t$ use only $\zeta|_{S_{t, d_u(t)}^{\mathcal{B}}}$. $\square$

Fix t and write $\zeta_g = \zeta|_{S_{t, g}^{\mathcal{B}}}$. The restrictions of $C^{\mathcal{B}}$ satisfy

$$C^{\mathcal{B}}\zeta = \sum_{g=1}^r C_{t, g}^{\mathcal{B}}\zeta_g. \quad (4.12)$$

Fixing each vector $C_{t,g}^{\mathcal{B}}\zeta_g$ therefore fixes $C^{\mathcal{B}}\zeta$ while imposing separate conditions on the r auxiliary coordinate groups. When all vectors $C_{t,g}^{\mathcal{B}}\zeta_g$ are fixed, $\langle \vec{P}_u(t), \zeta \rangle$ is determined by $\zeta|_{S_{t,d_u(t)}^{\mathcal{B}}}$. We will obtain the corresponding separation on the actual Boolean variables by placing (f, v) at a coordinate labeled (a_v, d_f) .

Remark 4.4. We illustrate how (4.10) confines dependence outside the assigned group to one linear measurement. Take $Q = R = 2$, outside the regime $R < Q$ used for the running-time bound, and $r = 2$. Use $\{0, 1\}$ as the group indices and $\mathcal{I} = \{0, 1\}$, so $f_{\mathcal{I}}(z) = 1 - 2z$. Starting with the label $(00, 11)$, the two exchanges generate four coordinates:

$$\zeta_0 = \zeta_{(00,11)}, \quad \zeta_1 = \zeta_{(10,01)}, \quad \zeta_2 = \zeta_{(01,10)}, \quad \zeta_{12} = \zeta_{(11,00)}.$$

Both exchanges preserve this set of four coordinates, so we can restrict the calculation to it. On this restriction, $M_0 = \Pi_1 + \Pi_2 - \Pi_1\Pi_2$ fixes the kernel of

$$c(\zeta) = \zeta_0 - \zeta_1 - \zeta_2 + \zeta_{12}.$$

The row indexed by $u = (00, 11)$ gives

$$\langle \vec{P}_u(T), \zeta \rangle = \zeta_0 - \frac{T}{2}\zeta_1 - T\zeta_2.$$

It equals ζ_0 at zero, while

$$\langle \vec{P}_u(1), \zeta \rangle = c(\zeta) + \frac{1}{2}\zeta_1 - \zeta_{12}, \quad \langle \vec{P}_u(2), \zeta \rangle = c(\zeta) - \zeta_2 - \zeta_{12}.$$

At $t = 1$, the labels of ζ_1, ζ_{12} satisfy $a_y(1) = 1 = d_u(1)$. At $t = 2$, the labels of ζ_2, ζ_{12} satisfy $a_y(2) = 1 = d_u(2)$. Fixing $c(\zeta)$ therefore leaves $\langle \vec{P}_u(t), \zeta \rangle$ dependent only on coordinates in its assigned group. Here $c(\zeta)$ plays the role of $C^{\mathcal{B}}\zeta$ on the four-coordinate restriction. The full construction must give this property for every $u \in \mathcal{B}$ with at most ϵD measurement rows.

4.4. Choosing the variable and factor groups. We now choose the families $(a_v)_{v \in [n]}$ and $(d_f)_{f \in \mathcal{F}}$ from Section 4.1, and set

$$u_{f,v} = (a_v, d_f) \quad ((f, v) \in \mathcal{A}).$$

These choices have two requirements. First, if a label occurs c_u times, its incidences need c_u different positions in copies of the auxiliary array. Thus the number of copies needed is

$$c_u = \#\{(f, v) \in \mathcal{A} : u_{f,v} = u\}, \quad h = \max_{u \in \mathcal{B}} c_u.$$

Second, the actual variable group $\{v : a_v(t) = g\}$ must be small. Bounding h alone would bound the number of incidences in a group, not give the required bound on its number of Boolean variables. The next lemma controls both quantities.

Lemma 4.5. *Let $(I_f)_{f \in \mathcal{F}}$ satisfy (4.1), with $n \leq M \leq bn$, and put $D = r^{2Q}$ for integers $r \geq 2$ and $Q \geq 1$. Suppose*

$$2(b+w)D^2 \leq M, \quad 2Qr^2 \leq n. \quad (4.13)$$

One can deterministically choose $a_v, d_f \in [r]^Q$, with $d_f(t) = 1$ for factors of empty scope, so that

$$\begin{aligned} \max_{u \in \mathcal{B}} \#\{(f, v) \in \mathcal{A} : (a_v, d_f) = u\} &\leq \frac{2M}{D}, \\ \#\{v \in [n] : a_v(t) = g\} &\leq \frac{2n}{r} \quad (t \in [Q], g \in [r]). \end{aligned}$$

The construction takes $\text{poly}(n, M, |\mathcal{F}|, D, Q, r, b, w)$ bit operations.

Proof. We first estimate the effect of random labels; we will then fix their values deterministically. For this estimate, all a_v and all d_f with nonempty I_f are independent and uniform on $[r]^Q$. For empty I_f , we set d_f identically one. Each incidence label $u_{f,v} = (a_v, d_f)$ is uniform on $\mathcal{B}$. Put $n_{t,g} = \#\{v : a_v(t) = g\}$, and consider

$$\Phi = \sum_{u \in \mathcal{B}} (c_u - M/D)^2, \quad \Xi = \sum_{t=1}^Q \sum_{g=1}^r (n_{t,g} - n/r)^2.$$

Their expansions as sums of indicators are

$$\begin{aligned} \Phi &= \sum_{(f,v),(f',v') \in \mathcal{A}} \mathbb{1}_{\{(a_v, d_f) = (a_{v'}, d_{f'})\}} - \frac{M^2}{D}, \\ \Xi &= \sum_{t=1}^Q \sum_{v, v' \in [n]} \mathbb{1}_{\{a_v(t) = a_{v'}(t)\}} - \frac{Qn^2}{r}. \end{aligned}$$

Two incidence labels are independent unless they share a factor or a variable. For each (f, v) , there are at most $b + w$ incidences sharing f or v , including (f, v) itself. Independent pairs have equality probability $1/D$; the other pairs have probability at most one. Hence

$$\mathbb{E}\Phi \leq M(b + w).$$

For each t and $v \neq v'$, the values $a_v(t), a_{v'}(t)$ are independent and uniform on $[r]$. The second expansion therefore gives

$$\mathbb{E}\Xi = Qn(1 - 1/r) \leq Qn.$$

To obtain the labels deterministically, we fix the a_v and the d_f for nonempty I_f one at a time. At each step we choose a value that does not increase the conditional expectation of

$$\frac{\Phi}{M(b + w)} + \frac{\Xi}{Qn}.$$

Such a choice exists because the current expectation is the average over the possible values of the next a_v or d_f . The final value is at most two. Since both summands are nonnegative,

$$c_u \leq \frac{M}{D} + \sqrt{2M(b + w)}, \quad n_{t,g} \leq \frac{n}{r} + \sqrt{2Qn}.$$

The two inequalities in (4.13) make the square-root terms at most M/D and n/r , respectively.

The following enumeration computes the conditional expectations in $\text{poly}(n, M, |\mathcal{F}|, D, Q, r, b, w)$ bit operations. Put $K = r^Q$, so $D = K^2$. Each indicator in the expansion of Φ depends on at most the four independent choices $a_v, d_f, a_{v'}, d_{f'}$; repeated indices contribute just one choice. Each indicator in Ξ depends on at most $a_v, a_{v'}$. After any partial fixing, enumerate at most $K^4 = D^2$ assignments to the remaining choices in one indicator. Its conditional probability has denominator dividing K^4 . Summing the $M^2 + Qn^2$ indicators computes both expectations. Including the normalizations $M(b + w)$ and Qn , all rational values in these calculations have numerators and denominators with

$$O(\log(2D) + \log(2M) + \log(2n) + \log(2Q) + \log(2 + b + w))$$

bits. There are at most $n + |\mathcal{F}|$ choices to fix, each with K candidate values. Since $K^4 = D^2$, the entire construction takes $\text{poly}(n, M, |\mathcal{F}|, D, Q, r, b, w)$ bit operations. $\square$

4.5. Placing copies of the variables in the auxiliary arrays. We now transfer the auxiliary rows $\vec{P}_u(T)$ and matrix $C^{\mathcal{B}}$ to the rows $\vec{P}_{f,v}(T)$ and matrices $C_{t,g}$ on the original variables required by Definition 3.1. Assume (4.13) and apply Lemma 4.5. With $h = \max_u c_u$, index h copies of the auxiliary coordinates by $[h] \times \mathcal{B}$. For each $(f, v) \in \mathcal{A}$, we assign $c(f, v) \in [h]$ so that all pairs $(c(f, v), u_{f,v})$ are distinct. This is possible because no label occurs more than h times, and

$$hD \leq 2M \leq 2bn.$$

Define the linear map $\iota : \mathbb{Q}^n \rightarrow \mathbb{Q}^{[h] \times \mathcal{B}}$ by

$$(\iota x)_{c,u} = \begin{cases} x_v, & (c, u) = (c(f, v), u_{f,v}) \text{ for some } (f, v) \in \mathcal{A}, \\ 0, & \text{otherwise.} \end{cases}$$

The occupied positions are distinct, so this definition is unambiguous. Different positions can contain the same x_v , and unused positions are fixed at zero. For $c \in [h]$, define the projection

$$\pi_c : \mathbb{Q}^{[h] \times \mathcal{B}} \rightarrow \mathbb{Q}^{\mathcal{B}}, \quad (\pi_c \eta)_u = \eta_{c,u}.$$

Now put

$$\begin{aligned} \vec{P}_{f,v}(T) &= \vec{P}_{u_{f,v}}(T) \pi_{c(f,v)} \iota, \\ S_{t,g} &= \{v \in [n] : a_v(t) = g\}, \quad \sigma_f(t) = d_f(t). \end{aligned} \tag{4.14}$$

The row $\vec{P}_{f,v}(T)$ acts on the actual variables $x \in \mathbb{Q}^n$. At $T = 0$, it extracts the value in position $(c(f, v), u_{f,v})$, which is x_v . Thus $\vec{P}_{f,v}(0) = \delta_v$, and its degree is at most $R - 1$. Lemma 4.5 gives $|S_{t,g}| \leq 2n/r$.

Define the measurement matrices by

$$\tilde{C} = \text{diag}(\underbrace{C^{\mathcal{B}}, \dots, C^{\mathcal{B}}}_{h \text{ copies}}), \quad \hat{C} = \tilde{C} \iota, \quad C_{t,g} = \hat{C}|_{S_{t,g}}. \tag{4.15}$$

The restriction keeps the columns in $S_{t,g}$ and all rows, including zero rows. Thus $\hat{C}x$ records the auxiliary measurements in each copy after its coordinates have been filled from x . The matrices $C_{t,g}$ are integral. Their entries need not belong to $\{0, 1, -1\}$, because coefficients at several positions containing the same x_v are added. Each row of $\tilde{C}$ has at most D nonzero entries, all equal to 1 or -1 . Multiplication by ι adds entries in a common column and discards unused positions, so it cannot increase the sum of their absolute values. Consequently every row of $C_{t,g}$ has absolute row sum at most D , and its value on a Boolean assignment lies in $[-D, D] \cap \mathbb{Z}$.

If (4.8) holds, each $C_{t,g}$ has $c_{t,g} = hc_{\mathcal{B}}$ rows. Hence

$$\sum_{g=1}^r c_{t,g} = rhc_{\mathcal{B}} \leq rh\epsilon D \leq 2r\epsilon M \leq 2r\epsilon bn. \tag{4.16}$$

Setting $\epsilon = \tau/(rb)$ makes this at most $2\tau n$.

To verify separation, fix $t \in [Q]$, $g \in [r]$ and $(f, v) \in \mathcal{A}$ with $g \neq \sigma_f(t)$. Suppose $\xi \in \mathbb{Q}^{S_{t,g}}$ satisfies $C_{t,g}\xi = \vec{0}$, and extend it by zero to $\tilde{\xi} \in \mathbb{Q}^n$. Equation (4.15) gives $\tilde{C}\tilde{\xi} = \vec{0}$. Thus, for every $c \in [h]$,

$$C^{\mathcal{B}} \pi_c \tilde{\xi} = \vec{0}.$$

Every nonzero coordinate of $\pi_c \tilde{\xi}$ contains a variable v' with $a_{v'}(t) = g$. Its auxiliary label has first sequence $a_{v'}$, so this vector is supported on $S_{t,g}^{\mathcal{B}}$. Lemma 4.3, applied in copy $c(f, v)$ with $d_{u_{f,v}}(t) = d_f(t) = \sigma_f(t)$, gives

$$\langle \vec{P}_{f,v}(t), \tilde{\xi} \rangle = \langle \vec{P}_{u_{f,v}}(t), \pi_{c(f,v)} \tilde{\xi} \rangle = 0.$$

This is the kernel condition in Definition 3.1. All inputs of factor f use the same assigned group $d_f(t)$, and the counting sum still ranges over exactly the n Boolean bits.

4.6. Coefficient bounds and the completed construction. The substitution rows $\vec{P}_{f,v}(T) \in \mathbb{Q}[T]^{1 \times n}$ in (4.14) have polynomial entries with rational coefficients. We now bound the bits needed to store those coefficients; these bounds will control exact arithmetic in Section 5.4. We first give a common denominator and coefficient bounds for the auxiliary rows $\vec{P}_u(T)$ defined in (4.10). Passing from $\vec{P}_u(T)$ to $\vec{P}_{f,v}(T)$ by (4.14) only adds coefficients in common columns or discards unused columns, so the same denominator and bound on the sum of absolute values apply.

Lemma 4.6. *Let $d_{\text{den}} = (R-1)!Q!$. Then $d_{\text{den}}\vec{P}_u(T) \in \mathbb{Z}[T]^{1 \times D}$ for every $u \in \mathcal{B}$. The sum of the absolute values of all coefficients in each row $d_{\text{den}}\vec{P}_u(T)$ is at most $2^{O(Q \log Q)}$. The same bound holds for $d_{\text{den}}\vec{P}_{f,v}(T)$. The matrices $M_0, C^{\mathcal{B}}$ and the rows $\vec{P}_u(T)$ can be constructed in $\text{poly}(D, Q, r)$ bit operations.*

Proof. For $0 \leq j < R$, the corresponding summand of $f_{\mathcal{I}}(W_0)$ equals

$$(-1)^{A_0}(-2)^j \binom{W_0 - A_0 I}{j} = \frac{(-1)^{A_0+j}}{j!} \prod_{a=0}^{j-1} (2W_0 - 2(A_0 + a)I).$$

Each matrix factor on the right is integral because $2W_0 = QI - \sum_s \Pi_s$. Also $j!$ divides $(R-1)!$. Multiplication by the permutation matrix V_0 therefore shows that $(R-1)!M_0$ is integral. The coefficients of $\varphi_{u,y}$ have denominators dividing $\prod_{s \in E_{u,y}} s$, which divides $Q!$. Equation (4.10) proves the asserted common denominator.

For the coefficient estimates, we use the matrix norm $\|A\|_{\infty} = \max_i \sum_j |A[i, j]|$ and the polynomial norm $\|\sum_j a_j T^j\|_1 = \sum_j |a_j|$. Since $\|W_0\|_{\infty} \leq Q$ and $0 \leq A_0 + a \leq Q$ for all factors above, each $2W_0 - 2(A_0 + a)I$ has norm at most $4Q$. It follows that

$$\|M_0\|_{\infty} \leq R(4Q)^{R-1}, \quad \|\varphi_{u,y}\|_1 \leq \prod_{s \in E_{u,y}} (1 + 1/s) \leq 2^Q.$$

The sum of the absolute values of all coefficients in the row $d_{\text{den}}\vec{P}_u$ is therefore at most

$$d_{\text{den}}(1 + (1 + 2^Q)R(4Q)^{R-1}).$$

Its logarithm is $O(Q \log Q)$ because $R < Q$. For $u = u_{f,v}$, multiplication of $d_{\text{den}}\vec{P}_u(T)$ by $\pi_{c(f,v)}^{\iota}$ adds coefficients in common columns and discards unused columns. By (4.14), the resulting row is $d_{\text{den}}\vec{P}_{f,v}(T)$, so it satisfies the same bound.

Enumerating the eigenbasis in Lemma 4.1, evaluating $f_{\mathcal{I}}(W_0)$, and forming the polynomials $\varphi_{u,y}$ require $\text{poly}(D, Q, r)$ arithmetic operations. Using the common denominators above, the coefficients stored during these matrix and polynomial calculations have numerators and denominators with $O(Q \log Q)$ bits. Thus the construction takes $\text{poly}(D, Q, r)$ bit operations. $\square$

The construction has three quantitative requirements. The two inequalities in (4.13) bound the number of auxiliary copies and the number of actual variables in each group. The tail bound with $\epsilon = \tau/(rb)$ gives the total measurement bound through (4.16). We collect the result for use in the counting algorithm.

Lemma 4.7. *Let $n, b, w \geq 1$ be integers, and let $(I_f)_{f \in \mathcal{F}}$ be subsets of $[n]$ of size at most w , with each $v \in [n]$ belonging to at most b sets. Put $M = \sum_f |I_f|$ and assume $n \leq M \leq bn$. Let $r \geq 2$ and $1 \leq R < Q$ be integers, let $0 < \tau \leq 1/2$, and put $D = r^{2Q}$. Suppose*

$$2(b+w)D^2 \leq M, \quad 2Qr^2 \leq n,$$

and an interval $\mathcal{I} = \{A_0, \dots, A_0 + R - 1\} \subseteq \{0, \dots, Q\}$ satisfies

$$\Pr[W \notin \mathcal{I}] \leq \frac{\tau}{rb}, \quad W \sim \text{Bin}\left(Q, \frac{r-1}{2r}\right).$$

One can deterministically construct all the data in Definition 3.1 in $\text{poly}(n, M, |\mathcal{F}|, D, Q, r, b, w)$ bit operations.

Proof. Lemma 4.1 constructs C^B with $c_B \leq \tau D/(rb)$. Lemmas 4.2 and 4.3 give the auxiliary polynomial rows. Lemma 4.5 chooses the variable and factor groups with $hD \leq 2M$ and $|S_{t,g}| \leq 2n/r$. The maps in (4.14) and (4.15) then give rows with $\tilde{P}_{f,v}(0) = \delta_v$, degree at most $R-1 \leq R$, and the required kernel implication. Every row of $C_{t,g}$ has absolute row sum at most D , and (4.16) gives $\sum_{g=1}^r c_{t,g} \leq 2\tau n$ for each $t \in [Q]$. Lemma 4.6 supplies the common denominator and coefficient bounds. Together, the constructions take $\text{poly}(n, M, |\mathcal{F}|, D, Q, r, b, w)$ bit operations, and the resulting arrays occupy $\text{poly}(n, M, |\mathcal{F}|, D, Q, r, b, w)$ bits. $\square$

4.7. Choosing the parameters. In the application, both the largest scope w and the maximum number b of factors containing a variable will be bounded by a fixed power of $\log n$. We now choose parameters satisfying Lemma 4.7 in that regime. The exponent of the counting cost depends on the three quantities

$$\frac{R}{Q} \log \frac{2Q}{R}, \quad \tau \log(2D+1), \quad \frac{1}{r}.$$

They account respectively for recovering Taylor coefficients, enumerating measurement values, and enumerating assignments in one group. The following choices bound all three at once.

Lemma 4.8. *Fix an integer $\beta \geq 1$. Let $n \geq 20$, put $\lambda = \lceil \log_2 n \rceil$ and $\nu = \lceil \log_2 \lambda \rceil$, and suppose $b, w \leq \lambda^\beta$ and $n \leq M \leq bn$. For all sufficiently large n , the parameters*

$$r = \left\lfloor \frac{\sqrt{\lambda}}{\nu^2} \right\rfloor, \quad Q = \left\lfloor \frac{\lambda}{128\nu} \right\rfloor, \quad R = (\beta+3)\lceil \sqrt{\lambda} \rceil, \quad (4.17)$$

$$\tau = \frac{1}{r\lambda}, \quad \epsilon = \frac{1}{br^2\lambda}, \quad D = r^{2Q},$$

together with the interval specified in the proof, satisfy all numerical hypotheses of Lemma 4.7. They also satisfy

$$R \geq 32, \quad 6R \leq Q, \quad D \leq 2^{\lambda/128},$$

and

$$\frac{R}{Q} \log \frac{2Q}{R} + \tau \log(2D+1) + \frac{1}{r} = O_\beta \left(\frac{\nu^2}{\sqrt{\lambda}} \right). \quad (4.18)$$

Given the numerical values n, M, b, w , the parameters and numerical hypotheses can be computed and checked exactly in $\text{poly}_\beta(\log n)$ bit operations. For fixed β , only finitely many n fail these checks.

Proof. For sufficiently large n , the parameters have orders

$$r = \Theta(\sqrt{\lambda}/\nu^2), \quad Q = \Theta(\lambda/\nu), \quad R = \Theta_\beta(\sqrt{\lambda}).$$

In particular, eventually $r \geq 2$, $R \geq 32$, and $6R \leq Q$. Since $\log_2 r \leq \nu/2$,

$$\log_2 D = 2Q \log_2 r \leq 2 \frac{\lambda}{128\nu} \frac{\nu}{2} = \frac{\lambda}{128}.$$

The two balance inequalities follow for sufficiently large n :

$$2(b+w)D^2 \leq 4\lambda^\beta 2^{\lambda/64} \leq n \leq M, \quad 2Qr^2 \leq n.$$

For the second, Q, r are bounded by fixed powers of λ , whereas $n > 2^{\lambda-1}$. The first follows from the same comparison.

Put

$$\mu = \frac{Q(r-1)}{2r}, \quad A_0 = \left\lfloor \mu - \frac{R-1}{2} \right\rfloor, \quad \mathcal{I} = \{A_0, \dots, A_0 + R-1\}.$$

Here $Q/4 \leq \mu \leq Q/2$. Since $R \leq Q/6$, the interval lies in $\{0, \dots, Q\}$. Its center differs from μ by less than one, and $R \geq 32$ ensures that it contains every integer within $R/4$ of μ . For $W \sim \text{Bin}(Q, (r-1)/(2r))$, Hoeffding's inequality gives

$$\Pr[W \notin \mathcal{I}] \leq 2e^{-R^2/(8Q)} \leq 2e^{-16(\beta+3)^2\nu} \leq 2^{-(\beta+2)\nu} \leq \lambda^{-(\beta+2)} \leq \frac{1}{br^2\lambda}.$$

The last inequality uses $b \leq \lambda^\beta$ and $r^2 \leq \lambda$. This proves (4.8) with $\epsilon = \tau/(rb)$. Also $0 < \tau \leq 1/2$ once $r \geq 2$.

Finally,

$$\frac{R}{Q} = O_\beta(\nu/\sqrt{\lambda}), \quad \log(2Q/R) = O(\nu), \quad \tau \log(2D+1) = O(1/r).$$

Together with $1/r = O(\nu^2/\sqrt{\lambda})$, these prove (4.18).

We finish by showing that the parameters in (4.17) can be constructed exactly within the stated bit bound. The algorithm finds $\lambda = \lceil \log_2 n \rceil$ as the least integer with $n \leq 2^\lambda$, and then finds $\nu = \lceil \log_2 \lambda \rceil$ in the same way. It obtains r as the largest integer satisfying $r^2\nu^4 \leq \lambda$, computes Q by integer division, and finds $\lceil \sqrt{\lambda} \rceil$ by an integer square-root search to obtain R . If $r < 2$ or $Q \leq R$, the algorithm reports failure at this stage. Otherwise it constructs τ, ϵ, D and the endpoints of $\mathcal{I}$ using integer or rational arithmetic. It then checks $\mathcal{I} \subseteq \{0, \dots, Q\}$, the two balance inequalities, $R \geq 32$, and $6R \leq Q$. The tail probability in (4.8) has denominator $(2r)^Q$, so its required bound can also be checked by integer arithmetic. These computations take $\text{poly}_\beta(\log n)$ bit operations. The preceding estimates show that all checks pass for sufficiently large n . $\square$

5. COUNTING COMMON ZEROS OF SPARSE POLYNOMIAL EQUATIONS

We now prove Theorem 2.2. The input is a system $\mathcal{P}$ of $m_{\text{test}} \leq cn$ polynomial equations $P_i(x) = 0$ over $\mathbb{F}_2$ in n Boolean variables. Each P_i is a list of at most $(\log n)^c$ monomials of degree at most k , for fixed k, c . Our target is the integer

$$N(\mathcal{P}) = \#\{x \in \{0, 1\}^n : P_i(x) = 0 \text{ for every } i\}. \quad (5.1)$$

First we build a product with one factor per equation whose value on the Boolean cube records the common-zero indicator. The difficulty is keeping its total input degree linear in n . Next we construct the replacement forms for these factors and use Section 3 to sum the product. Section 6 uses the resulting algorithm for arbitrary supplied systems, including systems that still accept nonsatisfying assignments of a formula.

5.1. An integer polynomial for one equation. On Boolean inputs we may replace each positive power x_v^j by x_v . Combine duplicate monomials modulo two. If $\mathcal{M}_i$ is the resulting list of variable sets for P_i , define its integer lift by

$$L_i(\vec{X}) = \sum_{M \in \mathcal{M}_i} \prod_{v \in M} X_v.$$

The empty monomial is 1 and the empty sum is 0. The value $L_i(x)$ counts the monomials equal to one, so $P_i(x) = 0$ precisely when $L_i(x)$ is even. Write $I_i = \bigcup_{M \in \mathcal{M}_i} M$. Although L_i can read $k(\log n)^c$ variables, it still has degree at most k .

Set $\ell = n + 1$. Since $0 \leq N(\mathcal{P}) \leq 2^n < 2^\ell$, it suffices to obtain its residue modulo 2^ℓ . We first represent the evenness indicator by an integer polynomial modulo this power of two. Multiplying m_{test} such indicators directly would give only a quadratic degree bound; the form of the next lemma will let us control the degree across all equations together.

Put $h(Y) = Y(Y - 1)$. Its value is even at every integer Y . Writing the indicator in powers of h exposes one factor of two for each copy of h in the product.

Lemma 5.1. *For every integer $\ell \geq 1$, one can construct polynomials $e_0, \dots, e_{\ell-1} \in \mathbb{Z}[Y]$, each of degree at most one, such that*

$$p_\ell(Y) = \sum_{j=0}^{\ell-1} e_j(Y) h(Y)^j, \quad p_\ell(y) \equiv \mathbf{1}_{\{2|y\}} \pmod{2^\ell} \quad (y \in \mathbb{Z}). \quad (5.2)$$

The coefficients of the e_j have $\text{poly}(\ell)$ bits, and the construction uses $\text{poly}(\ell)$ bit operations and $\text{poly}(\ell)$ bits of space.

Proof. Let F_ℓ be the modulus-amplifying polynomial of Beigel and Tarui [BT94, Section 2.3]. Chan and Williams apply F_ℓ to sums of clause-violation indicators to count satisfying assignments deterministically [CW21, Theorem 2.5 and proof of Theorem 1.6]. We need the indicator of even parity, so we take its complement:

$$p_\ell(Y) = 1 - F_\ell(Y) = (1 - Y)^\ell \sum_{j=0}^{\ell-1} \binom{\ell+j-1}{j} Y^j. \quad (5.3)$$

The finite sum is the inverse of $(1 - Y)^\ell$ modulo Y^ℓ . Indeed, the coefficient of Y^j in $(1 + Y + Y^2 + \dots)^\ell$ counts weak compositions of j into ℓ parts and equals $\binom{\ell+j-1}{j}$. Therefore

$$p_\ell(Y) \equiv 1 \pmod{Y^\ell}, \quad p_\ell(Y) \equiv 0 \pmod{(1 - Y)^\ell}.$$

For even y , the first congruence gives $p_\ell(y) \equiv 1 \pmod{2^\ell}$. For odd y , the second gives $p_\ell(y) \equiv 0 \pmod{2^\ell}$.

The polynomial p_ℓ has integer coefficients and degree less than 2ℓ . Successive division by the monic quadratic $h = Y^2 - Y$ writes it uniquely as $\sum_{j < \ell} e_j h^j$, with integer remainders e_j of degree less than two. Lemma B.1 in Appendix B bounds their coefficients and proves the $\text{poly}(\ell)$ bit-operation bound for this construction. $\square$

5.2. Keeping the product degree linear. The full product $\prod_{i=1}^{m_{\text{test}}} p_\ell(L_i(\vec{X}))$ has degree at most $km_{\text{test}}(2\ell - 1)$. Each factor $h(y)$ contributes a factor of two at integer arguments, so a term containing at least ℓ such factors can be discarded modulo 2^ℓ . This lets us retain only terms whose total power of h is less than ℓ , across all equations together. Introduce a formal variable V to record that power, using the same V for every equation. Its exponent records a guaranteed power of two dividing the corresponding term at integer arguments. It is not an additional Boolean variable to be summed over. This is where we choose the coefficient algebra of Section 3.1:

$$\mathcal{A}_\ell = \mathbb{Q}[V]/(V^\ell).$$

In this algebra every multiplication discards powers V^s with $s \geq \ell$. Once the product is formed, we add its retained coefficients using the linear map

$$\mathcal{S}_\ell \left(\sum_{s=0}^{\ell-1} b_s V^s \right) = \sum_{s=0}^{\ell-1} b_s.$$

This map need not preserve products, so it is applied only after the factors have been multiplied. We use the same notation for polynomial coefficients b_s .

Lemma 5.2. *Let $m_{\text{test}} \geq 0$ and $\ell \geq 1$ be integers. For integers $y_1, \dots, y_{m_{\text{test}}}$, put*

$$B_i(V) = \sum_{j=0}^{\ell-1} e_j(y_i) h(y_i)^j V^j.$$

Then

$$\mathcal{S}_\ell \left(\prod_{i=1}^{m_{\text{test}}} B_i(V) \bmod V^\ell \right) \equiv \prod_{i=1}^{m_{\text{test}}} \mathbf{1}_{\{2|y_i\}} \pmod{2^\ell}.$$

For any nonnegative integer d_* , if the y_i are instead polynomials over $\mathbb{Q}$ of total degree at most d_* , then each retained coefficient of V^s , and hence their sum, has total input degree at most $d_*(m_{\text{test}} + 2(\ell - 1))$.

Proof. Before truncation, a term contributing to the coefficient of V^s has the form

$$\prod_{i=1}^{m_{\text{test}}} e_{j_i}(y_i) h(y_i)^{j_i}, \quad \sum_{i=1}^{m_{\text{test}}} j_i = s.$$

At integer arguments it is divisible by 2^s . Dropping every coefficient with $s \geq \ell$ therefore leaves the sum of coefficients unchanged modulo 2^ℓ . Without truncation, that sum is $\prod_i p_\ell(y_i)$, which has the required congruence by Lemma 5.1.

For polynomial arguments, every retained term has $\sum_i j_i \leq \ell - 1$. Since $\deg e_{j_i} \leq 1$ and $\deg h = 2$, its degree is at most

$$\sum_{i=1}^{m_{\text{test}}} d_*(1 + 2j_i) \leq d_*(m_{\text{test}} + 2(\ell - 1)).$$

The bound holds coefficient by coefficient, as well as for their sum. $\square$

To see why the truncation uses the total order across all equations, take $\ell = 2$. Then

$$p_2(Y) = 1 - 3Y^2 + 2Y^3 = (1 - Y) + (2Y - 1)h(Y).$$

For two equations, multiplication modulo V^2 , followed by $\mathcal{S}_2$, retains

$$(1 - y_1)(1 - y_2) + (2y_1 - 1)h(y_1)(1 - y_2) + (1 - y_1)(2y_2 - 1)h(y_2).$$

Its degree is at most four. The omitted term $(2y_1 - 1)(2y_2 - 1)h(y_1)h(y_2)$ has degree six, but is divisible by four at integer inputs.

5.3. Counting with supplied replacement forms. For equation i supply Section 3 with the factor

$$\mathcal{B}_i((X_v)_{v \in I_i}, V) = \sum_{j=0}^{\ell-1} e_j(L_i(\vec{X})) h(L_i(\vec{X}))^j V^j \quad \text{in } \mathcal{A}_\ell[(X_v)_{v \in I_i}]. \quad (5.4)$$

Within a factor, every occurrence of X_v uses the same replacement form; different factors may use different forms for X_v . Even if each factor receives its own independent inputs, L_i has degree at most k in those inputs. Lemma 5.2 bounds the input degree of the product in $\mathcal{A}_\ell$ by $d = k(m_{\text{test}} + 2(\ell - 1))$. Thus the product satisfies precisely the degree hypothesis (3.2) of the common routine. Adding factors equal to one with scope $\{v\}$ ensures that all declared variables occur in the construction.

Let $H(T) \in \mathcal{A}_\ell[T]$ be (3.3) for these factors. Its V -coefficients at $T = 0$ are integers because all replacements restore their Boolean inputs at zero. By the same lemma, summing their values gives

$$\deg_T H \leq d_H := kR(m_{\text{test}} + 2(\ell - 1)), \quad \mathcal{S}_\ell(H(0)) \equiv N(\mathcal{P}) \pmod{2^\ell}. \quad (5.5)$$

Consequently the least nonnegative residue of $\mathcal{S}_\ell(H(0))$ is the exact count. Interpolation takes place over the rationals; the congruence is needed only on Boolean inputs at $T = 0$.

5.4. Cost of evaluating the factors.

Proposition 5.3. *Let a list of degree-at-most- k polynomial equations on n variables and a design satisfying Definition 3.1 for their factors be given. Let $S \geq 1$ bound the total length of the monomial lists and factor scopes, and use any ℓ with $2^\ell > 2^n$. With d_H from (5.5) and q, J from (3.5), $N(\mathcal{P})$ can be computed deterministically in*

$$\text{poly}(n, m_{\text{test}}, \ell, D, Q, R, J, r, k, S) C_q(2D + 1)^{2\tau n} 2^{2n/r}$$

bit operations and $\text{poly}(n, m_{\text{test}}, \ell, D, Q, R, J, r, k, S)$ bits of working space.

Proof. Apply Proposition 3.4 with coefficients in $\mathcal{A}_\ell$. At an evaluation point t , the routine uses $\mathcal{A}_\ell[z]/(z^J)$, so a stored element has just $J\ell$ rational coefficients. To evaluate factor i at a fixed Boolean substitution $\vec{X} = \mathbf{1}_U$ in (3.7), first evaluate each listed monomial by multiplying its at most k substituted inputs, then add the results to get L_i . Use this value in (5.4), successively multiplying by $h(L_i)$ to obtain its powers. This takes polynomially many rational operations in the displayed parameters and never enumerates assignments to the entire scope I_i . The polynomials e_j are computed once by Lemma 5.1.

For bit complexity, the rows have common denominator $d_{\text{den}} = (R-1)Q!$. Each substituted monomial has denominator dividing $\Delta = d_{\text{den}}^k$, every lifted factor has denominator dividing $\Gamma = \Delta^{2\ell-1}$, and their product has denominator dividing $\Omega = \Gamma^{m_{\text{test}}}$. Thus exponentially many summands can be accumulated over one denominator. Appendix B bounds the bit lengths in these sums by

$$B_{\text{sum}} = \text{poly}(n, m_{\text{test}}, \ell, k, Q, \log(D+2), \log(S+2)). \quad (5.6)$$

For each $0 \leq s < \ell$ we interpolate the rational polynomial $[V^s]H(T)$ from its QJ Taylor coefficients. The appendix verifies polynomial bit cost for this interpolation and the final integer remainder in (5.5). The $O(QJ\ell)$ stored coefficients, the design and the sequential enumerations in Proposition 3.4 give the stated space bound. $\square$

5.5. Constructing the forms and completing the proof. Each sparse polynomial reads few variables, but one variable might occur in many equations. We first fix the values of the few variables with the largest numbers of occurrences. On each resulting branch we can construct the design required by Proposition 5.3.

Proof of Theorem 2.2. Let $\mathcal{P}$ be the supplied polynomial system. An empty list of equations gives 2^n common zeros; a constant equation $1 = 0$ gives none. Discard equations $0 = 0$. Include a neutral factor equal to one with scope $\{v\}$ for each declared variable v , and write

$$M = \sum_{f \in \mathcal{F}} |I_f|$$

for the total number of factor–variable incidences. The input bounds imply $M = n(\log n)^{O_{k,c}(1)}$, and every scope has $(\log n)^{O_{k,c}(1)}$ variables. A particular variable, however, may occur in many of these scopes.

Put $\lambda = \lceil \log_2 n \rceil$ and

$$b_0 = \left\lceil \frac{M}{n} \right\rceil \lceil \sqrt{\lambda} \rceil.$$

Let $S \subseteq [n]$ be the variables belonging to more than b_0 scopes. For each $b \in \{0, 1\}^S$, we fix $x|_S = b$ and count the completions on $[n] \setminus S$; we call this restricted problem a branch. Since $|S| \leq M/b_0 \leq n/\sqrt{\lambda}$, there are at most $2^{n/\sqrt{\lambda}}$ branches. Each branch has the same number n' of free variables, with

$$n' \geq n - \frac{n}{\sqrt{\lambda}} \geq \frac{n}{2} \quad (n \geq 20).$$

Substitute the fixed bits in every P_i over $\mathbb{F}_2$ and combine duplicate monomials modulo two before forming its integer lift. Neither degree nor the number of monomials increases. A constant equation can again be handled immediately. The branches partition the Boolean cube, so their common-zero counts add to $N(\mathcal{P})$.

To apply Section 4, remove the fixed variables from the scopes and relabel the free variables by $[n']$. Write $I'_f \subseteq [n']$ for the restricted scopes and $M' = \sum_f |I'_f|$ for their total incidence count. Let b be the maximum factor incidence of a free variable and let $w = \max_f |I'_f|$. Then $b \leq b_0$, and both b and w are bounded by a fixed power of $\log n'$. The remaining neutral factors ensure $M' \geq n'$. Set

$$\lambda' = \lceil \log_2 n' \rceil, \quad \nu' = \lceil \log_2 \lambda' \rceil,$$

and choose the smallest integer $\beta \geq 1$ with $b, w \leq (\lambda')^\beta$. For fixed k, c , the possible values of β are bounded by a constant for all sufficiently large n . We apply Lemma 4.8 to the restricted scopes (I'_f) on $[n']$, with incidence count M' , using

$$r = \left\lfloor \frac{\sqrt{\lambda'}}{(\nu')^2} \right\rfloor, \quad Q = \left\lfloor \frac{\lambda'}{128\nu'} \right\rfloor, \quad R = (\beta + 3) \lceil \sqrt{\lambda'} \rceil.$$

If $r < 2$ or $Q \leq R$, use the exhaustive-search fallback below before forming τ , ϵ , or the parity interpolation interval $\mathcal{I}$. Otherwise set

$$\tau = \frac{1}{r\lambda'}, \quad \epsilon = \frac{1}{br^2\lambda'}, \quad D = r^{2Q},$$

and carry out the remaining feasibility checks from that lemma. For all sufficiently large n , Lemma 4.7 supplies the required rows and measurement matrices, with

$$r = \Theta\left(\frac{\sqrt{\log n'}}{(\log \log n')^2}\right), \quad Q = \Theta\left(\frac{\log n'}{\log \log n'}\right), \quad R = O_{k,c}(\sqrt{\log n'}), \quad D \leq (2n')^{1/128}.$$

On every branch we may use the original precision $\ell = n + 1$, since even the unrestricted count is smaller than 2^ℓ . As $n' \geq n/2$, both ℓ and m_{test} are $O_c(n')$. Thus $d_H = O_{k,c}(n'R)$, $J = O_{k,c}(n'R/Q) + 1$, and (3.16), in dimension n' , has exponent

$$O_{k,c}\left(\frac{n'(\log \log n')^2}{\sqrt{\log n'}}\right) = O_{k,c}\left(\frac{n(\log \log n)^2}{\sqrt{\log n}}\right).$$

All substitution data, coefficient arrays, and interpolation systems have total representation length $n^{O_{k,c}(1)}$ bits. Multiplying the per-branch running time by at most $2^{n/\sqrt{\lambda}}$ leaves the displayed exponential bound unchanged, so summing the branch answers gives the required running time.

To implement the construction for every input size, the algorithm checks its numerical hypotheses using integer arithmetic. In addition to the preliminary guards above, it checks $2(b+w)D^2 \leq M'$ and $2Qr^2 \leq n'$ from Lemma 4.5, the endpoints of $\mathcal{I}$, $R \geq 32$, $6R \leq Q$, and the binomial tail bound in (4.8), which can be tested exactly. If any check fails, the algorithm abandons the branched computation and checks the original equations directly on all 2^n assignments. For fixed k, c , the bounds just proved and Lemma 4.8 show that a failure can occur only for finitely many n . This fallback is therefore absorbed in the theorem's constants.

To combine the branch counts within the stated space bound, we process the branches one at a time. We initialize an integer accumulator $Z = 0$ and add each branch's answer to it. The construction and Proposition 5.3 use $n^{O_{k,c}(1)}$ bits of space on each branch. Disjointness gives $0 \leq Z \leq 2^n$ throughout, and the final value is $N(\mathcal{P})$, as claimed. $\square$

6. REDUCING $\#k$ -SAT TO COUNTING COMMON ZEROS

Given a sparse k -CNF F with at most Bn clauses, where B is polylogarithmic in n , we construct $O(n)$ sparse polynomial equations on its original n variables whose common zeros are exactly its satisfying assignments. One equation per clause could require Bn equations. Instead, each equation will test the parity of the violations of a short list of clauses. We use Theorem 2.2 both to choose these equations and to obtain the final count.

Let m be the number of indexed clauses and $v_e(x)$ the violation bit of clause e . As a polynomial over $\mathbb{F}_2$, v_e is a product of the factors $1 + x_v$ for positive literals and x_v for negative literals. A tautological clause has $v_e = 0$ and an empty clause has $v_e = 1$. After removing repeated literals, $\deg v_e \leq k$ and v_e expands to at most 2^k monomials. Put $v(x) = (v_1(x), \dots, v_m(x))$. For a binary matrix A with m columns, row i specifies the equation

$$P_i(x) := \sum_{e: A_{ie}=1} v_e(x) = 0 \quad \text{over } \mathbb{F}_2. \quad (6.1)$$

Write $\mathcal{P}_A$ for this system and $N(A) = N(\mathcal{P}_A)$ for its number of common zeros. Every satisfying assignment passes every row. Hence $N(A) = \#F + D(A)$, where $D(A)$ counts the nonsatisfying assignments still passing. We do not know $\#F$, but it is the same for every candidate next row: the row minimizing N also minimizes D .

We represent a test row by a list of pairs (e, c) with $e \in [m]$ and $c \in \{0, 1\}$, summing coefficients modulo 2 if a clause index is repeated. The length of this list counts all pairs, including repeated indices and pairs with coefficient zero. A row of list length s thus reads at most ks variables before cancellation.

6.1. Initial tests that exclude small violation sets. If an assignment violates only one of many clauses, a test sampling a short list of clauses is unlikely to notice it. We therefore first construct a matrix $A^{(0)}$ that rejects every nonzero violation vector with small support. Every nonsatisfying assignment that passes these initial tests must violate many clauses. A further test that samples a short list of clauses can then reject each such assignment with constant probability.

We use the standard edge-based Tanner code: bits are assigned to the edges of an expander, and the bits incident to each vertex must form a word of a prescribed local linear code (see Tanner [Tan81, Section II] for the local constraints and Sipser and Spielman [SS96, Sections IV and VI] for the expander construction). The usual local-distance and expansion argument gives the needed lower bound on the weight of a nonzero codeword. The proof below constructs the graph and local code deterministically, with parameters that meet our bounds on the number and length of the tests.

Lemma 6.1. *Let n, m, B be positive integers with $m \leq Bn$, and put*

$$\mu = \lfloor \log_2 m \rfloor + 1, \quad P = 20000B^2\mu^2, \quad D_0 = 320000B^2\mu^3.$$

There is a deterministic construction of a binary matrix $A^{(0)}$ with m columns and at most $3n$ rows such that every row has at most D_0 nonzero entries and

$$z \in \{0, 1\}^m \setminus \{\vec{0}\}, \quad A^{(0)}z = \vec{0} \pmod{2} \implies |\text{supp } z| \geq m/P.$$

The construction uses $\exp(\text{poly}(B, \log(m+2)))$ bit operations and $\text{poly}(n, m, B)$ bits of working space.

Proof. We construct a bipartite graph with between m and $2m$ edges, using its edges as temporary binary coordinates. Checks at each vertex will force every vertex touched by a nonzero edge to have many nonzero incident edges; a mixing bound will then force large total weight. At the end, we keep m edge coordinates and fix the rest to zero. Let

$$d_G = \min(m, D_0), \quad q_G = \min\{2^a : a \in \mathbb{Z}_{\geq 0}, 2^a d_G \geq m\}.$$

Then $m \leq q_G d_G < 2m$. Write $q_G = 2^{\ell_G}$ and take two copies of $\{0, 1\}^{\ell_G}$ as the vertex sets. For each left vertex u and each port $j \in [d_G]$, put an edge from u to $u \oplus s_j$, where $\oplus$ denotes bitwise addition modulo 2 and the shifts $s_1, \dots, s_{d_G} \in \{0, 1\}^{\ell_G}$ will be chosen below. Edges are indexed by the pairs (u, j) ; parallel edges remain distinct. Every vertex on either side has one incident edge at each port j , so the graph is d_G -regular.

To obtain the mixing estimate (6.3), we choose the shifts so that every parity of a nonempty set of coordinates is nearly balanced. For a nonzero vector $b \in \{0, 1\}^{\ell_G}$, the bit $b \cdot s_j \pmod{2}$ is the parity of those coordinates of s_j where b equals 1. With $\varepsilon = 1/(200B\mu)$, we require

$$\left| \sum_{j=1}^{d_G} (-1)^{b \cdot s_j} \right| \leq \varepsilon d_G \quad (b \in \{0, 1\}^{\ell_G} \setminus \{\vec{0}\}). \quad (6.2)$$

The sum in (6.2) is the number of indices j with even parity minus the number with odd parity. Thus, for every nonzero b , each parity occurs at a fraction of indices between $(1 - \varepsilon)/2$ and $(1 + \varepsilon)/2$. A list of shifts with this property is called ε -biased [NN93, Definitions 2.1–2.2].

If $q_G = 1$, there is no condition to check. Otherwise $d_G = D_0$. To show that suitable shifts exist, we choose them independently and uniformly at random. For each fixed nonzero b , the signs $(-1)^{b \cdot s_j}$ are independent and uniform in $\{-1, 1\}$. Hoeffding's inequality bounds the probability of violating (6.2) for this b by $2 \exp(-\varepsilon^2 d_G/2)$. A union bound over b gives total failure probability at most

$$2q_G \exp(-\varepsilon^2 d_G/2) = 2q_G \exp(-4\mu) < 1,$$

where $q_G < 2m < 2^{\mu+1}$. Exhaustively enumerating the shift tuples and checking (6.2) therefore finds a valid tuple in $\exp(O(d_G \log(m+2)))$ time.

We next derive the mixing estimate from (6.2). Let A_G be the $q_G \times q_G$ matrix with $A_G[u, v] = |\{j \in [d_G] : u \oplus s_j = v\}|$; its entries count edges from left vertices u to right vertices v . Both sides have the same index set $\{0, 1\}^{\ell_G}$, on which the functions $\chi_b(u) = (-1)^{b \cdot u}$ form an orthogonal basis. For every $b, u \in \{0, 1\}^{\ell_G}$,

$$(A_G \chi_b)(u) = \sum_{j=1}^{d_G} \chi_b(u \oplus s_j) = \chi_b(u) \sum_{j=1}^{d_G} (-1)^{b \cdot s_j}.$$

Thus each χ_b is an eigenvector: the constant function $\chi_{\vec{0}}$ has eigenvalue d_G , while (6.2) bounds all other eigenvalues in absolute value by εd_G . For sets U of left vertices and V of right vertices, let $e_G(U, V)$ count the edges between them, including multiplicity. Decomposing the indicator vectors of U, V into their constant and orthogonal parts and applying Cauchy–Schwarz gives

$$e_G(U, V) \leq \frac{d_G |U||V|}{q_G} + \varepsilon d_G \sqrt{|U||V|}. \quad (6.3)$$

To specify the local code, we put

$$c_G = \left\lceil \frac{d_G}{8B} \right\rceil, \quad \delta = \frac{1}{100B\mu}.$$

We need a binary matrix A_{loc} with c_G rows and d_G columns such that every nonzero word in its kernel modulo 2 has weight at least δd_G . To prove that such a matrix exists, we consider a matrix whose entries are independent uniform bits. For each fixed nonzero word, one of its nonzero coordinates makes the parity in each row uniform; independence of the rows therefore gives probability 2^{-c_G} that the word lies in the kernel. Set $h = \lceil \delta d_G \rceil - 1$. If $h = 0$, there are no nonzero words of weight less than δd_G . Otherwise their number is at most

$$\sum_{j=1}^h \binom{d_G}{j} \leq (h+1)(2d_G)^h \leq (2d_G)^{2h} < 2^{d_G/(25B)}.$$

For the last inequality, use $h < \delta d_G$, $d_G \leq m < 2^\mu$, and $\mu + 1 \leq 2\mu$. Since $c_G \geq d_G/(8B)$, a union bound shows that a suitable A_{loc} exists. Enumerating the $2^{c_G d_G}$ binary matrices and testing all words of weight less than δd_G finds one in $\exp(\text{poly}(B, \log(m+2)))$ bit operations.

At every vertex on both sides, impose the c_G rows of A_{loc} on its incident edge coordinates, ordered by their ports. These constraints define a Tanner code with local code $C_{\text{loc}} = \ker_{\mathbb{F}_2} A_{\text{loc}}$. Its incidence graph has one bit node for every edge of our graph and one local-code node for every vertex; each bit participates in the constraints at its two endpoints. Consider a nonzero edge word accepted by all these checks. Let U, V be the vertices touched by its nonzero edges on the two sides, and let h' be its weight. Each vertex in $U \cup V$ is incident to at least δd_G nonzero edges, so $h' \geq \delta d_G \max(|U|, |V|)$. Therefore

$$\delta d_G \sqrt{|U||V|} \leq h' \leq e_G(U, V) \leq \frac{d_G |U||V|}{q_G} + \varepsilon d_G \sqrt{|U||V|}.$$

Both U and V are nonempty. Dividing by $d_G \sqrt{|U||V|}$ yields $\sqrt{|U||V|} \geq (\delta - \varepsilon) q_G$, and hence

$$h' \geq \delta(\delta - \varepsilon) q_G d_G = \frac{q_G d_G}{20000 B^2 \mu^2} \geq \frac{m}{P}.$$

Shorten this Tanner code by keeping any m of the $q_G d_G$ edge coordinates, identifying them with $[m]$, and setting the remaining coordinates to zero. Restricting all the vertex checks to the retained coordinates gives $A^{(0)}$. Any word accepted by $A^{(0)}$ extends by zeros to an accepted edge word, so it has the required weight. Every row has at most $d_G \leq D_0$ nonzero entries. There are $2q_G c_G$ rows. If $q_G = 1$, then $d_G = m$ and

$$2c_G \leq \frac{m}{4B} + 2 \leq \frac{n}{4} + 2 \leq 3n.$$

If $q_G > 1$, then $d_G = D_0 \geq 8B$, and

$$2q_G c_G \leq \frac{q_G d_G}{4B} + 2q_G \leq \frac{q_G d_G}{2B} < \frac{m}{B} \leq n.$$

The searches over shifts, local matrices, and test words are sequential. Storing the current choices and the final rows uses $\text{poly}(n, m, B)$ bits of working space. $\square$

6.2. Choosing additional equations by counting common zeros. After the initial checks, every nonsatisfying assignment still passing violates at least m/P clauses. This lets a short random test catch each such assignment with constant probability. To make the choice deterministic we use the averaging principle underlying conditional expectations [Rag88].

Lemma 6.2. *With P, D_0 and $A^{(0)}$ from Lemma 6.1, a deterministic procedure appends $3n$ rows of list length P and obtains a matrix A with $N(A) = \#F$. It uses at most $3n(2m)^P$ common-zero counts. Every candidate system has at most $6n$ equations, each involving at most $2^k \max(D_0, P)$ monomials of degree at most k .*

Proof. At a stage with current matrix A , enumerate the ordered lists

$$\gamma = ((e_1, c_1), \dots, (e_P, c_P)) \in \mathcal{C} := ([m] \times \{0, 1\})^P.$$

The list γ represents the binary row h_γ with

$$(h_\gamma)_e = \left(\sum_{j: e_j=e} c_j \right) \bmod 2 \quad (e \in [m]).$$

There are $(2m)^P$ lists; distinct lists are still counted separately when they represent the same row. Write $A; h_\gamma$ for the matrix obtained by appending h_γ to A .

Fix a nonsatisfying assignment x accepted by A . The initial rows remain in A , so $|\text{supp } v(x)| \geq m/P$. For a uniform list γ , the probability that none of $e_1, \dots, e_P$ belongs to this support is at most

$$(1 - 1/P)^P \leq e^{-1} < 1/2.$$

Conditional on the indices hitting the support, choose one hit position j . After fixing every coefficient except c_j , the uniform bit c_j makes $h_\gamma \cdot v(x)$ uniform modulo 2. This remains true when some clause indices are repeated. Thus, for each such x , at least a quarter of the candidate lists give a row that rejects x .

Summing this bound over the nonsatisfying assignments still passing, we find

$$\mathbb{E}_\gamma D(A; h_\gamma) \leq \frac{3}{4} D(A).$$

For every candidate, compute $N(A; h_\gamma)$ by counting the common zeros of its polynomial system. Append a candidate attaining the minimum, breaking ties by enumeration order. Since $N(A; h_\gamma) = \#F + D(A; h_\gamma)$ for every γ , the chosen row leaves at most $\frac{3}{4} D(A)$ nonsatisfying assignments. After $3n$ rounds at most $(3/4)^{3n} 2^n = (27/32)^n < 1$ remain; their number is an integer, so it is zero. The procedure runs for a fixed number of rounds and never needs to know $D(A)$ or $\#F$.

The initial matrix has at most $3n$ rows, and every candidate is evaluated with at most $3n$ additional rows. A row listing s clauses expands, by (6.1), to at most $2^k s$ monomials on the original variables; duplicates cancel over $\mathbb{F}_2$. These observations give the claimed size and query bounds. $\square$

6.3. Counting one sparse formula. Suppose $n \geq 20$ and $1 \leq m \leq Bn$, where B is bounded by a fixed power of $\log n$ depending only on k . Construct the initial matrix by Lemma 6.1. This uses $\exp((\log n)^{O_k(1)})$ bit operations and polynomial space. Both P and D_0 are $(\log n)^{O_k(1)}$, so every candidate in Lemma 6.2 is an instance of Theorem 2.2 with a single fixed $c = c(k)$. This applies to discarded candidates as well as the final system: the counter requires only the number, degree and sparsity of the equations. Expanding each row and combining monomials costs polynomial time and space for fixed k .

At most $3n(2m)^P + 1 = \exp((\log n)^{O_k(1)})$ calls suffice, including one final count. Each call takes $2^{O_k(n(\log \log n)^2/\sqrt{\log n})}$ bit operations. Since any fixed power of $\log n$ is $o(n(\log \log n)^2/\sqrt{\log n})$, the total time, including construction and candidate comparisons, is

$$(n + m)^{O_k(1)} 2^{O_k(n(\log \log n)^2/\sqrt{\log n})}. \quad (6.4)$$

The final count is $\#F$ by the greedy lemma. A formula with no clauses or containing an empty clause is handled directly. Candidates are examined sequentially; only the current rows, a candidate, the best row and its count need be kept in addition to the counter's polynomial working space.

6.4. Returning to a general formula. We use disjoint sparsification to partition the satisfying assignments among sparse formulas on the same n declared variables. The preceding algorithm counts each of these formulas. The parameter σ controls a tradeoff: making it smaller reduces the number of formulas but increases their occurrence bound B . Our solver permits B to grow as a fixed power of $\log n$, so we can let σ tend to zero while keeping both the number of formulas and the cost per formula within the desired time bound.

Lemma 6.3. *Fix $k \geq 2$. Given a k -CNF on n declared variables with m clauses and a rational $0 < \sigma \leq 1$, a deterministic algorithm constructs a tree of k -CNFs on the same variables. Let L_σ be the total binary encoding length of the supplied numerator and denominator of σ . An assignment satisfies the input exactly when it satisfies one leaf formula, and it cannot satisfy two distinct leaves. The tree has*

$$\text{poly}_k(n, \sigma^{-1}) 2^{\sigma n}$$

nodes and depth $\text{poly}_k(n, \sigma^{-1})$. In every leaf, each variable occurs at most B times, where B is an explicitly computed integer satisfying

$$B = O_k\left(\left[\sigma^{-1} \log(2/\sigma)\right]^{k-1}\right).$$

Input normalization uses $\text{poly}_k(n, m)$ bit operations. Constructing the numerical parameters uses $\text{poly}_k(L_\sigma)$ bit operations. Thereafter, processing each node uses $\text{poly}_k(n, \log(2/\sigma))$ bit operations. The input count is the sum of the leaf counts.

Appendix A proves the lemma, including the disjointness that justifies adding the leaf counts.

Proposition 6.4. *For each fixed $k \geq 2$, the interpolation algorithm counts the satisfying assignments of a k -CNF on $n \geq 20$ declared variables with m clauses in $(n + m)^{O_k(1)} 2^{O_k(\rho(n))}$ bit operations and $(n + m)^{O_k(1)}$ bits of space.*

Proof. We first normalize the input in $(n + m)^{O_k(1)}$ time by deleting repeated literals, tautological clauses, duplicate clauses, and clauses containing another clause. We retain all n declared variables. An empty clause gives count zero, and a formula with no clauses gives count 2^n . We handle $n < 20$ by exhaustive search. For $n \geq 20$, we next select the sparsification parameter. Define

$$\lambda_n = \lceil \log_2 n \rceil, \quad \nu_n = \lceil \log_2 \lambda_n \rceil, \quad s = \left\lfloor \frac{\sqrt{\lambda_n}}{\nu_n^2} \right\rfloor.$$

We compute s by integer comparisons, taking the largest nonnegative integer satisfying $s^2 \nu_n^4 \leq \lambda_n$. If $s < 2$, we use exhaustive search; $\sqrt{\lambda_n}/\nu_n^2$ tends to infinity, so there are only finitely many such

sizes. Otherwise we apply Lemma 6.3 with $\sigma = 1/s$. The tree has at most $2^{n/s} \text{poly}_k(n, s)$ nodes, and its leaves have occurrence bound

$$B = O_k((s \log s)^{k-1}) = (\log n)^{O_k(1)}.$$

Each leaf uses the original n variables and has at most Bn nonempty clauses. We count each leaf by the sparse-formula algorithm of Section 6.3, whose running time is given by (6.4). Leaves with no clauses or containing an empty clause are handled directly. The clause count in that application is the number of clauses in the leaf; m in the final bound remains the input clause count.

For $s \geq 2$, the floor in its definition gives

$$\frac{n}{s} = O\left(\frac{n\nu_n^2}{\sqrt{\lambda_n}}\right) = O(\rho(n)).$$

Thus both the number of leaves and the time to count each leaf have exponent $O_k(\rho(n))$. Their polynomial factors, together with normalization and the work at the tree nodes, are bounded by $(n + m)^{O_k(1)}$. The total number of bit operations is therefore

$$(n + m)^{O_k(1)} 2^{O_k(\rho(n))}.$$

The finite exhaustive-search cases are absorbed in the constants. All parameters, initial checks, and additional rows are computed by the deterministic procedures above; every counting query uses the common-zero algorithm of Theorem 2.2.

To bound the space used by the full algorithm, we traverse the sparsification tree depth first and count one leaf at a time. The current path has $\text{poly}_k(n, s)$ nodes, and each normalized formula has a $\text{poly}_k(n)$ -bit encoding. Since $s \leq \sqrt{\lceil \log_2 n \rceil}$, storing this path and counting one leaf use $\text{poly}_k(n, m)$ bits. Disjointness keeps the sum of the processed leaf counts between zero and 2^n , so its accumulator uses at most $n + 1$ bits. Thus the total space is $\text{poly}_k(n, m)$ bits. $\square$

APPENDIX A. PARTITIONING THE SATISFYING ASSIGNMENTS

Fix $k \geq 2$ and a rational $0 < \sigma \leq 1$. We prove Lemma 6.3 by repeatedly splitting a formula into two cases. Both children use the same n variables, and every satisfying assignment of the parent satisfies exactly one child. We follow the rule for choosing a shared set of literals and the insertion accounting of [CIP06, Section 6]. The refinement that makes the two cases disjoint is standard in exact counting [DHM⁺14, Appendix A].

Throughout this appendix, a clause is a set of literals and denotes their disjunction. Its *width* is the number of literals it contains. We delete tautological clauses, repeated clauses, and clauses containing another clause. Thus the remaining clauses form an antichain under inclusion. We keep all n declared variables and do not propagate unit clauses. An empty clause gives count zero, and the empty formula gives count 2^n .

A.1. The branching rule. Fix an integer $M \geq 2$. We will obtain an occurrence bound $B = O_k(M^{k-1})$ at every leaf, then choose M so that the tree has at most $\text{poly}_k(n, \sigma^{-1}) 2^{\sigma n}$ nodes. Put

$$\theta_j = M^j \quad (0 \leq j < k).$$

Writing F for the current family of clauses, define

$$\deg_c(H) = \#\{C \in F : |C| = c, H \subseteq C\}$$

for each literal set H . We branch whenever a nonempty proper subset H of a width- c clause satisfies

$$\deg_c(H) \geq \theta_{c-|H|}. \tag{A.1}$$

The algorithm chooses the smallest possible c , and then the largest possible $|H|$; break any remaining ties deterministically. Thus a nonempty shared literal set H leaving $j = c - |H|$ other literals must occur in at least M^j clauses before we branch on it.

Let $H \cup P_1, \dots, H \cup P_z$ be all the width- c clauses containing the chosen H , and let G consist of the other clauses. Put $j = c - |H|$. Here P_a is the set of literals left after removing H from the a th selected clause, so each P_a has width j and $z \geq \theta_j$. The two children are

$$F_A = G \wedge H, \quad F_B = G \wedge \bigwedge_{a=1}^z P_a \wedge \bigwedge_{\ell \in H} \{\neg \ell\}. \quad (\text{A.2})$$

We restore the antichain condition in each child by removing clauses made redundant by its new clauses. The sets P_a may overlap. The satisfying assignments of the children are disjoint: F_A requires some literal of H to be true, whereas F_B requires every literal of H to be false.

Lemma A.1. *The children in (A.2) partition the satisfying assignments of their parent. Every new clause has width less than c . The A-child inserts one new clause, and the B-child inserts at least θ_j new clauses of width j . All newly inserted clauses survive the immediate normalization. Every old clause removed in either child contains a newly inserted strict subset.*

Proof. Every assignment satisfies exactly one of the conditions *some literal of H is true* and *every literal of H is false*. In the first case, all selected clauses are satisfied, leaving $G \wedge H$. In the second, each selected clause reduces to P_a , and the unit clauses $\{\neg \ell\}$ enforce this case. Thus the two children partition the parent's satisfying assignments.

The old antichain contains no clause included in H or in any P_a : such a clause would be a strict subset of $H \cup P_a$. Thus H and the distinct equal-width clauses P_a are new. No P_a contains a variable of H , since its parent is nontautological. Consequently, none of the unit clauses $\{\neg \ell\}$, $\ell \in H$, is a subset of any P_a . Any unit clause $\{\neg \ell\}$ already present is retained and is not counted as a new insertion. The remaining new unit clauses and the clauses P_a all survive. Each selected old clause contains the newly inserted clause H for child A , or its newly inserted subset P_a for child B . Every other removed clause contains a newly inserted strict subset by the normalization rule. $\square$

A.2. Counting insertions. To bound the tree, we count every clause introduced along one path, including clauses later deleted. Clauses present in the input are excluded from this count. By Lemma A.1, each step inserts a clause, so bounding the total number of insertions also bounds the path length.

For this accounting, a clause is marked when a branch introduces it. First we bound how many marked clauses can still be present and contain a given literal set. The smallest-width priority controls those present before a branch; the largest- $|H|$ priority controls those introduced by that branch.

Lemma A.2. *At every node, for $1 \leq i < j < k$ and every literal set S of size i , fewer than $2\theta_{j-i}$ marked clauses of width j that are still present contain S .*

Proof. Initially there are no marked clauses. It suffices to consider a branch that inserts clauses of width j . Its selected width c is larger than j . Before insertion, fewer than θ_{j-i} current width- j clauses contain S : otherwise the smallest-width priority would select a width at most j .

Choosing child A inserts only one clause. Suppose child B inserts clauses P_a of width j . If none of the P_a contains S , the number of new width- j clauses containing S is zero. Otherwise S lies in a petal and is therefore disjoint from H . In this case P_a contains S exactly when the selected clause $H \cup P_a$ contains $H \cup S$. Since $0 < i < j$, $H \cup S$ is larger than H and is a proper subset of a selected width- c clause. The priority for the largest shared set therefore gives

$$\#\{a : S \subseteq P_a\} = \deg_c(H \cup S) < \theta_{c-|H \cup S|} = \theta_{j-i}.$$

The other new clauses $\{\neg \ell\}$ have width one. Thus the old and new clauses of width j together satisfy the asserted bound. Deletion cannot increase it. $\square$

Lemma A.3. *Along any path or path prefix, the total number T_j of width- j insertions satisfies $T_j \leq 4(3M)^{j-1}n$ for $1 \leq j < k$.*

Proof. The case $n = 0$ is immediate, so assume $n \geq 1$. For the current path prefix, let L_j be the number of marked width- j clauses still present, and let D_j be the number of width- j insertions whose clauses have since been deleted. Then $T_j = L_j + D_j$. Units are never deleted, so $T_1 \leq 2n \leq 4n$. For $j \geq 2$, sum Lemma A.2 over the $2n$ single literals. This gives $L_j \leq 4n\theta_{j-1}$.

Whenever a marked clause of width j is deleted, assign that deletion to one newly introduced strict subset contained in it; Lemma A.1 guarantees such a subset. If the new clause has width $i < j$, Lemma A.2 shows that it was contained in fewer than $2\theta_{j-i}$ marked clauses of width j immediately before the branch. Thus one insertion of a width- i clause receives at most that many deletions. Summing over the widths i of the new clauses gives $D_j \leq 2 \sum_{i=1}^{j-1} \theta_{j-i} T_i$. Therefore

$$T_j = L_j + D_j \leq 4n\theta_{j-1} + 2 \sum_{i=1}^{j-1} \theta_{j-i} T_i. \quad (\text{A.3})$$

The first term counts clauses still present; the sum counts those that were deleted. Dividing by nM^{j-1} and applying induction gives

$$\frac{T_j}{nM^{j-1}} \leq 4 + 2 \sum_{i=1}^{j-1} 4 \cdot 3^{i-1} = 4 \cdot 3^{j-1},$$

as required. $\square$

A.3. The size of the tree. We now convert Lemma A.3 into a bound on the number of tree nodes. It controls both the length of a path and the number of times that path chooses the B -child in (A.2). Write these bounds as Cn and $\Gamma n/M$, respectively, where

$$C = 4 \sum_{j=0}^{k-2} (3M)^j, \quad \Gamma = 4 \sum_{j=0}^{k-2} 3^j = 2(3^{k-1} - 1).$$

Lemma A.3 gives depth at most Cn . A choice of child B with $|P_a| = j$ inserts at least M^j clauses of width j , so a path contains at most $4 \cdot 3^{j-1}n/M$ such choices. In total, a path chooses B at most $\Gamma n/M$ times.

A path with few choices of B has a short description. Because the branching rule is deterministic, a node is determined by its depth and the positions at which its path chooses B ; every other position chooses A . For $M \geq \Gamma$, this bounds the number of nodes by

$$(Cn + 1) \sum_{a \leq \lceil \Gamma n/M \rceil} \binom{Cn}{a} \leq (Cn + 1)^2 2^{(\Gamma n/M) \log_2(4CM)}.$$

Here $\Gamma n/M \leq n \leq Cn/4$. For a positive integer a , use $\binom{Cn}{a} \leq (4Cn/a)^a$; this upper bound increases for $a \leq Cn/4$. There are at most $Cn + 1$ summands. The case $\Gamma n/M < 1$ is immediate.

We choose M to make the exponent in this bound at most σn . Test powers of two $M \geq \Gamma$ in increasing order. For each choice, compute C and $h = \lceil \log_2(4CM) \rceil$, and stop when

$$\Gamma h \leq \sigma M.$$

All comparisons are exact. Since $C = O_k(M^{k-2})$ and Γ depends only on k , the first successful choice satisfies

$$M = O_k(\sigma^{-1} \log(2/\sigma)).$$

The tree therefore has at most $(Cn + 1)^2 2^{\sigma n}$ nodes.

At a leaf, (A.1) fails for every nonempty proper subset of a clause. A fixed literal belongs to fewer than θ_{c-1} width- c clauses when $c \geq 2$, and to at most one unit clause. Hence each variable occurs at most

$$B = 2 \sum_{j=0}^{k-1} M^j \leq 4M^{k-1} = O_k\left([\sigma^{-1} \log(2/\sigma)]^{k-1}\right).$$

Thus every leaf has the required bound on occurrences per variable. The repeated two-case partition also shows that each original satisfying assignment occurs at exactly one leaf, so summing the leaf counts gives the original count.

To implement the branching rule at a node, we compute $\deg_c(H)$ for its current formula. For each width- c clause, we enumerate its nonempty proper subsets H and increment the counter indexed by (c, H) . There are $O_k(n^k)$ possible nonempty normalized clauses, each with at most 2^k subsets. The clause counters use $O_k(\log(n+2))$ bits, and the thresholds use $O_k(\log M)$ bits. Computing the counts, choosing a branch, and normalizing its children therefore use $\text{poly}_k(n, \log M)$ bit operations and bits of working space per node. The bound on M gives $\log M = O_k(\log(2/\sigma))$.

With L_σ as in Lemma 6.3, the parameter search examines $O_k(L_\sigma)$ powers of two. Its integer operands have $O_k(L_\sigma)$ bits, so constructing the numerical parameters uses $\text{poly}_k(L_\sigma)$ bit operations and bits of working space.

APPENDIX B. EXACT ARITHMETIC AND SPACE

Section 4 bounds the rational coefficients of the substitution rows $\vec{P}_{f,v}(T)$. We complete the bit-cost and space estimates in Proposition 5.3 by bounding the lifted factors, the sums used to compute the Taylor polynomials of H , and the final interpolation. The essential point is that each sum uses a fixed denominator. Its numerator grows with the number of summands, so its bit length grows only with the logarithm of that number. For any polynomial $G = \sum_\alpha c_\alpha \vec{X}^\alpha$ in finitely many indeterminates, write $\|G\|_1 = \sum_\alpha |c_\alpha|$ for its coefficient norm.

B.1. Integer coefficients of the lifted indicator. We first bound the coefficients of the parity polynomials supplied by Lemma 5.1. They occur in every equation factor $\mathcal{B}_i$ in (5.4), so their sizes enter the bit-cost bounds for evaluating those factors. For an integer $\ell \geq 1$, recall $h(Y) = Y(Y-1)$ and the polynomials from Lemma 5.1:

$$p_\ell(Y) = (1-Y)^\ell \sum_{j=0}^{\ell-1} \binom{\ell+j-1}{j} Y^j = \sum_{j=0}^{\ell-1} e_j(Y) h(Y)^j, \quad \deg e_j \leq 1.$$

That lemma proves the parity congruence and constructs the e_j by successive monic division by h . Here we bound the integers occurring in this construction.

Lemma B.1. *Write $e_j(Y) = a_j + b_j Y$ for $0 \leq j < \ell$. Then $a_j, b_j \in \mathbb{Z}$ and*

$$|a_j|, |b_j| \leq 2^{C\ell^2} \quad (0 \leq j < \ell)$$

for an absolute constant C . Successive division of p_ℓ by h constructs these polynomials using $\text{poly}(\ell)$ bit operations and bits of working space.

Proof. The coefficient-norm inequality $\|fg\|_1 \leq \|f\|_1 \|g\|_1$, together with the binomial identity

$$\sum_{j=0}^{\ell-1} \binom{\ell+j-1}{j} = \binom{2\ell-1}{\ell-1}$$

gives

$$\|p_\ell\|_1 \leq 2^\ell \binom{2\ell-1}{\ell-1} \leq 2^{3\ell}.$$

Monic division by $h = Y^2 - Y$ preserves integer coefficients. Each leading-term elimination increases the sum of the coefficient norms of the current quotient and remainder by at most a factor of three. There are $O(\ell^2)$ eliminations in all the successive divisions. Thus every intermediate coefficient has absolute value at most $2^{O(\ell^2)}$, which proves the stated bound for a_j, b_j . Its signed binary encoding consequently uses $O(\ell^2)$ bits. Each coefficient vector has $O(\ell)$ entries. The successive divisions use $O(\ell^2)$ integer arithmetic operations on integers of $O(\ell^2)$ bits, giving the claimed $\text{poly}(\ell)$ bit-operation and space bounds. $\square$

B.2. Bit complexity of the sums over Boolean assignments. We now bound every intermediate value in the computation of $\mathcal{H}_t(z) = H(t + z) \bmod z^J$ from (3.13). Starting with the substitution rows $\vec{P}_{f,v}(T)$, we bound the substituted factors, their sums over each group of Boolean variables, and the accumulator T_t for the Taylor data in Proposition 5.3. Fix a polynomial system $\mathcal{P}$ in n Boolean variables, with m_{test} equations of degree at most k , precision ℓ such that $2^\ell > 2^n$, and a replacement design for its equation factors. Suppose the rows $\vec{P}_{f,v}$, partitions $S_{t,g}$, and matrices $C_{t,g}$ satisfy Definition 3.1, with parameters Q, R, r, D, τ . Recall the degree bound for H and the number of Taylor coefficients retained at each node:

$$d_H = kR(m_{\text{test}} + 2(\ell - 1)), \quad J = \left\lfloor \frac{d_H}{Q} \right\rfloor + 1.$$

All polynomials in the Taylor variable z and the shared truncation marker V are stored in their unique representative modulo (z^J, V^ℓ) . Removing terms outside these degree ranges cannot increase the coefficient norm.

The substitution rows have common denominator

$$d_{\text{den}} = (R - 1)!Q!.$$

We use an integer $B_P \geq 1$ such that every integer coefficient of every row $d_{\text{den}}\vec{P}_{f,v}(T)$ has absolute value at most 2^{B_P} . Definition 3.1 gives $B_P = O(Q \log Q + \log D)$. At a fixed node $t \in [Q]$ and for a fixed correction set $U \subseteq [n]$, the input to factor f at variable v is

$$\theta_{f,v}(z; x) = \langle \vec{P}_{f,v}(t), x \rangle + \langle \vec{P}_{f,v}(t + z) - \vec{P}_{f,v}(t), \mathbf{1}_U \rangle, \quad x \in \{0, 1\}^n.$$

Let $L_i(z; x)$ be the sum of the listed monomials of equation i after these substitutions. Each $d_{\text{den}}\theta_{i,v}(z; x)$ is an integer polynomial of coefficient norm at most

$$B_0 = d_{\text{den}} + 3n(R + 1)2^{B_P}(Q + 1)^R.$$

Indeed, each of the three row evaluations in $\theta_{i,v}$ contributes at most $n(R + 1)2^{B_P}(Q + 1)^R$ after denominators are cleared. The bound holds also for the representatives in the group sums.

Put $\Delta = d_{\text{den}}^k$. For a monomial of degree $u \leq k$, multiplying its substituted value by Δ gives an integer polynomial of norm at most $d_{\text{den}}^{k-u}B_0^u \leq B_0^k$, including the constant monomial. As S bounds the total list length,

$$B = \max\{\Delta, SB_0^k\} \quad \text{satisfies} \quad \|\Delta L_i(z; x)\|_1 \leq B.$$

Consequently $\log d_{\text{den}} = O(Q \log(Q + 2))$, $\log B_0 = O(Q \log(Q + 2) + \log(nD + 2))$, and $\log B = O(k \log B_0 + \log(S + 2))$.

The substituted factor of equation i is

$$\mathcal{B}_i(z, V; x) = \sum_{j=0}^{\ell-1} e_j(L_i(z; x)) h(L_i(z; x))^j V^j \pmod{(z^J, V^\ell)}.$$

Since e_j is integral and has degree at most one, its substitution has denominator dividing Δ ; the denominator of $h(L_i)^j$ divides Δ^{2j} . Thus, with

$$\Gamma = \Delta^{2\ell-1},$$

every coefficient of $\Gamma\mathcal{B}_i(z, V; x)$ is an integer. To bound the resulting integer numerators, put

$$E_\ell = \max\{1, \|e_0\|_1, \dots, \|e_{\ell-1}\|_1\}, \quad B_1 = \ell E_\ell (B + \Delta + 2)^{2\ell-1}.$$

For $A_i = \Delta L_i$, the two substituted factors have integer numerators $\Delta e_j(L_i)$ and $A_i(A_i - \Delta)$, with norms at most $E_\ell(B + \Delta)$ and $B(B + \Delta)$, respectively. Padding each term to denominator Γ therefore gives

$$\|\Gamma\mathcal{B}_i\|_1 \leq B_1, \quad B_1 \geq \Gamma, \quad \log B_1 = O(\ell^2 + \ell \log(B + \Delta + 2) + \log(\ell + 1)).$$

We next apply these estimates to the actual group sums. Write $S_g = S_{t,g}$ and $C_g = C_{t,g}$, and let c_g be the number of rows of C_g . Thus $c = \sum_g c_g \leq 2\tau n$, and every coordinate of a Boolean measurement $C_g x_g$ lies in $[-D, D] \cap \mathbb{Z}$. Let $m_{\text{test},g}$ be the number of equation factors assigned to group g at t , so $\sum_g m_{\text{test},g} = m_{\text{test}}$. Their product $A_{g,U}(x; z)$ from (3.11) is represented in $\mathbb{Q}[z, V]/(z^J, V^\ell)$. It has denominator dividing $\Gamma^{m_{\text{test},g}}$ and numerator norm, over that denominator, at most $B_1^{m_{\text{test},g}}$. The neutral constant-one factors do not affect this bound.

For a pair $U \subseteq E \subseteq [n]$ with $|E| < J$ and an attainable measurement tuple $\vec{y} = (y_1, \dots, y_r)$, choose Boolean representatives $\bar{x}_h$ with $C_h \bar{x}_h = y_h$. The group sum in (3.4) is

$$\Sigma_g(\vec{y}) = \sum_{\substack{x_g \in \{0,1\}^{S_g} \\ C_g x_g = y_g \\ (x_g)_v = 1 \ (v \in E \cap S_g)}} A_{g,U}\left(x_g + \sum_{h \neq g} \bar{x}_h; z\right).$$

Here vectors on individual groups are extended by zero. The sum has at most $2^{|S_g|}$ terms, all over the same denominator, so

$$\|\Gamma^{m_{\text{test},g}} \Sigma_g(\vec{y})\|_1 \leq 2^{|S_g|} B_1^{m_{\text{test},g}}.$$

For $1 \leq g \leq r$, define the partial product $\Pi_g(\vec{y}) = \prod_{h=1}^g \Sigma_h(\vec{y})$, with truncation after each multiplication. Its denominator divides

$$\Gamma^{\sum_{h \leq g} m_{\text{test},h}}.$$

Consequently every full contribution $\Pi_r(\vec{y})$ can be placed over the fixed denominator

$$\Omega = \Gamma^{m_{\text{test}}},$$

and it satisfies

$$\|\Omega \Pi_r(\vec{y})\|_1 \leq 2^n B_1^{m_{\text{test}}}.$$

Let $T_t(z, V)$ be the accumulator that adds these full contributions with sign $(-1)^{|E| - |U|}$ over all pairs and measurement tuples. By (3.13), its final value is the $\mathcal{A}_\ell$ -valued Taylor polynomial $\mathcal{H}_t(z)$. The number M_* of signed updates at one node is bounded by

$$M_* \leq \left(\sum_{s=0}^{\min(J-1, n)} 2^s \binom{n}{s} \right) (2D+1)^c \leq 3^n (2D+1)^c.$$

Every intermediate value of T_t , not just its final value, therefore satisfies

$$\|\Omega T_t\|_1 \leq M_* 2^n B_1^{m_{\text{test}}}.$$

Each coefficient of the integer polynomial ΩT_t has

$$O(\log(M_* + 1) + n + m_{\text{test}} \log B_1)$$

bits. This estimate also bounds the integer numerators of all group sums and partial products with the fixed denominators given above. The denominator satisfies

$$\log \Omega = k m_{\text{test}} (2\ell - 1) \log d_{\text{den}} = O(k(m_{\text{test}} \ell + 1) Q \log(Q + 2)).$$

Each contribution multiplies at most m_{test} lifted equation factors. Processing further pairs or measurement tuples adds new contributions; it does not multiply them into the previous ones.

This is why exponentially many iterations contribute only $\log(M_* + 1)$ to the bit estimate. Each stored polynomial has at most $J\ell \leq (d_H + 1)\ell$ coefficients. The enumeration retains its current indices and Boolean representatives, one group sum, one partial product, and the accumulator T_t , without storing a table of all assignments or all measurement tuples. These estimates give (5.6) and polynomial bit cost per rational operation. The $QJ\ell$ Taylor coefficients and the working state fit in $\text{poly}(n, m_{\text{test}}, \ell, D, Q, R, J, r, k, S)$ bits.

B.3. Interpolation and the integer answer. For $0 \leq s < \ell$, let $H_s(T) = [V^s]H(T) \in \mathbb{Q}[T]$. Section B.2 bounds the computed data $[z^j V^s]\mathcal{H}_t(z) = [z^j]H_s(t + z)$ for $t \in [Q]$, $0 \leq j < J$. Fix s . We interpolate this coordinate, using the same rational Hermite matrix for all ℓ coordinates. Put $M_H = QJ > d_H$. In the monomial basis $1, T, \dots, T^{M_H-1}$, the Hermite interpolation matrix is the integer matrix

$$V_{(t,j),u} = \begin{cases} \binom{u}{j} t^{u-j}, & u \geq j, \\ 0, & u < j, \end{cases} \quad t \in [Q], \quad 0 \leq j < J, \quad 0 \leq u < M_H.$$

It is nonsingular by the Hermite interpolation statement in Section 3. Its entries have $O(M_H \log(Q + 2))$ bits. Let $\mathbf{y}$ be the vector of computed coefficients $[z^j V^s]\mathcal{H}_t(z)$, ordered by (t, j) , and let $\mathbf{h}$ be the monomial coefficient vector of H_s , padded with zeros to length M_H . Thus $V\mathbf{h} = \mathbf{y}$. The vector $\mathbf{b} = \Omega\mathbf{y}$ is integral; write B_{data} for a bound on its entries' bit lengths. The bounds in Section B.2 give

$$B_{\text{data}} \leq \text{poly}(n, m_{\text{test}}, \ell, k, Q, \log(D + 2), \log(S + 2)).$$

Only the constant coefficient $h_0 = H_s(0)$ is needed. Let $V^{(0)}$ be obtained from V by replacing its column indexed by $u = 0$ with $\mathbf{b}$, and put

$$N_H = \det V^{(0)}, \quad D_H = \det V \neq 0.$$

Cramer's rule for $V(\Omega\mathbf{h}) = \mathbf{b}$ gives

$$H_s(0) = h_0 = \frac{N_H}{\Omega D_H}. \quad (\text{B.1})$$

We compute these two determinants by fraction-free Gaussian elimination [Bar68, Sections I–II and IV–V]. For an $M_H \times M_H$ integer matrix, this algorithm uses $O(M_H^3)$ integer arithmetic operations; pivoting allows it to handle singular matrices as well. The retained elimination entries are signed minors of the input matrix. Each update forms products and a difference of such entries, followed by an exact integer division. Consequently, bounds on the minors of V and $V^{(0)}$ also bound the intermediate integers used to compute D_H and N_H . Every minor of $(V \mid \mathbf{b})$ has absolute value at most

$$2^{O(M_H(B_{\text{data}} + M_H \log(Q+2) + \log(M_H+1)))}.$$

Indeed, its determinant is a sum of at most $M_H!$ products of at most M_H entries. The intermediate integers in fraction-free elimination therefore have

$$O(M_H(B_{\text{data}} + M_H \log(Q + 2) + \log(M_H + 1)))$$

bits in signed binary notation. Computing N_H, D_H thus uses $\text{poly}(M_H, B_{\text{data}})$ bit operations and bits of working space.

At $T = 0$ every $H_s(0)$ is an integer by construction, so the division in (B.1) is exact for each s . After computing these ℓ integers, put $u = \sum_{s=0}^{\ell-1} H_s(0)$. By (5.5), the least nonnegative residue

$$u - 2^\ell \left\lfloor \frac{u}{2^\ell} \right\rfloor = N(\mathcal{P})$$

is the exact count. Reusing the same workspace for successive coordinates, all ℓ interpolations, the sum and the remainder use $\text{poly}(M_H, B_{\text{data}}, \log(\Omega + 1), \ell)$ bit operations and polynomially many bits of working space.

B.4. Arithmetic for the technique overview. We also justify the bit-complexity and space bounds for the bounded-occurrence algorithm in Section 2. Lemma 4.6 bounds the bit lengths of the replacement coefficients by a polynomial in n . Solving the polynomial-size rational linear systems in Section 2.4 gives measurement coefficients $\vec{\lambda}_{e,v,h}$ of polynomial bit length as well. Let δ be the product of the denominators of all these coefficients and all replacement coefficients, over all evaluation points. There are only polynomially many, so δ also has polynomial bit length. Every clause product uses at most $M \leq 3n$ linear inputs; consequently, all coefficients in the restricted sums can be accumulated with common denominator δ^{3n} . Products of polynomially many input coefficients of polynomial bit length, and sums of exponentially many such terms over this common denominator, still have polynomial bit length. The exact Hermite interpolation above therefore has polynomial bit cost for these Taylor data as well. Finally, the enumerations can be performed sequentially, retaining only the construction, current indices, coefficient arrays, running sums, and the polynomially many Taylor coefficients needed for interpolation. This gives polynomial space and only polynomial bit-operation overhead.

APPENDIX C. ADDITIONAL CONSEQUENCES OF REFUTING ETH

Theorem 1.1 gives a circuit lower bound and subexponential-time algorithms for inverting Boolean maps with constant output locality. The fixed-density case gives the quantitative circuit lower bound in Corollary 1.4. Both circuit arguments use only the decision version of our 3-SAT algorithm. Exact preimage counting uses its full counting guarantee.

C.1. Circuit lower bounds for E^{NP} . For the qualitative bound below, write $SIZE(O(n))$ for the languages having nonuniform circuits with $O(n)$ bounded-fan-in AND, OR, and NOT gates.

Corollary C.1. *We have*

$$E^{NP} \not\subseteq SIZE(O(n)).$$

Proof. Williams proved that if ETH is false, then E^{NP} does not have linear-size circuits [Wil13, Theorem 6.1]. Apply this implication to the deterministic subexponential-time 3-SAT algorithm supplied by Theorem 1.1. $\square$

The connection is an instance of deriving circuit lower bounds from faster satisfiability algorithms. To encode a Boolean circuit with s gates and n inputs, introduce a variable for each gate, enforce its value from its inputs, and add a unit clause requiring the output bit to equal 1. The resulting 3-CNF has $O(n + s)$ variables and clauses and is satisfiable exactly when the circuit is. Our theorem therefore decides satisfiability of linear-size circuits in $2^{o(n)}$ time. Williams’s argument combines such a speedup with hypothetical small circuits for witnesses to contradict a nondeterministic time hierarchy. The corollary gives a language in E^{NP} without a linear-size circuit family; it does not assert a superpolynomial circuit lower bound for NP.

Proof of Corollary 1.4. We use Williams’s witness-circuit argument [Wil13, Theorem 6.1] with the local reduction of Jahanjou, Miles, and Viola [JMV13, Theorem 1 and Section 6].

By Žák’s nondeterministic time hierarchy [Žák83], in the form stated in [Wil13, Theorem 2.1 in the full version], fix a unary language $H \in \text{NTIME}(2^n) \setminus \text{NTIME}(2^{n/2})$ on multitape Turing machines. For all sufficiently large n apply the local reduction to its fixed machine on 1^n with locality parameter $r = 2$. A time allowance $T = 2^n n^{O(1)}$ includes the polylogarithmic overhead of simulating the machine in the reduction and gives $q = n + O(\log n)$ clause-index bits. In polynomial time the reduction constructs a decoder giving three variable labels and their signs on each index. Let φ_n consist of the clauses decoded on all 2^q indices. It is satisfiable exactly when $1^n \in H$.

Each of the decoder’s $O(n)$ output bits has a constant-depth decision tree, so it depends on at most a fixed number w of index bits. Evaluating the decoder on all strings of Hamming weight at most w recovers each support. Indeed, two strings witnessing dependence on one coordinate can be

set to zero off the support while retaining their differing outputs. The same evaluations give the truth table on each support. This constructs a decoder of $O(n)$ fan-in-two gates in polynomial time.

Pad each variable label to dn bits for a fixed positive integer d and declare all 2^{dn} variables. If $|i| = dn$ for such an n and φ_n is satisfiable, let $W(i)$ be bit i of its lexicographically first satisfying assignment, ordering variables by their labels. Set $W(i) = 0$ in all other cases. The length of i determines n . To decide W , write out the 2^q clauses, ask an NP oracle whether the formula is satisfiable, and, if so, recover the first assignment by asking whether its current prefix followed by zero has a satisfying extension. At most 2^{dn} prefix queries of length $2^{O(n)}$ suffice, so $W \in \mathbf{E}^{\text{NP}}$.

Put $a(t) = \sqrt{\log_2 t}$. Testing whether the count in the constant-density part of Theorem 1.1 is positive decides 3-SAT with V variables and at most $5V$ clauses in time $V^b 2^{AV/a(V)}$ for fixed $A, b > 0$. We absorb the change of logarithm base in A . The solver uses polynomial space, so a multitape simulation can scan its workspace for each bit access at a cost $V^{O(1)}$, absorbed in b .

Fix a positive rational $\eta < 1/(12A)$ and put $s_n = \lfloor \eta n \lfloor a(n) \rfloor \rfloor$ for $n \geq 2$. The threshold is computable using the binary length of n and the integer square root of $\lfloor \log_2 n \rfloor$. Suppose that $C_W(dn) \leq s_n$ for all sufficiently large n . To decide H , reject nonunary inputs. On 1^n , guess a circuit on dn index bits with at most s_n gates. Feed the decoder's three variable labels to three copies of this circuit and test whether the indexed clause is violated. Accept if the test circuit is unsatisfiable. When $1^n \in H$, a circuit computing W at length dn gives an accepting guess. When $1^n \notin H$, every guess violates a clause.

The test circuit has q inputs and at most $3s_n + O(n)$ gates. The encoding above uses one variable per input and gate, at most four clauses of width at most three per gate to enforce its truth table, and one unit clause to pin the output. Padding with unused variables gives $U_n = 3s_n + Bn$ variables and at most $5U_n$ clauses for a fixed integer B . Since $U_n = (3\eta + o(1))na(n)$ and $a(U_n) \sim a(n)$, the test takes $2^{(3A\eta + o(1))n}$ time. Guessing and building the circuits take polynomial time, so H is decidable nondeterministically in time $2^{n/2}$ for all sufficiently large n . The finitely many exceptional lengths can be hardwired, contradicting the choice of H . Thus $C_W(dn) > s_n$ for infinitely many n . Since $s_n \sim \eta n \sqrt{\log_2 n}$ and d is fixed, taking $L = W$ and any $0 < c < \eta/(d\sqrt{\log 2})$ proves (1.2). $\square$

C.2. Inverting Boolean maps with constant output locality. Fix a constant d . Consider a Boolean map $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$ whose i th output bit has the form

$$G(\vec{x})_i = g_i(\vec{x}|_{S_i}), \quad S_i \subseteq [n], \quad |S_i| \leq d.$$

The input description of G lists the sets S_i and the truth tables of the Boolean functions g_i . Thus each output inspects only constantly many input bits, although the sets S_i may overlap arbitrarily. Given a target $\vec{b} \in \{0, 1\}^m$, inversion asks for any $\vec{x}$ with $G(\vec{x}) = \vec{b}$.

Corollary C.2. *For every fixed d and $n \geq 20$, the number $|G^{-1}(\vec{b})|$ can be computed exactly, and a preimage can be found whenever one exists, in deterministic time*

$$(n + m)^{O_d(1)} 2^{O_d(\rho(n))}$$

and space $(n + m)^{O_d(1)}$.

Proof. For each i and each local assignment $\vec{\alpha} \in \{0, 1\}^{S_i}$ with $g_i(\vec{\alpha}) \neq b_i$, include the clause

$$\bigvee_{j \in S_i} \ell_{j, \alpha_j}, \quad \ell_{j, 0} = X_j, \quad \ell_{j, 1} = \neg X_j.$$

This clause forbids exactly the local assignment $\vec{\alpha}$. If S_i is empty, its constraint is constant and can be checked directly. The conjunction of the remaining clauses is a CNF on the original n declared variables, with width at most d and at most $2^d m$ clauses. Its satisfying assignments are exactly $G^{-1}(\vec{b})$; no auxiliary variables or multiplicity corrections are needed. Apply Theorem 1.1 with $k = \max\{2, d\}$. The usual variable-by-variable self-reduction finds a preimage using at most n

further counting queries, which only changes the polynomial factor. Each query may be discarded before the next, preserving polynomial space. $\square$

Conditional counts also allow an exactly uniform preimage to be sampled, when the preimage set is nonempty, with the same bound in expected time up to polynomial factors [JVV86]. In particular, the corollary applies to constant-locality versions of Goldreich’s candidate one-way functions, which evaluate local predicates on overlapping subsets of input bits [Gol00]. For polynomially many output bits, it gives a $2^{o(n)}$ -time inversion algorithm for every output in the image, ruling out $2^{\Omega(n)}$ inversion hardness for these families. This does not rule out one-wayness against polynomial-time algorithms.

APPENDIX D. A SHARPER BOUND FOR EXACT COUNTING

We prove Theorem 1.1 by contracting Boolean tensor networks. The two running times reflect the number of wires in the networks. A formula with $m \leq Dn$ clauses gives a network with $O_{k,D}(n)$ wires. For arbitrary formulas, we use the parity tests of Section 6. The degree bound of Lemma 5.2 lets us index a monomial retained in the product of their polynomial lifts with $O_k(n \log \log n)$ wires. We give both encodings before proving the contraction theorem they use.

D.1. Boolean tensor contraction. A Boolean tensor network is a bipartite multigraph G in which each edge carries a single bit. Each vertex v has an ordered list $\delta(v)$ of incident edges and a tensor T_v assigning a ring element to each Boolean assignment to those edges. Parallel edges have distinct ports. The *contraction* of the network is

$$Z = \sum_{x \in \{0,1\}^{E(G)}} \prod_{v \in V(G)} T_v(x|_{\delta(v)}).$$

An isolated vertex contributes its scalar entry. We measure the size of the network by its number of edges.

Theorem D.1 (Boolean tensor contraction). *Fix positive constants C, A, b . Let $s \geq 2$ be an integer, and let $\mathcal{L}$ be the total length of the graph and tensor descriptions of a bipartite tensor network with at most Cs edges and maximum degree at most $\lceil s^{9/10} \rceil$. Suppose that the number of vertices is bounded by a fixed polynomial in s , and that the entries belong to*

$$\mathcal{B}_{P_0, \ell} = (\mathbb{Z}/2^{P_0}\mathbb{Z})[V]/(V^\ell), \quad 1 \leq P_0, \ell \leq Cs.$$

Suppose, uniformly for every vertex and Boolean choice of its ports, that its entry can be computed from the description in time $(s + \mathcal{L})^b 2^{As^{9/10}}$ and space $(s + \mathcal{L})^b$. Then the contraction can be computed deterministically in

$$(s + \mathcal{L})^{O(1)} 2^{O(s/\sqrt{\log s})}$$

bit operations and $(s + \mathcal{L})^{O(1)}$ bits of space. Constants depend only on the fixed bounds in the hypotheses. In particular, $\ell = 1$ is permitted.

The algorithm in Section D.4 replaces a single global Taylor expansion by recursion on the wires. At each call it deforms the wire identifications so that, at a chosen interpolation node, only ports in the same group interact. Near that node, interactions between groups expand in variables called modes. A mode of weight j has coefficient divisible by 2^j , so only monomials of total weight below the working precision can contribute. Tapering the Hermite multiplicities with the two-adic valuation of the node bounds that precision. With M wires and u retained valuation levels, the number of mode monomials is $2^{O(M/\sqrt{u})}$, and each monomial factors into contractions with at most $M/8$ wires. At recursion depth d , $M = O(s8^{-d})$ and $u = \Theta((\log s)/2^d)$. The exponents therefore sum geometrically to $O(s/\sqrt{\log s})$.

D.2. Fixed clause density.

Proof of the fixed-density bound in Theorem 1.1. Give each clause a vertex carrying its Boolean satisfaction table, with one port for each distinct variable in the clause. A tautology may be dropped, and an empty clause gives count zero. For a variable with $d \geq 1$ ports, require their bits to agree. For $d = 1$ attach the unary tensor $(1, 1)$, and for $d = 2$ attach a binary equality tensor. For $d \geq 3$ take a tree with d external ports and trivalent equality tensors at its internal vertices. Such a tree has $d - 2$ internal vertices, and each common value on its ports extends uniquely to all its internal edges.

There are at most km clause ports. The graph has $O_k(n + m)$ edges and vertices, and maximum degree at most $\max\{k, 3\}$. Subdivide every edge once by a binary equality tensor. The old vertices and the new vertices form the two sides of a bipartite network. The two bits on a subdivided edge must be equal, so this preserves the contraction. Every satisfying assignment on the occurring variables contributes one and every other assignment contributes zero.

With $s = n + m$, apply Theorem D.1 over $\mathbb{Z}/2^{n+1}\mathbb{Z}$, taking $P_0 = n + 1$ and $\ell = 1$. The local tables have bounded size for fixed k . The degree and precision bounds hold for all sufficiently large s , and bounded instances can be enumerated directly. If u declared variables are unused, multiply the contraction by 2^u modulo 2^{n+1} . Its least nonnegative residue is $\#F$, since $0 \leq \#F \leq 2^n$. This includes the empty formula, for which the empty network has contraction one. The time is $s^{O_k(1)} 2^{O_k(s/\sqrt{\log s})}$ and the space is polynomial. When $m \leq Dn$, we have $n \leq s \leq (1 + D)n$, giving exponent $O_{k,D}(n/\sqrt{\log n})$. $\square$

D.3. Arbitrary clause density. The sparsification and greedy row selection in Section 6 reduce this case to counting common zeros of sparse polynomial equations. We retain the parity lift of Section 5. Its degree bound lets us encode a monomial by a subset of test-variable incidences using $O(n \log \log n)$ Boolean wires.

Lemma D.2 (Counting sparse polynomial systems by contraction). *Fix integers $k, c \geq 1$. Let $P_1, \dots, P_a$ be polynomials over $\mathbb{F}_2$ on $n \geq 20$ declared Boolean variables. Suppose $a \leq cn$ and each P_i is listed as at most $\lceil \log_2 n \rceil^c$ monomials of degree at most k . The number of common zeros is computable deterministically in $n^{O_{k,c}(1)} 2^{O_{k,c}(n \log \log n / \sqrt{\log n})}$ bit operations and $n^{O_{k,c}(1)}$ bits of space.*

Proof. Put $\lambda = \lceil \log_2 n \rceil$ and $\nu = \lceil \log_2 \lambda \rceil$. Within each monomial replace positive powers of a variable by its first power, and combine duplicate monomials modulo two. The scope of test i consists of the variables in its remaining monomials. It has size at most $k\lambda^c$. Let N_0 be the sum of the scope sizes, counting a pair (i, v) once if variable v belongs to test i . Thus $N_0 \leq ak\lambda^c$. Fix, in all possible ways, the variables belonging to more than

$$D_* = \max\{1, \lceil N_0/n \rceil\} \lceil \sqrt{\lambda} \rceil$$

scopes. There are at most $n/\sqrt{\lambda}$ such variables. Their assignments partition the original count into at most $2^{n/\sqrt{\lambda}}$ branch counts. On each branch substitute the fixed bits and combine duplicate monomials again. The free scope I_i of test i still has size at most $k\lambda^c$, and each free variable appears in at most $D_* = \lambda^{O_{k,c}(1)}$ scopes. Free variables absent from every scope remain declared. A constant-one equation makes the branch count zero. We may discard constant-zero equations, and if none remain the answer for the branch is two to the power of its number of free variables. Below a denotes the number of remaining equations and n the original variable count.

The polynomial on the incidences. Give each pair (i, v) with $v \in I_i$ its own indeterminate z_{iv} . Write $z_i = (z_{iv})_{v \in I_i}$. Form $\tilde{L}_i(z_i)$ by summing the remaining monomials of the normalized branch polynomial with integer coefficients, using z_{iv} for x_v . We use the ring

$$\mathcal{B} = (\mathbb{Z}/2^p\mathbb{Z})[V]/(V^\ell), \quad p = \ell = n + 1.$$

Take $h(Y) = Y(Y - 1)$ and the integer polynomials e_j from Lemma 5.1. Reduce their coefficients modulo 2^p , and form the factor

$$B_i(z_i, V) = \sum_{j=0}^{\ell-1} e_j(\tilde{L}_i(z_i)) h(\tilde{L}_i(z_i))^j V^j.$$

Let $\bar{B}_i$ be its multilinear representative in the z_i , obtained in each coefficient of V by replacing each positive power of an incidence indeterminate by its first power. It agrees with B_i on every Boolean assignment to z_i . Define $F(z, V) = \prod_i \bar{B}_i(z_i, V)$ in $\mathcal{B}[z]$. The incidence indeterminates for different tests are disjoint, so F is multilinear even before substituting the same Boolean x_v for all copies of variable v .

The coefficient of V^j in $\bar{B}_i$ has incidence degree at most $k(1 + 2j)$, since $\deg e_j \leq 1$, $\deg h = 2$, and multilinearization only lowers monomial degrees. Multiplication in $\mathcal{B}$ retains only terms with $\sum_i j_i \leq \ell - 1$. Consequently every incidence monomial in F has degree at most

$$K = k(a + 2(\ell - 1)) = O_{k,c}(n). \quad (\text{D.1})$$

Let $\mathcal{S}_\ell$ sum the coefficients of $1, V, \dots, V^{\ell-1}$ in $\mathcal{B}$. At $z_{iv} = x_v$, the multilinearized factor $\bar{B}_i$ agrees with B_i for the normalized branch polynomial. The integer $\tilde{L}_i((x_v)_{v \in I_i})$ has the same parity as the original P_i evaluated at x and the fixed bits. Applying Lemma 5.2 to these integers shows that the branch count is congruent modulo 2^p to

$$\mathcal{S}_\ell \left(\sum_x F((z_{iv} = x_v)_{iv}, V) \right), \quad (\text{D.2})$$

where x runs over all free Boolean assignments. We apply the coefficient-sum map after multiplying the test factors in the truncated ring.

Indexing the monomials. Partition the tests and the free variables separately into $J = \lceil n^{1/4} \rceil$ groups of sizes differing by at most one on each side. Write $\mathcal{T}_g$ for a test group and $\mathcal{X}_h$ for a variable group. A group contains $O_c(n^{3/4})$ objects, so the number of its incidences is at most $n^{3/4} \lambda^{O_{k,c}(1)}$ on either side. Let N_{gh} be the number of incidences from $\mathcal{T}_g$ to $\mathcal{X}_h$ and let $N = \sum_{g,h} N_{gh} \leq N_0$.

For an incidence monomial let d_{gh} be the number of its selected incidences joining groups g, h . Enumerate arrays of integers (d_{gh}) with $0 \leq d_{gh} \leq N_{gh}$ and $\sum_{g,h} d_{gh} \leq K$. There are at most

$$(K + 1)^{J^2} = 2^{O_{k,c}(\sqrt{n} \log n)}$$

arrays, which can be enumerated using J^2 counters. For a fixed array there are

$$C_{gh} = \binom{N_{gh}}{d_{gh}}$$

subsets of the specified size for this pair. Draw $\lceil \log_2 C_{gh} \rceil$ Boolean wires between the two group vertices to index them in lexicographic order. For $C_{gh} = 1$ both endpoints use the unique subset (possibly the entire incidence set) with no wires. Any unused bit string gives tensor entry zero. To decode a valid index, take the incidences in order and count the possible completions with or without each incidence using binomial coefficients. These have polynomial bit length, so decoding uses polynomial time and space.

The union of the subsets over all pairs specifies a squarefree monomial of degree at most K in N indeterminates. Including all monomials of this degree bound gives the upper bound $\binom{N+K}{K}$ by stars

and bars. Thus the number of wires M in the network for this array satisfies

$$\begin{aligned} M &\leq J^2 + \log_2 \prod_{g,h} C_{gh} \leq J^2 + \log_2 \binom{N+K}{K} \\ &\leq J^2 + K \log_2 \left(\frac{e(N+K)}{K} \right) = O_{k,c}(n\nu). \end{aligned} \quad (\text{D.3})$$

Here $K = \Theta_{k,c}(n)$ and $N \leq n\lambda^{O_{k,c}(1)}$. Also $\lceil \log_2 C_{gh} \rceil \leq N_{gh}$, so the degree of each vertex is at most its number of incidences, $n^{3/4}\lambda^{O_{k,c}(1)}$.

Tensor entries and contraction. A valid assignment of wire bits determines a subset $S_i \subseteq I_i$ at each test. Write $z_{i,S_i} = \prod_{v \in S_i} z_{iv}$. The entry at a test group g is the coefficient of its indicated monomial in $\prod_{i \in \mathcal{T}_g} \overline{B}_i(z_i, V)$. Since the z_i are disjoint, this is

$$\prod_{i \in \mathcal{T}_g} [z_{i,S_i}] \overline{B}_i(z_i, V) \quad \text{in } \mathcal{B}.$$

For a variable group h , put $b_v = |\{i : v \in S_i\}|$ for $v \in \mathcal{X}_h$. Its entry is

$$\sum_{x \in \{0,1\}^{\mathcal{X}_h}} \prod_{v \in \mathcal{X}_h} x_v^{b_v} = 2^{|\{v \in \mathcal{X}_h : b_v=0\}|} \pmod{2^p}.$$

Indeed a positive exponent requires $x_v = 1$ and a zero exponent leaves x_v free. This includes variables outside all test scopes.

Each monomial of F occurs in exactly one array and has exactly one valid index at each group pair. Its coefficient in F is the product of the test-group coefficients. After the incidences with the same v are set equal, its sum over Boolean assignments is the product of the variable-group entries. By (D.1), summing over these arrays exhausts all monomials. Consequently the element of $\mathcal{B}$ inside $\mathcal{S}_\ell$ in (D.2) is the sum of the contractions of these bipartite networks.

To evaluate one test-side coefficient, let $\mathbf{1}_T$ be the Boolean vector on I_i with support $T \subseteq S_i$. At $\mathbf{1}_T$, the multilinear polynomial evaluates to the sum of its coefficients indexed by subsets of T , and it agrees there with B_i . Inverting this subset sum gives

$$[z_{i,S_i}] \overline{B}_i(z_i, V) = \sum_{T \subseteq S_i} (-1)^{|S_i| - |T|} B_i(\mathbf{1}_T, V) \quad \text{in } \mathcal{B}.$$

Each term costs polynomially many bit operations. Evaluate $\widetilde{L}_i(\mathbf{1}_T)$ from the listed monomials and use the polynomials e_j of Lemma 5.1, which are constructible in polynomial time. Store the ℓ coefficients of $B_i(\mathbf{1}_T, V)$ modulo 2^p . There are at most $2^{|S_i|} \leq 2^{k\lambda^c}$ subsets to process. Sum one test at a time and multiply the resulting coefficients in $\mathcal{B}$ over the group. This evaluates a test-group entry in $2^{\lambda^{O_{k,c}(1)}}$ bit operations and polynomial space. The variable-side entries and checks for invalid wire indices take polynomial time. The graph, degree array, and entry procedures have polynomial-size descriptions.

Apply Theorem D.1 with size parameter $s = n\nu$, root precision $p = n + 1$, and truncation order $\ell = n + 1$. The graph has $2J$ vertices and $O_{k,c}(s)$ wires by (D.3). For all sufficiently large n its maximum degree is at most $\lceil s^{9/10} \rceil$, its precision and truncation order are $O(s)$, and its entry-evaluation time is at most $2^{O_{k,c}(s^{9/10})}$. Each contraction then takes $n^{O_{k,c}(1)} 2^{O_{k,c}(s/\sqrt{\log s})}$ bit operations and polynomial space. Here $s/\sqrt{\log s} = O(n\nu/\sqrt{\lambda})$. The high-incidence branches and degree arrays multiply the running time by at most

$$2^{n/\sqrt{\lambda} + O_{k,c}(\sqrt{n} \log n)}.$$

Both terms in this exponent are $O_{k,c}(n\nu/\sqrt{\lambda})$. Process the branches and arrays sequentially. On each branch, sum the contractions in $\mathcal{B}$ and apply $\mathcal{S}_\ell$. Adding the branch residues modulo 2^{n+1}

gives the original count, since it lies between 0 and 2^n . This proves the claimed bounds, using direct enumeration for bounded exceptional sizes. $\square$

Proof of the arbitrary-density bound in Theorem 1.1. Normalize the formula on its original n declared variables as in Section 6.4. Empty clauses and formulas with no clauses have immediate answers. For $n \geq 20$ apply Lemma 6.3 with $\sigma = 1/\lceil\sqrt{\lambda}\rceil$, where $\lambda = \lceil\log_2 n\rceil$. The tree has $n^{O_k(1)}2^{O(n/\sqrt{\lambda})}$ nodes and polynomial depth. Every leaf has occurrence bound $B = O_k((\sigma^{-1}\log(2/\sigma))^{k-1}) = \lambda^{O_k(1)}$ and at most Bn nonempty clauses.

For a nontrivial leaf with m_G clauses, use the initial parity checks of Lemma 6.1 and the greedy procedure of Lemma 6.2. Each initial row has support size at most D_0 , and each additional row has list length P . Both bounds are $\lambda^{O_k(1)}$. By (6.1), every row gives a degree-at-most- k equation with at most $2^k \max\{P, D_0\}$ monomials, and every queried system has at most $6n$ equations. A single constant $c = c(k)$ bounds both the ratio of equations to variables and the exponent of λ in the monomial bound for all sufficiently large n . At each greedy step, use Lemma D.2 to compute $N(A; h_\gamma)$ for every candidate row h_γ and append one attaining the minimum. The evaluator applies also to discarded candidates whose systems still accept nonsatisfying assignments. One final call gives $N(A)$ for the selected matrix, which equals the leaf count by Lemma 6.2.

Constructing the initial checks and enumerating candidates costs $2^{\lambda^{O_k(1)}}$ bit operations apart from the evaluations. There are at most $3n(2m_G)^P + 1 = 2^{\lambda^{O_k(1)}}$ evaluations per leaf. Since every fixed power of λ is $o(n \log \log n / \sqrt{\log n})$, these costs and the number of leaves are absorbed in the claimed time bound

$$(n + m)^{O_k(1)} 2^{O_k(n \log \log n / \sqrt{\log n})}.$$

Here m counts the input clauses. Traverse the tree depth first and process one query at a time. The leaf solution sets are disjoint, so their exact counts add with an $(n+1)$ -bit accumulator. Together with polynomial space per query and per tree path, this gives the asserted polynomial-space bound. $\square$

A single algorithm attains both bounds. With $\lambda = \lceil\log_2 n\rceil$ and $\nu = \lceil\log_2 \lambda\rceil$, use the clause network when $m \leq n\nu$ and the parity queries otherwise. In the first branch $n \leq s = n + m = O(n \log \log n)$, so its exponent is $O_k(n \log \log n / \sqrt{\log n})$ even at growing density. For each fixed D and all sufficiently large n , every input with $m \leq Dn$ takes that branch with $s = O_D(n)$.

D.4. Proof of the contraction theorem. Throughout the proof of Theorem D.1, $\log$ denotes a base-two logarithm and $\ln$ denotes a natural logarithm.

D.4.1. The coefficient ring and exact arithmetic. Write $\lambda = \lceil\log_2 s\rceil$ and $e = 4\lambda$. Choose a monic polynomial $f \in \mathbb{Z}[U]$ of degree e , with coefficients in $\{0, 1\}$ below its leading coefficient, whose reduction modulo two is irreducible. This can be done deterministically in polynomial time in s by enumerating the 2^e candidates and testing divisibility over $\mathbb{F}_2$ by every monic polynomial of positive degree at most $e/2$. Finite fields admit an irreducible polynomial of every positive degree [vzGG13]. Put

$$\mathcal{O} = \mathbb{Z}_{(2)}[U]/(f), \quad \mathcal{K} = \mathbb{Q}[U]/(f), \quad \mathcal{A} = \mathcal{O}[V]/(V^\ell), \quad \mathcal{A}_p = \mathcal{A}/(2^p).$$

Here $\mathbb{Z}_{(2)}$ consists of rational numbers with odd denominator. Irreducibility modulo two implies irreducibility over $\mathbb{Q}$, so $\mathcal{K}$ is a field. For a nonzero element of $\mathcal{K}$, let v_2 be the minimum of the ordinary two-adic valuations of its coordinates in $1, U, \dots, U^{e-1}$. Reduction of $\mathcal{O}$ modulo two is the field $\mathbb{F}_{2^e}$. Factoring out the minimum coordinate valuation therefore shows that

$$v_2(xy) = v_2(x) + v_2(y) \quad (x, y \neq 0).$$

Every element of $\mathcal{O}$ that is nonzero modulo two is a unit because the determinant of its multiplication matrix is invertible in $\mathbb{Z}_{(2)}$, so the inverse matrix has entries in that ring.

Arithmetic in $\mathcal{A}_p$ takes polynomial time and space in s, p, ℓ . We use the bases $1, U, \dots, U^{e-1}$ and $1, V, \dots, V^{\ell-1}$, and represent each coordinate by its integer residue between zero and $2^p - 1$.

Inverting a scalar unit can be done by inversion modulo two followed by Newton lifting, or by its multiplication matrix, using exact polynomial and integer operations [vzGG13]. Divisibility and precision for elements of $\mathcal{A}$ are always coordinatewise.

The binary-coordinate lifts of the 2^e residue-field elements give interpolation points with pairwise unit differences. Lagrange interpolation at any required number of these points is therefore valid over $\mathcal{A}_p$. We will use fewer than s^4 points. A contraction whose initial entries belong to $\mathcal{B}_{P_0, \ell}$ belongs to that subring after computation in $\mathcal{A}_{P_0}$, because it is a polynomial in those entries.

When the precision increases, we lift each tensor entry separately. For an entry originally specified modulo 2^p , its *canonical lift* is the element of $\mathcal{A}$ obtained from the integer representatives of all its coordinates. Canonical lifting need not preserve addition or multiplication. Each recursive call contracts its specified tensors modulo the precision of that call. To define its deformation, it first makes this entrywise choice of lifts. All subsequent evaluations in that call refer to these same lifts. More explicitly, each tensor description is tagged with its creation precision p_c (P_0 for an original tensor). At every later precision, its entry is evaluated modulo 2^{p_c} and those same integer representatives are used. Precisions strictly increase along a recursive path, so these representatives are also the canonical representatives at every later precision. In particular, a packet tensor retains its creation-time representatives throughout the recursion.

D.4.2. Parameters and collision packing. All estimates below hold for sufficiently large s , with the threshold depending only on the fixed constants. A fixed finite range of smaller sizes is handled by direct summation. The parameter inequalities can be tested directly. Set

$$H = \left\lceil \frac{s}{\lfloor \sqrt{\lambda} \rfloor} \right\rceil, \quad r = 64, \quad \varepsilon = \frac{1}{128 \cdot 1024}, \quad \Delta = \lceil s^{9/10} \rceil.$$

A call with $M \leq H$ edges is a leaf and is evaluated directly. At an active call at depth d , with the root at depth zero, use

$$u = u_d = \left\lfloor \frac{\varepsilon \lambda}{2^d} \right\rfloor, \quad a = a_d = 2 \sum_{j=0}^d u_j, \quad a_{-1} = 0, \quad L = 2^a, \quad D_{\text{arr}} = r^{2a}. \quad (\text{D.4})$$

We will arrange that every child has at most $M/8$ edges and maximum degree at most Δ . These assertions imply, on every active path,

$$H < M \leq C s 8^{-d}, \quad d \leq \frac{\log_2 \lambda}{6} + O_C(1).$$

Consequently

$$u = \Theta_C(\lambda/2^d), \quad \lambda^{4/5} \leq u \leq M, \quad a \leq 4\varepsilon \lambda, \quad L, D_{\text{arr}} \leq s^{1/1000} \quad (\text{D.5})$$

for all sufficiently large s . In particular, at each active call

$$M \geq 2D_{\text{arr}}^2 \Delta, \quad M \geq LD_{\text{arr}}, \quad 2L \leq \Delta. \quad (\text{D.6})$$

For example, the right side of the first inequality is $O(s^{9/10+2/1000}) = o(H)$. The other inequalities follow in the same way.

Lemma D.3 (Packing into packets). *Under (D.6), label the vertices on each side of the network by elements of $[r]^a$. There is a deterministically constructible labeling and an embedding of the edges into packets of L slots with the following properties.*

- (1) *There are at most $3M/L$ packets. Each edge occupies the same slot at its row and column ends. Unused slots have no ports.*
- (2) *In every packet and every coordinate, the unordered pair of row and column symbols is fixed. When these symbols differ, bit i of the slot number specifies their orientation in coordinate i , for $0 \leq i < a$.*

- (3) For every coordinate and every symbol, at most $3M/r$ real ports on each side belong to vertices with that symbol.

The construction uses polynomial time and space in $s + \mathcal{L}$.

Proof. For each pair of labels let X_γ be the number of edges with that pair. Independent uniform vertex labels give

$$\mathbb{E} \sum_{\gamma} (X_\gamma - M/D_{\text{arr}})^2 \leq 2M\Delta.$$

Indeed, expansion expresses this expectation as a sum over ordered pairs of edges of their label-agreement probability minus $1/D_{\text{arr}}$. Pairs with disjoint endpoints contribute zero. At most $2M\Delta$ ordered pairs share an endpoint, including equal edges, and each contributes at most one. Conditional expectation gives a labeling with this bound. To implement it, try each of the r^a labels of the next vertex and compute the conditional expectation by enumerating pairs of edges. Each probability concerns at most four vertices, so this is an exact polynomial-time calculation. This is the usual conditional-expectation method [Rag88].

Every load is at most $M/D_{\text{arr}} + \sqrt{2M\Delta}$. Pad all pair-label classes to a common number H_{pad} of slots, the maximum load rounded up to a multiple of L . The assumptions imply

$$H_{\text{pad}} \leq M/D_{\text{arr}} + \sqrt{2M\Delta} + L \leq 3M/D_{\text{arr}}.$$

Fix the unordered pair of symbols in each coordinate. If a' coordinates have unequal symbols, their orientations give $2^{a'}$ pair labels. Partition their padded slots into packets using $2^{a-a'}$ slots from each of these labels per packet. This is possible because H_{pad} is divisible by L . Number a packet's positions from 0 to $L - 1$. Its i th bit specifies the orientation when the i th pair is unequal and the copy when the pair is equal. Use one orientation convention throughout each packet.

There are $H_{\text{pad}}D_{\text{arr}}/L \leq 3M/L$ packets. In a fixed coordinate, a given symbol occurs in exactly $1/r$ of all padded pair labels on either side. Its number of real ports is therefore at most $H_{\text{pad}}D_{\text{arr}}/r \leq 3M/r$. $\square$

D.4.3. *A Cauchy deformation and integral local coordinates.* Fix an active call, its packets, and its entrywise lifted tensors. Use

$$S = \{-L, -L+1, \dots, L-1\}, \quad \mathcal{C}(T) = \prod_{t \in S} (T - t), \quad \alpha = 2^{a+1}.$$

Within a packet, for its row slot x and column slot y , put

$$W_{xy}(T) = \frac{\mathcal{C}(T)}{\mathcal{C}'(0)(T - (y - x))}. \quad (\text{D.7})$$

The factor in the denominator cancels a numerator factor, since $y - x \in S$. Thus this is a polynomial over $\mathbb{Q}$ of degree $2L - 1$. Set entries between different packets to zero and restrict to the real ports. Under the original edge identification, $W(0) = I$, since the diagonal entries are one and each other entry has an uncanceled zero factor.

Let $A(Y)$ and $B(X)$ be the products of the lifted row and column tensors, respectively, on independent port assignments $Y, X \in \{0, 1\}^M$. Write $l_0(z) = 1 - z$ and $l_1(z) = z$. Define

$$F(T) = \sum_{X, Y \in \{0, 1\}^M} A(Y)B(X) \prod_{\text{row ports } x} l_{Y_x}((W(T)X)_x). \quad (\text{D.8})$$

Its coefficients lie in $\mathcal{K}[V]/(V^\ell)$, its degree is at most $M(2L - 1)$, and $F(0)$ is the contraction of the lifted tensors. In particular, reducing $F(0)$ modulo the call's precision gives the required answer.

Lemma D.4. *For every $t \in S$, the entries of $W(t + \alpha z)$ belong to $\mathcal{O}[z]$, and $F(t + \alpha z)$ belongs to $\mathcal{A}[z]$. Given the e coordinates of ζ modulo 2^p , each entry $W_{xy}(t + \alpha\zeta) \bmod 2^p$ can be computed using polynomially many bit operations in s, p .*

Proof. The grid S is a complete residue system modulo 2^{a+1} . For $t \neq q$ in S , $v_2(t - q) \leq a$, and the counts of nonzero differences at every valuation are independent of t . Hence $v_2(C'(t)) = v_2(C'(0))$. In

$$\frac{C(t + \alpha z)}{C'(0)} = \frac{\alpha z \prod_{q \in S \setminus \{t\}} (t - q + \alpha z)}{C'(0)},$$

each factor indexed by $q \neq t$ is coefficientwise divisible by $2^{v_2(t-q)}$. After canceling these powers against the denominator, the displayed polynomial is divisible by 2^{a+1} . To obtain (D.7), omit just the factor indexed by $y - x$. If $y - x = t$, this removes αz and leaves an integral polynomial. Otherwise αz remains and the omitted factor has two-adic content $2^{v_2(t-y+x)}$ with $v_2(t - y + x) \leq a$, so the result is still divisible by two. Substitution in (D.8) proves the assertion for F .

For the evaluation, first divide each factor $t - q + \alpha z$ with $q \neq t$ by the known integer $2^{v_2(t-q)}$. Both coefficients of the resulting linear polynomial are integers. Cancel the remaining explicit powers of two against the denominator as above, and invert its odd part modulo 2^p . We can now evaluate the product directly at the supplied residue of ζ , including $\zeta = 0$. The input has ep bits, and the product has $O(L)$ factors. All coordinate operations take polynomial time in s, p . $\square$

D.4.4. *Tapered Hermite recovery.* The next lemma isolates the precision estimate. All power-series truncations in its statement are ordinary truncations in z .

Lemma D.5 (Tapered Hermite recovery). *Let integers u, M satisfy $1 \leq u \leq a$ and $M \geq u$, and put $h = \lceil 3M/u \rceil$. For $t \in S \setminus \{0\}$ set*

$$i = v_2(t), \quad q_t = q(i) = h \max(0, u - i), \quad D_t = L/2^i.$$

Suppose $F \in (\mathcal{K}[V]/(V^\ell))[T]$ has degree at most $M(2L - 1)$, $F(0) \in \mathcal{A}$, and $F(t + \alpha z) \in \mathcal{A}[z]$ for every $t \in S$. For $p \geq 1$, its value $F(0)$ modulo 2^p can be recovered from

$$F(t + \alpha z) \pmod{2^{p_t}}, \quad p_t = p + 8hD_t + 8M(a + 1), \quad q_t > 0.$$

Only the first q_t coefficients of each polynomial are needed. Recovery takes polynomial time and space in M, L, ℓ, e and the precisions.

Proof. There are L odd points in S , so $\sum_{t \neq 0} q_t \geq Lhu \geq 3ML > M(2L - 1)$. Also $h \leq 4M/u$, and hence $q_t \leq 4M$. For each t of positive multiplicity put

$$E_t(T) = \prod_{q \in S \setminus \{0, t\}} \left(\frac{T - q}{t - q} \right)^{q_q}.$$

Hermite interpolation gives

$$F(0) = \sum_{t: q_t > 0} E_t(0) \left(\left[\frac{F(t + \alpha z)}{E_t(t + \alpha z)} \right]_{< q_t} \right) (-t/\alpha). \quad (\text{D.9})$$

For completeness, replace zero in the right side by an indeterminate T , and $-t/\alpha$ by $(T - t)/\alpha$. Each summand has degree less than $\sum q_t$. At its own node it has the prescribed jet, and at every other node it vanishes to the required order. The difference from F is therefore divisible by $\prod_{t \neq 0} (T - t)^{q_t}$ and has smaller degree, so is zero. This is the usual Chinese-remainder construction of Hermite interpolation [vzGG13, Section 5.6].

Each factor of $E_t(t + \alpha z)$ is a power of $1 + \alpha z/(t - q)$. It is integral and has constant term one. Its reciprocal truncated at q_t is therefore integral, so forming the quotient in (D.9) loses no precision.

Fix $i = v_2(t)$ and $D = D_t$. Directly from the product,

$$\begin{aligned} v_2(E_t(0)) &= \sum_{q \neq 0, t} q_q (v_2(q) - v_2(t - q)) \\ &= D \sum_{j=1}^{a-i} \frac{j}{2^j} (q(i+j) - q(i)) \geq -6Dh. \end{aligned} \quad (\text{D.10})$$

To justify the second equality, points of valuation less than i contribute zero. For $1 \leq j \leq a - i$, there are $D/2^j$ points with $v_2(q) = i + j$, and the same number with $v_2(t - q) = i + j$. The latter have $v_2(q) = i$, and these levels exhaust the points other than t with that valuation. These counts follow from the complete residue grid, whose nonzero integer differences have valuation at most a . Finally, $q(i+j) - q(i) \geq -hj$ and $\sum_{j \geq 1} j^2/2^j = 6$.

Evaluation of a polynomial of degree below q_t at $-t/\alpha$ loses at most $4M(a+1)$ further powers of two. Thus replacing each quotient coefficient by any lift of its residue modulo 2^{p_t} changes its contribution in (D.9) by an element of $2^p \mathcal{A}$. Summing these errors loses no further precision. The reconstructed sum is integral, since it differs from the integral value $F(0)$ by an element of $2^p \mathcal{A}$.

One can perform this last combination by rational arithmetic in the specified bases. The multipliers $E_t(0)(-t/\alpha)^j$, $j < q_t$, are products of polynomially many rational factors of $O(a)$ bits, counting multiplicity. Their numerator and denominator lengths are therefore polynomial in M, L, a . There are polynomially many such coefficients, so even a product of their denominators is an adequate polynomial-bit common denominator. After addition, cancel its powers of two and reduce the resulting integral coordinates modulo 2^p . This proves the algorithmic assertion. $\square$

At a call with input precision p , apply this lemma with the parameters in (D.4). For every active t , obtain the entire polynomial $F(t + \alpha z)$ modulo 2^{p_t} by evaluation at $M(2L - 1) + 1$ distinct binary-coordinate lifts of residue-field elements and Lagrange interpolation. Lemma D.4 permits this interpolation over $\mathcal{A}_{p_t}$. The number of points is less than s^4 by (D.5). The evaluation algorithm below passes precision p_t to its children.

The input and child precisions satisfy

$$p \leq s\lambda^{O(1)}2^{a_d-1}, \quad p_t = O_C(MD_t/u). \quad (\text{D.11})$$

The first assertion holds at the root since $P_0 \leq Cs$. Along a path, the increments in precision are $8hD_t + 8M(a+1)$, with $h \leq 4M/u$, $M \leq Cs$, and $D_t \leq 2^{a_d}$. Summing them over $O(\log \lambda)$ depths proves the first assertion for every call, including leaves. For the second, $q_t > 0$ implies $i < u$ and hence

$$D_t \geq 2^{a_d-u} = 2^{a_{d-1}+u}.$$

Since $M > H$ and $u \geq \lambda^{4/5}$, this extra factor 2^u dominates every fixed power of λ . Thus the first bound on p , as well as $8M(a+1)$, is $O_C(MD_t/u)$. The remaining increment $8hD_t$ has that bound directly. All precisions remain polynomial in s because $a_d \leq 4\epsilon\lambda$.

The taper accounts for the factor $1/u$ in this estimate. For comparison, assign the same multiplicity $q_* = \Theta(M)$ to every level below u , and zero to the remaining levels. At the last retained level $i = u - 1$, the identity (D.10) would give the loss

$$-q_* D_t \sum_{j=1}^{a-i} \frac{j}{2^j} = -\Theta(MD_t).$$

Our multiplicities instead change by only $h = \Theta(M/u)$ per level, giving loss $O(MD_t/u)$. The cumulative choice of a_d provides the additional factor 2^u needed to absorb inherited precision and the evaluation loss. Thus the smaller bound holds at every retained level and every recursion depth.

D.4.5. *Modes for interactions between groups.* Fix an active t and an interpolation value $\zeta \in \mathcal{O}$ for z . In this subsection $T = t + \alpha\zeta$ and all calculations are modulo 2^{p_t} . Put $i = v_2(t) < u$ and $D = L/2^i$. Group vertices on both sides by their symbol in coordinate i . Entries of $W(T)$ whose two ports have the same group are called local and are kept unchanged.

This choice of coordinate makes $W(t)$ local. Indeed, $W_{xy}(t) = 0$ unless $y - x = t$. Adding an integer of valuation i flips bit i , so a surviving pair has different i th bits. When the packet's two symbols in that coordinate differ, this means that the row and column symbols agree. Equal pairs of symbols already lie in one group. Moving from t to $t + \alpha\zeta$ introduces the crossing terms that we now expand.

In a packet whose two symbols in coordinate i differ, a row position x and a column position y are in different groups exactly when their i th bits are equal. Partition columns into tiles

$$y = y_0 + 2^{i+1}j, \quad 0 \leq y_0 < 2^{i+1}.$$

The column group is constant on a tile. For a row crossing to that tile,

$$v_2(t - y_0 + x) \leq i.$$

Otherwise y_0 would agree with $x + 2^i$ modulo 2^{i+1} , contrary to equality of their i th bits. Adding $\alpha\zeta$ does not change this valuation. A geometric expansion consequently gives

$$W_{xy}(T) = \sum_{\substack{b \geq 0 \\ b+1 < p_t}} \frac{\mathcal{C}(T)}{\mathcal{C}'(0)(T - y_0 + x)} \left(\frac{2^{i+1}}{T - y_0 + x} \right)^b j^b \pmod{2^{p_t}}. \quad (\text{D.12})$$

Indeed the first fraction has valuation at least $a + 1 - i \geq 1$, and the ratio raised to the b th power has valuation at least one before taking that power. The geometric remainder is divisible by 2^{p_t} . These expressions are computable by factoring the known powers of two and using only unit inverses.

For each such packet, tile, and b introduce a mode variable Z of weight $w_Z = b + 1$, to be replaced by

$$L_Z(X) = \sum_{\text{real columns } y \text{ in the tile}} j^b X_y.$$

As usual $j^0 = 1$, including $j = 0$. Its source is the tile's column group and its target is the other group in that packet's coordinate pair. Formula (D.12) expresses each row form as its local part plus a linear combination of its targeted modes. The coefficient of Z is divisible by 2^{w_Z} . There are at most

$$(3M/L)2^{i+1} = 6M/D \quad (\text{D.13})$$

variables of any one weight.

In the product of the affine factors $l_{Y_x}((W(T)X)_x)$, every coefficient of a mode monomial Z^β is divisible by $2^{\sum_Z w_Z \beta_Z}$. Consequently only the exponent tuples

$$\sum_Z w_Z \beta_Z < p_t \quad (\text{D.14})$$

need be considered. The number of these tuples is

$$\exp(O(\sqrt{Mp_t/D})) = \exp(O_C(M/\sqrt{u})). \quad (\text{D.15})$$

Here is the complete counting estimate. For any $\theta > 0$, the generating product and (D.13) bound the number by

$$\exp \left(\theta p_t + \frac{6M}{D} \sum_{j \geq 1} -\ln(1 - e^{-\theta j}) \right).$$

Expanding the logarithm and summing the geometric series gives

$$\sum_{j \geq 1} -\ln(1 - e^{-\theta j}) = \sum_{h \geq 1} \frac{1}{h(e^{\theta h} - 1)} \leq \frac{1}{\theta} \sum_{h \geq 1} \frac{1}{h^2}.$$

Take $\theta = \sqrt{M/(Dp_t)}$ and use (D.11). All mode variables and their exponents have polynomial-size descriptions. Tuples satisfying (D.14) can be enumerated with polynomial delay and space by ordering the variables and successively choosing each exponent subject to the remaining budget. Every feasible partial choice has the all-zero continuation, so the number of visited partial choices is at most the number of complete tuples times the number of variables, up to an additive constant.

D.4.6. The child networks and their descriptions. For one tuple β in (D.14) construct a child network for each group $g \in [r]$. Place all current vertices in group g , from both sides of the current network, on one side of the child. Retain their ports and their chosen lifted entries, now reduced modulo 2^{p_t} . On the other side put one vertex for each packet, with an edge to every retained port of that packet. Its indices are precisely the row bits Y and column bits X of group g in the packet.

To define this new packet tensor, form the product of the row factors l_{Y_x} for group g in the packet, with their arguments expressed in local column bits and formal modes. Extract the coefficient with exponents β_Z for all modes targeted at g in that packet. Multiply it by

$$\prod_{\substack{Z \text{ in this packet} \\ \text{source}(Z)=g}} L_Z(X)^{\beta_Z}.$$

Empty products equal one. If there are no target rows, the extracted coefficient is zero whenever a target exponent is positive. If the source tile of Z has no real column, then $L_Z(X) = 0$ and its power contributes one for $\beta_Z = 0$ and zero otherwise.

Every mode has one target and one source. Coefficient extraction therefore factors over the packet-group row products, and the substitution for each mode depends only on bits in its source group. Distributing the product in (D.8), and then summing the independent bits within groups, proves the exact identity

$$F(t + \alpha\zeta) = \sum_{\beta: \sum w_Z \beta_Z < p_t} \prod_{g=1}^r Z_{g,\beta} \pmod{2^{p_t}}, \quad (\text{D.16})$$

where $Z_{g,\beta}$ is the contraction of the child just defined. The omitted monomials vanish by their coefficient divisibility.

By Lemma D.3, a child has at most $6M/r = 3M/32 < M/8$ edges. Old vertex degrees are unchanged, and a new packet vertex has at most $2L \leq \Delta$ ports. Thus the invariants used in (D.5) hold inductively.

All new tensors are given implicitly. To evaluate a packet tensor entry, expand its product of at most L affine row factors. Each factor has polynomially many terms in s , including its local term. Enumerating these choices, checking the desired exponents, and summing the resulting terms takes

$$\exp(O(L \log s))$$

time and polynomial space. The factors $L_Z(X)^{\beta_Z}$ are computed by ordinary repeated squaring in $\mathcal{A}_{p_t}$. Because $L \leq s^{1/1000}$, this entry cost is $\exp(O(s^{9/10}))$.

A description of a new tensor consists of its packet data, the tuple β , the interpolation value, and the creation precision. These data and the local kernel suffice to compute its entry directly. A retained tensor is evaluated from its original description, or from its own creation-time description if it was created earlier in the recursion. Its residue at that precision is then canonically lifted to the current precision. The old descriptions are immutable and shared by reference. Thus no composition of entry routines builds up with the depth, and all entry routines continue to have the required time

and space bounds. Only polynomially many vertices and polynomially much new description data are added along a path.

D.4.7. Time and space bounds.

Proof of Theorem D.1. Embed the original entries into $\mathcal{A}_{P_0}$ and perform the recursive algorithm above. At an active call, for every node of positive multiplicity and every required residue-distinct interpolation value, compute (D.16) by its child contractions. Interpolate $F(t + \alpha z)$ and apply Lemma D.5. At a leaf, sum directly over all Boolean edge assignments. Lemma D.4, the recursion identity, and the recovery lemma prove correctness by induction from the leaves.

The number of interpolation nodes, values, and groups at a call is polynomial in s , independently of $\mathcal{L}$. By (D.15), the number of children of an active depth- d call is at most

$$s^{O(1)} \exp \left(O_C \left(\frac{s}{\sqrt{\lambda}} \left(\frac{\sqrt{2}}{8} \right)^d \right) \right).$$

Indeed $M \leq Cs8^{-d}$ and $u_d = \Theta_C(\lambda/2^d)$. Multiplying these bounds over the $O_C(\log \lambda)$ depths, or bounding the number of nodes on each level in this way, gives at most

$$\exp \left(O_C(s/\sqrt{\lambda}) + O_C(\lambda \log \lambda) \right) = \exp(O_C(s/\sqrt{\lambda}))$$

calls in total. At a leaf, direct summation uses at most 2^H assignments and polynomially many entry evaluations per assignment. Its cost is bounded by a polynomial in $s + \mathcal{L}$ times

$$\exp(O_C(H) + O_{C,A}(s^{9/10})) = \exp(O_{C,A}(s/\sqrt{\lambda})).$$

Local arithmetic, packing, descriptions, and recovery use polynomial time per generated term, apart from the already bounded entry computations. The description-dependent factor bounds the work per call. Since the branching bound is independent of $\mathcal{L}$, the total work retains a fixed polynomial factor in $s + \mathcal{L}$. This proves the time bound.

For space, traverse the recursion depth first, accumulating one sum over β and one product over groups at a time. Store the polynomially many Taylor data needed for a call's recovery, its current interpolation and mode indices, and its graph and new tensor descriptions. Their sizes are polynomial in $s + \mathcal{L}$, including the precisions by (D.11). There are only $O_C(\log \lambda)$ stack levels. The common-denominator calculation in Lemma D.5 also uses only polynomial-bit integers. Each call accumulates its contraction terms modulo its own fixed precision. Original entry routines use polynomial space by hypothesis, and the new routines have the same property. This proves the space bound. Finally extract the base coordinates in $\mathcal{A}_{P_0}$, which give the answer in $\mathcal{B}_{P_0,\ell}$. $\square$

REFERENCES

- [ABHS19] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. *SETH-Based Lower Bounds for Subset Sum and Bicriteria Path*. 30th ACM–SIAM Symposium on Discrete Algorithms, 2019, 41–57. 5
- [AW14] Amir Abboud and Virginia Vassilevska Williams. *Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems*. 55th IEEE Symposium on Foundations of Computer Science, 2014, 434–443. 5
- [AGP23] Josh Alman, Yunfeng Guan, and Ashwin Padaki. *Smaller Low-Depth Circuits for Kronecker Powers*. 34th ACM–SIAM Symposium on Discrete Algorithms, 2023, 4159–4187. 6
- [AW15] Josh Alman and Ryan Williams. *Probabilistic Polynomials and Hamming Nearest Neighbors*. 56th IEEE Symposium on Foundations of Computer Science, 2015, 136–150. 6
- [AW17] Josh Alman and Ryan Williams. *Probabilistic Rank and Matrix Rigidity*. 49th ACM Symposium on Theory of Computing, 2017, 641–652. 6
- [AR94] Noga Alon and Yuval Roichman. *Random Cayley Graphs and Expanders*. Random Structures & Algorithms 5(2), 1994, 271–284. 7
- [BI16] Arturs Backurs and Piotr Indyk. *Which Regular Expression Patterns Are Hard to Match?* 57th IEEE Symposium on Foundations of Computer Science, 2016, 457–466. 5

- [BI18] Arturs Backurs and Piotr Indyk. *Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)*. SIAM Journal on Computing **47**(3), 2018, 1087–1097. Conference version in STOC 2015. 5
- [Bar23] Boaz Barak. *Introduction to Theoretical Computer Science*. Online textbook, 2023. Chapter 15, Lemmas 15.9–15.10. Archived 2023 edition (v0.95). 2, 3
- [Bar68] Erwin H. Bareiss. *Sylvester’s Identity and Multistep Integer-Preserving Gaussian Elimination*. Mathematics of Computation **22**(103), 1968, 565–578. 50
- [BI13] Chris Beck and Russell Impagliazzo. *Strong ETH Holds for Regular Resolution*. 45th ACM Symposium on Theory of Computing, 2013, 487–494. 4
- [BT94] Richard Beigel and Jun Tarui. *On ACC*. Computational Complexity **4**(4), 1994, 350–366. 36
- [BGKM23] Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. *Fast, Algebraic Multivariate Multipoint Evaluation in Small Characteristic and Applications*. Journal of the ACM **70**(6), 2023, 42:1–42:46. Conference version in STOC 2022. 6
- [Bjö16] Andreas Björklund. *Below All Subsets for Some Permutational Counting Problems*. 15th Scandinavian Symposium and Workshops on Algorithm Theory, LIPIcs **53**, 2016, 17:1–17:11. 5
- [BKW19] Andreas Björklund, Petteri Kaski, and Ryan Williams. *Generalized Kakeya Sets for Polynomial Evaluation and Faster Computation of Fermionants*. Algorithmica **81**, 2019, 4010–4028. Conference version in IPEC 2017. 5
- [Bri14] Karl Bringmann. *Why Walking the Dog Takes Time: Fréchet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails*. 55th IEEE Symposium on Foundations of Computer Science, 2014, 661–670. 5
- [BK15] Karl Bringmann and Marvin Künnemann. *Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping*. 56th IEEE Symposium on Foundations of Computer Science, 2015, 79–97. 5
- [Cal09] Chris Calabro. *The Exponential Complexity of Satisfiability Problems*. Ph.D. dissertation, University of California, San Diego, 2009. Chapter 5. 6, 18
- [CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. *A Duality between Clause Width and Clause Density for SAT*. 21st IEEE Conference on Computational Complexity, 2006, 252–260. 6, 18, 44
- [CGI⁺16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. *Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility*. 7th Innovations in Theoretical Computer Science Conference, 2016, 261–270. 4
- [CCK⁺20] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. *From Gap-Exponential Time Hypothesis to Fixed Parameter Tractable Inapproximability: Clique, Dominating Set, and More*. SIAM Journal on Computing **49**(4), 2020, 772–810. Conference version in FOCS 2017. 4
- [CW21] Timothy M. Chan and R. Ryan Williams. *Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov–Smolensky*. ACM Transactions on Algorithms **17**(1), Article 2, 2021, 1–14. Authors’ manuscript, Theorems 1.6 and 2.5. 4, 36
- [CHKX06] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. *Strong Computational Lower Bounds via Parameterized Complexity*. Journal of Computer and System Sciences **72**(8), 2006, 1346–1367. 5
- [CLWYY25] Zongchen Chen, Aditya Lonkar, Chunyang Wang, Kuan Yang, and Yitong Yin. *Counting Random k -SAT near the Satisfiability Threshold*. 57th ACM Symposium on Theory of Computing, 2025, 867–878. Theorem 1.2 is numbered as in the authors’ full version. 4
- [Coo71] Stephen A. Cook. *The Complexity of Theorem-Proving Procedures*. 3rd ACM Symposium on Theory of Computing, 1971, 151–158. 1
- [Cur18] Radu Curticapean. *Block Interpolation: A Framework for Tight Exponential-Time Counting Complexity*. Information and Computation **261**, Part 2, 2018, 265–280. Conference version in ICALP 2015. 4
- [CDL⁺16] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. *On Problems as Hard as CNF-SAT*. ACM Transactions on Algorithms **12**(3), Article 41, 2016, 1–24. 4, 5
- [DHM⁺14] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. *Exponential Time Complexity of the Permanent and the Tutte Polynomial*. ACM Transactions on Algorithms **10**(4), Article 21, 2014, 1–32. 4, 6, 18, 44
- [Din16] Irit Dinur. *Mildly Exponential Reduction from Gap 3SAT to Polynomial-Gap Label-Cover*. Electronic Colloquium on Computational Complexity, Report TR16-128, 2016. 4
- [DL20] Zeev Dvir and Allen Liu. *Fourier and Circulant Matrices are Not Rigid*. Theory of Computing **16**(20), 2020, 1–48. 6
- [vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd edition, Cambridge University Press, 2013. 57, 58, 60

- [Gol00] Oded Goldreich. *Candidate One-Way Functions Based on Expander Graphs*. Electronic Colloquium on Computational Complexity, Report TR00-090, 2000. 53
- [Gol08] Oded Goldreich. *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Author’s draft, June 14, 2008, Proposition 4.7. The Cambridge University Press edition appeared in 2010. 19
- [HKZZ19] Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. *Faster k -SAT Algorithms Using Biased-PPSZ*. 51st ACM Symposium on Theory of Computing, 2019, 578–589. 3
- [Her14] Timon Hertli. *3-SAT Faster and Simpler—Unique-SAT Bounds for PPSZ Hold in General*. SIAM Journal on Computing **43**(2), 2014, 718–729. Conference version in FOCS 2011. 3
- [Hus11] Thore Husfeldt. *Invitation to Algorithmic Uses of Inclusion–Exclusion*. 38th International Colloquium on Automata, Languages, and Programming, Part II, Lecture Notes in Computer Science **6756**, 2011, 42–59. 5
- [IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. *A Satisfiability Algorithm for AC^0* . 23rd ACM–SIAM Symposium on Discrete Algorithms, 2012, 961–972. Publisher’s version. 4
- [IP01] Russell Impagliazzo and Ramamohan Paturi. *On the Complexity of k -SAT*. Journal of Computer and System Sciences **62**(2), 2001, 367–375. 1, 4
- [IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. *Which Problems Have Strongly Exponential Complexity?* Journal of Computer and System Sciences **63**(4), 2001, 512–530. 1, 5, 6, 18
- [JMV13] Hamid Jahanjou, Eric Miles, and Emanuele Viola. *Local Reductions*. Electronic Colloquium on Computational Complexity, Report TR13-099, revision 2, 2013. Conference version in ICALP 2015, 749–760. 51
- [JVV86] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. *Random Generation of Combinatorial Structures from a Uniform Distribution*. Theoretical Computer Science **43**(2–3), 1986, 169–188. 2, 53
- [JC26] Tao Jiang and Shaowei Cai. *A Better Analysis For PPSZ For 3-SAT*. arXiv:2607.10697, 2026. 3
- [Koi06] Mikko Koivisto. *Optimal 2-Constraint Satisfaction via Sum–Product Algorithms*. Information Processing Letters **98**(1), 2006, 24–28. 5
- [KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. *High-Rate Codes with Sublinear-Time Decoding*. Journal of the ACM **61**(5), 2014, 28:1–28:20. 6
- [KW16] Ioannis Koutis and Ryan Williams. *Algebraic Fingerprints for Faster Algorithms*. Communications of the ACM **59**(1), 2016, 98–105. 5
- [Kut26] Konstantin Kutzkov. *Complementary Branching for #SAT: Bounded-Degree Bounds and Faster #3-SAT*. 35th International Joint Conference on Artificial Intelligence, 2026, 2282–2290. 4
- [Li21] Ray Li. *Settling SETH vs. Approximate Sparse Directed Unweighted Diameter (up to (NU)NSETH)*. 53rd ACM Symposium on Theory of Computing, 2021, 1684–1696. 4
- [Liu18] Sixue Liu. *Chain, Generalization of Covering Code, and Deterministic Algorithm for k -SAT*. 45th International Colloquium on Automata, Languages, and Programming, LIPIcs **107**, 2018, 88:1–88:13. 3
- [LMS18] Daniel Lokshantov, Dániel Marx, and Saket Saurabh. *Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal*. ACM Transactions on Algorithms **14**(2), Article 13, 2018, 1–30. Conference version in SODA 2011. 5
- [MR17] Pasin Manurangsi and Prasad Raghavendra. *A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs*. 44th International Colloquium on Automata, Languages, and Programming, LIPIcs **80**, 2017, 78:1–78:15. 4
- [Mar10] Dániel Marx. *Can You Beat Treewidth?* Theory of Computing **6**, 2010, 85–112. 5
- [NN93] Joseph Naor and Moni Naor. *Small-Bias Probability Spaces: Efficient Constructions and Applications*. SIAM Journal on Computing **22**(4), 1993, 838–856. 7, 40
- [Ope26] OpenAI, *The Unique Games Theorem*, OpenAI Math Release preprint [OAI:The-Unique-Games-Theorem-September-23-2026](#), 2026. 2
- [PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. *An Improved Exponential-Time Algorithm for k -SAT*. Journal of the ACM **52**(3), 2005, 337–364. Conference version in FOCS 1998. 3
- [Rag88] Prabhakar Raghavan. *Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs*. Journal of Computer and System Sciences **37**(2), 1988, 130–143. 7, 42, 59
- [Rag08] Prasad Raghavendra. *Optimal Algorithms and Inapproximability Results for Every CSP?* 40th ACM Symposium on Theory of Computing, 2008, 245–254. 2
- [RV13] Liam Roditty and Virginia Vassilevska Williams. *Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs*. 45th ACM Symposium on Theory of Computing, 2013, 515–524. 5
- [Sch24] Dominik Scheder. *PPSZ is better than you think*. TheoretCS **3**, Article 5, 2024, 1–37. 3
- [Sch99] Uwe Schöning. *A Probabilistic Algorithm for k -SAT and Constraint Satisfaction Problems*. 40th IEEE Symposium on Foundations of Computer Science, 1999, 410–414. 3

- [SS09] Alexander D. Scott and Gregory B. Sorkin. *Polynomial Constraint Satisfaction Problems, Graph Bisection, and the Ising Partition Function*. ACM Transactions on Algorithms **5**(4), 2009, 45:1–45:27. 5
- [SKC94] Bart Selman, Henry A. Kautz, and Bram Cohen. *Noise Strategies for Improving Local Search*. 12th National Conference on Artificial Intelligence, 1994, 337–343. 3
- [SS96] Michael Sipser and Daniel A. Spielman. *Expander Codes*. IEEE Transactions on Information Theory **42**(6), 1996, 1710–1722. 6, 40
- [Tan81] R. Michael Tanner. *A Recursive Approach to Low Complexity Codes*. IEEE Transactions on Information Theory **27**(5), 1981, 533–547. 6, 40
- [Tov84] Craig A. Tovey. *A Simplified NP-Complete Satisfiability Problem*. Discrete Applied Mathematics **8**(1), 1984, 85–89. 18
- [Val77] Leslie G. Valiant. *Graph-Theoretic Arguments in Low-Level Complexity*. Mathematical Foundations of Computer Science 1977, Lecture Notes in Computer Science **53**, 1977, 162–176. 6
- [Val79] Leslie G. Valiant. *The Complexity of Enumeration and Reliability Problems*. SIAM Journal on Computing **8**(3), 1979, 410–421. 4
- [VW19] Virginia Vassilevska Williams. *On Some Fine-Grained Questions in Algorithms and Complexity*. Proceedings of the International Congress of Mathematicians (ICM 2018), World Scientific, 2019, 3447–3487. 5
- [VW21] Nikhil Vyas and Ryan Williams. *On Super Strong ETH*. Journal of Artificial Intelligence Research **70**, 2021, 473–495. 4
- [Wah08] Magnus Wahlström. *A Tighter Bound for Counting Max-Weight Solutions to 2SAT Instances*. 3rd International Workshop on Parameterized and Exact Computation, Lecture Notes in Computer Science **5018**, 2008, 202–213. 4
- [Wil05] Ryan Williams. *A New Algorithm for Optimal 2-Constraint Satisfaction and Its Implications*. Theoretical Computer Science **348**(2–3), 2005, 357–365. 5
- [Wil13] Ryan Williams. *Improving Exhaustive Search Implies Superpolynomial Lower Bounds*. SIAM Journal on Computing **42**(3), 2013, 1218–1244. Conference version in STOC 2010. The numbering of Lemma 3.2 and Theorem 6.1 follows the [author’s full version](#). 51
- [Wil14] Richard Ryan Williams. *The Polynomial Method in Circuit Complexity Applied to Algorithm Design (Invited Talk)*. 34th International Conference on Foundation of Software Technology and Theoretical Computer Science, LIPIcs **29**, 2014, 47–60. 5
- [Wil16] Richard Ryan Williams. *Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation*. 31st Conference on Computational Complexity, LIPIcs **50**, 2016, 2:1–2:17. 4
- [WY07] David Woodruff and Sergey Yekhanin. *A Geometric Approach to Information-Theoretic Private Information Retrieval*. SIAM Journal on Computing **37**(4), 2007, 1046–1056. 6
- [Žák83] Stanislav Žák. *A Turing Machine Time Hierarchy*. Theoretical Computer Science **26**(3), 1983, 327–333. 51