Some exposition of

"The Exponential Time Hypothesis Is False"*

October 30, 2026

1 Goal and outline of the proof

We explain the sparse-permanent approach to the supplied manuscript [OAI26]. The main result is an exact deterministic algorithm for the permanent of an $n \times n$ rational matrix with $O(1)$ nonzero entries in each row and column, running in time

$$2^{O\left(\frac{n(\log\log n)^2}{\sqrt{\log n}}\right)}. \tag{1.1}$$

Classical linear-size reductions give the same bound for counting satisfying assignments of a 3-CNF with n variables and $O(n)$ clauses. The main text proves this sparse case. Appendix A gives polynomial space. Appendix B extends the argument to arbitrary 3-CNFs by a randomized reduction, with running time $2^{O(n(\log\log n)^3/\sqrt{\log n})}$.

The algorithm will recover $\mathrm{per}(A)$ by interpolating the univariate polynomial $P(t) = \mathrm{per}(A(t))$. The construction of $A(t)$ and the computation of its permanent are separate parts of the argument.

1. **Compute structured permanents.** Ryser's formula or subset dynamic programming computes an arbitrary permanent in $2^{O(n)}$ time. For a rank-r matrix, a dynamic program recording r type counts takes

$$\mathrm{poly}(n, r)\binom{n+r}{r} = n^{O(r)}$$

time. The two ideas combine: if B has diagonal blocks with at most s rows and columns each, and $\mathrm{rank}(E) \le r$, then $\mathrm{per}(B + E)$ costs $2^{O(s)}n^{O(r)}$, up to polynomial factors. The same bound holds for *every square submatrix*.

2. **Compute derivatives from submatrix permanents.** Suppose every square submatrix permanent of $A(t_0)$ costs at most T. For $1 \le h \le n/2$, we can compute

$$P(t_0), P'(t_0), \ldots, P^{(h-1)}(t_0) \quad \text{in time} \quad T\,\mathrm{poly}(n)(n/h)^{O(h)}.$$

The derivative matrices need not be structured. The permanent addition formula keeps the large permanents inside submatrices of $A(t_0)$.

3. **Construct a family with many easy points.** For sparse A, construct a matrix polynomial satisfying

$$A(0) = A, \qquad \deg A(t) \le d = O(\sqrt{\log n}). \tag{1.2}$$

At each of the $\ell = \Theta(\log n/\log\log n)$ points $i = 1, \ldots, \ell$, the permanent of *every square submatrix* of $A(i)$ should be computable in time $T_0 = 2^{O(n/\log^{10} n)}$. Here the degree of a matrix polynomial is the maximum degree of an entry. The special-point algorithm comes from a decomposition $A(i) = B_i + E_{\mathrm{out}}$, with small blocks in B_i and low rank in E_{out}.

*Produced by odonnell@cs.cmu.edu through interaction with LLMs. The idea of focusing the exposition on permanent came thanks to a Bluesky post of Nutan Limaye. Thanks to Bernardo Subercaseaux for telling me about his result on NTIME(T). Thanks to Ewin Tang for discussions about notation.

4. **Interpolate using derivatives.** Since $\deg P \le nd$, derivative orders $0, \dots, h-1$ at ℓ distinct points determine P whenever $\ell h > nd$. Taking

$$h = \left\lfloor \frac{nd}{\ell} \right\rfloor + 1 = O\left(\frac{n \log\log n}{\sqrt{\log n}} \right)$$

gives total time

$$2^{O(n/\log^{10} n + h \log(n/h))} = 2^{O\left(\frac{n(\log\log n)^2}{\sqrt{\log n}} \right)}.$$

The interpolation itself has only polynomial cost. Thus the saving comes from $d/\ell \ll 1$, which keeps the required derivative order well below n.

5. **Give rows and columns word labels.** Use labels in $[q]^\ell$, where $q = \lceil (\log n)^{10} \rceil$ and $Q = q^\ell$ is roughly $\sqrt{n}$, up to a factor of q. A balanced greedy labeling ensures that each ordered pair of labels contains at most one nonzero entry, and that the two labels of every nonzero disagree in every coordinate.

6. **Factor through copies of the Boolean cube.** The intermediate coordinates are ordered word pairs that disagree everywhere. Their number is

$$M = (q(q-1))^\ell \le Q^2 = O(n \log^{20} n).$$

Fixing the unordered symbol pair at each position leaves ℓ orientation bits. Hence the coordinates split into copies of $\{0,1\}^\ell$, and

$$A = L \mathcal{J} R', \qquad \mathcal{J} = I_C \otimes X^{\otimes \ell}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The set C indexes the copies. We first explain the complete factorization when $\ell = 1$, starting with $A = LR$, inserting I, and only then reversing the labels in R.

7. **Leave one coordinate alone.** Call K *i-nice* if it has the form $I_2 \otimes B$ after moving coordinate i to the first tensor factor. Replacing $X^{\otimes \ell}$ by such a K makes

$$L(I_C \otimes K)R'$$

block-diagonal after independently grouping rows and columns by their ith label symbol. There are q blocks, each with $O(n/q)$ rows and columns.

8. **Allow a low-rank correction.** It is enough to construct a local matrix polynomial $J(t)$ with

$$J(0) = X^{\otimes \ell}, \qquad J(i) = K_i + E, \qquad K_i \text{ } i\text{-nice}, \qquad \operatorname{rank}(E) \le 2^\ell / \ell^{100}.$$

Indeed, setting $A(t) = L(I_C \otimes J(t))R'$ gives an output correction of rank at most M/ℓ^{100}. The same E works at every special point, and the correction is computed exactly, not discarded.

9. **Obtain the polynomial from parity.** There is a rational polynomial p of degree $d = O(\sqrt{\ell \log \ell})$ such that $p(x_1 + \cdots + x_\ell)$ equals $x_1 \cdots x_\ell$ on all but an ℓ^{-100} fraction of $\{-1,1\}^\ell$. Let X_j flip bit j. The required family is

$$\boxed{J(t) = X^{\otimes \ell} + p\left(\sum_{j=1}^{\ell} (1 - t/j) X_j \right) - p\left(\sum_{j=1}^{\ell} X_j \right).} \tag{1.3}$$

At $t = i$, the middle term does not involve X_i, so it is i-nice. A Hadamard change of basis identifies the remaining error's rank with the number of inputs where p fails to equal parity.

2 Computing structured permanents

We begin with the algorithms needed at the interpolation points. They do not use sparsity.

Conventions. All logarithms are base two, except ln. In informal descriptions the exponent is $n/\sqrt{\log n}$ with powers of $\log \log n$ suppressed; displayed running-time bounds retain them. All matrices in the algorithm have rational entries. Arithmetic-operation bounds suppress polynomial factors; for polynomial-bit input, the numbers used below have polynomial bit length. We work with sufficiently large n; bounded input sizes can be handled directly.

2.1 Ordinary permanent and the addition formula

The permanent is the weighted sum of perfect matchings in a bipartite graph:

$$\operatorname{per}(A) = \sum_{\pi \in S_n} \prod_{u=1}^{n} A_{u,\pi(u)}.$$

The usual subset dynamic program processes the rows and records the set of columns already used. It takes $O(n2^n)$ arithmetic operations. Ryser's inclusion–exclusion formula gives the same bound [Rys63]:

$$\operatorname{per}(A) = \sum_{S \subseteq [n]} (-1)^{n-|S|} \prod_{u=1}^{n} \sum_{v \in S} A_{uv}. \tag{2.1}$$

For later use, expanding each selected entry of $B + C$ according to its source gives

$$\operatorname{per}(B + C) = \sum_{\substack{S,T \subseteq [n] \\ |S|=|T|}} \operatorname{per}(B[S^c, T^c]) \operatorname{per}(C[S, T]). \tag{2.2}$$

Here S, T are the rows and columns matched using C. The remaining endpoints are matched using B. The permanent of the empty matrix is 1.

2.2 Low rank: count colors, not subsets

Let $E = UV^{\mathsf{T}}$ have rank at most r, so

$$E_{ij} = \sum_{a=1}^{r} U_{ia} V_{ja}.$$

Think of each selected edge as choosing one of r colors. For a fixed color, its weight factors into a left-endpoint weight and a right-endpoint weight. This lets us record color counts instead of endpoint subsets. A rank factorization can be found by rational Gaussian elimination; the following is the dynamic-programming form of the low-rank permanent algorithm [Bar96].

Rank one. If $E_{ij} = u_i v_j$, every perfect matching has the same weight, so

$$\operatorname{per}(E) = n! \prod_i u_i \prod_j v_j.$$

Rank two: red and blue. Write $E_{ij} = u_i v_j + w_i z_j$. To see the counting interpretation, first suppose the four lists consist of nonnegative integers. Between left vertex i and right vertex j, place $u_i v_j$ red edges and $w_i z_j$ blue edges. A red edge is specified by a label from $[u_i]$ at its left endpoint and a label from $[v_j]$ at its right endpoint; blue edges are labeled analogously.

Let $D_L(j,k)$ count the ways to color and label the first j left vertices, exactly k of them red. Then

$$D_L(0,0) = 1, \qquad D_L(j,k) = u_j D_L(j-1,k-1) + w_j D_L(j-1,k), \tag{2.3}$$

with invalid states zero. Define $D_R(j,k)$ by the same recurrence with v_j, z_j. Once the endpoints have been colored and labeled, k red vertices on each side can be paired in $k!$ ways, and the blue vertices in $(n-k)!$ ways. Therefore

$$\operatorname{per}(E) = \sum_{k=0}^{n} k!(n-k)!\, D_L(n,k) D_R(n,k). \tag{2.4}$$

The two tables have $O(n^2)$ states, so this takes $O(n^2)$ arithmetic operations. The identities remain valid for arbitrary rational factors: the same computation is then a weighted sum, possibly with negative weights. Positivity of E itself is not an assumption about the signs of a chosen rank factorization.

Rank r. For a vector $\nu = (\nu_1, \dots, \nu_r)$ of nonnegative color counts, let $D_L(j,\nu)$ be the total weight of colorings of the first j left vertices with counts ν. The recurrence is

$$D_L(j,\nu) = \sum_{a=1}^{r} U_{ja} D_L(j-1, \nu - e_a), \qquad D_L(0,0) = 1,$$

and similarly on the right. Put $\nu! = \prod_a \nu_a!$. Matching separately within each color gives

$$\operatorname{per}(E) = \sum_{|\nu|=n} \nu!\, D_L(n,\nu) D_R(n,\nu). \tag{2.5}$$

The number of states over all layers is

$$\sum_{j=0}^{n} \binom{j+r-1}{r-1} = \binom{n+r}{r}.$$

Thus, given the factors, the algorithm takes $O(r\binom{n+r}{r})$ arithmetic operations. For fixed r this is $O(rn^r)$; for $r \le n$ it is $n^{O(r)}$. Even $r = n$ gives $2^{O(n)}$ time, although not the 2^n base of Ryser's formula.

2.3 Small blocks plus low rank

Say that B has *corresponding diagonal blocks of size at most s* if its rows are partitioned into $I_1, \dots, I_b$ and its columns into $J_1, \dots, J_b$, with

$$B[I_a, J_c] = 0 \quad (a \neq c), \qquad |I_a|, |J_a| \le s.$$

The corresponding row and column groups may have different sizes. If B alone has a nonsquare block, then $\operatorname{per}(B) = 0$; otherwise its permanent is the product of its block permanents.

Lemma 2.1 (Structured permanents)**.** Let $A = B + E$, where B has corresponding diagonal blocks of size at most s and $\operatorname{rank}(E) \le r$. The permanent of A, and that of every square submatrix of A, can be computed in

$$\operatorname{poly}(n)\, 2^{O(s)} n^{O(r)}$$

arithmetic operations.

Proof. Factor $E = UV^{\mathsf{T}}$ and use the r colors above. In each block $I_a \times J_a$, choose subsets $S \subseteq I_a, T \subseteq J_a$ with $|S| = |T|$. Match these endpoints using B, contributing $\operatorname{per}(B[S,T])$. All remaining endpoints will be matched using E; assign them colors and their corresponding endpoint weights.

Let $W_a(\nu, \xi)$ be the total weight of these choices with left color counts ν and right color counts ξ among the remaining endpoints. There are at most 2^{2s} choices of S, T. For each choice, use subset DP for $\mathrm{per}(B[S,T])$ and the color-count DP for the remaining endpoints. This computes W_a in $2^{O(s)}n^{O(r)}$ operations.

Combine the blocks by adding their count vectors, that is, by ordinary table convolution. Let $W(\nu, \xi)$ be the resulting table. Every table has at most $(n+1)^{2r}$ entries, so even naive convolutions take $n^{O(r)}$ operations per block. Finally,

$$\mathrm{per}(B+E) = \sum_{\nu} \nu!\, W(\nu, \nu). \tag{2.6}$$

The factorials pair the still-unmatched endpoints within each color. Each term in the expansion of $\mathrm{per}(B+E)$ is counted exactly once: its B-edges determine the local matchings, and its E-edges determine the endpoint colors and final pairings.

Deleting rows and columns preserves the block partition and restricts the factors U, V without increasing rank. Therefore the same algorithm works for every square submatrix. The rank-zero case is the ordinary block computation. □

The two separately easy pieces B and E do not by themselves make their sum easy. The useful fact is that the color-count tables can be combined *one block at a time*.

3 Derivatives and interpolation

Let $A(t)$ be an $n \times n$ polynomial matrix and $P(t) = \mathrm{per}(A(t))$. The next lemma explains why efficient submatrix permanents are the right hypothesis.

Lemma 3.1 (Derivatives from submatrix permanents)**.** Fix t_0 and suppose every square submatrix permanent of $A(t_0)$ is computable in time T. For $1 \le h \le n/2$, all values

$$P(t_0), P'(t_0), \ldots, P^{(h-1)}(t_0)$$

can be computed together in time $T\,\mathrm{poly}(n)(n/h)^{O(h)}$, provided the polynomial entries have polynomial-size descriptions and degree.

Proof. Write $A_0 = A(t_0)$. The first derivative already displays the mechanism:

$$P'(t_0) = \sum_{u,v} A'_{uv}(t_0)\,\mathrm{per}(A_0 \text{ with row } u \text{ and column } v \text{ deleted}).$$

Thus it is the submatrix permanents of A_0, not the permanent of $A'(t_0)$, that matter.

For higher derivatives, put $\Delta(z) = A(t_0+z) - A_0$. Every entry of $\Delta(z)$ is divisible by z. The permanent addition formula gives

$$P(t_0+z) = \sum_{a=0}^{n} \sum_{\substack{S,T\subseteq[n]\\|S|=|T|=a}} \mathrm{per}(A_0[S^c,T^c])\,\mathrm{per}(\Delta(z)[S,T]).$$

Since the last permanent is divisible by z^a, an order-j derivative receives contributions only from $a \le j$. Explicitly,

$$P^{(j)}(t_0) = j! \sum_{a=0}^{j} \sum_{\substack{S,T\subseteq[n]\\|S|=|T|=a}} \mathrm{per}(A_0[S^c,T^c])\,[z^j]\,\mathrm{per}(\Delta(z)[S,T]). \tag{3.1}$$

For each $a < h$ and each S, T, compute the large structured permanent in time T. Compute the small $a \times a$ permanent by subset DP or Ryser over $\mathbb{Q}[z]/(z^h)$, costing $2^{O(a)}\,\mathrm{poly}(n)$. Accumulate its contributions to all $j < h$. The total cost is at most

$$T\,\mathrm{poly}(n) \sum_{a<h} 2^{O(a)} \binom{n}{a}^2 \le T\,\mathrm{poly}(n)(n/h)^{O(h)}.$$

□

3.1 Recovering the value at zero

If every entry of $A(t)$ has degree at most d, then $\deg P \le nd$. A polynomial of degree less than ℓh is determined by its derivatives of orders $0, \dots, h-1$ at ℓ distinct points: the difference of two candidates would be divisible by $\prod_{i=1}^{\ell}(t-i)^h$, and hence would be zero. Solving the resulting nonsingular linear system is ordinary Hermite interpolation.

Proposition 3.2 (Recovery from structured values)**.** Suppose $A(0) = A$ and $\deg A(t) \le d$. For every $i \in [\ell]$, suppose $A(i)$ is a sum of a block matrix of block size at most s and a matrix of rank at most r. Let $h = \lfloor nd/\ell \rfloor + 1 \le n/2$. Then $\mathrm{per}(A)$ can be computed in

$$\mathrm{poly}(n)\, 2^{O(s)} n^{O(r)} (n/h)^{O(h)} = \mathrm{poly}(n)\, 2^{O(s + r\log n + h\log(n/h))} \tag{3.2}$$

time, assuming ℓ, d and the exact input descriptions are polynomially bounded.

Proof. Apply Lemma 2.1 to all square submatrices at a special point, and Lemma 3.1 to compute the first h derivative values there. Repeat for all ℓ points. Since $\ell h > nd$, interpolate P and evaluate $P(0)$. Polynomial factors cover the number of points and the final linear system. □

More generally, any known upper bound D on $\deg P$ may replace nd: take $h = \lfloor D/\ell \rfloor + 1$ in the same proof. Appendix B will use a degree bound substantially smaller than the matrix dimension times the entry degree.

We have now separated the algorithmic task from the algebraic one: construct $A(t)$ with many structured values and with d/ℓ small.

4 A factorization that turns coordinate identities into blocks

The input is an $n \times n$ matrix with at most a fixed constant Δ nonzeros in each row and column. We first work with ordinary colors. This is the entire structural idea in its simplest form.

4.1 One color per row and column

For this subsection take $q = \Theta(\sqrt{n})$, with a sufficiently large constant depending on Δ. An elementary greedy procedure colors the rows and columns from $[q]$ so that each color occurs at most $O(n/q)$ times on either side, every nonzero has distinct endpoint colors, and different nonzeros have different ordered pairs of endpoint colors. The proof is the $\ell = 1$ case of Lemma 4.1 below.

Order the rows and columns by color. Then

$$A = \begin{pmatrix} 0 & A_{12} & A_{13} & \cdots & A_{1q} \\ A_{21} & 0 & A_{23} & \cdots & A_{2q} \\ A_{31} & A_{32} & 0 & \cdots & A_{3q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{q1} & A_{q2} & A_{q3} & \cdots & 0 \end{pmatrix}. \tag{4.1}$$

Each block has $O(\sqrt{n})$ rows and columns and contains at most one nonzero entry. The entries need not be 0 or 1; only their locations matter for the coloring.

First, $A = LR$. Make one intermediate coordinate e for each nonzero A_{uv}. Put A_{uv} in position (u, e) of L, and put 1 in position (e, v) of R, leaving the rest of that column and row zero. Multiplying through e reproduces precisely this entry, so $LR = A$.

Now name each coordinate by its ordered pair of colors (a, b). The names are distinct because each color block contains at most one nonzero. Include a zero column of L and a zero row of R for each empty block. The intermediate dimension becomes

$$M = q(q-1), \qquad L \in \mathbb{Q}^{n \times M}, \qquad R \in \mathbb{Q}^{M \times n}.$$

For an entry $A_{uv} = w$ in block (a, b), our definitions say

$$L[u, (a, b)] = w, \qquad R[(a, b), v] = 1. \tag{4.2}$$

In particular, column (a, b) of L uses rows of color a, while row (a, b) of R uses columns of color b.

Next, $A = LIR$. Insert the $M \times M$ identity and order the intermediate coordinates in pairs:

$$(1,2),(2,1); \quad (1,3),(3,1); \quad \ldots; \quad (q-1,q),(q,q-1).$$

Thus I consists of 2×2 identity blocks, one for each unordered color pair. Nothing has changed yet.

Then, $A = L\mathcal{J}R'$. Let

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \mathcal{J} = \operatorname{diag}(X, \ldots, X).$$

Reverse every consecutive pair of rows of R, obtaining $R' = \mathcal{J}R$. Since $\mathcal{J}^2 = I$, we have

$$\boxed{A = LR = LIR = L\mathcal{J}R'.} \tag{4.3}$$

The entry in (4.2) now appears as $R'[(b, a), v] = 1$. The middle matrix reverses the label back, so the product still reconstructs the same entry of A.

Finally, replace $\mathcal{J}$ by I. Keep L and R' fixed and define the new matrix $B = LIR'$. Its support is best read from the following table:

Intermediate coordinate	Column of L: row color	Row of R: column color	Row of R': column color
(a, b)	a	b	a
(b, a)	b	a	b

After the reversal, L and R' both use the *first* color in a coordinate's label. Therefore

$$B_{uv} = \sum_{a \neq b} L[u, (a, b)] R'[(a, b), v]$$

can be nonzero only when row u and column v have the same color. Hence

$$\boxed{B = LIR' = \begin{pmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_q \end{pmatrix}.} \tag{4.4}$$

These are new blocks, not the zero diagonal blocks of the original A. Each has $O(\sqrt{n})$ rows and columns, so $\mathrm{per}(B)$ is computable in $2^{O(\sqrt{n})}$ time. If any corresponding row and column counts differ, its permanent is zero.

One pair of colors, in ordinary matrix notation. Write $A_{ab} = u_{ab}v_{ab}^{\mathsf{T}}$ and $A_{ba} = u_{ba}v_{ba}^{\mathsf{T}}$, using the weighted row indicator and column indicator of the unique entry, or zero vectors for an empty block. The pieces for $\{a, b\}$ are

$$L_{\{a,b\}} = \begin{pmatrix} u_{ab} & 0 \\ 0 & u_{ba} \end{pmatrix}, \quad R_{\{a,b\}} = \begin{pmatrix} 0 & v_{ab}^{\mathsf{T}} \\ v_{ba}^{\mathsf{T}} & 0 \end{pmatrix}, \quad R'_{\{a,b\}} = \begin{pmatrix} v_{ba}^{\mathsf{T}} & 0 \\ 0 & v_{ab}^{\mathsf{T}} \end{pmatrix}.$$

Consequently,

$$L_{\{a,b\}}XR'_{\{a,b\}} = \begin{pmatrix} 0 & u_{ab}v_{ab}^{\mathsf{T}} \\ u_{ba}v_{ba}^{\mathsf{T}} & 0 \end{pmatrix},$$

$$L_{\{a,b\}}I_2R'_{\{a,b\}} = \begin{pmatrix} u_{ab}v_{ba}^{\mathsf{T}} & 0 \\ 0 & u_{ba}v_{ab}^{\mathsf{T}} \end{pmatrix}.$$

Summing these contributions over all unordered pairs gives (4.3) and (4.4). The identity pairs opposite blocks' row and column data, so a contribution may disappear when the reverse block is empty.

One easy matrix is not enough. The permanent of B need not resemble that of A. A linear interpolation from one matrix to the other has a permanent polynomial of degree up to n, ordinarily requiring $n + 1$ values. We have woefully too few easy points. We will obtain more by replacing each color with a word. The resulting easy blocks will be larger, but small enough for the algorithm.

4.2 Balanced labels by words

Now choose a smaller alphabet $[q]$ and assign row and column labels

$$\alpha(u), \beta(v) \in [q]^{\ell}, \qquad Q = q^{\ell}.$$

We require three properties:

(i) Each word labels at most $D = \lceil 2n/Q \rceil$ rows and at most D columns.

(ii) Different nonzeros have different ordered label pairs $(\alpha(u), \beta(v))$.

(iii) If $A_{uv} \neq 0$, then $\alpha(u)_i \neq \beta(v)_i$ for every $i \in [\ell]$.

Thus A has a $Q \times Q$ grid of blocks with at most one nonzero per block. Moreover, a block is zero whenever its two word labels agree in even one coordinate.

Lemma 4.1 (Balanced greedy labeling)**.** For fixed Δ and sufficiently large n, the labels can be found deterministically in polynomial time provided

$$4\Delta\sqrt{n} \leq Q = q^{\ell} \leq n, \qquad q \geq 4\Delta\ell.$$

Proof. A word becomes unavailable on a side once it has been used D times; before the last vertex is labeled, fewer than $Q/2$ words are full. Label the rows first, requiring any two rows sharing a nonzero column to have distinct labels. A row conflicts with at most $\Delta(\Delta - 1)$ earlier rows, leaving an available label.

Next label the columns. A column meets at most Δ rows, whose labels are distinct. For each neighboring row label, at most ΔD column labels have already been paired with it. Avoiding repeated ordered pairs therefore forbids at most $\Delta^2 D$ labels. Coordinatewise disagreement with the neighboring row labels forbids at most $\Delta\ell Q/q \leq Q/4$ further labels. Our bounds imply $\Delta^2 D \leq Q/4$ for sufficiently large n. Together with fewer than $Q/2$ full labels, the exclusions leave a valid choice. Choose the first available word. □

For each position i and symbol a, define

$$I_{i,a} = \{u : \alpha(u)_i = a\}, \qquad J_{i,a} = \{v : \beta(v)_i = a\}.$$

There are Q/q words with a prescribed symbol at position i, so

$$|I_{i,a}|, |J_{i,a}| \le \frac{Q}{q}\left\lceil \frac{2n}{Q} \right\rceil \le \frac{2n+Q}{q} \le \frac{3n}{q}. \tag{4.5}$$

This coordinatewise balance, rather than the size of an individual word class, will control the easy blocks.

4.3 The intermediate coordinates are copies of a Boolean cube

Let

$$\Omega = \{(a,b) \in [q]^\ell \times [q]^\ell : a_i \ne b_i \text{ for every } i\}, \qquad M = |\Omega| = (q(q-1))^\ell.$$

As in the one-color construction, use one intermediate coordinate for every possible label pair, including unused pairs, and define

$$L[u, (\alpha(u), \beta(v))] = A_{uv}, \qquad R[(\alpha(u), \beta(v)), v] = 1 \quad (A_{uv} \ne 0).$$

All other entries are zero. Uniqueness of the ordered label pairs gives $A = LR$.

An element $(a,b) \in \Omega$ specifies ℓ unordered pairs

$$\{a_1, b_1\}, \dots, \{a_\ell, b_\ell\}$$

and an orientation bit at each position. Give each unordered pair its increasing order; bit 0 uses that order and bit 1 reverses it. Hence

$$\Omega \cong C \times \{0,1\}^\ell, \qquad C = \binom{[q]}{2}^\ell, \qquad |C| = \binom{q}{2}^\ell. \tag{4.6}$$

Fixing the unordered pairs gives one Boolean cube. Reversing the two words complements every orientation bit and never changes the cube.

Let

$$\mathcal{J} = I_C \otimes X^{\otimes \ell}, \qquad R' = \mathcal{J}R.$$

The notation $I_C \otimes K$ simply means a block-diagonal matrix with one identical copy of K for each cube. Since $\mathcal{J}^2 = I$, the same sequence of equalities gives

$$\boxed{A = LR = LIR = L\mathcal{J}R'.} \tag{4.7}$$

In particular, L stores an entry at its ordered word pair, while R' stores the column indicator at the reversed pair. All input dependence lies in L, R'; the middle blocks are identical.

4.4 The useful notion of i-niceness

Definition 4.2. A matrix K indexed by $\{0,1\}^\ell$ is *i-nice* if, after moving tensor factor i to the first position, it has the form $I_2 \otimes B$ for some matrix B on the other $\ell - 1$ coordinates. Equivalently,

$$K[x,y] = \mathbf{1}\{x_i = y_i\}\, B[x_{-i}, y_{-i}]. \tag{4.8}$$

Thus K preserves bit i and its action on the other bits is independent of that bit's value. Grouping indices by bit i gives two *identical* diagonal blocks:

$$K = \begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}.$$

For $\ell = 1$, these are exactly the scalar multiples of I_2. In particular, I is i-nice for every i, while $X^{\otimes \ell}$ is i-nice for none.

Lemma 4.3 (Niceness gives block structure)**.** For every i-nice K, the matrix $L(I_C \otimes K)R'$ has corresponding diagonal blocks indexed by $a \in [q]$, with row groups $I_{i,a}$ and column groups $J_{i,a}$. Each block has at most $3n/q$ rows and columns.

Proof. A potentially nonzero contribution to entry (u, v) uses a nonzero entry $A_{uv'}$ from L and a nonzero entry $A_{u'v}$ from R'. Their intermediate labels are

$$(\alpha(u), \beta(v')) \qquad \text{and} \qquad (\beta(v), \alpha(u')).$$

The middle matrix stays within a cube, so at position i the unordered pairs agree. It preserves orientation bit i, so the *ordered* pairs agree. Comparing their first symbols gives

$$\alpha(u)_i = \beta(v)_i.$$

Thus an entry vanishes unless its row and column have the same ith symbol. The size bound is (4.5). □

Only preservation of bit i was needed in this proof. The stronger identity-on-a-factor definition is convenient because it is exactly what the polynomial construction produces.

Low rank survives the factorization. If $K = K_i + E$, where K_i is i-nice and $\operatorname{rank}(E) \le \varepsilon 2^\ell$, then

$$L(I_C \otimes K)R' = B_i + E_{\text{out}}, \qquad E_{\text{out}} = L(I_C \otimes E)R', \qquad \operatorname{rank}(E_{\text{out}}) \le \varepsilon M. \tag{4.9}$$

The rank bound follows by adding the ranks of the separate error blocks and then multiplying by L, R'. The useful structure belongs to B_i; only the correction has low rank.

5 The matrix polynomial

We now work entirely on a single copy of $\{0,1\}^\ell$. The input matrix A plays no role in this section.

5.1 Why an exact low-degree deformation is impossible

An exact degree-ℓ construction is easy:

$$\bigotimes_{j=1}^{\ell} \big((1 - t/j)X + (t/j)I_2\big).$$

At zero it is $X^{\otimes \ell}$, and at $t = i$ its ith factor is I_2. But degree ℓ would require about n derivatives per interpolation point, giving no useful saving.

In fact, exact niceness forces degree at least ℓ. For any fixed x, the entry indexed by $(x, \bar{x})$ must be 1 at zero and 0 at every $i = 1, \dots, \ell$. It therefore has at least ℓ roots. The low-rank allowance is what lets us lower the degree.

Theorem 5.1 (Low-degree deformation)**.** For sufficiently large ℓ, there is an explicitly constructible matrix polynomial $J(t) \in \mathbb{Q}[t]^{2^\ell \times 2^\ell}$ of degree

$$d = O(\sqrt{\ell \log \ell})$$

with

$$J(0) = X^{\otimes \ell}, \qquad J(i) = K_i + E \quad (i = 1, \dots, \ell),$$

where each K_i is i-nice and the *same* matrix E satisfies

$$\operatorname{rank}(E) \le \frac{2^\ell}{\ell^{100}}.$$

The construction takes polynomial time in 2^ℓ and uses polynomial-bit rational coefficients. The exponent 100 may be replaced by any fixed constant, changing only the constant in the degree bound.

5.2 Three elementary matrices

Let

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Direct multiplication gives $H^2 = I_2$ and $HZH = X$. For $j \in [\ell]$, put

$$X_j = I_2^{\otimes(j-1)} \otimes X \otimes I_2^{\otimes(\ell-j)}, \qquad Z_j = I_2^{\otimes(j-1)} \otimes Z \otimes I_2^{\otimes(\ell-j)}.$$

The X_j commute; X_j flips bit j and leaves the others fixed. With $H_\ell = H^{\otimes\ell}$,

$$H_\ell Z_j H_\ell = X_j, \qquad H_\ell Z^{\otimes\ell} H_\ell = X^{\otimes\ell}. \tag{5.1}$$

Consequently, for any polynomial p and scalars c_j,

$$H_\ell \, p\left(\sum_j c_j Z_j\right) H_\ell = p\left(\sum_j c_j X_j\right). \tag{5.2}$$

This is just conjugation preserving addition and multiplication.

Functions as diagonal matrices. For $g : \{-1,1\}^\ell \to \mathbb{R}$, define D_g by

$$D_g[b,b] = g((-1)^{b_1}, \ldots, (-1)^{b_\ell}), \qquad b \in \{0,1\}^\ell.$$

The joint diagonal entries of $Z_1, \ldots, Z_\ell$ run through $\{-1,1\}^\ell$ exactly once. Hence $D_g = g(Z_1, \ldots, Z_\ell)$, interpreting g through its multilinear polynomial, and $H_\ell D_g H_\ell = g(X_1, \ldots, X_\ell)$.

Two facts will be used. First, if g is independent of coordinate i, then, after placing that coordinate first, $D_g = I_2 \otimes D_{g'}$. It follows that

$$H_\ell D_g H_\ell = I_2 \otimes (H_{\ell-1} D_{g'} H_{\ell-1}), \tag{5.3}$$

which is i-nice. Second, since H_ℓ is invertible,

$$\operatorname{rank}(H_\ell (D_g - D_h) H_\ell) = |\{x \in \{-1,1\}^\ell : g(x) \neq h(x)\}|. \tag{5.4}$$

Thus agreement on most inputs becomes low-rank difference after a change of basis.

5.3 A polynomial agreeing with parity on most inputs

Let $\chi(x) = x_1 \cdots x_\ell$. If x has exactly k negative coordinates, then

$$x_1 + \cdots + x_\ell = \ell - 2k, \qquad \chi(x) = (-1)^k.$$

Set $d = \lceil 100\sqrt{\ell \log \ell} \rceil$. For sufficiently large ℓ, interpolate

$$p(\ell - 2k) = (-1)^k \tag{5.5}$$

for $d+1$ consecutive integer values of k centered at $\ell/2$. This gives $p \in \mathbb{Q}[z]$ of degree at most d. The chosen interval contains every integer k with $|k - \ell/2| \le d/4$. For uniform x, its number of negative coordinates is distributed as $\mathrm{Bin}(\ell, 1/2)$, so the elementary binomial tail bound gives

$$\mathbb{P}_x\left[p\left(\sum_j x_j\right) \neq \chi(x)\right] \le 2\exp\left(-\frac{d^2}{8\ell}\right) \le \ell^{-100}. \tag{5.6}$$

This is *exact agreement* away from a small exceptional set, not a uniform approximation in value. On the exceptional inputs, p may be large. Lagrange interpolation at the integer points in (5.5) yields rational coefficients with polynomial bit length.

5.4 Make one coefficient disappear at each special point

Define

$$\widetilde{J}(t) = p\left(\sum_{j=1}^{\ell}(1 - t/j)X_j\right), \qquad E = X^{\otimes\ell} - \widetilde{J}(0), \qquad J(t) = \widetilde{J}(t) + E. \tag{5.7}$$

This is the formula (1.3). It immediately gives $\deg J(t) \le d$ and $J(0) = X^{\otimes\ell}$.

At $t = i$, the coefficient of X_i vanishes. Every remaining X_j acts as the identity on tensor factor i, as does every polynomial in them. Therefore

$$K_i = \widetilde{J}(i) = p\left(\sum_{j\neq i}(1 - i/j)X_j\right)$$

is i-nice. More explicitly, if the ith tensor factor is placed first, the matrix inside p is $I_2 \otimes S_i$, so $K_i = I_2 \otimes p(S_i)$.

It remains to bound the rank of E. Conjugating by H_ℓ gives

$$H_\ell E H_\ell = Z^{\otimes\ell} - p\left(\sum_j Z_j\right). \tag{5.8}$$

This matrix is diagonal, with one entry

$$x_1 \cdots x_\ell - p(x_1 + \cdots + x_\ell)$$

for each $x \in \{-1, 1\}^\ell$. By (5.6), at most $2^\ell/\ell^{100}$ of these entries are nonzero. This proves Theorem 5.1.

The polynomial matrices can be formed directly from (5.7) by rational matrix arithmetic. The factors $1/\sqrt{2}$ appear only in the proof of the rank bound, not in the algorithm.

6 Parameters and consequences

We now combine the construction with the permanent algorithms. Fix the input maximum degree $\Delta = O(1)$ and choose

$$q = \lceil(\log n)^{10}\rceil, \qquad \ell = \left\lceil \log_q(16\Delta\sqrt{n})\right\rceil, \qquad d = \lceil 100\sqrt{\ell \log \ell}\rceil. \tag{6.1}$$

Then $16\Delta\sqrt{n} \le Q = q^\ell < 16\Delta q\sqrt{n}$, so the coloring lemma applies for sufficiently large n. In particular,

$$\ell = \Theta\left(\frac{\log n}{\log\log n}\right), \quad M \le Q^2 = O(n\log^{20} n), \quad d = \Theta(\sqrt{\log n}).$$

Construct L, R' and the local polynomial $J(t)$, and put

$$A(t) = L(I_C \otimes J(t))R'. \tag{6.2}$$

We have $A(0) = A$ and $\deg A(t) \le d$. At every $i \in [\ell]$, Lemma 4.3 and (4.9) give

$$A(i) = B_i + E_{\text{out}}, \qquad s \le \frac{3n}{q}, \qquad r := \operatorname{rank}(E_{\text{out}}) \le \frac{M}{\ell^{100}}.$$

The same output correction works at all points. The quantitative bounds are

$$s = O(n/\log^{10} n), \qquad r\log n = O\left(\frac{n(\log\log n)^{100}}{(\log n)^{79}}\right) = o(n/\log^{10} n).$$

Hence every special-point square submatrix permanent costs $2^{O(n/\log^{10} n)}$. For interpolation take

$$h = \left\lfloor \frac{nd}{\ell} \right\rfloor + 1 = \Theta\left(\frac{n \log\log n}{\sqrt{\log n}} \right).$$

The derivative cost dominates, since

$$h \log(n/h) = O\left(\frac{n(\log\log n)^2}{\sqrt{\log n}} \right).$$

Proposition 3.2 now proves the main result.

Theorem 6.1 (Sparse permanent)**.** For every fixed Δ, the permanent of an $n \times n$ rational matrix with at most Δ nonzero entries per row and column, and polynomial-bit entries, can be computed exactly and deterministically in time (1.1).

Exact arithmetic. The auxiliary dimensions and polynomial degrees are polynomial in n. The interpolant p, its substitutions, and the factors L, R' have polynomial-bit rational descriptions. Rank factorizations and Hermite interpolation use rational linear systems of polynomial dimension, whose determinant bounds give polynomial-bit solutions. The permanent DPs sum products of polynomially many such numbers; factorial weights add only polynomially many bits. Common denominators can be chosen with polynomial bit length throughout. Thus exact bit arithmetic contributes only polynomial overhead.

From sparse 3-CNF to sparse permanent. The weighted permanent reductions of Valiant and Ben-Dor–Halevi [Val79, BH93] send a 3-CNF with n variables and $O(n)$ clauses to a matrix of dimension $O(n)$ with $O(n)$ nonzeros and bounded integer weights. Its permanent is a known nonzero multiple of the satisfying-assignment count.

To bound individual degrees, view the support as a bipartite graph. Replace a degree-k vertex by a path of $2k - 1$ vertices, placing its original incident edges at the odd positions and giving each path edge weight 1. In any perfect matching exactly one external edge is used: two used odd positions would leave an odd internal interval, and no used odd position would leave an odd path. For each choice there is a unique internal completion. Replacing vertices on both sides therefore preserves the weighted matching sum and gives maximum degree three with linear size. Theorem 6.1 consequently counts sparse 3-CNFs within (1.1).

Standard counting sparsification already extends this to arbitrary 3-CNFs with a smaller logarithmic saving. Appendix B gives a randomized extension with the same $\sqrt{\log n}$ saving and one additional factor of $\log\log n$ in the exponent.

A consequence for nondeterministic time. The schedule-conditioned encoding reported in [Sub26] reduces accepting computations of a fixed-tape nondeterministic multitape machine running in time T to at most $2^{O(T \log\log T/\log T)}$ formulas, each with $O(T \log\log T)$ variables and clauses. Applying the sparse-formula algorithm gives, for the usual time-constructible bounds,

$$\mathrm{NTIME}(T) \subseteq \mathrm{DTIME}\left(2^{O(T(\log\log T)^3/\sqrt{\log T})} \right).$$

The nontrivial ingredient in this consequence is that schedule-conditioned encoding; it is separate from the permanent algorithm proved here.

A Polynomial space

The color-count tables in Lemma 2.1 may require superpolynomial space. We can evaluate the same counting identity by numerical interpolation, one evaluation at a time. This preserves the time bound.

A.1 The counting identity as a polynomial pairing

Write $A = B + UV^{\mathsf{T}}$ with r columns in U, V. Introduce tuples $z, w \in \mathbb{Q}^r$ and linear forms

$$u_i(z) = \sum_{a=1}^{r} U_{ia} z_a, \qquad v_j(w) = \sum_{a=1}^{r} V_{ja} w_a.$$

For each corresponding block $I_b \times J_b$, form

$$F_b(z, w) = \sum_{\substack{S \subseteq I_b,\ T \subseteq J_b \\ |S| = |T|}} \operatorname{per}(B[S, T]) \prod_{i \in I_b \setminus S} u_i(z) \prod_{j \in J_b \setminus T} v_j(w). \tag{A.1}$$

The coefficient of $z^\nu w^\xi$ in $F = \prod_b F_b$ is exactly the table entry $W(\nu, \xi)$ from (2.6). Define a linear functional by

$$\mathcal{L}_r(z^\nu w^\xi) = \begin{cases} \nu!, & \nu = \xi, \\ 0, & \nu \neq \xi. \end{cases} \tag{A.2}$$

Then the structured-permanent identity is simply

$$\operatorname{per}(B + UV^{\mathsf{T}}) = \mathcal{L}_r(F). \tag{A.3}$$

At fixed numerical z, w, evaluate F_b by enumerating S, T in (A.1), using Ryser for the small permanent. There are at most 2^{2s} pairs, so one evaluation of F costs $2^{O(s)}\operatorname{poly}(n)$ time and polynomial space. It remains to apply $\mathcal{L}_r$ without storing the coefficients of F.

A.2 Interpolate the functional, not the coefficient table

Every individual variable of F has degree at most n. Let

$$\Lambda_a(t) = \prod_{\substack{0 \le c \le n \\ c \neq a}} \frac{t - c}{a - c} \qquad (a = 0, \dots, n)$$

be the univariate Lagrange basis, and precompute

$$\kappa_{ab} = \sum_{k=0}^{n} k!\, [t^k]\Lambda_a(t)\, [t^k]\Lambda_b(t). \tag{A.4}$$

Multivariate interpolation, followed by applying (A.2) separately in each coordinate, gives

$$\mathcal{L}_r(F) = \sum_{a,b \in \{0,\dots,n\}^r} \left(\prod_{c=1}^{r} \kappa_{a_c b_c} \right) F(a, b). \tag{A.5}$$

To verify this, expand F in the tensor-product Lagrange basis. The functional on $\Lambda_{a_c}(z_c)\Lambda_{b_c}(w_c)$ is $\kappa_{a_c b_c}$, and the factorial pairing factors across coordinates.

There are $(n+1)^{2r}$ grid points. For each point, compute its weight and $F(a, b)$, add the product to an accumulator, and discard the evaluation. The running time is $2^{O(s)} n^{O(r)} \operatorname{poly}(n)$ and the space is polynomial. The weights, evaluations, and accumulated sums all have polynomial bit length by the same degree and denominator bounds used in the main algorithm.

A.3 Space for the complete permanent algorithm

Use (A.5) for each large submatrix permanent in (3.1). Enumerate the special points and the row and column subsets sequentially. The small perturbation permanent is computed by Ryser over $\mathbb{Q}[z]/(z^h)$, storing only polynomially many coefficients. Retain the ℓh derivative values for the final interpolation, whose linear system also has polynomial size. Thus Theorem 6.1 holds in polynomial space.

B Arbitrary #3SAT: keep the number of factors small

The permanent algorithm in the main text is sensitive to the degree of the polynomial it interpolates. This appendix exploits that fact: a large permanent matrix can still have a low-degree permanent polynomial.

There are two reductions. First, replace a conjunction of many clauses by a short product of sparse cubics. Second, encode the Boolean sum of that product as a permanent, with one choice of a monomial per factor:

$$\#F \quad \longrightarrow \quad Z = \sum_{x\in\{0,1\}^n} \prod_{j=1}^{R} f_j(x) \quad \longrightarrow \quad 2^{-a} \operatorname{per} A.$$

The first arrow is randomized; the second is an exact identity over $\mathbb{Q}$. The important parameter is $R = O(n \log\log n)$. The matrix may have n polylog(n) rows, but its permanent will have degree only $4R$ in the entries we deform.

Theorem B.1 (Arbitrary #3SAT)**.** There is a randomized algorithm for 3-CNFs on n declared variables that always returns an upper bound on the satisfying-assignment count and is exact with probability at least $3/4$. Its running time is

$$2^{O\left(n(\log\log n)^3/\sqrt{\log n}\right)}, \tag{B.1}$$

and its space usage is polynomial.

We normalize by deleting duplicate and tautological clauses; thereafter there are $O(n^3)$ clauses, so polynomial factors are absorbed in the displayed bound. All arithmetic below is ordinary integer or rational arithmetic.

B.1 From many clauses to few factors

Counting sparsification, in its disjoint form [DHMTW14, Appendix A] with the quantitative bounds of [CIP06, Section 6], reduces us to $2^{O(n/\sqrt{\log n})}$ formulas on the original variables. In each formula, the clause count is $m = n$ polylog(n) and every variable has polylogarithmic occurrence. Use sparsification parameter $1/\lceil\sqrt{\log n}\rceil$. Its precise value is not important: the number of leaves must fit our time budget, and the occurrence bound must be a fixed polylogarithm. Process the leaves one at a time. Empty formulas and empty clauses are immediate; a leaf with $m \le n$ is covered by the main theorem.

Fix a remaining leaf. Let $v_e(x)$ be the 0–1 indicator that clause e is violated. It is a product of at most three literals x_i or $1 - x_i$. For a set T of clauses, put

$$g_T(x) = 1 - \sum_{e\in T} v_e(x).$$

This is first-order inclusion–exclusion. It equals 1 if no sampled clause is violated, and 0 if exactly one is violated. Therefore *one sample isolating a violation suffices to kill a false assignment*.

We use a sparse specialization of the classical probabilistic polynomial for OR of Beigel–Reingold–Spielman and Tarui [BRS91, Tar93]; see also [BHMS18, Section 2]. With probability at least $3/4$, we can choose $r_0 = O(n \log\log n)$ sets T_j such that every false assignment has an isolated violation in some T_j, every set has polylogarithmic size, and every clause occurs in $O(\log n)$ sets. The last two conditions are efficiently checkable.

The sampling argument. Choose a dyadic $p_0 = \Theta(\log m/n)$, and successively halve it down to order $1/m$. At each rate take Cn independent samples, for a sufficiently large constant C. There are $O(1 + \log(m \log m/n)) = O(\log \log n)$ sampling rates. A rate-p sample isolates an element of a fixed w-element violation set with probability $wp(1-p)^{w-1}$.

If $wp_0 \le 1$, this probability is $\Omega(wp_0)$; failure in all first-rate samples is $m^{-\Omega(Cw)}$. Union-bound over the at most m^w sets of size w, and then over these small w. If $wp_0 > 1$, some rate has $wp \in [1/2, 1]$. Isolation then has constant probability, so failure after Cn samples is $\exp(-\Omega(Cn))$; union-bound now over the at most 2^n assignments. Finally, a sample has expected size $O((m/n)\log m)$, and each clause participates in an expected $Cn \sum_j p_j = O(\log m)$ samples. Chernoff bounds give the stated size and participation bounds. Increasing C and the checkable thresholds gives the joint success probability above.

The $\log \log n$ number of scales affects the running time. Sample sizes and occurrence bounds, in contrast, may be *any fixed polylogarithms*. The guarantee concerns all violation vectors *realized by assignments*, not the entire punctured m-dimensional Boolean cube.

Now square the product:

$$G(x) = \prod_{j=1}^{r_0} g_{T_j}(x)^2, \qquad Z = \sum_{x \in \{0,1\}^n} G(x). \tag{B.2}$$

For every trial, $G = 1$ on satisfying assignments and $G \ge 0$ everywhere, so $Z \ge \#F$. A successful trial has $G = \mathbf{1}_F$, hence $Z = \#F$. Regard each square as two copies of its cubic factor, giving $R = 2r_0$ factors.

Expand each cubic separately into monomials, using $x_i^2 = x_i$ on Boolean inputs:

$$Z = \sum_{x \in \{0,1\}^n} \prod_{j=1}^{R} f_j(x), \qquad f_j(x) = \sum_{\mu} c_{j\mu} x^{S_{j\mu}}, \qquad |S_{j\mu}| \le 3. \tag{B.3}$$

A clause indicator expands into at most eight monomials, so this changes only constants. Each factor has polylogarithmically many monomials, and each variable belongs to polylogarithmically many monomial occurrences. We do *not* expand the product of the factors.

B.2 An ordinary permanent for a short product

Expanding the product in (B.3) means choosing one monomial from each factor. If the chosen monomials together use the variable set S, exactly $2^{n-|S|}$ assignments make them all equal to 1. Thus

$$Z = \sum_{\mu_1, \dots, \mu_R} \left(\prod_{j=1}^{R} c_{j\mu_j} \right) 2^{n - |\bigcup_j S_{j\mu_j}|}. \tag{B.4}$$

For example, the two terms of $f(x) = 2 - x_1 x_2$ contribute 8 and -1, giving $\sum_x f(x) = 7$. We encode these choices by a Valiant-style permanent reduction [Val79, BH93]. The explicit construction also gives the degree bound we need.

The monomial-choice matrix W. Make one selector s_j for each factor. For every monomial of f_j, make three private positions, labeled by its variables and padded with 1's. Join them into a cycle

$$s_j \longrightarrow o_1 \longrightarrow o_2 \longrightarrow o_3 \longrightarrow s_j. \tag{B.5}$$

Put the monomial coefficient on the first edge and weight 1 on the others. Alternatives share only their selector. Let W be the adjacency matrix of these cycles.

The zero-variable matrix V. For each variable x_i, cyclically join all positions labeled x_i, with weight 1 on every edge. Selecting this cycle will mean $x_i = 0$. An unused variable gets one isolated extra position with a

weight-one loop; it has no W-edges. Give every selector a loop of weight -1. Padding positions have no V-edges. Both matrices are indexed by the same set of $a = n\,\mathrm{polylog}(n)$ positions and selectors.

To combine the choices, use the single block matrix

$$A(V,W) = \begin{pmatrix} I+V & 0 & I & 0 \\ 0 & W & 0 & I \\ -I & I & I & I \\ I & I & I & -I \end{pmatrix}. \tag{B.6}$$

Its basic identity is

$$\boxed{2^{-a}\,\mathrm{per}\,A(V,W) = \sum_{\substack{I,J\subseteq[a]\\ I\cap J=\varnothing}} \mathrm{per}\,V[I,I]\ \mathrm{per}\,W[J,J].} \tag{B.7}$$

In words: choose a collection of V-cycles and a collection of W-cycles, with disjoint vertex sets.

Why the identity holds. The entire local calculation is

$$\mathrm{per}\begin{pmatrix} 1+\alpha & \beta & 1 & 0 \\ \gamma & \delta & 0 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} = 2(1+\alpha+\delta). \tag{B.8}$$

Expand (B.6) according to the entries selected from V and W, then sum the remaining entries separately at each index. The complementary minors recorded in (B.8) allow an index to be used by V, by W, or by neither, with internal weight 2 in every case. Simultaneous usage, or taking the incoming edge from one matrix and the outgoing edge from the other, gives zero. This proves (B.7), including its validity for arbitrary rational or polynomial edge weights.

Why this counts the Boolean sum. If a W-cover omits a selector, selecting or omitting that selector's -1 loop in V gives canceling contributions. Thus every selector must be used by W: precisely one monomial is chosen per factor. A variable cycle may be selected in V exactly when no selected monomial uses that variable. Each such variable has two choices, while a variable used by a monomial is forced to 1. This is exactly the factor $2^{n-|S|}$ in (B.4). Therefore

$$\boxed{\mathrm{per}\,A(V,W) = 2^a Z.} \tag{B.9}$$

The -1 loops are simply inclusion–exclusion enforcing that every selector is used. The weight-one loops for unused variables supply their factors of 2.

B.3 The degree is controlled by the number of choices

Every selector has level 0; a monomial's three positions have levels $1, 2, 3$. All W-edges go

$$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 0. \tag{B.10}$$

Unused-variable positions are outside these levels. Allow the entries of W to change arbitrarily, provided they keep these four transitions. Equation (B.7) remains valid, and the selector-loop cancellation still forces every selector into the W-cover.

A cycle cover on a subset of these four levels gives a bijection between each level and the next. Hence the four selected counts are equal. There are exactly R selectors, all selected, so such a cover uses exactly $4R$ entries of W. We have proved

$$\boxed{\deg_t \mathrm{per}\,A(V,W(t)) \le 4Rd \quad \text{whenever } \deg W(t) \le d \text{ and } W(t) \text{ keeps the four levels.}} \tag{B.11}$$

This is a polynomial identity after the cancellations in (B.7); no bound on individual uncancelled covers is required. The deformation may mix different monomial alternatives. Padding was only to make every alternative have the same constant length; degree-k factors would use $k + 1$ levels.

Now set $B = A(V, 0)$. It has small diagonal blocks: group the four copies of each variable's positions together, and group the four copies of each selector and its padding positions together. There are $b = n+R = O(n \log\log n)$ groups, each of polylogarithmic size. Only the W-entries connect different groups, and each group is connected to polylogarithmically many others. Thus

$$A(V, W) = B + C, \qquad N = 4a = n\,\mathrm{polylog}(n), \tag{B.12}$$

where B has small fixed blocks and C is sparse between those blocks. The powers hidden in these polylogarithms are immaterial. The quantity that must remain small is R.

B.4 Create the same easy points as in the main proof

We need only the following grouping variant of the main construction.

Lemma B.2 (Keep small groups together)**.** Suppose $B, C \in \mathbb{Q}^{N\times N}$ have a common partition into b groups, each of size at most $K = \mathrm{polylog}(b)$. Suppose B is block-diagonal in these groups, and each group is connected by C to at most $\Delta = \mathrm{polylog}(b)$ other groups. One can deterministically construct a polynomial $C(t)$ with

$$C(0) = C, \qquad \deg C(t) = d = O(\sqrt{\log b}), \qquad \ell = \Theta(\log b/\log\log b)$$

such that, at $i = 1, \dots, \ell$, the matrix $B + C(i)$ is block-diagonal plus low rank, with

$$s + r \log N = O(b/\log^2 b). \tag{B.13}$$

Here s bounds the new block size and r the error rank. A prescribed constant number of row-type/column-type restrictions can also be preserved, including the four levels above. Fixed polylogarithmic powers affect only the constants.

The point of the lemma is compatibility: *color entire groups*. The new blocks at a special point are unions of those groups, so the fixed matrix B automatically respects them. There is no new permanent algorithm.

Proof. Let $\Delta \geq 1$ be a degree upper bound. Color the graph of groups by words in $[q]^\ell$, with $Q = q^\ell$, so that each word is used at most $\lceil 2b/Q \rceil$ times, adjacent words disagree everywhere, and different edges have different unordered pairs of words. The main text's greedy argument works with the additional rule that vertices at distance at most two receive different words. Besides fewer than $Q/2$ full classes, the forbidden words number at most

$$\Delta^2 + \Delta^2\lceil 2b/Q \rceil + \Delta\ell Q/q.$$

The terms prevent distance-two conflicts, repeated edge-label pairs, and coordinate agreements. For $Q \geq 16\Delta\sqrt{b}$ and $q \geq 8\Delta\ell$, their sum is less than $Q/2$ for sufficiently large b. Choose the first available word.

Keep within-group entries of C fixed. Each ordered word pair now specifies at most one $K \times K$ block. Factor that block as itself times an identity, padding to inner dimension K. The scalar factorization of Section 4 becomes

$$C_{\mathrm{out}} = L(I_C \otimes X^{\otimes\ell} \otimes I_K)R', \qquad M = K(q(q-1))^\ell \leq KQ^2.$$

Replace $X^{\otimes\ell}$ by the same $J(t)$ from Theorem 5.1, with exceptional fraction ℓ^{-A}. At $t = i$, its nice part connects only groups with the same ith symbol. Each new block contains at most $3b/q$ old groups. Consequently,

$$s \leq 3bK/q, \qquad r \leq M/\ell^A.$$

Both the fixed part and the nice part respect these very same blocks.

Choose q to be a sufficiently large fixed power of $\log b$, and take the least ℓ with $q^\ell \geq 16\Delta\sqrt{b}$. Then $Q < 16\Delta q\sqrt{b}$, so $M = b\,\mathrm{polylog}(b)$ and $\ell = \Theta(\log b/\log\log b)$. The choice of q makes s small; choose the fixed constant A afterwards to make $r\log N$ small. This proves (B.13), with $d = O_A(\sqrt{\ell\log\ell}) = O(\sqrt{\log b})$. The power 2 in (B.13) is only convenient slack; any larger fixed power is equally possible.

Finally, preserve the levels by deleting forbidden entries. If $\widetilde{C}(t)$ is the family just constructed and Π_j projects onto level j, set

$$C(t) = \sum_{j=0}^{3} \Pi_j \widetilde{C}(t)\Pi_{j+1}, \qquad \Pi_4 = \Pi_0.$$

The projections are zero outside the choice indices. They preserve every block partition, keep $C(0) = C$, and multiply the error rank by at most four. Degree is unchanged. The same argument handles any fixed list of permitted row-type/column-type pairs. All matrices and rational coefficients have polynomial-size descriptions, as in the main construction. □

B.5 Finish by permanent interpolation

Apply the grouping lemma to (B.12), and let $P(t) = \mathrm{per}(B + C(t))$. Its level-preserving deformation gives $P(0) = 2^a Z$ and $\deg P \leq 4Rd$. Every square submatrix permanent at a special point costs $2^{O(s)}N^{O(r)}\,\mathrm{poly}(N)$. These are exactly the hypotheses needed by Section 3. Choose

$$h = \left\lfloor \frac{4Rd}{\ell} \right\rfloor + 1 = O\left(\frac{R\log\log n}{\sqrt{\log n}}\right).$$

The permanent addition formula computes the first h Taylor coefficients at every special point; Hermite interpolation recovers $P(0)$. We use the actual bound on $\deg P$, not the generic estimate Nd. For sufficiently large n, $h \leq N/2$, and the time is

$$\begin{aligned} 2^{O(s + r\log N + h\log(N/h))} &= 2^{O(R(\log\log n)^2/\sqrt{\log n})} \\ &= 2^{O(n(\log\log n)^3/\sqrt{\log n})}. \end{aligned}$$

Indeed, $b = n + R = O(n\log\log n)$ makes the easy-point cost (B.13) negligible, and $N = n\,\mathrm{polylog}(n)$ gives $\log(N/h) = O(\log\log n)$. Divide by 2^a to obtain Z.

The three logarithmic factors have distinct sources: sampling uses $\log\log n$ scales; $d/\ell = O(\log\log n/\sqrt{\log n})$ determines the derivative order; and enumerating derivative terms contributes $\log(N/h) = O(\log\log n)$. Any fixed polylogarithmic increase in matrix size, group size, or interaction degree is harmless. Increasing the number of factors R directly increases the exponent.

For each sparsification leaf, initialize an estimate to 2^n. Perform $O(n)$ independent sampling trials, ignore those failing the checkable size or participation bounds, and take the minimum of the exactly evaluated values Z and the initial estimate. Every value is an upper bound and each trial has constant probability of being exact. A union bound over the leaves gives total success probability at least $3/4$. Repeat the *whole trial*; do not enlarge its product of factors. The leaf count and this polynomial repetition overhead are absorbed in (B.1).

All matrix constructions and evaluations after the clause sampling are deterministic. Use Appendix A for the structured permanents and process leaves, trials, and derivative terms sequentially; exact arithmetic has polynomial bit overhead and the space remains polynomial. A deterministic extension is possible with additional counting and coding arguments [OAI26]; we do not develop it here.

References

[Bar96] Alexander I. Barvinok. Two Algorithmic Results for the Traveling Salesman Problem. *Mathematics of Operations Research* 21(1):65–84, 1996. Section 3, especially Algorithm 3.2 and Theorem 3.3. doi:10.1287/moor.21.1.65.

[BH93] Amir Ben-Dor and Shai Halevi. Zero-One Permanent Is #P-Complete, A Simpler Proof. In *ISTCS*, pages 108–117, 1993. The linear-size signed construction is in Section 4. Author's manuscript.

[BHMS18] Siddharth Bhandari, Prahladh Harsha, Tulasimohan Molli, and Srikanth Srinivasan. On the Probabilistic Degree of OR over the Reals. In *FSTTCS*, LIPIcs 122, pages 5:1–5:12, 2018. Section 2. doi:10.4230/LIPIcs.FSTTCS.2018.5.

[BRS91] Richard Beigel, Nick Reingold, and Daniel A. Spielman. The Perceptron Strikes Back. In *6th IEEE Conference on Structure in Complexity Theory*, pages 286–291, 1991. doi:10.1109/SCT.1991.160270.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A Duality between Clause Width and Clause Density for SAT. In *CCC*, 2006. Section 6. Author's manuscript.

[DHMTW14] 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. Counting sparsification and weighted-permanent reductions. arXiv:1206.1775.

[OAI26] OpenAI. *The Exponential Time Hypothesis Is False*. Supplied manuscript dated October 17, 2026; revised October 25, 2026.

[Rys63] Herbert John Ryser. *Combinatorial Mathematics*. Carus Mathematical Monographs, volume 14. Mathematical Association of America, 1963.

[Sub26] Bernardo Subercaseaux. Observation on nondeterministic time and schedule-conditioned SAT. Social media post, 2026. Source cited in the supplied exposition.

[Tar93] Jun Tarui. Probabilistic Polynomials, AC^0 Functions, and the Polynomial-Time Hierarchy. *Theoretical Computer Science* 113(1):167–183, 1993. Preliminary version in STACS 1991. doi:10.1016/0304-3975(93)90214-E.

[Val79] Leslie G. Valiant. The Complexity of Computing the Permanent. *Theoretical Computer Science* 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.