

Exact Derandomization of Logarithmic Space: $L = RL = BPL$

OpenAI

September 23, 2026

Abstract

We prove $L = RL = BPL$ for polynomial-time randomized logarithmic-space machines. More generally, for each fixed such machine, a uniform deterministic transducer approximates its acceptance probability to additive error 2^{-q} using $O(\log(n+2)+q)$ work space and $(n+2)^{O(1)}2^{O(q)}$ time, where n is the input length and $q \geq 1$ is supplied in unary. The construction maintains a reward and a remaining transition contribution whose sum is the original acceptance probability at the designated start. Matrix corrections and copies of vertices shrink the transition entries faster than the number of vertices incident to positive transitions grows, making that contribution small. The estimates compare endpoints using short random keys; we control the errors even when later endpoints depend on earlier comparisons. We bound the combined storage of all recursive estimator calls, including suspended callers. Enumerating a finite family of computations that all halt and taking the median of their estimates gives the approximation without an acceptance-probability promise.

Contents

1	Introduction	2
2	The transition model and resource parameters	4
3	From reward estimates to deterministic approximation	7
4	Sampling retained vertex sets	11
5	The exact correction and copy hierarchy	13
6	Sample tables and their access operations	22
7	Conditional averaging in the shared environment	32
8	Additive estimation and conditional prefix bounds	36
9	Compression of additive row estimates	47
10	Adaptive fingerprints within budget blocks	57
11	Finite numerical queries with bounded suspended storage	68
12	Uniform controller implementation	79
13	Finite parameter choices and certified accepting computations	99
14	The effective compiler	102
	References	109

1 Introduction

A logarithmic-space machine can remember a bounded number of positions in its input, but cannot keep a polynomial-length computation or random tape. Can fresh random bits increase what such a machine can decide? We answer this question by giving a deterministic logarithmic-space simulation of every polynomial-time randomized logarithmic-space computation with bounded error.

We use machines with finite control, an endmarked read-only input, finitely many work tapes, and fresh independent fair bits. Input heads are confined between the endmarkers. All randomized running-time bounds hold on every random tape. Space counts traversed writable cells in bits, including every simultaneously live counter, numerical register and suspended call; visiting a blank work cell counts toward space. A transducer has a one-way write-only output tape. Throughout, n is the input length and logarithms have base two.

The class L consists of languages decidable by deterministic machines in $O(\log(n+2))$ work space. Such a halting deterministic machine has only polynomially many configurations and hence runs in polynomial time. The class RL permits polynomial-time randomized machines in this space, with acceptance probability zero on no inputs and at least $1/2$ on yes inputs. For BPL , the corresponding probabilities are at most $1/3$ and at least $2/3$.

Theorem 1.1 (Exact logarithmic-space derandomization). *For the machine and resource conventions above,*

$$L = RL = BPL.$$

The proof approximates arbitrary acceptance probabilities. This gives the two-sided statement directly, and also applies to computations whose acceptance probabilities have no decision gap. Its quantitative form separates the memory needed for the input from the memory needed for the requested accuracy.

Theorem 1.2 (Acceptance-probability approximation). *For each fixed randomized machine $\mathcal{M}$ with polynomial worst-case running time and $O(\log(n+2))$ work space, there is a uniform deterministic transducer with the following guarantee. On input $(x, 1^q)$, where $n = |x|$ and q is an integer at least one, it outputs an integer $a \in \{0, \dots, 2^{q+2}\}$ such that*

$$\left| a2^{-(q+2)} - p_{\mathcal{M}}(x) \right| \leq 2^{-q}, \quad p_{\mathcal{M}}(x) = \Pr[\mathcal{M}(x) \text{ accepts}].$$

The transducer uses $O(\log(n+2) + q)$ work space and $(n+2)^{O(1)}2^{O(q)}$ time, with constants depending only on $\mathcal{M}$.

In particular, every fixed inverse-polynomial accuracy is attained in logarithmic space and polynomial time. The precision parameter is unary; the running-time bound is exponential in general q . We first prove the fixed inverse-polynomial result and then obtain the single variable-precision algorithm by padding. Section 3 also derives total deterministic separators for bounded-error promise problems; Section 13 constructs accepting runs when the machine's acceptance probability has a specified inverse-polynomial lower bound.

The simulation is also effective at the level of machine descriptions. Given a source program and supplied polynomial-time and logarithmic-space bounds, one terminating compiler produces a total deterministic machine with explicit numerical bounds for its own space and time (Theorem 14.1). When the source satisfies the supplied bounds and its bounded-error promise, the emitted machine decides the same language. Section 14 constructs this compiler by specializing one fixed deterministic library.

1.1 Relation to earlier methods

Acceptance-probability estimation is a standard formulation of space-bounded derandomization. Hoza obtains deterministic space $O(\log^{3/2} n / \sqrt{\log \log n})$ for general bounded-error logarithmic-space computation [8, Theorem 1.6 and Corollary A.5]. For a product of T arbitrary stochastic matrices of dimension $w \times w$, with $T \geq w$, Cohen, Doron, Sberlo and Ta-Shma obtain space $\tilde{O}(\log T + \sqrt{\log T} \log w)$ at error $2^{-\text{polylog}(T)}$ [6, Theorems 1.1 and 6.1]. Their precision-alternating recursion already reduces the cost of storing intermediate approximations, and their generalized composition lemma shares one global location in a formula while charging local work along its paths [6, full version, Lemma 6.13]. When $\log T = \Theta(\log w)$, however, this bound remains superlogarithmic. The issue addressed here is to reach $O(\log n)$ total space for general polynomial-time computations, including all information retained while an estimate is recomputed.

Matrix inversion also supplies a close point of comparison. Ahmadijad, Kelner, Murtagh, Peebles, Sidford and Vadhan use a finite inverse on a path-layered graph to encode random-walk probabilities [1, Section 1.2, Equation (1)]. Their nearly logarithmic-space estimation theorem applies to Eulerian graphs; they also give a larger-space high-precision algorithm for general Markov chains [1, Theorems 1.1 and 1.3]. Thus both general transition matrices and their inverse representation already occur in this literature. Our time-layered configuration matrix need not be Eulerian. We update its reward directly while reducing the remaining transition contribution. The reward estimates use row arrays represented by a finite list of exceptional entries and a common value at every omitted endpoint.

Universal estimation has a separate antecedent. Pyne, Raz and Zhan give one deterministic algorithm that approximates the acceptance probability of every ordered branching program of length and width at most n to error less than $1/n$; for each space-constructible $S(n) \geq \log n$, its space is $O(S(n))$ exactly when $\text{prBPL} \subseteq \text{SPACE}[O(S(n))]$ [10, full version, Theorem 5.1]. Here prBPL denotes bounded-error promise problems in randomized logarithmic space and polynomial time. Our compiler specializes the fixed separator established here and computes explicit numerical space and time bounds from the supplied source bounds. Its construction uses the probability approximation theorem also on inputs outside a decision promise.

1.2 Proof strategy

Deterministic approximation. Put $B = \Theta(\log(n+2))$. We construct a finite family of estimates, each specified by $O(B)$ bits. Most estimates approximate the desired probability, and every estimate can be evaluated by a total deterministic $O(B)$ -space computation, including choices for which it is inaccurate. Section 3 gives the deduction from these two properties: condition on retaining the start vertex, enumerate the conditional family, and take its median. It also proves the padding argument for variable precision.

Exact target. The construction acts directly on the time-layered configuration matrix; no Eulerian hypothesis is needed. For a current transition matrix C and a nonnegative correction E on the same vertex set, the identity

$$D = C - E + EC, \quad I - D = (I + E)(I - C)$$

shows how to update a reward without changing the continuation vector it represents: if $W = (I - C)p$, then $(I + E)W = (I - D)p$. We suppress correction in columns with too many substantial entries and divide the remaining large entries among copies of their destination vertices. Counting those substantial entries bounds the number of parents needing extra active copies. The number of active

vertices—those incident to a positive transition—grows by at most a factor of two, while the entry bound falls by 2^H , for fixed $H \geq 4$. The remaining transition mass therefore decays geometrically. Section 5 turns this mechanism into a reward approximating the original probability after $O(B)$ stages.

Estimator accuracy. To avoid storing full endpoint identifiers at every recursive level, the estimates compare their short random keys. The endpoints being compared depend on earlier comparisons. For the error analysis, group statistical budgets into consecutive intervals and replace key comparisons by exact endpoint equality in all calls within one interval. Calls below it keep the actual algorithm. The comparison calculation’s candidate endpoint set is independent of that interval’s key. If an actual path leaves this set, one of its lower calls exposes an entry where a bounded witness function is one in the actual calculation and zero in the comparison. The probability of leaving the set is therefore bounded by errors in those lower witness functions. Inside the fixed set, a union bound over pairs controls the probability that distinct endpoints have the same key. This decomposition closes the error induction for the actual compressed estimates in Section 10. The conditional table identities and compression estimates supply its quantitative inputs.

Space bound. The space obstacle is that the neighbor operations used in a configuration tree traversal are themselves recursive estimator queries. Cook and McKenzie’s contour construction assumes access to ordered incidences and edge reversal [7, Proposition 1]; those operations must coexist here with suspended callers. We construct finite program graphs that represent each completed lower restoring query or paired motion by one transition. A predecessor probe copies only the calling program’s local records, while the lower query occupies its own stack segment. The digit procedures release long numerical words before requesting operand digits at larger indices. Each suspended caller’s records can then be bounded by a fixed linear combination of the decrease in its assigned storage allowance and the difference between the caller’s and callee’s digit positions. Summing these differences along the physical call chain cancels the intermediate allowances and digit positions, leaving a bound in terms of the outermost call. Section 12.8 proves totality and the simultaneous $O(B)$ space bound.

The conditional averaging used in the estimates is derived from Shalom’s property (T) and relative property (T) statements [12, Theorems 2.1 and 2.6]; Section 7 checks the finite affine actions and their conditional measures. The numerical and controller arguments are proved in Sections 11 and 12.

2 The transition model and resource parameters

We first express acceptance probability as a finite matrix inverse. Time labels make the transition matrix nilpotent; keeping the complete configuration at each vertex gives local access in both directions. The construction will use the resulting probability vector at every configuration, as well as its value at the designated start.

Attach an integer time to each vertex of a finite set V . We use matrices whose nonzero entries go only to strictly later times. Order vertices by time, breaking ties by their encodings; the matrices are then strictly upper triangular. Throughout, $C(x, y)$ denotes a transition from row x to column y . A matrix is *row-substochastic*, or simply *substochastic*, when its entries are nonnegative and each row sums to at most one, following the row convention of [6, Section 3.1]. Vectors such as e and p_0 are column vectors, so a transition matrix acts on continuation values by multiplication on the left.

For local edge access, a vertex and a port from a fixed finite set specify an edge. A *paired* edge operation returns the other endpoint together with its inverse port; applying the inverse operation restores the first endpoint. Absent ports are explicitly recognized.

Lemma 2.1 (Configuration reduction). *Fix a randomized machine with polynomial worst-case running time and $O(\log(n+2))$ work space. Its acceptance probability on each length- n input has a uniform logarithmic-space representation*

$$p_0(x_0), \quad p_0 = (I - S)^{-1}e \in [0, 1]^{V_0},$$

where V_0 is a polynomial-size time-ordered vertex set, $x_0 \in V_0$, $e \in \{0, 1\}^{V_0}$, and S is strictly upper triangular and row-substochastic. The incoming and outgoing port degrees of S are bounded by machine-dependent constants. Vertex encodings, port validity, edge weights, and paired edge operations are computable uniformly in logarithmic space.

For an RL computation this representation gives $p_0(x_0) = 0$ on no inputs and $p_0(x_0) \geq 1/2$ on yes inputs.

Proof. Let $T(n)$ be a fixed polynomial upper bound on the running time. A vertex records a time in $\{0, \dots, T(n)\}$, the finite control, all input and work head positions, and the contents of a sufficiently large box of $O(\log(n+2))$ work cells. Take elementary machine steps with fixed dyadic coin probabilities. A transition advances the time by one. Invalid configurations or moves leaving the box have no corresponding transition. The box is chosen large enough that this never affects a computation from the specified start.

Each elementary step has constantly many successor choices. To find predecessors, guess the old finite control, locally overwritten symbols, head movements, and coin choice, then check that the forward step gives the specified configuration. There are constantly many such guesses. Combine transitions with the same ordered pair of endpoints by adding their probabilities. Order the distinct successors and predecessors by their configuration encodings; their positions give paired ports for the resulting matrix entries. These lists have constant length, so duplicate removal, summed weights, and the inverse port are computed in logarithmic space by local enumeration. This base computation makes no recursive graph queries. All records use $O(\log(n+2))$ bits, and the read-only input can be consulted during these checks.

All input-head coordinates in the vertex universe are restricted to a fixed polynomial interval containing the positions reachable within $T(n)$ local steps from their specified initial positions. Configurations and predecessor candidates outside that interval are invalid. To obtain a simulated head's scanned symbol, compare its stored coordinate with the input boundaries, supply any required endmarker or off-data blank according to the source tape convention, and otherwise scan the real endmarked input to the designated position using a logarithmic-space counter. A position forbidden by the source tape convention remains invalid. The physical scan stays between the real endmarkers, even when a valid simulated coordinate is outside the data. This routine makes no recursive graph query.

An already halted computation keeps its complete tape and head data while its time advances. In particular, halted configurations are not merged into a single vertex. Let e be one precisely at accepting configurations at time $T(n)$, and zero elsewhere. Final-time rows have no outgoing transitions. Thus S is strictly upper triangular and substochastic, and it is nilpotent. Its inverse expression is the finite sum

$$(I - S)^{-1}e = \sum_{j=0}^{T(n)} S^j e.$$

For a configuration v at time τ , the only possibly nonzero summand is $(S^{T(n)-\tau}e)(v)$: each transition advances time by one and e is supported at time $T(n)$. This summand is the mass of accepting paths of the required length, with any missing transition mass treated as rejection. It lies in $[0, 1]$ because S is substochastic. At x_0 no transition was removed by a resource check, so this mass is exactly the machine's acceptance probability. $\square$

The coordinatewise bound on p_0 will also control the values assigned to vertex copies: each copy receives its parent's continuation value multiplied by a factor in $[0, 1]$. The bound holds at every configuration, independently of the acceptance probability at the designated start. Thus the matrix construction and the probability approximation below apply to arbitrary randomized computations satisfying the resource bounds.

2.1 Resource parameters

Fix an integer parameter B , a power of two, comparable with $\log_2(n + 2)$ and at least a sufficiently large fixed constant. For example, take the least power of two at least the maximum of that constant and $\lceil \log_2(n + 2) \rceil$. This choice is computable in logarithmic space and includes all bounded input lengths in the same construction. All constants throughout the construction are independent of the input; they may depend on the fixed machine and, for fixed inverse-polynomial accuracy, its fixed exponent. The uniform precision parameter of Theorem 1.2 is handled separately by padding. The number of stages is $L = O(B)$. At stage l , a vertex has a base identifier followed by l symbols from a fixed radix D_0 . The radix will be selected by the hierarchy. Encode each base vertex by a fixed-width radix- D_0 word and prepend the digit 1; unused base words are invalid. Appending one copy digit at each stage gives injective numerical identifiers even across different stage lengths. Indeed, a base width w places stage- l identifiers in $[D_0^{w+l}, 2D_0^{w+l})$, and these intervals are disjoint. Appending a fixed suffix is an affine map on identifiers.

Every stage has size at most a common bound

$$N = 2^{C_N B}.$$

We may increase the fixed constant C_N as needed. This is a bound on one stage, not a claim that there are only N pairs of vertices. Integrated entry errors below carry a factor $1/N$ while summing over both endpoints.

Fix integers $H \geq 4$ and $2 \leq h_0 < H$, and put

$$b_k = 2^{-Hk} \quad (k \geq 0).$$

These are the nominal sampling rates. Stage l will have density $f = b_l$. The sample-dependent tables will use rates from this same grid for their source and terminal retention tests. The error power in the later analysis is always $s = 8$.

All thresholds needed by the algorithm can be chosen rational, with fixed gaps, and all discrete constants can be chosen integer. The estimates below are uniform in the ambient finite universe: the sampler works on any finite identifier and rate range once its prime field is large enough. The hierarchy uses $O(B)$ copy digits and rates with $k = O(B)$. Section 13.1 verifies the same rate bound for recursive estimator calls and gives the compatible order of constant choices. We then choose the field-size exponent so that the field has size $2^{O(B)}$ and covers these ranges. Its elements and all identifiers occupy $O(B)$ bits; enlarging the exponent leaves the probabilistic constants of Section 4 unchanged.

3 From reward estimates to deterministic approximation

The proof of the main theorems reduces to approximating one deterministic reward by a finite family of total computations. We give that deduction here, using the outputs of the hierarchy, estimator analysis, and controller. The subsequent sections establish these three inputs.

We retain the transducer conventions of the introduction: output contents are never available as work memory. Write $p_{\mathcal{M}}(x)$ for the acceptance probability of a fixed randomized machine $\mathcal{M}$ on input x .

Theorem 3.1 (Fixed inverse-polynomial accuracy). *For every fixed randomized machine $\mathcal{M}$ with polynomial worst-case running time and $O(\log(n+2))$ work space, and every fixed integer $d \geq 1$, there is a uniform deterministic transducer that outputs, on each length- n input x , a dyadic rational $z_x \in [0, 1]$ satisfying*

$$|z_x - p_{\mathcal{M}}(x)| \leq (n+2)^{-d}.$$

The transducer uses $O(\log(n+2))$ work space, outputs $O(\log(n+2))$ bits, and runs in polynomial time. All constants may depend on $\mathcal{M}, d$ and are independent of x .

3.1 The three inputs to the approximation

Fix the configuration representation of Lemma 2.1 and the size parameter B of Section 2.

The hierarchy constructs a fixed radix D_0 , copy sets $V_l = V_0 \times \{0, \dots, D_0 - 1\}^l$, and deterministic rewards $W_l \in [0, 1]^{V_l}$. Write $\bar{x}_l$ for the copy of x with all copy symbols zero. Proposition 5.7 gives, for every constructed stage,

$$0 \leq p_0(x) - W_l(\bar{x}_l) \leq |V_0|2^{-(H-1)l}.$$

For $L = O(B)$ all these domains have a common bound $N = 2^{C_N B}$. Their addresses use $O(B)$ bits.

For the final reward, put $f = 2^{-HL}$. The construction supplies a finite environment σ , uniform over specified matrix encodings and original fair-bit strings, with $O(B)$ bits in total. Its retention indicator $X_f(z)$ has probability $\pi_f \asymp f > 0$ by Lemma 4.1. At any admissible integer statistical budget $M_{\text{top}} = O(B)$, the actual estimate $\bar{W}_L(z; \sigma)$ is in $[0, 1]$. Let i_* be the positive integer rank of this reward task in the dependency order of Section 8; it is $O(L+1)$. For fixed constants $K_{\text{prec}}, A_*, C_b$, its precision is

$$p = M_{\text{top}}/K_{\text{prec}} - HL - A_* i_*.$$

This is Equation (8.8) at $a = u = f$. Theorem 10.4 gives

$$\frac{1}{N} \mathbb{E}_{\sigma} \sum_{z \in V_L} \frac{X_f(z)}{f} |\bar{W}_L(z) - W_L(z)|^8 \leq C_b^8 2^{-8p}.$$

The bound applies to the actual endpoint-grouping rule, including both the exact-grouping error and the additional error from fingerprint comparisons.

Finally, each fixed estimator value has digits $w_j \in \{-6, \dots, 6\}$ with $\bar{W}_L(z) = \sum_{j \geq 0} w_j 2^{-j}$ and tail at most $6 \cdot 2^{-K}$ after place K . With the digit interface of Theorem 11.3, Theorem 12.1 implements every finite query in $O(B + M_{\text{top}} + j + 1)$ space. Every query terminates on every environment. Its physical input heads stay between the real endmarkers, and its structural choices are independent of the requested digit place. These guarantees are needed for enumeration of the entire conditional law, including its inaccurate estimates.

With the constants and finite admissible ranges chosen in the order of Section 13.1, the linear stage count and budget can be increased for any fixed accuracy exponent.

3.2 Conditioning and the median

Let $W \in [0, 1]^V$ be deterministic and $\bar{W}(v; \sigma)$ a real estimator under the shared environment law. If

$$\frac{1}{N} \mathbb{E} \sum_{v \in V} \frac{X_f(v)}{f} |\bar{W}(v) - W(v)|^8 \leq C_b^8 2^{-8p},$$

then for every fixed $z \in V$ and $\eta > 0$,

$$\mathbb{E}[|\bar{W}(z) - W(z)|^8 \mid X_f(z) = 1] \leq N \frac{f}{\pi_f} C_b^8 2^{-8p}, \quad (3.1)$$

$$\Pr[|\bar{W}(z) - W(z)| > \eta \mid X_f(z) = 1] \leq N \frac{f}{\pi_f} C_b^8 2^{-8p} \eta^{-8}. \quad (3.2)$$

Here f is an admissible rate and $\pi_f = \Pr[X_f(z) = 1] > 0$.

To obtain these inequalities, keep only the nonnegative summand at z . Since $\Pr[X_f(z) = 1] = \pi_f > 0$,

$$\frac{\pi_f}{f} \mathbb{E}[|\bar{W}(z) - W(z)|^8 \mid X_f(z) = 1] = \mathbb{E}\left[\frac{X_f(z)}{f} |\bar{W}(z) - W(z)|^8\right] \leq N C_b^8 2^{-8p}.$$

Dividing by π_f/f proves Equation (3.1). In particular, choosing one specified coordinate costs the factor N from the integrated norm. Markov's inequality gives Equation (3.2).

Proof of Theorem 3.1. Apply Lemma 2.1 to $\mathcal{M}$, with no promise on its acceptance probability. Set

$$r = d \lceil \log_2(n+2) \rceil, \quad \Delta = 2^{-r}, \quad L = \left\lceil \frac{C_0 B + r + 3}{H - 1} \right\rceil,$$

where $|V_0| \leq 2^{C_0 B}$ is an explicit configuration bound. Since d is fixed, $r, L = O(B)$. Let z be the all-primary copy of the start at stage L . Proposition 5.7 gives

$$0 \leq p_{\mathcal{M}}(x) - W_L(z) \leq \Delta/8. \quad (3.3)$$

This bound holds for every acceptance probability. Choose a fixed C_N large enough to bound all the resulting stage domains by $N = 2^{C_N B}$.

For the final reward task take $a = u = f = b_L$. Its rank satisfies $i_* = O(L+1)$, and Equation (8.8) reads

$$p = M_{\text{top}}/K_{\text{prec}} - HL - A_* i_*.$$

By Lemma 4.1, fix an integer $C_{\text{tail}} \geq 1$ bounding $(f/\pi_f)C_b^8$ throughout the admissible range. Choose the fixed linear coefficient of $M_{\text{top}} = O(B)$ large enough that

$$p \geq r + 3 + \frac{C_N B + \lceil \log_2 C_{\text{tail}} \rceil + 3}{8}. \quad (3.4)$$

The top-budget and field-size choices in Section 13.1 allow this enlargement; Equation (13.1) still bounds every descendant rate. Theorem 10.4 supplies the reward moment estimate. The conditioning calculation above, with $\eta = \Delta/8$, now shows that

$$\Pr[|\bar{W}_L(z) - W_L(z)| > \Delta/8 \mid X_f(z) = 1] \leq 1/8. \quad (3.5)$$

Thus the factor N lost on selecting one coordinate has been paid by the statistical precision. Explicitly, the bound in Equation (3.2) is at most

$$C_{\text{tail}} 2^{C_N B - 8p + 8r + 24} \leq 2^{-3},$$

by Equation (3.4).

We next turn the estimator values into a finite list of dyadic numbers. For each master environment, Theorem 11.3 gives the digits $w_j \in \{-6, \dots, 6\}$ of its one fixed value $\bar{W}_L(z)$. Structural choices do not depend on the requested digit. Put $K = r + 6$, compute

$$A_\sigma = \sum_{j=0}^K w_j 2^{K-j}, \quad a_\sigma = \min\{2^K, \max\{0, A_\sigma\}\}, \quad v_\sigma = a_\sigma 2^{-K}.$$

The integer A_σ is obtained by starting at zero and applying $A \leftarrow 2A + w_j$ in order, so it uses $O(K)$ bits. By Equation (11.2), the truncation error is at most $6 \cdot 2^{-K} < \Delta/8$. Clipping cannot increase error relative to $W_L(z) \in [0, 1]$. Equations (3.3) and (3.5) imply that, on more than three quarters of the environments conditioned on $X_f(z) = 1$,

$$|v_\sigma - p_{\mathcal{M}}(x)| < 3\Delta/8. \quad (3.6)$$

To enumerate the conditional law in Equation (3.5), range each field-matrix entry over $\{0, \dots, P-1\}$ and enumerate the original salt bit strings. Retain exactly the encodings whose positive-degree coefficient matrices have full column rank and for which $X_f(z) = 1$. These encodings are equally weighted. Raw salt strings that induce the same fingerprint remain separate, so their multiplicities reproduce the original law. At least one encoding is retained: a matrix with linearly independent positive-degree columns exists, and its independent constant column can place the designated evaluation in the nonempty retention box. The matrices have fixed dimensions and $\log P = O(B)$; together with Equation (10.4), this gives $O(B)$ bits per master environment and at most $2^{O(B)}$ encodings to enumerate.

Let $E \geq 1$ be the number of retained encodings, found by one pass, and let $h = \lfloor E/2 \rfloor + 1$. Binary-search the integer interval $[0, 2^K]$ for the least a such that at least h retained encodings satisfy $a_\sigma \leq a$. Each count is computed by a fresh pass, recomputing each numerator a_σ . This finds the upper median of the finite multiset, including all repetitions. Since more than three quarters of its members satisfy Equation (3.6), so does its median: fewer than $E/4$ values can lie below the indicated interval, and fewer than $E/4$ can lie above it. The algorithm retains only the current count and search registers; it never stores the multiset. Output the integers (a, K) representing $z_x = a 2^{-K}$. Then

$$|z_x - p_{\mathcal{M}}(x)| < 3\Delta/8 \leq (n+2)^{-d}.$$

We account for the memory held during this computation. The master environment, current cursor, catalyst, shared scratch, and suspended controller records together use $O(B)$ bits by Theorem 12.1, because $M_{\text{top}}, K = O(B)$. The outer computation keeps only a fixed number of additional $O(B)$ -bit registers: its top parameters and designated start, the environment enumeration index, E , h , the current count, the two binary-search endpoints and candidate, the digit index, and the signed numerator accumulator. Each count is bounded by the number of environment encodings, so it also has $O(B)$ bits. These outer registers remain live while a digit query runs; in particular, both its environment-enumeration state and its numerator accumulator are included. They are present once for the whole computation, not once per recursive level. Every query terminates on every environment, including the inaccurate environments, by the same realization theorem. The output is written only after the median is computed.

Each digit query is a halting deterministic computation with $O(B)$ work bits and physical input heads confined between the real endmarkers. Counting its finite control, work-tape contents, and all work and input head positions gives at most $2^{O(B)}$ complete configurations on its fixed input. A repeated complete configuration would force a deterministic loop, so totality bounds the query by $2^{O(B)}$ steps. There are $2^{O(B)}$ environment encodings, $O(K + 1)$ binary-search passes, and $K + 1$ digit queries per numerator. Their product is still $2^{O(B)} = (n + 2)^{O(1)}$. All finite parameters are generated uniformly, and all constants are fixed for $(\mathcal{M}, d)$. $\square$

3.3 The class equality and promise separation

Proof of Theorem 1.1. Let $\mathcal{M}$ recognize a language in BPL, with acceptance probability at most $1/3$ on no inputs and at least $2/3$ on yes inputs. Apply Theorem 3.1 with $d = 3$. For every input length, including $n = 0$, its error is at most $(n + 2)^{-3} \leq 1/8$. Hence the dyadic answer is at most $1/3 + 1/8 = 11/24$ on a no input and at least $2/3 - 1/8 = 13/24$ on a yes input. Comparing this answer with $1/2$ decides the language uniformly in logarithmic space. This proves $\text{BPL} \subseteq \text{L}$ directly from probability approximation.

A deterministic machine may ignore its coins, giving $\text{L} \subseteq \text{RL}$. Two independent repetitions of an RL machine, accepting if either repetition accepts, have acceptance probability zero on no inputs and at least $3/4$ on yes inputs. They use logarithmic space and polynomial time, so $\text{RL} \subseteq \text{BPL}$. The three inclusions prove $\text{L} = \text{RL} = \text{BPL}$. $\square$

A promise problem is a pair (Y, N) of disjoint sets of inputs. Here **PromiseBPL**, denoted **prBPL** in the introduction, requires one polynomial-time randomized logarithmic-space machine that halts on every input, accepts each input in Y with probability at least $2/3$, and accepts each input in N with probability at most $1/3$. The class **PromiseL** consists of pairs separated by a total deterministic logarithmic-space decider; there is no correctness requirement outside $Y \cup N$.

Corollary 3.2. *For promise problems whose randomized machine is total on all inputs, and whose deterministic separator is also required to be total, one has*

$$\text{PromiseBPL} = \text{PromiseL}.$$

Proof. The same $d = 3$ approximation and threshold $1/2$ used in the preceding proof accept every input in Y and reject every input in N . The resulting machine halts also outside the promise because its approximation queries are total. The reverse inclusion follows by ignoring the coins. $\square$

3.4 Accuracy supplied as part of the input

Increasing the numerical clock alone approximates the same estimator value more accurately; it does not reduce the statistical error. To handle an arbitrary requested accuracy while retaining the condition $M_{\text{top}} = O(B)$, we apply the fixed-accuracy theorem to a longer input represented implicitly.

Proof of Theorem 1.2. Fix a padded machine $\mathcal{M}_{\text{pad}}$ as follows. On a string beginning with $1^n 0x$, where $|x| = n$, it simulates $\mathcal{M}(x)$ and ignores all remaining symbols. If the delimiter or the following n symbols are absent, it rejects. This is one fixed randomized machine with polynomial worst-case running time and logarithmic work space in its full input length. On every string with the specified prefix its acceptance probability is $p_{\mathcal{M}}(x)$. Fix once and for all the transducer of Theorem 3.1 for $\mathcal{M}_{\text{pad}}$ and $d = 1$.

On input $(x, 1^q)$, with $n = |x|$ and $q \geq 1$, set

$$m = 2^{q + \lceil \log_2(2n+4) \rceil + 4}.$$

Simulate that fixed transducer on the virtual length- m string

$$y = 1^n 0 x 0^{m-(2n+1)}.$$

An input query is answered by comparing its address with the prefix boundaries and either returning the appropriate bit of x or a fixed bit. At each completed simulated transition, the control and work tapes equal those of the fixed transducer on the literal string y , and the stored virtual input coordinate equals its input-head position. The callback returns exactly the symbol at that coordinate, so induction preserves this simulation invariant even though m depends on q . The simulated input-head address and all comparisons use $O(\log m)$ bits; the string y is never stored. The simulated transducer returns a dyadic rational $z \in [0, 1]$ satisfying

$$|z - p_{\mathcal{M}}(x)| \leq (m + 2)^{-1} < 2^{-q-4}.$$

Its numerator and exponent have $O(\log m)$ bits and can be retained in work space while its finite output is simulated.

Compute by exact dyadic arithmetic

$$a = \left\lfloor 2^{q+2} z + \frac{1}{2} \right\rfloor.$$

Then $0 \leq a \leq 2^{q+2}$, and rounding gives

$$\left| a 2^{-(q+2)} - p_{\mathcal{M}}(x) \right| \leq 2^{-q-3} + 2^{-q-4} < 2^{-q}.$$

Write the binary numerator a to the output tape. It has at most $q + 3$ bits and represents the denominator specified in the theorem. The simulator state, virtual input address, actual input counters, captured dyadic answer, and rounding operands occupy a fixed number of records of total size

$$O(\log m) = O(\log(n + 2) + q).$$

The simulation and the synthesis of each input bit take polynomial time in m , so the total time is $m^{O(1)} = (n + 2)^{O(1)} 2^{O(q)}$. The padded machine and its exponent-one approximator were fixed before reading q . Thus their constants are independent of the requested accuracy. Throughout their simulation, the original size convention is $B = \Theta(\log(m + 2))$, exactly as required by the estimator and controller theorems. $\square$

4 Sampling retained vertex sets

The hierarchy will use random retained sets to detect columns with many substantial entries. The table construction will use the same sets to represent matrix entries. Section 7 later constructs averaging walks that preserve specified retention conditions.

The environment consists of the independent sampling channels below together with a finite string of independent uniform auxiliary bits. The length and uses of that string will be specified when needed.

We use a fixed number of independent sampling channels, including channels denoted X, Y, Z . Extra channels, when needed, are also a fixed finite number. A channel is a uniformly random matrix

$$A = (a_0, F) \in \mathbb{F}_P^{H \times h_0}$$

conditioned on the last $r = h_0 - 1$ columns F being linearly independent, so F has full column rank. The notation $\mathbb{F}_P$ denotes the field with P elements, and P is prime. It is chosen larger than every identifier and sufficiently large for all rate comparisons below. For a numerical identifier y , set

$$v(y) = (1, y, \dots, y^{h_0-1})^\top, \quad a(y) = Av(y) \in \mathbb{F}_P^H.$$

Field elements in a retention test use their representatives in $\{0, \dots, P-1\}$. At nominal rate b_k , let

$$L_k = \lceil P2^{-k} \rceil, \quad \pi_{b_k} = (L_k/P)^H.$$

The identifier is retained if all coordinates of $a(y)$ lie in $\{0, \dots, L_k-1\}$. Write $X_a(y)$ for the indicator on channel X at rate a , and similarly for other channels. The same notation permits a specified fixed suffix in the channel argument. Retention boxes are nested. All rates in this section belong to the grid $b_k = 2^{-Hk}$, and the admissible range satisfies $P2^{-k} \geq 1$. The choices of H, h_0 and the identifier range are those of Section 2.

Lemma 4.1 (Retention probabilities and conditional bounds). *For the conditioned channel law above, the following bounds hold uniformly over every admissible identifier and rate range. Their positive constants depend only on H, h_0 .*

1. At rate a , each identifier has retention probability π_a , with $c_\pi a \leq \pi_a \leq C_\pi a$.
2. For distinct identifiers x, y and admissible rates u, a ,

$$\Pr[X_a(y) = 1 \mid X_u(x) = 1] \leq C_{\text{pair}} a.$$

3. For a deterministic set A_0 of $q > 0$ distinct identifiers,

$$\Pr \left[\sum_{y \in A_0} X_a(y) < q\pi_a/2 \right] \leq \frac{C_{\text{var}}}{q\pi_a}.$$

4. For $\alpha, \beta \in \mathbb{F}_P$ with $\alpha \neq 0$, replacing all arguments by $y \mapsto \alpha y + \beta$ preserves their joint evaluation law. This includes appending a common fixed radix suffix when the radix is nonzero modulo P .

Proof. The probability q_{fr} that an unrestricted H -by- r matrix has full column rank is

$$q_{\text{fr}} = \prod_{j=0}^{r-1} (1 - P^{j-H}) \geq c_{\text{fr}} > 0.$$

For fixed positive-degree coefficients, a_0 is uniform, so any one evaluation is uniform in $\mathbb{F}_P^H$ and independent of the full-column-rank event. This proves the exact marginal retention probability. In the ranges used, take $P2^{-k} \geq 1$. Then $P2^{-k} \leq L_k \leq 2P2^{-k}$, proving comparability. The prime exponent can in fact make all these sides much larger than one.

Before conditioning on full column rank, two evaluations at distinct field points are independent uniform vectors: the two evaluation functionals have rank two since $h_0 \geq 2$. Consequently

$$\Pr\{X_a(y) = 1, X_u(x) = 1 \mid \text{rank}(F) = r\} \leq \frac{\pi_a \pi_u}{q_{\text{fr}}}.$$

The first marginal after conditioning is still π_u , which proves the second assertion.

For the third assertion, in the unrestricted distribution the hit count has mean $q\pi_a$ and variance at most $q\pi_a$, by pairwise independence. Chebyshev's inequality bounds the stated lower-tail event by $4/(q\pi_a)$. Conditioning on the full-column-rank event increases this probability by at most $1/q_{\text{fr}}$.

To check the affine change for the whole channel, write $a(y) = \sum_{k=0}^r a_k y^k$, with vector coefficients $a_k \in \mathbb{F}_P^H$. For arbitrary $\alpha \neq 0$ and β , the transformed coefficients are

$$a'_j = \alpha^j \sum_{k=j}^r \binom{k}{j} \beta^{k-j} a_k \quad (0 \leq j \leq r), \quad \sum_{j=0}^r a'_j y^j = a(\alpha y + \beta). \quad (4.1)$$

The positive-degree block is multiplied on the right by a triangular matrix with diagonal $\alpha, \dots, \alpha^r$, so it retains rank r . The constant column changes to $a'_0 = a_0 + \sum_{k \geq 1} \beta^k a_k$. For each positive-degree block this is a translation of its independent uniform constant column. The full transformation is a bijection of the rank-conditioned coefficient arrays and hence preserves their uniform law. This verifies the joint evaluation law for every collection of arguments, including the interaction between the shift and the constant column. Appending a fixed suffix of length j and numerical value c uses $\alpha = D_0^j$ and $\beta = c$; $P > D_0$ makes α invertible. $\square$

5 The exact correction and copy hierarchy

At each stage we reduce the transition entries and transfer their contribution to a reward. Given a transition matrix C and a nonnegative correction matrix E indexed by the same finite vertex set, put

$$D = C - E + EC.$$

Here I is the identity matrix on that vertex set. Then $I - D = (I + E)(I - C)$. For column vectors W, p on the same set, a relation $W = (I - C)p$ becomes $(I + E)W = (I - D)p$: the same vector p is represented by the new transition and the updated reward. The correction must make D nonnegative and keep its row sums at most one. It must also reduce entries sufficiently fast that, after a controlled increase in the number of active vertices, the remaining transition contribution becomes small. The construction uses only the model reduction, with no one-sided acceptance promise.

At one stage the input is a finite totally ordered set V , a rate $0 < f \leq 1$ on the grid of Section 4, and a nonnegative matrix C with

$$C(x, y) = 0 \quad \text{unless } x < y, \quad C\mathbf{1} \leq \mathbf{1}, \quad C(x, y) \leq f. \quad (5.1)$$

All matrix and vector inequalities are entrywise. Write

$$\mathcal{R}(C) = \{x : \text{some } C(x, y) > 0\}, \quad \mathcal{T}(C) = \{y : \text{some } C(x, y) > 0\}, \quad \mathcal{A}(C) = \mathcal{R}(C) \cup \mathcal{T}(C).$$

Put $q = 2^H$. The stage will reduce the entry bound from f to f/q and increase $|\mathcal{A}(C)|$ by at most a factor of two. Its usable output is a nonnegative correction E , its substochastic residual $D = C - E + EC$, and a transition C^+ on $V^+ = V \times \{0, \dots, D_0 - 1\}$ for one fixed radix D_0 . Writing $x_i = (x, i)$, the row factors satisfy $0 \leq r_i(x) \leq 1$ and $r_0(x) = 1$. The weighted parent projection is $P^r(x_i, z) = r_i(x) \mathbf{1}_{\{z=x\}}$. The matrix C^+ is nonnegative and strictly upper triangular in the parent-first order. The conclusions used by the next stage are

$$C^+ \mathbf{1} \leq \mathbf{1}, \quad C^+(x_i, y_j) \leq f/q, \quad |\mathcal{A}(C^+)| \leq 2|\mathcal{A}(C)|, \quad C^+ P^r = P^r D.$$

By Lemma 5.4 and Proposition 5.5, if $W = (I - C)p$, then

$$P^r(I + E)W = P^r(I - D)p = (I - C^+)P^r p.$$

Thus the new reward represents the projected continuation vector. Iterating this identity with the entry and active-count bounds gives the reward error bound used in Section 3 (Section 5.5). The correction bounds hold for every fixed column multiplier. A finite detector calculation selects the multiplier for the hierarchy, and copies divide the entries in columns where correction is not fully enabled.

5.1 Correction with a fixed column multiplier

Fix rational numbers $0 < \theta_1 < \theta_2 < q^{-1}/32$ and a nondecreasing Lipschitz function $Q : [0, \infty) \rightarrow [0, \infty)$ satisfying

$$(t - \theta_2)_+ \leq Q(t) \leq (t - \theta_1)_+. \quad (5.2)$$

Here $r_+ = \max(r, 0)$ denotes the positive part. The eighth root of Q is required to be Lipschitz on bounded intervals. Here and below the scalar cutoffs can be chosen piecewise polynomial with rational coefficients. To see this explicitly, let β be the normalized integral of $t^7(1 - t)^7$ on $[0, 1]$, extended by zero to the left and one to the right. Both $\beta^{1/8}$ and $(1 - \beta)^{1/8}$ are Lipschitz: near their vanishing endpoints the corresponding functions vanish to order eight, and away from those endpoints the assertion follows by differentiation. For example,

$$Q(t) = (t - \theta_1)_+ \beta \left(\frac{t - \theta_1}{\theta_2 - \theta_1} \right)$$

has all the stated properties. Near θ_1 its eighth root is of order $(t - \theta_1)_+^{9/8}$; elsewhere its root has bounded derivative on every bounded interval. Let L_Q be a fixed Lipschitz constant for Q . For real arguments, extend Q by $Q(t) = 0$ when $t < 0$. This extension preserves Lipschitz continuity of Q and of its eighth root and permits evaluation on real approximations. The exact hierarchy itself uses only nonnegative arguments.

Fix a column multiplier $g : V \rightarrow [0, 1]$. Starting from zero, define

$$E_0 = 0, \quad G'_i = C + E_{i-1}C, \quad E_i(x, y) = fQ(G'_i(x, y)/f)g(y) \quad (i \geq 1). \quad (5.3)$$

The cutoff leaves a positive residual at each corrected entry. Keeping g fixed makes the update map monotone. A fixed number of iterations suffices for every such multiplier, independently of the size of V : the fixed point bounds the truncation error of the finite iteration used by the hierarchy.

Lemma 5.1 (Fixed point and factorial truncation). *Let C, f, V satisfy Equation (5.1), and let Q be the nondecreasing Lipschitz cutoff fixed above, including Equation (5.2). For every fixed $g : V \rightarrow [0, 1]$, the equation*

$$E_*(x, y) = fQ((C + E_*C)(x, y)/f)g(y) \quad (5.4)$$

has a unique nonnegative strictly upper triangular solution E_* . The iterates in Equation (5.3) are nonnegative and satisfy, for every integer $k \geq 0$ and $x, y \in V$,

$$0 \leq E_*(x, y) - E_k(x, y) \leq f e^{1/\theta_1} \frac{(L_Q/\theta_1)^k}{k!}. \quad (5.5)$$

In particular, one fixed positive integer k_* makes the normalized upper bound at most $q^{-1}/32$, independently of C, f, V, g .

Proof. Use the given total order on V . For a fixed row x , the entry $(E_*C)(x, y)$ involves only $E_*(x, z)$ with $z < y$. Equation (5.4) therefore defines the row successively, in increasing column order. Columns at or before x receive zero. This proves existence and uniqueness in the stated class, without a contraction hypothesis. Monotonicity of the update map proves $E_i \leq E_*$ and the nonnegativity of the differences.

Set $T = C - E_* + E_*C$. If $a = (C + E_*C)(x, y)$, then Equation (5.2) implies

$$0 \leq E_*(x, y) \leq (a - \theta_1 f)_+.$$

Thus $T \geq 0$, and at every positive position of E_* one has $T(x, y) \geq \theta_1 f$. In a column with $g(y) = 1$, the lower bound on Q also gives $T(x, y) \leq \theta_2 f$. This is the fixed-point entry bound transferred to the finite residual in Lemma 5.4. Also

$$T\mathbf{1} = C\mathbf{1} - E_*(\mathbf{1} - C\mathbf{1}) \leq C\mathbf{1} \leq \mathbf{1}. \quad (5.6)$$

Consequently, for each fixed row x , its set $A_x = \{y : E_*(x, y) > 0\}$ has cardinality $m_x \leq 1/(\theta_1 f)$.

Unroll the inequality $E_* \leq C + E_*C$ along this row's support. Each resulting term for $E_*(x, y)$ is a product of entries of C along a sequence

$$x < z_1 < \cdots < z_j < y, \quad z_1, \dots, z_j \in A_x.$$

Every substitution keeps the source coordinate equal to x : the recursive factor is $E_*(x, z)$, while only its terminal coordinate changes. All intermediate vertices therefore belong to the same set A_x . Strict ordering makes a choice of intermediate vertices specify at most one such sequence. There are at most $\binom{m_x}{j}$ such sequences, and each product is at most f^{j+1} . The expansion terminates because every sequence is strictly increasing. Therefore

$$E_*(x, y) \leq f \sum_{j=0}^{m_x} \binom{m_x}{j} f^j = f(1+f)^{m_x} \leq f e^{1/\theta_1}. \quad (5.7)$$

Only the support of the single fixed row x was used here; the number of vertices in the whole graph does not enter the exponent.

Let $\Delta_i = E_* - E_i$. These matrices are nonnegative, and their row- x supports are contained in A_x . The Lipschitz bound gives

$$\Delta_i(x, y) \leq L_Q \sum_{z < y} \Delta_{i-1}(x, z) C(z, y).$$

Outside A_x the left side is zero, so when this inequality is unrolled it suffices to use intermediate vertices from A_x at every step. After k steps there are at most $\binom{m_x}{k}$ strictly increasing sequences of intermediate vertices. Their initial differences are at most $f e^{1/\theta_1}$ by Equation (5.7), and their k transition factors are at most f^k . Thus

$$\Delta_k(x, y) \leq f e^{1/\theta_1} L_Q^k f^k \binom{m_x}{k} \leq f e^{1/\theta_1} \frac{(L_Q/\theta_1)^k}{k!}.$$

For $k > m_x$ the first upper bound is zero. This proves Equation (5.5). The factorial eventually dominates the fixed exponential factor, giving the stated fixed choice of k_* . $\square$

The iteration count k_* is now fixed independently of the detectors. We next choose g so that correction is supported on columns with only boundedly many substantial entries. A preliminary iteration selects this multiplier; the actual recurrence then uses that one multiplier for all k_* steps.

5.2 A preliminary iteration chooses the column multiplier

All expectations defining the detectors are true expectations; an estimate is never used to define its own target.

For the detectors we use independent channels X, Y from Section 4. Denote their common retention probability at rate f by π_f . The consequences of Lemma 4.1 needed here can be written, with fixed positive constants, as

$$c_\pi f \leq \pi_f \leq C_\pi f, \quad \Pr\{Y_f(w) = 1 \mid Y_f(y) = 1\} \leq C_{\text{pair}} f \quad (w \neq y), \quad (5.8)$$

$$\Pr\left\{\sum_{x \in A} X_f(x) < \frac{1}{2}|A|\pi_f\right\} \leq \frac{C_{\text{var}}}{|A|\pi_f} \quad (\emptyset \neq A \subseteq V \text{ deterministic}). \quad (5.9)$$

These constants are uniform in the stage, the admissible moduli, and common fixed padding suffixes. No independence between different rows of a detector trial will be assumed.

All threshold choices and ranks below are fixed constants. Let K, J be positive integers, to be chosen by the bounds below. The preliminary iteration uses the iteration count from Lemma 5.1. Choose

$$0 < \delta_0 < \delta_1 < \theta_1/100, \quad 0 < \eta_- < \eta_+ < \delta_0/3.$$

If other row scores require still smaller detector thresholds, make those choices before fixing the ranks. Let ϕ be a nondecreasing cutoff which is zero through δ_0 and one from δ_1 on. Let ρ be a nonincreasing cutoff which is one through η_- and zero from η_+ on. Finally, let $\gamma : [0, 1] \rightarrow [0, 1]$ be nonincreasing, one on $[0, 1/2]$, and zero on $[4/5, 1]$. These functions and their complements, when used, have the Lipschitz continuity of eighth roots just described.

For a finite multiset of nonnegative numbers, rk_j means its j -th largest member, with enough additional zeros that this is always defined. Starting from $\hat{E}_0 = 0$, perform k_* preliminary steps. At step i , first set

$$G_i = C + \hat{E}_{i-1}C. \quad (5.10)$$

For a draw of the terminal channel, set

$$R_i(x) = \rho(\text{rk}_K\{G_i(x, w)/f : Y_f(w) = 1\}). \quad (5.11)$$

Define the deterministic column number

$$U_i(y) = \mathbb{E}[\text{rk}_J\{\phi(G_i(x, y)/f)R_i(x) : X_f(x) = 1\} \mid Y_f(y) = 1]. \quad (5.12)$$

Then put

$$\hat{g}_i(y) = \gamma\left(\max_{1 \leq r \leq i} U_r(y)\right), \quad \hat{E}_i(x, y) = fQ(G_i(x, y)/f)\hat{g}_i(y). \quad (5.13)$$

The order of these definitions is essential: G_i is already deterministic when its detector is formed, and U_i is defined before $\hat{E}_i$. The running maximum makes $\hat{g}_i$ nonincreasing. Its final value is therefore no larger than any earlier preliminary multiplier; monotonicity of Q will let us bound the actual corrections by the preliminary corrections.

After the preliminary iteration, set $g = \hat{g}_{k_*}$ and specialize the fixed-multiplier recurrence of Equation (5.3) to the actual correction

$$E_0 = 0, \quad G'_i = C + E_{i-1}C, \quad E_i(x, y) = fQ(G'_i(x, y)/f)g(y) \quad (1 \leq i \leq k_*). \quad (5.14)$$

Write

$$E = E_{k_*}, \quad D = C - E + EC. \quad (5.15)$$

Lemma 5.2 (Bounds for the preliminary iteration). *For every preliminary and actual step $1 \leq i \leq k_*$, the four matrices $G_i, \hat{E}_i, G'_i, E_i$ are nonnegative and strictly upper triangular, with row sums at most i and entries at most if . They satisfy*

$$E_{i-1} \leq E_i \leq \hat{E}_i, \quad G'_i \leq G_i.$$

Each has row support contained in $\mathcal{R}(C)$ and column support contained in $\mathcal{T}(C)$.

Proof. Equation (5.2) gives $Q(0) = 0$ and $0 \leq Q(t) \leq t$ for $t \geq 0$. Consequently $\hat{E}_i \leq G_i$. If $\hat{E}_{i-1}\mathbf{1} \leq (i-1)\mathbf{1}$, then

$$G_i\mathbf{1} = C\mathbf{1} + \hat{E}_{i-1}C\mathbf{1} \leq \mathbf{1} + \hat{E}_{i-1}\mathbf{1} \leq i\mathbf{1}.$$

For an individual entry,

$$G_i(x, y) \leq f + f \sum_z \hat{E}_{i-1}(x, z) \leq if.$$

This proves the preliminary bounds by induction.

The sequence $\hat{g}_i$ is nonincreasing, so $g \leq \hat{g}_i$ for every i . Induction in Equations (5.10)–(5.14), using monotonicity of Q , proves $E_i \leq \hat{E}_i$ and $G'_i \leq G_i$. The map $F \mapsto fQ((C + FC)/f)g$, applied entrywise after forming the matrix product, is monotone. Its iteration from zero is therefore nondecreasing. The actual bounds follow from the preliminary bounds.

A product of strictly upper triangular matrices is strictly upper triangular. A zero row of C remains a zero row in all these formulas, by induction from the zero correction. A zero column of C is a zero column of every product of the form FC and hence of every input and correction. This proves the remaining assertions. $\square$

Lemma 5.3 (The detector bounds overloaded columns). *Choose an integer $K \geq 160(1 + C_{\text{pair}}k_*/\eta_-)$, also large enough for the other finitely many row-gate estimates. For any fixed positive integer J , take*

$$R_* \geq c_\pi^{-1} \max(4J, 40C_{\text{var}}). \quad (5.16)$$

For each $1 \leq i \leq k_*$ and $y \in V$, the detector satisfies

$$\#\{x : G_i(x, y) \geq \delta_1 f\} > R_*/f \implies U_i(y) \geq 19/20,$$

$$U_i(y) > 1/4 \implies \#\{x : G_i(x, y) > \delta_0 f\} > \frac{J}{4C_\pi f}.$$

Consequently, $\hat{g}_i(y) > 0$ implies $\#\{x : G_i(x, y) \geq \delta_1 f\} \leq R_*/f$, and $g(y) > 0$ implies $\#\{x : G'_i(x, y) \geq \delta_1 f\} \leq R_*/f$ for every actual step.

Proof. Put $D_{\text{row}} = 1 + C_{\text{pair}}k_*/\eta_-$. Fix i, y and condition on $Y_f(y) = 1$. In a fixed row, the set $\{w : G_i(x, w) > \eta_- f\}$ has cardinality at most $k_*/(\eta_- f)$ by Lemma 5.2. Its expected number of retained members is at most D_{row} : the specified terminal y , if in this set, contributes at most one, and Equation (5.8) applies to every other member. If the row gate is not full, at least K such members have been retained. Markov's inequality therefore gives

$$\Pr\{R_i(x) < 1 \mid Y_f(y) = 1\} \leq D_{\text{row}}/K. \quad (5.17)$$

Let $A = \{x : G_i(x, y) \geq \delta_1 f\}$ and $n = |A|$. Let N_A count its retained starts, and let L_A count those retained starts whose row gate is not full. Independence of the two channels and Equation (5.17) imply

$$\mathbb{E}[L_A \mid Y_f(y) = 1] \leq n\pi_f D_{\text{row}}/K.$$

Thus

$$\Pr\{L_A > n\pi_f/4 \mid Y_f(y) = 1\} \leq 1/40.$$

The law of the start channel is unchanged by the conditioning, so Equation (5.9) gives

$$\Pr\{N_A < n\pi_f/2 \mid Y_f(y) = 1\} \leq C_{\text{var}}/(n\pi_f).$$

If $n > R_*/f$, Equation (5.16) ensures $n\pi_f \geq \max(4J, 40C_{\text{var}})$. Outside an event of probability at most $1/20$, there are at least J retained high rows with full gates. Each has score exactly one, so the trial's J -th largest score is one. All trial scores lie in $[0, 1]$, proving the first assertion. This argument used no independence among the row-gate failures.

For the second assertion let $A_0 = \{x : G_i(x, y) > \delta_0 f\}$, and let N_0 be its number of retained starts. A positive score must come from A_0 . Pointwise the trial is at most $\mathbf{1}_{\{N_0 \geq J\}}$, and hence at most N_0/J . Its conditional expectation is at most $|A_0|\pi_f/J \leq |A_0|C_\pi f/J$. The asserted lower bound on $|A_0|$ follows.

Finally, a positive $\widehat{g}_i(y)$ implies $U_i(y) < 4/5$, so the first assertion excludes a high-entry count greater than R_*/f . If $g(y) > 0$, all preliminary multipliers at y are positive. Apply the same bound to every G_i and then use $G'_i \leq G_i$ from Lemma 5.2. $\square$

5.3 The finite residual

The factorial estimate fixes the iteration count for the correction $E = E_{k_*}$. Its residual is substochastic and has small entries on columns where correction is fully enabled. It also dominates its unsigned defining terms up to a fixed factor, allowing their estimation errors to be bounded term by term.

Lemma 5.4 (Residual bounds and unsigned-term domination). *The finite residual $D = C - E + EC$, with $E = E_{k_*}$ and k_* chosen in Lemma 5.1, is nonnegative, strictly upper triangular, and row-substochastic. It obeys*

$$D(x, y) \leq (k_* + 1)f, \quad g(y) = 1 \implies D(x, y) \leq (\theta_2 + q^{-1}/32)f < q^{-1}f/16, \quad (5.18)$$

and the entrywise comparison

$$D \geq \frac{\theta_1}{2k_*}(C + E + EC). \quad (5.19)$$

Its row and column supports are contained in $\mathcal{R}(C)$ and $\mathcal{T}(C)$, respectively.

Proof. Set $A = G'_{k_*}$ and $\delta = (E - E_{k_*-1})C \geq 0$. Then

$$D = A - E + \delta, \quad A - E \geq \min(A, \theta_1 f). \quad (5.20)$$

Here and in the minimum the scalar $\theta_1 f$ denotes the constant entrywise bound. The second inequality follows from $E \leq (A - \theta_1 f)_+$. It proves $D \geq 0$. Since $C\mathbf{1} \leq \mathbf{1}$ and $E \geq 0$,

$$D\mathbf{1} = C\mathbf{1} - E(\mathbf{1} - C\mathbf{1}) \leq C\mathbf{1} \leq \mathbf{1}.$$

Also $D \leq C + EC$, whose entries are at most $f + f \sum_z E(x, z) \leq (k_* + 1)f$ by Lemma 5.2. Strict upper triangularity and the support claims follow from the same formulas.

Because $A(x, y) \leq k_* f$, Equation (5.20) gives $A - E \geq (\theta_1/k_*)A$. On the other hand,

$$C + E + EC = A + E + \delta \leq 2A + \delta.$$

Since $\theta_1/k_* < 1$, these two inequalities prove Equation (5.19).

For the good-column bound use the fixed point with this same multiplier g , and set $\Delta = E_* - E$. With $T = C - E_* + E_*C$,

$$D = T + \Delta - \Delta C \leq T + \Delta.$$

If $g(y) = 1$, the lower bound in Equation (5.2) gives $T(x, y) \leq \theta_2 f$. Equation (5.5) gives $\Delta(x, y) \leq q^{-1}f/32$. This proves Equation (5.18). $\square$

The domination in Equation (5.19) is also an estimation bound. It lets us compare the three unsigned contributions C , E , and EC separately against their estimates, measuring each error relative to the same nonnegative target D . Their signs are then combined by a triangle inequality in Lemma 8.3. The estimated raw arrays may have negative entries; positivity here is a property of the exact target.

For the nonnegative input formulas $G_i = C + \widehat{E}_{i-1}C$ and $G'_i = C + E_{i-1}C$, each summand is bounded by the input itself.

5.4 Copies reduce density while controlling active vertices

The residual already has small entries in columns with $g(y) = 1$. We distribute the remaining large entries among copies of their destinations. The detector incidence bound will limit the number of parents with active extra copies, and a weighted projection will relate the new transition to the residual.

Choose an integer $D_0 > (k_* + 1)q$, and let $\tau_0 = q^{-1}/8$, $\tau_1 = q^{-1}/4$. Choose a rising cutoff χ which is zero through τ_0 and one from τ_1 on. For $0 \leq j < D_0$ set

$$s_0(t) = 1 - (1 - D_0^{-1})\chi(t), \quad s_j(t) = D_0^{-1}\chi(t) \quad (j > 0). \quad (5.21)$$

They are nonnegative, sum to one, and have Lipschitz eighth roots on bounded ranges. Let κ be a rising cutoff, zero through $1/4$ and one from $1/2$ on, and put

$$r_0(x) = 1, \quad r_j(x) = \kappa\left(\max_{1 \leq r \leq k_*} U_r(x)\right) \quad (j > 0). \quad (5.22)$$

On $V^+ = V \times \{0, \dots, D_0 - 1\}$ write $x_i = (x, i)$ and set

$$C^+(x_i, y_j) = r_i(x)D(x, y)s_j(D(x, y)/f). \quad (5.23)$$

Order the copies first by their parent and then by their copy symbol.

Proposition 5.5 (One-stage density and active-count reduction). *With the routing fractions and row factors in Equations (5.21)–(5.23), choose*

$$J \geq \frac{2C_\pi(D_0 - 1)k_*(k_* + 1)}{\delta_0} \quad (5.24)$$

and R_* as in Equation (5.16). Then C^+ is strictly upper triangular and row-substochastic on the specified ordered copy set V^+ , and

$$C^+(x_i, y_j) \leq f/q, \quad |\mathcal{A}(C^+)| \leq 2|\mathcal{A}(C)|.$$

The weighted parent projection $P^r \in \mathbb{R}^{V^+ \times V}$,

$$P^r(x_i, z) = r_i(x)\mathbf{1}_{\{z=x\}}, \quad (5.25)$$

intertwines the copy transition and this residual:

$$C^+P^r = P^rD. \quad (5.26)$$

Proof. The row sum at x_i is $r_i(x) \sum_y D(x, y) \leq 1$. Strict forwardness follows from that of D . Put $t = D(x, y)/f$. If $t < \tau_1$, then every routed entry is at most $ft < f/q$. If $t \geq \tau_1$, all fractions equal $1/D_0$, so every entry is at most $(k_* + 1)f/D_0 < f/q$. This proves the density bound.

If a nonprimary fraction $s_j(D(x, y)/f)$ is positive on a positive entry, then $D(x, y)/f > \tau_0 > \theta_2 + q^{-1}/32$. Lemma 5.4 therefore implies $g(y) < 1$. Since γ equals one through $1/2$, this gives $\max_{1 \leq r \leq k_*} U_r(y) > 1/2$, and hence $r_j(y) = 1$. The primary factor is always one. Thus for every positive entry of D ,

$$\sum_{j=0}^{D_0-1} s_j(D(x, y)/f) r_j(y) = 1. \quad (5.27)$$

For a zero entry no identity of its fractions is needed. For an arbitrary child row x_i and parent column y , matrix multiplication gives the required relation in its two different index sets:

$$(C^+ P^r)(x_i, y) = r_i(x) D(x, y) \sum_j s_j(D(x, y)/f) r_j(y) = r_i(x) D(x, y) = (P^r D)(x_i, y).$$

The destination factors $r_j(y)$ are the ones supplied by the detector implication above; the source factor $r_i(x)$ stays outside the sum. This proves Equation (5.26).

It remains to count active vertices. Put

$$\mathcal{O} = \{y : \max_{1 \leq r \leq k_*} U_r(y) > 1/4\}, \quad n' = |\mathcal{A}(C)|.$$

Every $y \in \mathcal{O}$ has, for some preliminary step i , more than $J/(4C_\pi f)$ entries with $G_i(x, y) > \delta_0 f$, by Lemma 5.3. Counting these incidences over all steps and all active rows, and using Lemma 5.2, gives

$$\begin{aligned} |\mathcal{O}| \frac{J}{4C_\pi f} &\leq \sum_{i=1}^{k_*} \#\{(x, y) : G_i(x, y) > \delta_0 f\} \\ &\leq \frac{n'}{\delta_0 f} \sum_{i=1}^{k_*} i = \frac{n' k_* (k_* + 1)}{2\delta_0 f}. \end{aligned}$$

The assertion is also valid when $n' = 0$, since then all matrices and detectors vanish. Thus

$$|\mathcal{O}| \leq \frac{2C_\pi n' k_* (k_* + 1)}{\delta_0 J}.$$

Every active child has an active parent, by the support conclusion of Lemma 5.4. A nonprimary child active as a row has $r_i(x) > 0$, so its parent belongs to $\mathcal{O}$. A nonprimary child active as a column has a positive extra routing fraction and, as shown above, its parent has $\max_{1 \leq r \leq k_*} U_r > 1/2$, so again belongs to $\mathcal{O}$. Therefore

$$|\mathcal{A}(C^+)| \leq n' + (D_0 - 1)|\mathcal{O}| \leq 2n',$$

where the last inequality is Equation (5.24). $\square$

Remark 5.6 (Order of the one-stage constants). The bounds used to select k_* in Lemma 5.1 hold for every multiplier in $[0, 1]$. Thus $\theta_1, \theta_2, Q, k_*$ and D_0 can be fixed before any detector rank. Next fix all finitely many correction, detector, and access-score threshold ladders, taking δ_0, δ_1 below the correction support thresholds as required. Choose K large enough for the corresponding row estimates. Choose J by Equation (5.24), and then R_* by Equation (5.16). An incoming rank used later can be chosen after R_* : its full-gate probability is controlled by the column bound in Lemma 5.3, independently of the auxiliary row gates. There is therefore no requirement to enlarge K again in response to that incoming rank. Subsequent estimator constants may depend on these already fixed ranks and thresholds.

5.5 Reward transport and the final deterministic target

The copy step controls transition entries and active vertices. We now iterate it and transport the terminal reward through the same weighted projections. The resulting reward will approximate the original acceptance probability, including probabilities strictly between zero and one.

Starting with $C_0 = S$ on V_0 , repeat the one-stage construction with

$$f_l = q^{-l}, \quad V_{l+1} = V_l \times \{0, \dots, D_0 - 1\}, \quad C_{l+1} = C_l^+.$$

Let E_l, D_l, P_l^r denote the correction, residual, and weighted parent projection at stage l . Let e be the base final-success reward and $p_0 = (I - S)^{-1}e \in [0, 1]^{V_0}$. Define $W_0 = e$ and, for $x_i = (x, i) \in V_{l+1}$, update the reward by

$$W_{l+1}(x_i) = r_i(x)(W_l(x) + (E_l W_l)(x)). \quad (5.28)$$

Here the row factor r_i is the one chosen at stage l . Write $\bar{x}_l$ for the copy of $x \in V_0$ obtained by appending l primary symbols.

Proposition 5.7 (Reward approximation). *The rewards defined by Equation (5.28) satisfy $W_l \in [0, 1]^{V_l}$ and, for every constructed stage l and $x \in V_0$,*

$$0 \leq p_0(x) - W_l(\bar{x}_l) \leq |V_0| 2^{-(H-1)l}. \quad (5.29)$$

In particular, an integer $L = O(B)$ makes this upper bound strictly less than $1/16$ at every base vertex. For $0 \leq l \leq L$, the full stage domains have size $2^{O(B)}$ and $O(B)$ -bit addresses.

Proof. We track the continuation vector through each weighted projection. The useful invariant is that the transported reward equals the transported continuation value minus its remaining transition contribution. This will give positivity from the reward construction and a small error from the residual row mass. Define matrices from V_l to V_0 by

$$P_0^c = I, \quad P_{l+1}^c = P_l^r P_l^c, \quad B_0 = I, \quad B_{l+1} = P_l^r (I + E_l) B_l. \quad (5.30)$$

The superscript c denotes the cumulative projection. The reward recursion gives $W_l = B_l e$. We first prove the identity

$$C_l P_l^c = P_l^c - B_l (I - S). \quad (5.31)$$

The assertion at $l = 0$ is $S = I - (I - S)$. The residual identity can be written as

$$D_l = I - (I + E_l)(I - C_l).$$

Using Equation (5.26) and then the inductive hypothesis,

$$\begin{aligned} C_{l+1} P_{l+1}^c &= P_l^r D_l P_l^c \\ &= P_l^r (P_l^c - (I + E_l)(P_l^c - C_l P_l^c)) \\ &= P_{l+1}^c - P_l^r (I + E_l) B_l (I - S) \\ &= P_{l+1}^c - B_{l+1} (I - S). \end{aligned}$$

This proves Equation (5.31).

Every B_l is nonnegative. Every P_l^r has at most one nonzero entry per row, of value in $[0, 1]$, and thus P_l^c is substochastic. Apply Equation (5.31) to p_0 , using $(I - S)p_0 = e$. This gives

$$0 \leq W_l = P_l^c p_0 - C_l P_l^c p_0 \leq 1. \quad (5.32)$$

The lower bound follows from $B_l e \geq 0$; its upper bound follows from $W_l \leq P_l^c p_0 \leq 1$. Equation (5.28) is Equation (5.30) applied to e .

By Proposition 5.5, induction gives

$$|\mathcal{A}(C_l)| \leq 2^l |\mathcal{A}(S)| \leq 2^l |V_0|, \quad C_l(x, y) \leq q^{-l}.$$

There are at most $|\mathcal{A}(C_l)|$ columns containing any positive entry, so each row sum of C_l is at most $|V_0| 2^l q^{-l} = |V_0| 2^{-(H-1)l}$. The all-primary projection has every row factor equal to one and hence $(P_l^c p_0)(\bar{x}_l) = p_0(x)$. The identity in Equation (5.32), together with $0 \leq P_l^c p_0 \leq 1$, now proves Equation (5.29).

Choose an integer L such that $(H-1)L > \log_2(16|V_0|)$. Since $\log_2 |V_0| = O(B)$, this is $O(B)$. If $p_0(x) = 0$, nonnegativity forces $W_L(\bar{x}_L) = 0$. If $p_0(x) \geq 1/2$, the strict error bound gives $W_L(\bar{x}_L) > 7/16$. Finally,

$$|V_l| = |V_0| D_0^l = 2^{O(B)} \quad (0 \leq l \leq L),$$

because D_0 is fixed. Parent projection deletes one radix symbol, and appending a specified symbol is affine in the numerical ID. Thus these domain changes preserve the address convention required by the sampler and by the later implementation. $\square$

The remainder of the proof will estimate the finite targets just constructed. In particular, it need only implement the k_* preliminary and k_* actual updates at a stage and the reward update in Equation (5.28); it does not evaluate E_* . The same error bound in Equation (5.29) gives any fixed inverse-polynomial accuracy by increasing the constant in $L = O(B)$. Section 3 makes that choice together with the statistical and numerical precisions.

6 Sample tables and their access operations

The exact hierarchy uses matrix entries, correction products, and column detector expectations. We represent each of these by sample-dependent tables: bins represent entries of the transitions and residuals, correction tables represent the preliminary and actual corrections, and score tables represent the detector trials. Each normalized table has an earlier preparatory table that supplies its denominator.

The true tables satisfy exact conditional-mean identities. Applying their row formulas to computed raw arrays preserves the support and access bounds even when those arrays are inaccurate, but need not preserve unbiased normalization. Errors in incoming gates and column multipliers are controlled by the assembly bounds below. Port enumeration and pairing remain implementation obligations.

Weighted pseudorandom generators give a related representation of expectations with few random bits and real weights [8, Section 2.2, Definition 2.5][5, Section 1.3]. The tables below have exact conditional-mean identities. Their estimates use a common environment, so the error analysis bounds products and normalization under that joint law.

6.1 Table domains and the normalization principle

Throughout this section $s = 8$. Fix a stage, write $f = b_l$, and let G be one of its nonnegative raw matrix inputs. The hierarchy gives absolute constants M_G, K_G such that

$$\sum_y G(x, y) \leq M_G, \quad G(x, y) \leq K_G f. \quad (6.1)$$

These bounds include preliminary and actual inputs and the residual D ; see Lemmas 5.2 and 5.4. For a correction with entries $F(x, y) = fQ(G(x, y)/f)g_0(y)$, where $g_0 : V \rightarrow [0, 1]$ is a fixed column multiplier, the additional input is

$$g_0(y) > 0 \implies \#\{x : G(x, y) \geq \delta_1 f\} \leq R_*/f, \quad (6.2)$$

from Lemma 5.3.

A row task has start rate $u \leq f$, end scale a , and, when needed, a usage rate $t \leq a$, all on the rate grid. Its domain is $X_u(x) = Y_a(y) = 1$, with distinct start and terminal channels. We use $a \leq K_f f$ for an absolute K_f , and use $a = f$ for corrections and detector scores. Here and in the estimation and compression sections, t denotes a sampling rate; the numerical clock t is introduced separately in Section 11. The true first raw signal is

$$d_y = G(x, y)/a. \quad (6.3)$$

An optional second raw signal is $\zeta \mathbf{1}_{\{Y_t(y)=1\}} q_y$, where $q_y = q(x, y) \in [0, 1]$ is a deterministic conditional mean and $\zeta > 0$ is a fixed sufficiently small constant. That mean will be specified separately for each normalized task.

All parameters of a table are common to the whole table: its stage and iteration, task type, channel roles, channel padding suffixes, rates u, a, t , and subsequently its budget and numerical convention. They do not depend on the row or column at which the table is queried.

Use rk_k for the k th largest value, with unlimited zero padding, as in Section 5. For two arrays on the same padded index set,

$$|\text{rk}_k(v) - \text{rk}_k(w)| \leq \sup_z |v_z - w_z|. \quad (6.4)$$

Indeed, if every coordinate changes by at most ϵ , at least k coordinates at or above the old k th value remain at or above that value minus ϵ ; apply this in both directions.

The normalization can be seen before choosing the row functions. Fix a pair (x, y) on its conditioning slice, a deterministic nonnegative value $k(d_y)$, and a random row gate $R \in [0, 1]$. Choose a preparatory factor h that is one wherever k is positive. If

$$q(x, y) = \mathbb{E}[h(d_y)R \mid X_u(x) = Y_t(y) = 1] \geq \frac{1}{2}$$

on that support, then $k(d_y)R/q(x, y)$ has conditional mean $k(d_y)$ there. Indeed, on the support of k , $h(d_y) = 1$ and $k(d_y)$ is constant under this conditioning, so

$$\mathbb{E} \left[\frac{k(d_y)R}{q(x, y)} \mid X_u(x) = Y_t(y) = 1 \right] = \frac{k(d_y)}{q(x, y)} \mathbb{E}[R \mid X_u(x) = Y_t(y) = 1] = k(d_y).$$

The denominator must therefore be the mean on this same usage slice. A clipped denominator keeps the formula bounded at every other position and for computed inputs. The constructions below arrange a uniform positive lower bound on this mean while the row gate limits the number of

nonzero retained entries. Corrections use an additional incoming gate, with the same normalization principle.

For real $\alpha < \beta$, let $\chi_{\alpha,\beta}$ be a fixed increasing switch, zero on $(-\infty, \alpha]$ and one on $[\beta, \infty)$. Choose this piecewise polynomial switch by rescaling the normalized integral of $z^{s-1}(1-z)^{s-1}$ on $[0, 1]$. Both $\chi_{\alpha,\beta}^{1/s}$ and $(1 - \chi_{\alpha,\beta})^{1/s}$ are Lipschitz: at either endpoint the vanishing function has a zero of order s , and on the interior ordinary differentiation applies. Products of bounded functions with Lipschitz s th roots again have Lipschitz s th roots.

Every scalar saturation below uses fixed min/max clipping,

$$\text{clip}_{[\alpha,\beta]}(v) = \min\{\beta, \max\{\alpha, v\}\},$$

which is 1-Lipschitz. We specify its bounds when defining the function. Functions initially defined for nonnegative first entries are extended by zero below zero; each already vanishes below a positive support threshold. These conventions apply only to evaluation of the row functions. Ranks and access tests always use the full prescribed raw arrays before scalar clipping.

Throughout, a bounded table value or bounded row support means a bound by an absolute constant independent of the stage, rates, and input size. Here is one explicit set of margins used below. Let $\vartheta > 0$ be a fixed lower support threshold of a main row function, and put $\rho = \text{rk}_K(v)$ for its first raw row v . Define

$$\begin{aligned} h_\vartheta(v) &= \chi_{\vartheta/2, 3\vartheta/4}(v), \\ R_\vartheta(\rho) &= 1 - \chi_{\vartheta/1000, \vartheta/500}(\rho), \\ I_\vartheta(v, \rho) &= \chi_{\vartheta/8, \vartheta/4}(v)(1 - \chi_{\vartheta/100, \vartheta/50}(\rho)). \end{aligned} \tag{6.5}$$

We take $\vartheta = \theta_1$ for corrections, $\vartheta = 1/2$ for bin tables, and $\vartheta = \delta_0$ for the detector score. For that detector we make the allowed hierarchy choices

$$\eta_- = \delta_0/1000, \quad \eta_+ = \delta_0/500, \quad \rho(t) = 1 - \chi_{\delta_0/1000, \delta_0/500}(t) = R_{\delta_0}(t).$$

Here ρ denotes the cutoff in the hierarchy, and the same switch χ is used in both occurrences. These choices satisfy $0 < \eta_- < \eta_+ < \delta_0/3$. Thus the detector row gate here is exactly R_i in Equation (5.11). The detector does not require h_ϑ . The hierarchy's choices may and do satisfy $\delta_1 < \theta_1/100$ and $0 < \delta_0 < \delta_1$. Thus the correction auxiliary score I_{θ_1} can be positive only when $G(x, y) > \delta_1 f$.

6.2 Entry bins and preparatory normalization

We first split a raw entry into contributions at the rate-grid scales. At each scale a row gate bounds the number of retained entries, and an exact preparatory conditional mean corrects the loss caused by that gate.

Recall $q = 2^H$, and let $\chi = \chi_{1/2, 1}$. For $w \geq 0$ define

$$\begin{aligned} \omega_0(w) &= \chi(w), \\ \omega_k(w) &= \chi(w/b_k) - \chi(w/b_{k-1}) \quad (k \geq 1). \end{aligned} \tag{6.6}$$

The top bin has no preceding switch. The weights are nonnegative and sum to one for $w > 0$, since their partial sums telescope to $\chi(w/b_k) \rightarrow 1$. Hence

$$w = \sum_{k \geq 0} w \omega_k(w). \tag{6.7}$$

At scale $a = b_k$, the bin in units of the table scale is

$$k_a(d) = d\omega_k(ad). \quad (6.8)$$

For $k \geq 1$ its support is contained in $1/2 < d < q$. Its two switch transition intervals are disjoint, and

$$\chi(d) - \chi(d/q) = \chi(d)(1 - \chi(d/q)),$$

which proves directly that its sth root is Lipschitz. For $k = 0$, fix $B_{\text{top}} > \max\{K_G, 1\}$ and evaluate the defining expression at $\text{clip}_{[0, B_{\text{top}}]}(d)$. This makes the top-bin function bounded with Lipschitz root. It preserves every true value, since $a = b_0 = 1$ and $G(x, y) \leq K_G f \leq K_G$. For $k \geq 1$, retain the zero extension outside the support $(1/2, q)$; no upper clipping changes a zero bin value. Bins with $a > 2K_G f$ vanish on the true input, so a fixed $K_f \geq 2K_G$ suffices for the permitted scale range.

For the routed residual bin use instead

$$k_a^{(j)}(d) = k_a(d)s_j(ad/f). \quad (6.9)$$

The routing fractions have bounded Lipschitz roots. Since $a/f \leq K_f$ and the bin is zero above a fixed argument range (or is top-bin clipped), these functions have uniformly bounded Lipschitz roots as well. The same zero-extension and clipping conventions apply to routed bins. Summing Equation (6.9) over scales gives the routed residual contribution exactly on every true input.

Choose $R_{1/2}$ and $h = h_{1/2}$ from Equation (6.5). For the true row set

$$\rho_x = \text{rk}_K\{G(x, z)/a : Y_a(z) = 1\}, \quad R = R_{1/2}(\rho_x).$$

Write

$$q_y = \mathbb{E}[h(d_y)R \mid X_u(x) = Y_t(y) = 1], \quad c(q) = \min\{1, \max\{1/2, q\}\}. \quad (6.10)$$

The preparatory table $h(d_y)R$ is defined on the broad domain Y_a and uses only the first raw signal. Averaging it on the narrower usage slice gives the deterministic denominator q_y in Equation (6.10); this denominator is zero when $d_y \leq 1/4$. The second raw signal carries its Y_t membership mask. Both domains belong to the interface: Y_a specifies where the preparation is available, while the fixed tuple's usage rate t specifies the conditional mean needed for normalization. Changing that slice can change the denominator.

Lemma 6.1 (Conditional mean of a bin table). *Fix the admissible task parameters $u, a = b_k, t$ above and the preparatory mean q_y in Equation (6.10). For a sufficiently large fixed K , define on $X_u(x) = Y_a(y) = 1$*

$$\mathcal{B}_a(x, y) = \mathbf{1}_{\{Y_t(y)=1\}} \frac{k_a(d_y)R}{c(q_y)}, \quad (6.11)$$

and extend it by zero elsewhere. Its values are bounded and each row has at most $K - 1$ nonzero entries. For every x, y whose conditioning slice is nonempty,

$$\mathbb{E}[a\mathcal{B}_a(x, y) \mid X_u(x) = Y_t(y) = 1] = G(x, y)\omega_k(G(x, y)).$$

Replacing k_a by $k_a^{(j)}$ defines $\mathcal{B}_a^{(j)}$ with the same bounds and with conditional mean multiplied by $s_j(G(x, y)/f)$. Apart from start membership, these true tables use only the terminal channel.

Proof. The row gate can fail to be full only if at least K sampled columns have $G(x, z)/a > r$, where $r = (1/2)/1000$ is fixed. There are at most $M_G/(ar)$ such deterministic columns. Conditional on $Y_t(y) = 1$, the specified column contributes at most one hit and each other column has Y_a -hit

probability at most Ca , by Lemma 4.1. Thus the expected count is at most $1 + CM_G/r$. Conditioning also on the independent source channel does not alter this bound. Markov's inequality makes the failure probability at most $1/8$ by a sufficiently large K .

Where $k_a(d_y) > 0$, we have $h(d_y) = 1$. Therefore $q_y = \mathbb{E}[R \mid Y_t(y) = 1] \in [7/8, 1]$ there, and the clipping does nothing. The conditional mean of Equation (6.11) is consequently $k_a(d_y)$. Outside that support both sides vanish. If a row output is positive, its first entry exceeds $1/2$ and its K th largest first entry is below $1/1000$. Hence there are at most $K - 1$ positive entries. Boundedness follows from the clipped denominator. The routed fraction is deterministic at truth and obeys the same argument. $\square$

6.3 Corrections with an incoming gate

Corrections also need a bound on incoming degree. Their deterministic column-support bound permits an additional incoming gate while keeping the probability that both gates are full bounded away from zero.

Consider $F = fQ(G/f)g_0(y)$ and put $a = f$, $\vartheta = \theta_1$. Fix $B_Q > \max\{K_G, \theta_2\}$. In the table formulas evaluate Q at $\text{clip}_{[0, B_Q]}(d)$. This bounded scalar function has a Lipschitz sth root on the real line and agrees with $Q(d)$ for every true argument $d = G(x, y)/f$; it is zero for $d < 0$. The hierarchy itself continues to use Q on its full domain. Let $R = R_{\theta_1}(\rho_x)$, $h = h_{\theta_1}$, and $I = I_{\theta_1}$ be the functions in Equation (6.5), where the true row rank is $\rho_x = \text{rk}_K\{G(x, z)/f : Y_f(z) = 1\}$. On a true or computed raw table, form $I(x, y)$ from that table's own row entry and own rank. Let K_c be a positive integer, to be chosen below. With $\chi_{1/4, 1/2}$ as above, define

$$T_y = 1 - \chi_{1/4, 1/2}(\text{rk}_{K_c}\{I(x, y) : X_u(x) = 1\}). \quad (6.12)$$

Thus $T_y = 1$ through rank $1/4$ and $T_y = 0$ from rank $1/2$. This rank is taken over the fixed retained source set before any incoming port filtering; there is no gate-dependent change of its index set. The true correction preparation is $h(d_y)RT_y$ on Y_a . Define its conditional mean by

$$q_y = \mathbb{E}[h(d_y)RT_y \mid X_u(x) = Y_t(y) = 1]. \quad (6.13)$$

The number q_y is deterministic; as before, its second raw signal carries the Y_t membership mask. Define the normalized true correction using the clipped expression everywhere:

$$\mathcal{F}(x, y) = \mathbf{1}_{\{Y_t(y)=1\}} \frac{Q(\text{clip}_{[0, B_Q]}(d_y))RT_y}{c(q_y)} g_0(y). \quad (6.14)$$

Here $q_y = 0$ whenever $d_y \leq \theta_1/2$. Thus both kinds of second signal are supported above a fixed positive first-signal threshold. Choose $\zeta \leq 10^{-6} \min\{\theta_1, \delta_0, 1/2\}$, decreasing it later if a fixed compression margin requires this. The second signals are then uniformly tiny on the first-signal threshold scale.

Lemma 6.2 (Conditional mean of a correction table). *For the correction $F = fQ(G/f)g_0$ satisfying Equation (6.2), choose K sufficiently large and then K_c sufficiently large. With the exact preparation, usage slice, and clipped denominator in Equations (6.13) and (6.14),*

$$\mathbb{E}[f\mathcal{F}(x, y) \mid X_u(x) = Y_t(y) = 1] = F(x, y) \quad (x, y \in V).$$

This identity includes columns with $g_0(y) = 0$. The true gates and denominator depend only on the role channels X, Y ; the multiplier g_0 is deterministic.

Proof. The row count argument in Lemma 6.1, with the correction thresholds, gives $\Pr(R \neq 1 \mid \text{the two hits}) \leq 1/8$. If $g_0(y) > 0$, every source with $I(x', y) > 0$ belongs to the deterministic set in Equation (6.2). Conditional on $X_u(x) = 1$, the number of sampled members of that set has expectation at most

$$1 + CuR_*/f \leq 1 + CR_*.$$

The bound remains valid after conditioning on the independent terminal hit; it ignores the auxiliary row gate and therefore needs no additional independence between rows. If fewer than K_c such sources are sampled, the K_c th auxiliary rank is zero and $T_y = 1$. Choose $K_c \geq 8(1 + CR_*)$ to make $\Pr(T_y \neq 1 \mid \text{the two hits}) \leq 1/8$. The union bound, without independence of the gates, gives $\mathbb{E}[RT_y \mid \text{the two hits}] \geq 3/4$.

Since $h(d_y) = 1$ on Q 's support, at positions with $Q(d_y) > 0$ and $g_0(y) > 0$ we have

$$q_y = \mathbb{E}[RT_y \mid X_u(x) = Y_t(y) = 1] \geq 3/4. \quad (6.15)$$

Thus $c(q_y) = q_y$ on these positions. No lower bound on this mean is needed in a column with $g_0(y) = 0$. The claimed conditional identity follows by division by this exact conditional mean. If $Q(d_y) = 0$ or $g_0(y) = 0$, both sides of the identity are zero. Even when $g_0(y) = 0$ and the unclipped mean vanishes, the clipped ratio $Q(\text{clip}_{[0, B_Q]}(d_y))R/c(q_y)$ remains bounded and well defined. This is the comparison target before multiplication by the incoming gate T_y and column multiplier. $\square$

6.4 Detector scores

The unnormalized detector table is

$$\mathcal{S}_i(x, y) = \phi(G_i(x, y)/f)R_{\delta_0}(\text{rk}_K\{G_i(x, w)/f : Y_f(w) = 1\}) \quad \text{on } X_f(x) = Y_f(y) = 1. \quad (6.16)$$

Its column trial is the J th largest such score over retained starts. By the identical cutoff choice, its conditional mean given $Y_f(y) = 1$ is exactly $U_i(y)$ from Equation (5.12). This operation ranges over sources x at the fixed terminal y . The bounded support of each row gives no uniform bound on the number of scores in that column, and the rank query requires no such bound.

6.5 Actual ports and estimated assembly

The conditional-mean identities now give the true contributions. A product estimate will query a correction at (x, z) and then a bin at (z, y) . Bounded outgoing degree limits the possible midpoints; bounded incoming degree is needed when errors at the new source z are summed over old sources x . We therefore specify access relations as well as nonzero table supports. The access relations may contain entries of weight zero.

Passing through one copy projection is partitioned by its fixed source and destination symbols. If an inherited suffix has length j and value c , appending copy symbol i before that suffix changes a parent identifier y to $D_0^{j+1}y + iD_0^j + c$. Equation (4.1) therefore applies with this full affine map. The source and terminal channels may use different fixed symbols; applying their coefficient bijections separately preserves their product law and all their specified retention rates. The task tuple records these fixed symbols and suffixes. By definition, neither a right factor's task parameters nor a column vector's task parameters retain an earlier source address. The environment supplied to the table is common to all its entries. The vertices x, y are arguments of that table, rather than additional parameters. Thus two paths that discover the same entry in that environment query the same value; the path used to discover it is not part of its mathematical definition.

We will apply these row functions to arrays represented by a finite list of exceptional values and one common value at every omitted endpoint. Call that common value the *default*. Adjoin arbitrarily many dummy coordinates with the same default, and include them in every rank; they are never matrix entries or ports. For the true row arrays, dummy values are zero. Section 9 constructs the finite-list representations. The elementary rank convention below lets us prove support and access bounds before choosing that construction.

Remark 6.3 (Ranks of arrays with a common default). Suppose a row array equals a common default outside a finite list, and arbitrarily many dummy coordinates have that same default. Its k th rank can be computed from the listed coordinates and k copies of the default, with the stipulated zero padding. Indeed, every omitted coordinate merely supplies another copy of a value already available with multiplicity at least k . Additional listed real coordinates equal to the default therefore do not change the rank. The same observation applies to a rank of entry sizes when each omitted entry has the same default vector.

This is an equality of ranks for one prescribed full array. The dummy coordinates are never matrix entries or ports. A common componentwise clamp of the entries and their default preserves this representation: an omitted position still equals the clamped default. No assertion that different calculations select identical lists or ports is needed.

Lemma 6.4 (Support margins and row access classes). *Fix $\vartheta > 0$ and the functions in Equation (6.5). For any real first-signal row, with rank ρ computed using its prescribed default and dummy coordinates, let k be bounded and nonnegative with $k(v) = 0$ for $v \leq \vartheta$. The supports of $k(v)R_\vartheta(\rho)$ and $h_\vartheta(v)R_\vartheta(\rho)$ are contained in one access class of at most $K - 1$ real entries, on which $I_\vartheta(v, \rho) = 1$.*

The score I_ϑ itself has an access class of at most $K - 1$ real entries containing its support. That class has a bounded witness with bounded row support and a Lipschitz sth root and value one on the whole class. These assertions require no accuracy or sign condition on the raw row.

Proof. For the main and preparatory products a nonzero entry has $v > \vartheta/2$ and $\rho < \vartheta/500$. There cannot be K such entries, since then the K th rank would exceed $\vartheta/2$.

For an explicit class choose consistently computed approximations $\tilde{v}, \tilde{\rho}$ with absolute errors at most $\eta = \vartheta/10000$, and set

$$A = \{y : Y_a(y) = 1, \quad \tilde{v}_y > 3\vartheta/8, \quad \tilde{\rho} < \vartheta/200\}. \quad (6.17)$$

Every positive main or preparatory entry belongs to A . An entry in A satisfies $v_y > (3/8 - 1/10000)\vartheta > \vartheta/4$ and $\rho < (1/200 + 1/10000)\vartheta < \vartheta/100$, so its witness I_ϑ equals one. Its entry threshold exceeds its rank threshold, proving $|A| \leq K - 1$.

The support of I_ϑ requires $v > \vartheta/8$ and $\rho < \vartheta/50$, again giving the same row bound. To access this score itself, the class

$$A_I = \{y : Y_a(y) = 1, \quad \tilde{v}_y > \vartheta/10, \quad \tilde{\rho} < \vartheta/40\}$$

contains its support and has at most $K - 1$ entries. A witness with a Lipschitz sth root that equals one on A_I is

$$J_\vartheta(v, \rho) = \chi_{82\vartheta/1000, 90\vartheta/1000}(v)(1 - \chi_{26\vartheta/1000, 27\vartheta/1000}(\rho)).$$

The approximation margins imply $A_I \subseteq \{J_\vartheta = 1\}$. Moreover $82/1000 > 3(27/1000)$, so this witness also has bounded row support and the required entry versus rank separation. The witness is an analytical output; its support need not be broadened recursively.

These arguments use the rank of the prescribed entire raw array. In particular, if at least K dummy positions have a common virtual default β , then $\rho \geq \beta$. A real position still equal to that

default cannot meet either access class: the required lower bound on $v = \beta$ exceeds the required upper bound on ρ . Thus omitted default positions do not create unlisted ports. $\square$

Each computed version uses its own final raw array and one repeatable approximation and tie convention for every access query. The fixed margins make the support assertions valid for any such convention, without testing equality of exact reals or assuming that the raw array is accurate. Different versions may therefore expose different zero-weight entries. Auxiliary-only access classes need a row bound; the incoming bound will apply to capped correction access below. Raw-array accuracy enters the later error estimates.

An estimated table applies the bounded row functions to its *final* raw arrays. A normalized task divides its computed second input, which approximates $\zeta \mathbf{1}_{\{Y_t(y)=1\}} q_y$, by ζ and applies c . This second input is formed by averaging the earlier preparatory table before selecting an endpoint; selection triggers no new averaging. The preparation has its own raw arrays, scores, and gates and targets the unnormalized table hR or hRT on Y_a . Its usage parameter t selects the averaging slice. The second raw signal and normalized main output carry the resulting Y_t mask.

For a correction, the auxiliary incoming scores use that task's own final first-signal rows, and Equation (6.12) forms their incoming gate. The correction also multiplies by a clipped vector estimate of the prescribed U -cutoff $g_0(y)$, queried with a common broad Y_f -domain tuple. Bin tables, their preparations, and detector tables have no additional incoming multiplier. Because c takes values in $[1/2, 1]$, every such row function before incoming multiplication has a globally Lipschitz sth root as a function of its saturated first entry, first rank, and scaled second entry. The constants may depend on the fixed ζ , but not on the stage, rates, or budget.

The compression argument uses the thresholds in Table 1. For a row function H evaluated before incoming multiplication, its entry factor vanishes through e_H , and its row gate vanishes from rank g_H onward. Every listed pair satisfies $3g_H < e_H$, including the broadest witness, for which $82\vartheta/1000 - 3(27\vartheta/1000) = \vartheta/1000 > 0$. All gates are full in a neighborhood of rank zero. The correction preparation in this table contributes the factor hR ; multiplication by its incoming gate T_y follows afterward.

Row function before incoming multiplication	e_H	g_H
Bin table, including routed bins	$1/2$	$1/1000$
Bin preparation hR	$1/4$	$1/1000$
Correction clipped-ratio function	θ_1	$\theta_1/500$
Correction preparation hR	$\theta_1/2$	$\theta_1/500$
Detector score	δ_0	$\delta_0/500$
Auxiliary score I_ϑ	$\vartheta/8$	$\vartheta/50$
Witness J_ϑ	$82\vartheta/1000$	$27\vartheta/1000$

Table 1: Entry and row-gate thresholds. The entry factor vanishes through e_H , and the row gate vanishes from rank g_H onward. Here $\vartheta \in \{1/2, \theta_1, \delta_0\}$ as appropriate.

The bounded first-entry ranges and gate transition intervals already specified give fixed sensitivity ranges for these functions. If w is a computed second raw entry, its only use is through $c(w/\zeta)$; clipping w to $[\zeta/2, \zeta]$ leaves this quantity unchanged. Its denominator stays in $[1/2, 1]$ even for negative or inaccurate inputs. Usage masks only remove entries and do not change any of these support or continuity bounds.

Lemma 6.5 (Support and degree bounds for table access). *For each true or computed version of the tables, use one fixed task tuple and that version's final raw arrays for all values, ranks, auxiliary scores, and access tests. The main tables, preparations, scores, and witnesses then have bounded values and row support. Each required access class has bounded size and a bounded witness with bounded row support and a Lipschitz sth root that equals one on every exposed entry, including entries of weight zero.*

For a correction or its preparation, let $A \subseteq V \times V$ be the row-access relation of Equation (6.17). Keep a pair $(x, y) \in A$ precisely when

$$\#\{x' : X_u(x') = 1, (x', y) \in A\} \leq K_c. \quad (6.18)$$

The resulting relation still covers the table's nonzero entries, with at most $K - 1$ outgoing and K_c incoming pairs at each vertex. Its entries can therefore be indexed by distinct row and column ports paired at the same pair. Auxiliary-score access has the row bound only; witnesses require no further access class.

Proof. The row and witness assertions follow from Lemma 6.4. The cap is defined from the same final rows that define I and T_y , so its deletion criterion can be compared pointwise with the incoming gate. Clipping and multiplication by bounded factors do not enlarge supports. If a column has more than K_c entries in A , its auxiliary score is one at all those entries, so its K_c th auxiliary rank is one. Equation (6.12) makes $T_y = 0$ there. Consequently every value deleted by Equation (6.18) was already zero. The surviving relation has the stated two degree bounds even when zero weights are retained. The argument applies to arbitrary raw arrays, so it applies equally to the estimated and true versions.

The cap removes every precursor port at an overfull column; it does not choose K_c of them. It is applied even when the particular entry has weight zero. Every surviving port still lies in A , so its row witness I equals one regardless of a zero usage mask, incoming gate, or column multiplier. The broader auxiliary class A_I is used to enumerate scores for an incoming rank. It is not the capped correction relation used as a paired left factor in a product.

The classes are relations on row and column IDs, rather than multisets of paths. Choosing one port per relation element gives the stated combinatorial interface. Establishing its single-address enumeration and port pairing within the space bound is an implementation obligation; the degree argument does not itself grant random access to both IDs. $\square$

The implementation must enumerate each relation element once and pair its two port indices consistently. This requirement applies to each computed version separately.

For later reference, the full interface supplied here is the following. Each task has fixed common parameters, and first ranks include the prescribed dummy coordinates. Exposed ports identify distinct real entries; at each such entry the corresponding bounded witness function equals one. Correction main and preparatory ports obey the actual incoming cap. Values, auxiliary scores, incoming ranks, and column multipliers are all queried at the same supplied environment and tuple. The path by which an entry is discovered must not change that environment or tuple for its assembly. Only completed earlier output types, or stated fixed local functions and aggregations of those types, may be used as raw inputs of a later task. No prefix still waiting for assembly is such an output. These requirements distinguish finite-support analytical bounds from the access properties used in the subsequent implementation. Completing a raw array here means defining its entries from the allowed earlier outputs, rather than storing the whole array at once. Its auxiliary scores and precursor classes are defined before its incoming gate and cap, so their definitions do not call that same gate or cap.

These contracts apply directly to each computed array after compression and clamping. The later comparison in Lemma 9.3 preserves relevant row-function values under its stated common-increment hypothesis. It need not preserve the finite-accuracy access descriptors of two calculations. Each calculation uses its own consistent classes, and Lemma 6.5 gives their support and degree bounds separately.

6.6 Integrated losses and assembly bounds

It remains to control how errors in the row functions, incoming scores, and detector vectors affect an assembled table. The incoming degree bound lets us sum these errors by columns without multiplying by the number of possible source vertices.

For a real sample-dependent row array $Z(x, y; \sigma)$, and for two nonnegative row arrays H, H' on the same domain, define

$$\|Z\|_{u,a;s}^s = \frac{1}{N} \mathbb{E}_\sigma \sum_x \frac{X_u(x)}{u} \sum_{y: Y_a(y)=1} |Z(x, y)|^s, \quad \mathfrak{d}_{u,a}(H, H') = \|H^{1/s} - (H')^{1/s}\|_{u,a;s}. \quad (6.19)$$

Here σ has the product law of the sampling channels and independent auxiliary bits specified in Section 4. All table values are extended by zero outside their membership domain. The denominator u is the nominal rate, and actual-rate comparability in Lemma 4.1 absorbs the resulting absolute constants. For a vertex array on a channel with rate u , write

$$\|v\|_{u;s}^s = \frac{1}{N} \mathbb{E}_\sigma \sum_x \frac{X_u(x)}{u} |v(x)|^s; \quad (6.20)$$

the indicated role channel replaces X when necessary. These are also the comparison norms for two sample-dependent versions. All required row functions before incoming multiplication and the specified support witnesses are included among the row outputs whose discrepancies are to be bounded.

Bounded row support and range imply a uniform bound on these row-output norms, since $|V_i| \leq N$ and $\mathbb{E} X_u/u = O(1)$. Bounded vertex arrays have the analogous uniform bound. Ordinary error is controlled by root error on bounded nonnegative outputs, since

$$|z - w| \leq sC^{(s-1)/s} |z^{1/s} - w^{1/s}| \quad (0 \leq z, w \leq C). \quad (6.21)$$

Lemma 6.6 (Assembly comparisons). *Compare two versions at the same environment, rates, and task tuple, using each version's own final raw arrays and capped access relation. Write their normalized correction outputs as $H = PTg$ and $H' = P'T'g'$. Here P, P' include the usage masks and clipped ratios; T, T' are the incoming gates formed from their respective scores I, I' ; and g, g' are the prescribed cutoffs of bounded vertex arrays $(U_i)_i, (U'_i)_i$, with Lipschitz s th roots. Then*

$$\mathfrak{d}_{u,f}(H, H')^s \leq C \mathfrak{d}_{u,f}(P, P')^s + C \mathfrak{d}_{u,f}(I, I')^s + C \frac{f}{u} \sum_i \|U_i - U'_i\|_{f;s}^s. \quad (6.22)$$

For correction preparations, use the unnormalized P, P' and omit the last term. Bin and detector row functions have no incoming gate or column-multiplier cost.

For the detector trials $V(y) = \text{rk}_J\{\mathcal{S}_i(x, y) : X_f(x) = 1\}$ and $V'(y)$ defined from $\mathcal{S}'_i$ on $Y_f(y) = 1$,

$$\|V - V'\|_{f;s}^s \leq C \mathfrak{d}_{f,f}(\mathcal{S}_i, \mathcal{S}'_i)^s. \quad (6.23)$$

The vertex norms here use the terminal channel Y , and C depends only on the fixed table bounds and cutoffs.

Proof. All factors are bounded. On the union E of the two final nonzero supports, the triangle inequality for the products of s th roots gives

$$|H^{1/s} - (H')^{1/s}| \leq C|P^{1/s} - (P')^{1/s}| + C|T^{1/s} - (T')^{1/s}| + C|g^{1/s} - (g')^{1/s}|.$$

Outside E the left side is zero. The discrepancy before incoming multiplication may be charged on its entire row domain; it needs no incoming cap. Each column has at most $2K_c$ edges in E , by Lemma 6.5. By Equation (6.4), Lipschitz continuity of the incoming cutoff's s th root, and Equation (6.21),

$$|T_y^{1/s} - (T'_y)^{1/s}|^s \leq C \sup_{x: X_u(x)=1} |I(x, y)^{1/s} - I'(x, y)^{1/s}|^s.$$

Consequently its integrated contribution is at most

$$\frac{C}{Nu} \mathbb{E} \sum_{y: Y_f(y)=1} \sum_{x: X_u(x)=1} |I(x, y)^{1/s} - I'(x, y)^{1/s}|^s,$$

which is $C\mathfrak{d}_{u,f}(I, I')^s$.

The maximum over a fixed number of U_i values is Lipschitz, and the multiplier cutoff has Lipschitz root. Thus the last discrepancy is bounded in s th power by $C \sum_i |U_i(y) - U'_i(y)|^s$. Summing over E costs at most $2K_c/u$ per retained column. The vertex metric has weight $1/f$ instead, giving precisely the factor f/u in Equation (6.22). All columns in this comparison are in the common Y_f domain, including those whose final values carry a narrower usage mask. The common-tuple condition ensures that the vertex discrepancy at y is the same quantity for every old start. Explicitly, if $m_E(y)$ counts the edges of E entering y , then for each i its contribution obeys

$$\frac{1}{Nu} \mathbb{E} \sum_{y: Y_f(y)=1} m_E(y) |U_i(y) - U'_i(y)|^s \leq 2K_c \frac{f}{u} \|U_i - U'_i\|_{f;s}^s.$$

This is the column-counting step; it uses no independence between the two row tables and the vector estimates.

For a detector trial, Equation (6.4) bounds the trial discrepancy by the supremum of the input score discrepancies over retained starts. Its s th power is at most the sum of their s th powers. The factor $Y_f(y)/f$ in the vertex metric is exactly the factor obtained by reversing the two sums in the row metric with $u = a = f$. Equation (6.21) gives Equation (6.23). $\square$

The required task types are therefore the bin tables $\mathcal{B}_b$ of C_l , the routed bins $\mathcal{B}_b^{(j)}$ of D_l , detector scores $\mathcal{S}_i$, and the preliminary and actual correction tables $\mathcal{F}$, together with one earlier unnormalized preparation for each normalized type. Their true conditional-mean identities are Lemmas 6.1 and 6.2. Their computed versions use the same bounded row functions and the assembly contract above. The next sections bound their discrepancies through the additive schedule and implement their accesses without storing two unrestricted vertex addresses.

7 Conditional averaging in the shared environment

The conditional-mean identities for the tables use conditional expectations on slices that retain specified source and terminal vertices. To approximate these expectations, we need a walk that mixes on each slice. We construct such a walk using the channels of Section 4, then express its steps by transformations that can be specified before the retained identifiers are known. Validation tests those identifiers when their vertices are visited.

For each designated channel, fix an identifier and condition on its retention. A conditional slice imposes these events and fixes any designated frozen bits; the remaining independent bit coordinates are free. The *full environment law* already imposes full column rank on every channel. Retention and frozen-bit conditions further restrict that law, which is then normalized for the slice's averaging operator and L_s norms.

The only external group-theoretic inputs are property (T) of $\mathrm{SL}_H(\mathbb{Z})$ for $H \geq 3$ and relative property (T) of

$$(\mathrm{SL}_2(\mathbb{Z}) \ltimes \mathbb{Z}^2, \mathbb{Z}^2).$$

We use Shalom's quantitative statements: [12, Theorem 2.6, p. 156] for the linear group and [12, Theorem 2.1, p. 152] for the relative property. From these inputs we obtain a walk on every finite transitive set on which the affine group acts, with a spectral gap bounded below uniformly over those sets. We then use that walk to average the environments satisfying each specified retention condition.

Lemma 7.1 (Conditional averaging). *Fix a product of the sampling channels above and condition on one retained identifier in each designated channel. Let a specified set of independent fair bits be free, padding its length to a multiple of H , and fix all remaining bits. On this slice there is a rational Markov operator K with the uniform conditional law as its stationary distribution such that, for every fixed $2 \leq s < \infty$ and integer $d \geq 0$,*

$$\|K^d - \Pi\|_{L_s \rightarrow L_s} \leq C_s 2^{-c_s d}.$$

Here Π averages precisely the channel and bit coordinates that are free on the slice. The positive constants C_s, c_s are independent of the field size, retention side lengths, free-bit length, retained identifiers, and frozen-bit values.

The conditional operator is obtained from a fixed affine walk whose gap is independent of the field, retention box, and free-bit length (Lemma 7.2).

Lemma 7.2 (A uniform affine spectral gap). *For each fixed integer $H \geq 3$, the affine group $G_H = \mathrm{SL}_H(\mathbb{Z}) \ltimes \mathbb{Z}^H$, with its usual left action on the translation subgroup, admits a lazy rational walk supported on a fixed finite symmetric generating multiset. On every finite transitive G_H -set, this walk has a spectral gap at least a positive constant depending only on H .*

Proof of Lemma 7.1. We identify the conditional slice with a transitive affine set, so the gap will follow from the fixed group walk without depending on the size of the slice. On a channel conditioned at y with side L , put $u_y = (y, \dots, y^r)^\top$ and use the coordinates

$$(a_0, F) \longleftrightarrow (F, w), \quad w = a_0 + F u_y, \quad a_0 = w - F u_y \pmod{P}.$$

This is a bijection from the conditioned channel to the product of full-column-rank matrices F and the box $w \in \{0, \dots, L-1\}^H$. Each F, w has exactly one preimage, so the conditional uniform law becomes the product uniform law. Identify the box with $(\mathbb{Z}/L\mathbb{Z})^H$. For $(M, c) \in G_H$ act by

$$(F, w) \longmapsto (MF, Mw + c),$$

using reduction modulo P on the first coordinate and modulo L on the second. This is one action even when $P \neq L$: for arbitrary $(M_1, c_1), (M_2, c_2)$, the successive transformations send (F, w) to

$$(M_1 M_2 F, M_1 M_2 w + M_1 c_2 + c_1),$$

with each coordinate reduced in its own modulus. This is precisely the semidirect-product law. Every transformation is invertible and hence preserves the product uniform law.

This action is transitive. Reduction of $\mathrm{SL}_H(\mathbb{Z})$ contains the elementary generators of $\mathrm{SL}_H(\mathbb{F}_P)$. That finite group acts transitively on ordered linearly independent r -tuples, with $r < H$: extend either ordered tuple to a basis and adjust a complementary column to make the change-of-basis determinant one. Once the ordered tuple has been moved, a translation adjusts w to any specified value. The two moduli need not agree. On an unconditioned channel use a public identifier and side P , giving the same description.

If the free bit length is Hq , identify its strings with $(\mathbb{Z}/2^q\mathbb{Z})^H$ and use the affine action on that coordinate alone. It is transitive. All currently free bit blocks can be concatenated into this one component; frozen bits are not transformed. There are a bounded number of components. Apply the lazy walk from Lemma 7.2 to a uniformly selected component. If there are J_{comp} components and the single-component gap is at least γ_H , the product walk has gap at least $\gamma_H/J_{\mathrm{comp}}$. Here J_{comp} is bounded independently of the field and bit lengths. The walk preserves the desired product measure.

Writing its gap as $\gamma > 0$, we have $\|K^d - \Pi\|_{2 \rightarrow 2} \leq (1 - \gamma)^d$ and $\|K^d - \Pi\|_{\infty \rightarrow \infty} \leq 2$. The diagonal case of the Riesz–Thorin interpolation theorem applies to this same finite-space operator [11, Theorems II and II', p. 472]. In the weighted formulation, take both sets of positive weights to be the conditional point probabilities. The reciprocal input and output exponents move together along the segment from $(1/2, 1/2)$ to $(0, 0)$; the point $(1/s, 1/s)$ has weight $2/s$ at the L_2 endpoint. Consequently

$$\|K^d - \Pi\|_{s \rightarrow s} \leq 2^{1-2/s} (1 - \gamma)^{2d/s},$$

which has the required form. $\square$

7.1 Proof of the affine gap

Proof of Lemma 7.2. Let $\mathcal{S}$ consist of the elementary matrices with a single ± 1 off the diagonal, acting linearly, and the translations by $\pm e_i$, where e_i are the standard basis vectors of $\mathbb{Z}^H$. This is a fixed finite symmetric generating set of G_H . For a unitary representation ρ of G_H on a Hilbert space and a vector ξ in that space, put

$$\delta = \max_{g \in \mathcal{S}} \|\rho(g)\xi - \xi\|.$$

We first bound the distance from ξ to the invariant subspace.

For each coordinate pair $\{i, j\}$, embed the two-dimensional affine group on those coordinates, and let P_{ij} be the orthogonal projection onto the subspace fixed by its translations. This subspace and its orthogonal complement are invariant under that pair group, because its translation subgroup is normal. The eight generators in [12, Theorem 2.1, p. 152] belong to $\mathcal{S}$. To pass from that theorem's existence conclusion to a distance bound, put $\eta = (I - P_{ij})\xi$. The restricted representation on this orthogonal complement has no nonzero translation-invariant vector. The contrapositive of the relative Kazhdan statement, applied to $\eta/\|\eta\|$ when $\eta \neq 0$, therefore gives

$$\frac{\|\eta\|}{10} \leq \max_{g \text{ in the pair generators}} \|\rho(g)\eta - \eta\| \leq \delta.$$

The last inequality uses $\rho(g)\eta - \eta = (I - P_{ij})(\rho(g)\xi - \xi)$ and the fact that orthogonal projection is contractive. Hence $\|\xi - P_{ij}\xi\| \leq 10\delta$, including $\eta = 0$. This is the projection argument in the proof of [12, Corollary 2.3, p. 154].

The projections P_{ij} commute: each is the strong limit of Cesàro averages over the corresponding two commuting coordinate translations, and all coordinate translations commute with one another. These averages also commute with each P_{kl} ; passing to the limit preserves that commutation. Their product P_{tr} projects onto the subspace fixed by all translations. With $q = \binom{H}{2}$, telescoping the product gives

$$\|\xi - P_{\text{tr}}\xi\| \leq \sum_{i < j} \|\xi - P_{ij}\xi\| \leq 10q\delta.$$

Since the full translation subgroup is normal, P_{tr} commutes with every $\rho(g)$. Its range carries a representation of $G_H/\mathbb{Z}^H = \text{SL}_H(\mathbb{Z})$. Let P_{inv} project onto the full G_H -invariant subspace, and let $\kappa_H > 0$ be the Kazhdan constant for the elementary generators from [12, Theorem 2.6, p. 156]; adjoining inverse generators preserves the bound. Applying that constant on the range of P_{tr} gives

$$\|P_{\text{tr}}\xi - P_{\text{inv}}\xi\| \leq \kappa_H^{-1}\delta, \quad \|\xi - P_{\text{inv}}\xi\| \leq C_H\delta, \quad C_H = 10q + \kappa_H^{-1}.$$

Thus the same generating set controls distance to invariants in every unitary representation, with a constant depending only on H .

Now let ρ be the permutation representation of a finite transitive action, with the uniform probability measure, and put $m = |\mathcal{S}|$. The rational Markov operator

$$K_H = \frac{1}{2}I + \frac{1}{2m} \sum_{g \in \mathcal{S}} \rho(g)$$

is self-adjoint, and its spectrum lies in $[0, 1]$. The invariant functions are exactly the constants. For f orthogonal to constants, the preceding inequality and the Dirichlet identity give

$$\langle f, (I - K_H)f \rangle = \frac{1}{4m} \sum_{g \in \mathcal{S}} \|\rho(g)f - f\|_2^2 \geq \frac{1}{4mC_H^2} \|f\|_2^2.$$

This proves the uniform spectral gap. □

7.2 Generating paths without retaining their endpoints

The coordinates in the preceding proof used the indicated identifier. The next lemma is the implementation form of the same averaging, and also supplies changes of variables for later error estimates.

Lemma 7.3 (Validated walk variants). *Each length- d walk word from Lemma 7.1 admits variants with $O(d)$ -bit descriptions over a fixed alphabet. Every fixed variant is a measure-preserving bijection of the full environment law and is independent of the conditioned identifiers. For each environment in the specified slice, exactly one variant of each word passes all slice and integer-wrap tests; it gives the conditional walk action of that word. Weighting each valid variant by its original word probability therefore gives exactly the conditional average.*

The tests may be made when their specified vertices are visited. If field elements, identifiers, and the total auxiliary bit string have $O(B)$ bits and $d = O(B)$, transformations and tests use $O(B)$ shared scratch, while the variant description uses $O(d)$ retained bits.

Proof. Consider a fixed affine generator (M, c) on a channel with retention side L . Enumerate the transformations

$$A' = MA + (c + Lz)e_1^T \pmod{P}, \tag{7.1}$$

where e_1 selects the constant column and z ranges over a fixed finite box of integer vectors. That box contains all possible wraps: for $w \in \{0, \dots, L-1\}^H$, every coordinate of $Mw + c$ is bounded in absolute value by a fixed multiple of L plus a fixed constant. Since $L \geq 1$ and the generators are fixed, one bound for z works for every side length.

Each transformation in Equation (7.1) is invertible. Left multiplication preserves the full-column-rank condition, and the added term changes only the constant column. It therefore preserves uniform measure on the full channel. It does not refer to the indicated identifier.

At that identifier, let $w = Av(y)$ in its integer representatives. Check first that w is in the retention box, and then that $Mw + c + Lz$ lies in the same box coordinatewise, using integer arithmetic before reduction modulo P . The unique valid wrap has coordinates $z_i = -\lfloor (Mw + c)_i / L \rfloor$. For this vector the evaluation of A' is the representative of the intended affine action modulo L . Applying these checks at every step gives the unique valid wrap sequence for a whole word.

The construction applies separately to the bounded number of channels. On a channel of side P the ordinary field reduction suffices; alternatively, the same wrap validation may be used at its public point. The free-bit torus has ordinary invertible affine operations and requires no unknown endpoint. Frozen bits remain fixed. Thus a word has only constantly many generator and wrap choices per step, giving the $O(d)$ -bit description.

For each word and initial environment in the slice, exactly one variant survives. Its weight must be the weight of that word. In particular, the sum is not divided by the number of guessed wraps. This proves the asserted exact identity. All matrices have fixed dimensions and $O(B)$ -bit entries; evaluating a bounded-degree polynomial or a word's individual steps needs $O(B)$ arithmetic scratch. A test depending on a vertex can be delayed until that vertex is current, with the required matrix state regenerated from the recorded transformations. Theorem 12.1 accounts for reversible restoration and the simultaneous space used when these operations are nested. $\square$

Remark 7.4 (Effective environment coordinates). Conditional averaging is expressed in the channel and bit coordinates of the environment supplied to that invocation. A fixed variant from Lemma 7.3 is a measure-preserving change of variables on the full environment; its retention and wrap tests remain explicit masks. Thus free and frozen bits always refer to the supplied coordinates, which may differ from those before the change of variables.

Remark 7.5. Two distinct uses of the variants should be kept separate. For centered-noise estimates, one first recognizes the sum of valid variants as the conditional walk and applies Lemma 7.1. For crude bounds and changes of variables, each fixed variant is a measure-preserving bijection, and one may then pay for their $2^{O(d)}$ number. Counting variants before a spectral estimate would needlessly weaken the useful contraction.

8 Additive estimation and conditional prefix bounds

The bounded sample-dependent tables of Section 6 represent the exact hierarchy through their conditional means. Earlier estimates serve as input tables and vectors with those interfaces. Under the corresponding child-error hypotheses, their sampled raw prefixes have two errors: the conditional mean can differ from the deterministic target, and a finite averaging walk leaves noise around that mean.

These prefixes are intermediate arrays. Final row outputs result from applying the table functions to completed, compressed rows; Section 10 completes the recursive definition, including its endpoint-grouping rule.

The scale and source-domain comparisons first determine the accuracy needed from each child. We then average differences between consecutive child budgets. Their conditional means telescope, while their small absolute norms allow finite conditional walks to suppress the noise. The resulting prefix estimates are the input to the compression analysis in Section 9. When contributions with the same endpoint are grouped exactly, that section bounds the error of an active task by half its allowed estimate error. Section 10 constructs the comparison calculation to which this bound applies, leaving the other half for the error caused by key comparisons.

8.1 Tasks and dependency order

A row task specifies its input matrix, stage, iteration label, start rate u , end scale a , optional usage rate $t \leq a$, channel roles, and fixed padding suffixes. These parameters are common to its whole table. A vector task specifies the analogous vertex domain. Its argument is a vertex, rather than an additional parameter retained by another task. Each task has a nonnegative integer statistical budget M ; m will denote a child budget. Write M_{top} for the budget of the outermost task. These budgets are distinct from the numerical digit places introduced in Section 11. The ordinary and root discrepancy norms are defined in Equations (6.19) and (6.20). We always take $s = 8$.

The source weight $X_u(x)/u$ has bounded expectation, and the factor $1/N$ normalizes the sum over source vertices. Endpoint sums remain explicit: bounded row supports or incoming degrees, as appropriate, will control them. A bound at one specified start consequently costs a factor N in moment power; Section 3.2 makes and pays this conversion at the end of the proof.

Here is a dependency order, with only a fixed number of task types at each stage. First come the bin tasks for C_l . At each preliminary step come its input-score task $\mathcal{S}_i$, the detector vector U_i , and then the preliminary correction task. Next come the actual correction tasks in iteration order, the routed bin tasks for D_l , and the reward update. Place the unnormalized preparation immediately before each normalized task of the same input. The raw first channels of both these tasks refer only to earlier inputs in this order; the normalized task has in addition its preparation as the source of its second channel.

For a correction task or its preparation, first complete its raw arrays and scores computed before incoming multiplication, then form its incoming gate from those scores. This assembly uses the current task's completed arrays; every recursive input to their raw calculation is an earlier task in the order just specified.

8.2 Raw sample terms

For a fixed row and endpoint let

$$\mathcal{C}_{xy}^{u,a} = \{X_u(x) = 1, Y_a(y) = 1\}.$$

We use conditional means and conditional walks on this slice. For each bin scale b in the rate grid, write $\mathcal{B}_b$ for a normalized bin table and $\mathcal{B}_b^{(j)}$ for a bin with destination routing fraction already included; $\mathcal{F}$ denotes a normalized correction table. Their values are in units of the table scale and are zero off their specified membership domains. In all formulas below put $\tau = \min(a, b)$.

The direct bin and correction-bin product terms, before division by the current first-channel scale a , are respectively

$$T_b(x, y) = b \frac{\pi_a}{\pi_\tau} \mathcal{B}_b(x, y), \quad (8.1)$$

$$T_b^{\text{prod}}(x, y) = b \frac{\pi_a}{\pi_\tau} \frac{f}{\pi_f} \sum_{z: Z_f(z)=1} \mathcal{F}(x, z) \mathcal{B}_b(z, y). \quad (8.2)$$

In the product, the left task has roles X, Z and rates $(u; f, f)$, listed as start, end, and usage. The right task has roles Z, Y and rates $(f; b, \tau)$. In the direct term the start rate stays u . The term $-E$ in D is the negative of Equation (8.1) with $b = f$ and $\mathcal{F}$ in place of $\mathcal{B}_b$. Summing the direct and product terms gives the raw inputs G_i, G'_i ; adding the negative term gives D .

For a copy projection x_i, y_j from stage $l + 1$ to stage l , the raw term is

$$r_i(x)b\frac{\pi_a}{\pi_\tau}\mathcal{B}_b^{(j)}(x, y). \quad (8.3)$$

Partition by the fixed symbols i, j . Append those symbols to the native hash arguments before the inherited suffixes on their respective channels. The parent-stage source multiplier is obtained from its detector vectors on the broad parent-stage vertex domain. No child tuple contains the old source or a particular right endpoint as an extra parameter.

For the second raw channel, use ζ times the preparatory table, take its conditional average on $\mathcal{C}_{xy}^{u,t}$, and extend the result by zero off $Y_t(y)$. For a detector vector use the trial

$$\text{rk}_J\{\mathcal{S}_i(x, y) : X_f(x) = 1\} \quad \text{on the slice } Y_f(y) = 1. \quad (8.4)$$

Ranks include arbitrary zero padding. For rewards, writing $f = b_l$, the two terms for $W_{l+1}(x_i)$ on its inherited source slice are

$$r_i(x)W_l(x), \quad r_i(x)\frac{f}{\pi_f} \sum_{z:Z_f(z)=1} \mathcal{F}(x, z)W_l(z). \quad (8.5)$$

The first W_l call keeps the inherited source domain; the second restarts on the midpoint domain. Vector answers are finally clipped into $[0, 1]$.

Lemma 8.1 (True sample means). *For true tables with the role channels, rates, usage masks, and copy suffixes specified in Equations (8.1) –(8.5), conditional averaging gives exactly their prescribed hierarchy contributions. Summing the first-channel terms and dividing by a gives $G(x, y)/a$; averaging the preparation on $\mathcal{C}_{xy}^{u,t}$ gives $\zeta q_y \mathbf{1}_{\{Y_t(y)=1\}}$. The detector and reward formulas give $U_i(y)$ and $W_{l+1}(x_i)$ on their indicated vertex slices. All quantities have the specified zero extensions outside those slices.*

Proof. If $b \geq a$, the usage hit is already the conditioning hit. The conditional normalization of a bin therefore turns $b\mathcal{B}_b$ into its deterministic bin contribution. If $b < a$, nested origin boxes give

$$\mathbb{E} \left[\frac{\pi_a}{\pi_b} \mathbf{1}_{Y_b(y)} V \mid \mathcal{C}_{xy}^{u,a} \right] = \mathbb{E}[V \mid \mathcal{C}_{xy}^{u,b}]$$

for every integrable V on the narrower slice. Thus the importance factor performs exactly the required change of conditioning.

For a fixed midpoint in Equation (8.2), first make this terminal change when necessary and then condition on $Z_f(z) = 1$. Apart from membership indicators, the true right bin gate and denominator use only channel Y . The true left correction uses only channels X, Z . These channels are independent, and the remaining true multipliers are deterministic. The two normalizations can therefore be applied separately. In particular, after the midpoint hit is fixed, the shared role Z does not occur in the right bin's remaining random value. This is an identity for the true tables; the estimates will instead be compared by their joint error norms. Write F for the true correction matrix represented by $f\mathcal{F}$, and $A_b(z, y) = C(z, y)\omega_k(C(z, y))$ for the true contribution of its right bin at $b = b_k$, where C is the right-factor transition at this stage. Conditional on the source and

terminal hits and on $Z_f(z) = 1$, the two mean factors are $F(x, z)/f$ and $A_b(z, y)/b$. Thus, after the terminal change of conditioning above,

$$\mathbb{E}[T_b^{\text{prod}}(x, y) \mid \mathcal{C}_{xy}^{u, a}] = \sum_z b \frac{f}{\pi_f} \pi_f \frac{F(x, z)}{f} \frac{A_b(z, y)}{b} = (FA_b)(x, y).$$

Here π_f is the midpoint-hit probability; its cancellation is separate from the terminal factor π_a/π_τ . This proves the product formula for both $b \geq a$ and $b < a$. The projection multiplies by deterministic $r_i(x)$ and includes the routing fraction in the bin, so the same argument applies on each fixed-symbol part. Fixed-suffix invariance in Lemma 4.1 justifies all inherited padding conventions.

The second channel is its defining preparatory conditional mean, on its own narrower slice; it has no importance factor. The detector mean is the definition of U_i . True reward factors are deterministic, so the same midpoint argument proves Equation (8.5). $\square$

To form a budget- m sample term, replace all earlier inputs in the corresponding formula by budget- m estimates, with their domains and common task parameters unchanged. This includes the detector inputs to r_i ; the primary factor $r_0 = 1$ remains constant. Products, source functions, and ranks are evaluated before averaging. We first compare this sample's conditional mean with the same formula using true tables. The scale and source-domain costs of that comparison will determine how much accuracy a budget must provide.

8.3 Mean comparisons and the cost of releasing a source

Lemma 8.1 identifies the exact conditional means. For estimated factors, the comparison has two distinct costs. The factor b/a converts a bin contribution to the current entry units. A right factor at a midpoint also replaces the old source domain of rate u by the stage domain of rate f . The following estimates account for both changes without assuming independence of the errors.

For a first-channel error e_y measured against the true signal $d_y = G(x, y)/a$, its clipped relative norm means $\|\min(1, |e|/(1 + d))\|_{u, a; s}$, with all operations entrywise. Other raw errors use the ordinary norms of Equations (6.19) and (6.20).

For nonnegative H, H' define

$$\mathcal{L}(H, H') = \begin{cases} |H - H'|^s / (H + H')^{s-1}, & H + H' > 0, \\ 0, & H = H' = 0. \end{cases} \quad (8.6)$$

Lemma 8.2 (Homogeneous loss). *For nonnegative scalars H, H' , the loss $\mathcal{L}(H, H')$ in Equation (8.6) is comparable to $|H^{1/s} - (H')^{1/s}|^s$, with constants depending only on s . For any nonnegative weights, the loss of the corresponding sums or integrals is at most the same weighted sum or integral of the individual losses, whenever these quantities are finite. If also $G \geq 0$ and $H \leq C_0 G$, then for every $a > 0$,*

$$a \min \left\{ 1, \frac{|H - H'|}{a + G} \right\}^s \leq C \mathcal{L}(H, H'), \quad (8.7)$$

where C depends only on s, C_0 .

Proof. Put $v = \max(H^{1/s}, (H')^{1/s})$ and $w = \min(H^{1/s}, (H')^{1/s})$. Factoring $v^s - w^s$ bounds it between $v^{s-1}(v - w)$ and $sv^{s-1}(v - w)$; the denominator is between $v^{s(s-1)}$ and $2^{s-1}v^{s(s-1)}$. This proves comparability. For a mixture, apply Hölder to $|H - H'| = \mathcal{L}(H, H')^{1/s}(H + H')^{(s-1)/s}$ and then integrate; the triangle inequality first bounds the absolute difference of the integrals. This proves contraction, including finite sums.

If $H' \leq 2H + a$, then $H + H' \leq 3C_0G + a \leq C'(a + G)$, which proves Equation (8.7) directly. If $H' > 2H + a$, then $|H - H'| > H'/2$, $H + H' < 3H'/2$, and $H' > a$. Thus $\mathcal{L}(H, H') \geq cH' \geq ca$, while the left side is at most a . $\square$

Every unsigned true term of the signed formula for D is bounded by a constant times D , by Lemma 5.4; the corresponding assertion for the other raw inputs follows from their nonnegative sums. Consequently the preceding lemma can be applied separately to all true contributions. Triangle inequalities combine their clipped relative errors even when a contribution has a negative sign.

Lemma 8.3 (Scale and domain comparison). *Replace the true factors of one raw sample term by budget- m estimates satisfying the table support, incoming-cap, and common-tuple conditions. In the s th power of its clipped relative first-channel mean error, each child root-discrepancy power is multiplied by at most a fixed constant times*

$$\begin{cases} b/a, & b \geq a, \\ (b/a)^{s-1}, & 0 < b < a. \end{cases}$$

A child that releases the source of rate u and uses the stage source domain of rate f incurs the further factor f/u . Children retaining the source incur no such factor. Second-channel and same-scale vector mean errors have constant scale cost, with f/u only for source release. These bounds use the prescribed joint environment and require no independence of estimated factors.

Proof. First suppose $b \geq a$. Apply Equation (8.7) after conditional expectation, and then the mixture contraction of Lemma 8.2 before it. Homogeneity supplies a factor b/a after division by a . The factor f/π_f in a product is bounded above and below by fixed constants. For bounded nonnegative tables a product-root difference splits into the factor-root differences, multiplied by fixed constants times support indicators of the unaffected factor in one of the two versions.

To determine the source-domain factor, write the root discrepancies as $\Delta_s H = H^{1/s} - (H')^{1/s}$ for each compared pair H, H' , in particular $H = \mathcal{F}$ or $H = \mathcal{B}_b$. After integration on the source/end slice, the relevant sums, besides $O(b/a)/N$, have the forms

$$\begin{aligned} & \mathbb{E} \sum_{x,z,y} \frac{X_u(x)}{u} Z_f(z) Y_a(y) |\Delta_s \mathcal{F}(x, z)|^s \mathbf{1}_{\mathcal{B}_b(z,y) \neq 0}, \\ & \mathbb{E} \sum_{x,z,y} \frac{X_u(x)}{u} Z_f(z) Y_a(y) \mathbf{1}_{\mathcal{F}(x,z) \neq 0} |\Delta_s \mathcal{B}_b(z, y)|^s. \end{aligned}$$

Either support indicator can come from either compared version. In the first sum the right row cap eliminates y at constant cost. In the second sum the left incoming cap eliminates x at cost $O(1/u)$. The right child's row metric has weight $1/f$, leaving exactly $O(f/u)$. Its tuple is the same for every old source, so this elimination compares the same right error throughout. More explicitly, for either left support relation $\mathcal{P}$ with incoming degree at most K_c and a common right discrepancy $e(z, y)$, the relevant inequality is

$$\begin{aligned} & \frac{1}{N} \mathbb{E} \sum_x \frac{X_u(x)}{u} \sum_{z:(x,z) \in \mathcal{P}} Z_f(z) \sum_{y: Y_b(y)=1} |e(z, y)|^s \\ & \leq \frac{K_c}{Nu} \mathbb{E} \sum_z Z_f(z) \sum_{y: Y_b(y)=1} |e(z, y)|^s \\ & = K_c \frac{f}{u} \|e\|_{f,b;s}^s, \end{aligned}$$

where the source channel in the last norm is Z . The incoming cap applies pointwise, so the discrepancy can depend on the same environment as $\mathcal{P}$. Using $Y_a \leq Y_b$ enlarges the remaining endpoint domain to the child's own domain. A direct term simply retains its source and endpoint sums. In a projected term, the source-multiplier error eliminates y by a row cap and changes the vertex weight from $1/u$ to $1/f_{\text{par}}$, where $f_{\text{par}} = b_l$ is the parent stage rate for a projection from stage $l + 1$ to stage l ; this costs $O(f_{\text{par}}/u)$. Partitioning the fixed copy symbols keeps every tuple common during these sums.

For $b < a$, change the terminal conditioning to Y_b *before* taking a power. The normalized coefficient inside is then b/a . There are only constantly many nonzero left paths in the union of the two rows. Ordinary bounded-factor difference inequalities and the finite-sum power inequality therefore apply with constant cost. The integrated change from Y_a to Y_b leaves π_a/π_b outside the power. Its total factor is

$$(b/a)^s \frac{\pi_a}{\pi_b} = O((b/a)^{s-1}).$$

The same row and incoming sums as above then apply, now with Y_b . Ordinary differences of bounded nonnegative values are controlled by their root differences. Absolute mean error bounds clipped relative mean error, so this proves the fine-bin assertion.

For the second raw channel, Jensen is applied directly on $\mathcal{C}_{xy}^{u,t}$. Its integrated endpoint mask is Y_t , already contained in the preparatory norm's Y_a domain. There is no importance factor and hence no π_a/π_t penalty. The fixed scaling ζ does not increase the cost.

A rank discrepancy is at most the supremum of its input discrepancies. For the detector trial, summing its power over Y_f/f is bounded by the score error sum over X_f/f and Y_f ; the rates match exactly. For rewards, the finite left-path count gives an ordinary product comparison, and the same incoming cap eliminates old starts for a right-vector error. Source functions have bounded Lipschitz roots in their clipped detector inputs, so their costs are the broad vertex costs already computed. Every estimate here is pointwise or follows from Jensen and summation; none uses independence of its errors. $\square$

The preceding comparison measures errors in a retained sample term. Omitting a fine bin instead discards its whole true contribution; the next bound measures that loss.

Lemma 8.4 (Fine omission). *For an omitted bin of scale $0 < b < a$, its true direct, routed, or correction-bin product contribution, divided by the current scale a , has ordinary row norm at most $C(b/a)^{(s-1)/s}$. The constant is independent of the two scales.*

Proof. Let F be any finite preliminary or actual correction at a stage with transition C . For $b > 0$ and a nonnegative matrix A_b on that stage satisfying $A_b \mathbf{1} \leq \mathbf{1}$ and $A_b(x, y) \leq c_b b$, one has

$$(FA_b)\mathbf{1} \leq k_* \mathbf{1}, \quad (FA_b)(x, y) \leq k_* c_b b.$$

If $A_b \leq C$, then also $FA_b \leq FC$. The constants are independent of b . Use $F\mathbf{1} \leq k_* \mathbf{1}$ from Lemma 5.2: $(FA_b)\mathbf{1} \leq F\mathbf{1}$, and $(FA_b)(x, y) \leq c_b b \sum_z F(x, z)$. The last assertion follows by multiplying $A_b \leq C$ on the left by the nonnegative matrix F .

The deterministic contribution H has entries at most Cb and bounded row mass. For a product, the inequalities just proved give these bounds for its nonnegative bin matrix A_b . Therefore

$$\sum_y \pi_a |H(x, y)/a|^s \leq \frac{\pi_a}{a^s} (Cb)^{s-1} \sum_y H(x, y) \leq C' (b/a)^{s-1}.$$

Integrating source hits gives $\pi_u/u = O(1)$, and summing at most N rows cancels the normalization $1/N$. $\square$

8.4 Choosing precision to pay for scale and domain changes

The preceding bounds explain the rate weights in the accuracy parameter. Write $h_b = \log_2(b/a)$ for the offset of a bin scale. In norm, the coarse-bin cost is $(b/a)^{1/s}$ and the fine-bin cost is $(b/a)^{(s-1)/s}$. Since $s = 8$, a weight $1/2$ between these two exponents leaves a geometric margin in both directions. The same weight pays for the source-release cost $(f/u)^{1/s}$. We now make this accuracy convention precise; the additive schedule will then trade lower child budgets for longer averaging walks.

Assign positive integer ranks i_* in the task order fixed above, so every dependency decreases rank by at least one. The finite number of preliminary steps and of preparations is an absolute constant, so $i_* = O(l + 1)$. A task at integer budget $M \geq 0$ has implicit precision

$$p = \frac{M}{K_{\text{prec}}} - \frac{1}{2} \log_2 \frac{1}{a} - \frac{1}{2} \log_2 \frac{1}{u} - A_* i_*. \quad (8.8)$$

For a stage- l vector use $a = f = b_l$; detector vectors also have $u = f$, whereas rewards retain their inherited start rate. The constants K_{prec} and A_* will be chosen below. If $p \leq 0$, return the zero table or vector and empty access classes. Otherwise the task is called active. The intended induction bound is $C_b 2^{-p}$ for every required ordinary vector norm and every required root discrepancy norm, including the scores computed before incoming multiplication and witnesses. A fixed C_b makes this assertion trivial for inactive tasks: values and row supports are uniformly bounded, and $\mathbb{E}X_u/u = \pi_u/u = O(1)$.

At a normalized correction task, the column multiplier is assembled from earlier detector-vector estimates at budget $M - c_{\text{col}}$, for a fixed positive integer c_{col} . This is a separate final assembly call. All other calls used in one additive sample term have the common budget denoted by m .

Suppose a child at budget m has parameters a', u', i'_* . Subtracting Equation (8.8) for the parent gives the exact identity

$$p' - p = -\frac{M - m}{K_{\text{prec}}} + \frac{1}{2} \log_2 \frac{a'}{a} + \frac{1}{2} \log_2 \frac{u'}{u} + A_*(i_* - i'_*). \quad (8.9)$$

In a bin comparison $a' \geq cb$ for a fixed $c > 0$: a bin child has $a' = b$, while a correction has $a' = f$ and $b \leq K_f f$. The start is either retained or reset to the indicated broad stage rate. Fixed-stage changes in vector calls only change fixed constants. Thus, after taking s -th roots of the costs in Lemma 8.3, the scale contribution times child accuracy is bounded by

$$CC_b 2^{-p - A_*(M - m)/K_{\text{prec}} - 3|h_b|/8}. \quad (8.10)$$

Indeed for $h_b \geq 0$ the exponent is $h_b/s - h_b/2 = -3h_b/8$; for $h_b < 0$ it is $(s - 1)h_b/s - h_b/2 = 3h_b/8$. For a child that restarts at rate f , the source-domain comparison costs $(f/u)^{1/s}$ in norm, while its precision gains $\frac{1}{2} \log_2(f/u)$. Multiplying the cost by the resulting accuracy factor gives $(f/u)^{1/s - 1/2} = (f/u)^{-3/8} \leq C$, using the allowed start-rate condition $u \leq f$.

For the separate final column-vector call, $a = f$, its child uses $a' = u' = f$, and $M - m = c_{\text{col}}$. Thus its norm cost times child accuracy is at most

$$CC_b 2^{-p} 2^{c_{\text{col}}/K_{\text{prec}} - A_*} (f/u)^{-3/8}.$$

The fixed rank separation pays this fixed budget loss. This calculation also explains why the source-domain ratio is charged in the error analysis rather than inserted as a numerical coefficient in a sample.

8.5 The additive schedule

Every budget-zero task is inactive under the preceding precision rule. Hence a budget-zero sample term is zero: it contains at least one estimated table or vector factor. Exact base terms $S(x, y)/a$ and $W_0(x)$ are inserted separately.

For every first-channel term labelled by scale b , set $c_b = \lceil c_0(1 + |h_b|) \rceil$, where c_0 is a fixed positive constant. This rule includes the direct correction term $-E$ with $b = f$, as well as direct bins, products, and projection terms. Every child factor in a term, including the detector-vector inputs to a projected source multiplier, uses that term's common budget m and cutoff c_b . Use zero scale offset only for second-channel preparatory means, detector-vector trials, and reward terms with fixed stage-rate changes. The separate final column assembly retains its fixed drop c_{col} . Denote the corresponding normalized sample term at budget m by $T_{b,m}$, including division by a for the first row channel. For $1 \leq m \leq M - c_b$ form

$$T_{b,m} - T_{b,m-1}. \quad (8.11)$$

Apply to this difference the conditional walk of length $\ell(M - m)$, where ℓ is a sufficiently large fixed integer. Use the same generator word and the same guessed wraps for its two parts.

Group increments by dyadic slack $d \leq M - m < 2d$. Add groups in decreasing powers of two $d \leq M$ down to 1, including empty groups. Put exact base contributions in the first group. Let S_j be the uncompressed prefix through the group with slack d_j . The conditional mean prefix is obtained by replacing each walk by uniform averaging on its slice; noise is the difference from that mean. For an available bin the last budget reached by this mean prefix is

$$m_j(b) = \max\{0, M - \max(d_j, c_b)\}. \quad (8.12)$$

This exact cutoff will matter in the error bound. Bins with $c_b \geq M$ have no layers and are omitted.

The walk is applied to an array indexed by actual endpoints. Thus, for each term and pair, its input is the single function

$$\sigma \mapsto T_{b,m}(x, y; \sigma) - T_{b,m-1}(x, y; \sigma),$$

with all contributions at that endpoint already summed. A path description is only a way to enumerate those contributions. In particular, the two budgets must be grouped at the same actual endpoint before their difference is measured. The following lemma establishes this identity for the path enumeration and also records the description length needed later by the controller.

Lemma 8.5 (Immediate paths). *For a row task of budget M , each immediate candidate in a slack- d group has an $O(d)$ -bit description, and every estimator child it uses has budget at most $M - d$. Summing valid candidates by their actual endpoints gives exactly the conditional-walk increments of the additive schedule, with the original word weights. A product first reaches its midpoint through a paired correction port; all subsequent right-factor queries there use the fixed right-table tuple, independent of the old source.*

Proof. A descriptor records the finitely many formula and copy-part choices, its scale and budget offsets, its sign in Equation (8.11), the generator word and guessed wraps, and a bounded number of constituent ports. An included scale obeys $|h_b| = O(d)$, and the walk has length $O(d)$. Lemma 7.3 therefore bounds all these records by $O(d)$ bits. Port alphabets are fixed by Lemma 6.5. There are only $O((1 + d)^2)$ choices of budgets and scales before the exponential word/variant choices.

Check each constituent port, the outer source and endpoint memberships, and the guessed wraps at the vertices where their coordinates can be evaluated. For a fixed endpoint and original

generator word, unique wrap validation leaves exactly its genuine conditional transformation. Give the surviving term its original walk-word probability, multiplied by the formula coefficients and sign. Guessed wraps do not change that probability. Constituent ports enumerate entries once. Hence summing over true equal endpoints is precisely the walk of the difference, even though its endpoint was not known when the variant was specified.

The first factor of every product is a correction, whose retained ports have bounded degree in both directions. Its paired move supplies the midpoint and a bounded return token. The right factor can then be queried using the common Z, Y tuple prescribed above; that tuple contains no old source. A direct terminal motion without an inverse port has no further factor query at its endpoint. These are combinatorial access requirements; the controller section implements the pairing and terminal operations. Finally, the slack definition gives $m \leq M - d$, also for the $m - 1$ term. $\square$

All reachable rate logarithms remain $O(B)$ when $M_{\text{top}} = O(B)$: a scale extension at slack d changes a reciprocal-rate logarithm by $O(d)$ and reduces budget by at least d . Stage resets have only their fixed stage rates, usage rates are minima of available rates, and inherited padding has at most one symbol per stage above the task. Summing along a dependency chain bounds all extensions by $O(M_{\text{top}})$. The field in the shared environment can therefore be chosen large enough for every used scale, including budget-zero endpoints of available differences. Omitted finer deterministic contributions need no hash tables.

8.6 The prefix guarantee

Compression uses conditional rarity to control the mean error and an absolute norm bound to control the noise. The prefix bounds below combine the mean estimate with conditional averaging and absolute bounds on the paired differences; their proof is in Section 8.8.

Lemma 8.6 (Conditional prefix bounds). *Assume exact endpoint grouping and the earlier-task bounds $C_b 2^{-p'}$ in the ordinary vector and root row metrics of Equation (8.8). Assume also that each sample function depends only on coordinates varied by its conditional walk. Let S_j be the prefix through slack d_j , S_j^{mean} its uniform slice mean, and $S_j^{\text{noise}} = S_j - S_j^{\text{mean}}$. For the true first row $d = (G(x, y)/a)_y$,*

$$\left\| \min \left(1, \frac{|S_j^{1, \text{mean}} - d|}{1 + d} \right) \right\|_{u, a; s} \leq c_* 2^{-p + B_1 d_j}, \quad (8.13)$$

$$\|S_j^{\text{noise}}\|_{u, a; s} \leq c_* 2^{-p - K_n d_j}. \quad (8.14)$$

The noise bound applies componentwise. Channel two has the same mean bound in its ordinary norm, extended by zero off Y_t ; vectors have both bounds in their vertex norm.

The exponent B_1 is fixed. Any prescribed K_n and arbitrarily small positive c_* can be attained by fixed task-separation and walk-length choices. Slice means at a specified pair or vertex are deterministic, apart from the stipulated zero-extension mask.

We can now estimate each mean prefix by telescoping the conditional means to budget $m_j(b)$. Its deterministic omission error and its child-estimation error are treated separately. The remaining noise will be bounded by applying the conditional spectral estimate to the endpoint functions defined in that proof.

Lemma 8.7 (Mean prefixes). *Assume the earlier task bounds with coefficient C_b in the ordinary vector and root row metrics of the precision convention. For every active task and prefix ending at slack d_j , the mean prefix differs from its target by at most $c_* 2^{-p+B_1 d_j}$. Use the clipped relative norm for channel one and the ordinary norm for channel two and vectors, with the prescribed membership masks.*

One fixed B_1 works independently of the later choice of task separation A_* . Increasing A_* makes c_* arbitrarily small.

Proof. Available scales. For an available scale, conditional expectations of Equation (8.11) telescope to its term at $m = m_j(b)$ in Equation (8.12). If this budget is zero, the term is zero and the inactive-child bound still applies. More generally, the child bound remains valid whenever its precision is nonpositive, by the bounded-output estimate for inactive tasks. In every case

$$M - m \leq d_j + c_0(1 + |h_b|) + 1.$$

Choose K_{prec} so that $c_0/K_{\text{prec}} < 1/8$. Equation (8.10) is then bounded by

$$C' C_b 2^{-p-A_*+d_j/K_{\text{prec}}} 2^{-|h_b|/4}.$$

The rate grid is geometric, so summing this bound over available scales costs a fixed constant. There are only a fixed number of formula and factor types. Unbinned calls use the same calculation without the scale sum. Minkowski and subadditivity of clipped relative discrepancies combine all these terms. Exact base contributions are already present in every prefix and have no error.

Wholly omitted scales. A bin with $c_b \geq M$ contributes no layer. We bound its deterministic contribution directly by Lemma 8.4. First we show that every such bin is fine. Put $r = \log_2(1/a) \geq 0$. A coarse bin has $0 \leq h_b \leq r$ because all grid rates are at most one. Activity gives $M > K_{\text{prec}}(r/2 + A_* i_*)$, even after discarding the nonnegative start-rate cost. With $K_{\text{prec}}/2 > c_0$ and A_* sufficiently large, this implies $M > \lceil c_0(1+r) \rceil$, so a wholly omitted bin is fine. For the direct correction term $b = f$, the same argument gives availability when $f \geq a$. When $f < a$, the admissibility condition $a \leq K_f f$ gives $|h_f| \leq \log_2 K_f$, and increasing A_* ensures $M > c_f$. Thus this term is never wholly omitted in an active task. The inequality $c_b \geq M$ implies

$$|h_b| \geq (M - 1)/c_0 - 1.$$

Lemma 8.4 and a geometric sum consequently bound all such omissions by $C 2^{-(s-1)M/(sc_0)}$. Since $p \leq M/K_{\text{prec}}$ and $K_{\text{prec}} > sc_0/(s-1)$, this is at most $C 2^{-p-\eta M}$ for some fixed $\eta > 0$, after absorbing the fixed rounding offset. Finally $M > K_{\text{prec}} A_* i_* \geq K_{\text{prec}} A_*$ makes its coefficient arbitrarily small as A_* increases. Taking, for example, any fixed $B_1 > 1/K_{\text{prec}}$ proves the claim, including every last-budget and no-budget cutoff loss. $\square$

8.7 Absolute difference bounds and averaging noise

The conditional-mean errors are now bounded. To control the noise left by finite averaging, we need an absolute norm bound on each paired difference before the walk is applied. This coarser bound will also be used for the fingerprint comparison.

Lemma 8.8 (Absolute differences). *When the earlier tasks have error bounds with coefficient C_b , the ordinary raw norm of each unaveraged paired difference in a slack- d group is at most*

$$CC_b 2^{-p-A_*+C_0d}, \quad (8.15)$$

with C_0 independent of the walk-length coefficient. For two versions of a budget- r sample term, a bound on their child discrepancies alone suffices: if those discrepancies are at most $2^{-p_r}\lambda_r$ in their respective prescribed metrics, their raw discrepancy is at most $2^{-p+C'_0d}\lambda_r$, summed over its finitely many child types. Here p'_r is each child's precision from Equation (8.8). For a paired layer this gives $2^{-p+C'_0d}(\lambda_m + \lambda_{m-1})$, or the corresponding maximum over children after enlarging the constant; all these children have budget at most $M - d$.

The bounds also hold after a fixed measure-preserving variant and restriction to its valid endpoints. Summing separately over all variants may enlarge C'_0 by a constant depending on the walk length.

Proof. Subtract the same true sample-table term from the budget- m and budget- $(m-1)$ terms, and use the triangle inequality. A product has only constantly many left paths in the union of the compared rows. Ordinary bounded-factor differences and the row and incoming caps therefore give the sums used in Lemma 8.3, without any averaging. For a coarse bin the absolute coefficient costs b/a in the norm rather than its s -th root. For a fine bin the importance factor can stay inside the power. Both are bounded by $2^{O(|h_b|)}$. Equation (8.9) pays the only separate possibly large domain ratio, f/u , as before. All other scale and accuracy changes are $2^{O(d)}$ because $|h_b| = O(d)$ and $M - m = O(d)$ in this group. Ranks and bounded source functions obey the same pointwise comparisons. This proves Equation (8.15). Replacing comparison to truth by a comparison of the two coupled child versions gives the second assertion by the identical inequalities; independence is still unnecessary.

For a fixed guessed variant φ , let V_φ be its zero-one validity mask. For an inherited source, validity gives

$$X_u(x; \sigma) V_\varphi(x, y; \sigma) \leq X_u(x; \varphi \sigma).$$

The corresponding endpoint and midpoint hits are likewise retained by validity. In an error sum evaluated at $\varphi \sigma$, replace these masks by the required transformed child-domain masks and discard the remaining guards. The substitution $\eta = \varphi \sigma$ then preserves full-environment expectation by Lemma 7.3; the nominal weights $1/u$ and $1/N$ are unchanged. This gives the child-domain sums just bounded. When a source is released at a midpoint, the subsequent incoming-cap sum supplies the separate factor f/u . When combining the paired terms, combine their contributions at the same actual endpoint before measuring the difference. The number of distinct variant descriptors is $2^{O(d)}$, so treating them separately merely enlarges the crude exponent. The estimate before such separate counting is a statement about the single-seed sample function and has no dependence on the eventual walk length coefficient. $\square$

8.8 Proof of the conditional prefix bound

Proof of Lemma 8.6. The mean assertion is Lemma 8.7. For a difference layer, write F_{xy} for its sample function on $\mathcal{C}_{xy}^{u,a}$, $\mathcal{A}_{xy}$ for the conditional walk operator of length $\ell(M - m)$, and Π_{xy} for uniform averaging on that slice. Put $w_{xy} = \Pr(\mathcal{C}_{xy}^{u,a})/(Nu)$ and $r_{M-m} = C2^{-c\ell(M-m)}$. Lemma 7.1 gives a uniform operator bound, so

$$\sum_{x,y} w_{xy} \|(\mathcal{A}_{xy} - \Pi_{xy})F_{xy}\|_{L_s}^s \leq r_{M-m}^s \sum_{x,y} w_{xy} \|F_{xy}\|_{L_s}^s.$$

The slice weights make the sum on the right precisely the integrated raw norm to the s th power, bounded by Lemma 8.8. Thus the uniform operator estimate applies directly to the integrated input bound, even when different slices have different input norms. For the second channel replace Y_a by Y_t in the slice and its weight; the mask stays Y_t throughout. Vectors use the same calculation on their one-vertex slices. No extra conditioning factor appears.

There are $O((1+d)^2)$ budget/scale terms in a group. Choose ℓ so that the contraction dominates $2^{C_0 d}$ and this polynomial, leaving at least $2^{-(K_n+1)d}$. Summing groups with slack at least d_j gives Equation (8.14). Exact base terms have no noise. This contraction applies to the paired difference as one function of the same seed. The sum of valid variants is the conditional walk, so the gap applies to that sum before any separate variant counting. Increasing A_* supplies an arbitrarily small coefficient in both this estimate and the mean bound. $\square$

The determinism hypothesis in Lemma 8.6 applies at each fixed pair or vertex, in the task’s effective environment as in Remark 7.4. Uniform averaging must integrate every coordinate on which its sample function depends. In Section 10, the comparison calculation removes dependence on the current block’s salt, while the parent walk varies all earlier salts used by lower-block calculations; Lemma 10.2 verifies this hypothesis. The resulting first-channel mean is a deterministic number, and the second-channel mean has only its prescribed Y_t membership mask. These are the means used in the conditional heavy-count argument of Lemma 9.7.

9 Compression of additive row estimates

The additive schedule produces a raw prefix on all retained endpoints after each slack group. We replace each array by a finite list of exceptions to a common default. Our objective is to control the bounded row functions, including their scores and support witnesses; a large raw entry need not have a small absolute error.

The update changes only small entries until its first overflow, when list capacity forces its threshold above the prescribed floor. Before then, the total earlier change is much smaller than the next floor. At the first overflow, either many retained endpoints are truly heavy or the prefix errors account for the excess. The heavy-count event is rare even after conditioning on one specified endpoint. We use this rarity for the deterministic conditional-mean errors, and use the absolute norm bound for the remaining noise. To control later updates, we compare them with a calculation that starts from the true target plus the error from earlier compression and adds no further increments. The remaining compression steps preserve that reference calculation’s row-function values.

We first fix the increments supplied by exact endpoint grouping and prove the comparison without range clamps. We then show that clamping preserves the relevant row functions and assemble the exact-grouping error reserve. Section 10 compares the actual and comparison calculations with those clamps in place.

9.1 The row interface

Fix a row task, its start rate u , and its first-channel endpoint rate a . Write $d_y \geq 0$ for its true first raw input, in units of a . Its second raw input, when present, is $v_y = \zeta \mathbf{1}_{\{Y_t(y)=1\}} q_y$. Here q_y is the deterministic conditional mean, $0 \leq q_y \leq 1$, and $q_y = 0$ whenever d_y is below the positive-start threshold of the preparatory function h . Thus d_y and q_y are deterministic at each specified endpoint; v_y carries the prescribed narrower membership mask. There is a fixed constant C_{mass} such that

$$a \sum_y d_y \leq C_{\text{mass}}. \quad (9.1)$$

The endpoint sets in this section are the retained sets, with the stipulated common padding on their hash arguments.

We apply a fixed finite family of row functions before incoming multiplication to a raw array. Denote this family by $\mathcal{H}$. For a raw row array Z , $H(Z; y)$ denotes the value of $H \in \mathcal{H}$ at endpoint y , using the entry of Z there and the rank of its entire first row. For a two-component row (d, v) we also write $H(d, v; y)$. The family includes the main pre-functions, auxiliary scores, and witnesses required by the preceding construction. Each function uses the first entry, possibly the second entry, and the K th largest first entry. For each member $H \in \mathcal{H}$ there are fixed positive thresholds g_H, e_H , with $3g_H < e_H$, such that its row gate is zero when that rank is at least g_H , and its entry factor is zero when the first entry is at most e_H . The gate is constant and full in a neighborhood of zero. The s th roots of the output functions are bounded and Lipschitz in their relevant entry and rank arguments, where $s = 8$. All these arguments can be clipped to fixed sensitivity ranges without changing the functions. These are exactly the threshold buffers and saturations imposed when constructing the tables; the thresholds for the required functions are listed in Table 1. In particular, for every real raw array, including arrays with negative entries,

$$\sum_{H \in \mathcal{H}} \#\{y : H(y) \neq 0\} \leq C_{\text{supp}}, \quad (9.2)$$

where C_{supp} is fixed. Indeed a nonzero gate implies that fewer than K entries exceed its close threshold, and every positive output entry exceeds the larger entry threshold. This observation applies also to the broader witnesses.

Choose $\eta > 0$ below all positive entry thresholds, all positive-start thresholds for rank sensitivity, and the threshold for h , with enough fixed separation that perturbations of size several times η remain inside these flat regions. Choose $\zeta \leq \eta/100$. These constants are fixed throughout the proof.

Let $d_1 > d_2 > \dots > d_{m_*} = 1$ be the decreasing powers of two indexing slack groups. Empty groups are included. Let $S_i = (S_i^1, S_i^2)$ be the uncompressed prefix through group i , and let $\Delta_i = S_i - S_{i-1}$, with $S_0 = 0$. Define the clipped endpoint errors and their row supremum by

$$e_i(y) = \max \left\{ \min \left(1, \frac{|S_i^1(y) - d_y|}{1 + d_y} \right), \min(1, |S_i^2(y) - v_y|) \right\}, \quad (9.3)$$

$$e_i = \sup_y e_i(y).$$

omitting channel two when absent. Write $\beta_i(y)$ for the maximum of the mean discrepancies, with the same clipping and first-channel normalization as in Equation (9.3), and $\gamma_i(y)$ for the maximum absolute prefix noise. Thus $e_i(y) \leq \beta_i(y) + \gamma_i(y)$. At a specified row and endpoint, the first-channel mean discrepancy is deterministic. The second-channel mean discrepancy is a deterministic magnitude times its narrower membership mask. Prefix noise may depend on the shared environment. The deterministic assertion is a hypothesis of the compression argument: the slice average defining a mean must vary every environment coordinate on which its sample function depends. Lemma 8.6 states this requirement, and Lemma 10.2 verifies it for the comparison calculations to which we will apply the result. We will use conditional rarity to control these fixed mean-error magnitudes; the noise will be controlled by its unconditional norm bound.

For row integration use

$$\int F d\mu = \frac{1}{N} \mathbb{E} \sum_x \frac{X_u(x)}{u} F(x).$$

Endpoint integration means inserting the sum over retained endpoints in this expression. The prefix

estimates used as hypotheses below are

$$\left(\int \sum_y \beta_i(y)^s d\mu \right)^{1/s} \leq c_* 2^{-p+B_1 d_i}, \quad \left(\int \sum_y \gamma_i(y)^s d\mu \right)^{1/s} \leq c_* 2^{-p-K_n d_i}. \quad (9.4)$$

Lemma 8.6 supplies these estimates for the additive schedule. Here and below endpoint sums use the relevant masks. In particular these bounds control the $L_s(\mu)$ norms of the corresponding row suprema, which we denote by β_i and γ_i .

9.2 The update map

A compressed array has default $(z, 0)$ and a finite list of exceptions; initially $z = 0$ and the list is empty. Adjoin arbitrarily many dummy positions with zero targets and increments and with current value equal to the default. By Remark 6.3, a rank of order r is computed from the list and at least r default copies. Extra listed positions equal to the default do not change it. Dummies are used only for ranks: error sums count real retained IDs, and Lemma 6.4 prevents default positions from exposing row ports. At group j add Δ_j by actual endpoint equality, obtaining w_y , and set

$$\begin{aligned} \rho_j &= 2^{-c_1 d_j}, & n_j &= 2^{c_2 d_j}, & m_y &= \max_\nu |w_y^\nu|, \\ u_0 &= \max\{z, \rho_j, \text{rk}_{n_j}(m)\}, \\ z' &= \max\{u_0, \min(\text{rk}_K(w^1), 3u_0)\}. \end{aligned} \quad (9.5)$$

Here c_1, c_2 are fixed sufficiently large positive integers, and c_2 is chosen also so that $n_j \geq K$ in every group. With $b = (z', 0)$, the updated array is

$$T_j^\nu(y) = b^\nu + \text{clip}(w_y^\nu - b^\nu, \pm 7(m_y - 2u_0)_+). \quad (9.6)$$

The notation $\text{clip}(r, \pm A) = \max(-A, \min(r, A))$ is used.

Lemma 9.1 (Local compression properties). *At group j , the update in Equations (9.5)–(9.6) replaces entries of size at most $2u_0$ by the new default and leaves entries of size at least $3u_0$ unchanged. Every updated coordinate lies between its old value and the new default, and the default is nondecreasing.*

As a map of the full array and old default, the update is uniformly Lipschitz in the sup norm. All exceptions to its new default fit in a list of at most n_j real positions drawn from the previous list and the new candidates. These conclusions hold for arbitrary real input values under the stated default and rank conventions.

Proof. The zero-radius assertion is immediate. Since $z' \leq 3u_0$, if $m_y \geq 3u_0$ then

$$|w_y^1 - z'| \leq m_y + 3u_0 \leq 7(m_y - 2u_0), \quad |w_y^2| \leq 7(m_y - 2u_0),$$

which proves identity there. Clipping a displacement toward zero proves the coordinatewise assertion. Also $z' \geq u_0 \geq z$.

If two input arrays and their defaults differ by at most δ , their size arrays, size ranks, u_0 's, and first-coordinate ranks differ by at most δ . Their new defaults differ by at most 3δ , and their clipping radii by at most 21δ . The elementary bound

$$|\text{clip}(r, \pm A) - \text{clip}(r', \pm A')| \leq |r - r'| + |A - A'|$$

therefore gives, for example, a Lipschitz constant 28 for the full update. This constant is independent of the number and magnitudes of entries.

Fewer than n_j positions have size strictly larger than u_0 . A coarse test supported above u_0 and full above $2u_0$ retains all possible exceptions within the stated cap. A position outside the old list and new candidate set has input $(z, 0)$, has size at most u_0 , and hence becomes the new default. No other position need be inspected. The selected list may retain some real positions that become exactly default after compression. Such positions still represent the same full array. No padding dummy is selected, since its input size is at most u_0 . $\square$

In particular an absolute perturbation introduced at group i propagates through the remaining updates and the final root functions with factor at most

$$CC_1^{m_*-i+1} \leq C(d_i + 1)^{\log_2 C_1}, \quad (9.7)$$

for a fixed $C_1 > 1$, enlarged below as needed. This statement concerns full arrays and defaults; it does not require their exceptional lists to agree.

9.3 The compression estimate and its use

The integrated compression bound below controls the row-function errors used in the fingerprint comparison. Under the common-increment and range-margin conditions of Lemma 9.3, clamping preserves the values of the row functions covered by this bound. The compression bound is proved in Sections 9.6 and 9.7.

Proposition 9.2 (Integrated compression bound). *Assume the row-interface hypotheses, including deterministic mean-error magnitudes, the mass bound (9.1), and prefix bounds (9.4). Choose the floor exponent c_1 sufficiently large, then the capacity exponent c_2 , and a sufficiently large noise exponent K_n . The final unclamped compressed row satisfies*

$$\left(\int \sum_{H \in \mathcal{H}} \sum_y |H(T_{m_*}; y)^{1/s} - H(d, v; y)^{1/s}|^s d\mu \right)^{1/s} \leq C_{\text{buf}} c_* 2^{-p}. \quad (9.8)$$

The constant C_{buf} is independent of the number of groups, row width, and precision. Thus reducing c_* gives any prescribed fixed error coefficient.

9.4 Bounding numerical ranges

In exact endpoint grouping, the additive increments are independent of which exceptions are stored. The path description and bounded child ranges give a fixed D for which $\|\Delta_i\|_\infty \leq 2^{Dd_i}$, after increasing D to include immediate term counts. This also bounds a group when all its same-key increments are summed into one coordinate, with a possibly larger fixed D . Fix a common gate closure level $G \geq \max_H g_H$, and then choose $L > 10G$ above every sensitivity threshold. Let $R_i = 2^{B_2 d_i}$ and choose B_2 sufficiently large that

$$R_i \geq 10L + 10 \sum_{k>i} \|\Delta_k\|_\infty \quad \text{for every group } i. \quad (9.9)$$

Dyadic grouping permits this uniform choice because $d_{i+1} = d_i/2$. After each compression, clip every coordinate and the baseline componentwise to $[-R_i, R_i]$. An off-list coordinate is clipped exactly as the baseline is, so it remains a default coordinate. These clamps therefore preserve the finite-list representation.

Lemma 9.3 (Range clamping). *Compare the compression calculation with and without the componentwise range clamps $[-R_i, R_i]$, including their defaults, using the same increment arrays. If Equation (9.9) holds, their final values H agree for every $H \in \mathcal{H}$. Each common clamp is also nonexpansive in the sup norm, so it preserves the Lipschitz propagation bound for calculations with possibly different increments.*

For a vector sum, a clamp beyond its final sensitivity range plus all remaining additive variation likewise preserves the final clipped value. Its comparison bound is nonexpansive.

Proof. Compare the clamped and unclamped calculations with the same increments. For the fixed level $G < L/10$, once both baselines are at least G , all gates are zero because of the dummies, and remain so: compression cannot lower the baseline and every clamp bound exceeds G . It remains to compare the calculations before this happens.

Maintain the following invariant. Their baselines agree while below G . Their coordinates either agree, or have matching signs and magnitudes well beyond L ; any such differing component has enough remaining margin that the total future increments cannot bring it into $[-L, L]$. Suppose a component is first changed by the clamp at group i . Its new magnitude is R_i , and the unclamped magnitude is at least R_i , with the same sign. As long as the baselines are below G , a differing position has size above $L > 3G$ and compression is the identity there. After all later additions its magnitude is therefore at least

$$R_i - \sum_{k>i} \|\Delta_k\|_\infty > L,$$

by Equation (9.9). A subsequent clamp supplies the same inequality with its own index. The margin therefore holds throughout the remaining updates.

After an addition, entries whose components differ therefore have size larger than L in both calculations. Their size-rank statistics either are equal below G or are both at least G : differing values lie above this comparison range and all other size values agree. In the latter case both new baselines close the gates. In the former case the two values of u_0 agree and are below G . Only the first-coordinate rank clipped at $3u_0$ is needed to compute z' . Differing positive first components are above this range; differing negative first components are below zero and do not affect its nonnegative clipped value. Consequently the two new baselines agree as well. If this baseline is at least G , both calculations close. Otherwise every differing position lies in the compression identity region because its size exceeds $L > 3G$; equal coordinates at other positions undergo the same update. Thus the invariant continues.

Before closure, all rank arguments and all entry arguments within their sensitivity ranges agree. Differing large components have matching signs and identical saturated behavior. The final row functions therefore agree. A scalar clipping map is 1-Lipschitz, which gives the comparison assertion. For a vector sum, the same tail-margin argument ensures that a clipped value stays on the same saturated side until the final bounded clipping; no rank argument is needed. $\square$

The equality assertion of Lemma 9.3 is used for the comparison error bound: exact endpoint grouping supplies the same increment arrays regardless of the chosen list presentation. It preserves the values of every relevant $H \in \mathcal{H}$, including scores and the specified support witnesses. It makes no assertion that finite-accuracy access tests or selected descriptors agree with those of the unclamped calculation. In the adaptive comparison of Section 10, both the actual and comparison calculations are clamped from their definition. Their possibly different bucket increments are compared directly, and the nonexpansiveness part of Lemma 9.3 preserves the local perturbation bound. This use allows different real representatives, including retained positions whose current value equals the default.

9.5 The error reserve with exact endpoint grouping

The prefix estimates and the stated compression bound give the current task's error estimate under exact endpoint grouping. The hypotheses still require each conditional average to vary every random coordinate used by its sample function. We combine the compressed bounds before incoming multiplication with the table assembly comparison; the next section will verify this averaging hypothesis for the comparison calculation.

Proposition 9.4 (Error reserve with exact endpoint grouping). *Assume all earlier tasks satisfy the table support and common-parameter conditions and have error at most $C_b 2^{-p'}$ in their prescribed metrics. In the current task, group endpoints exactly and let every conditional walk vary all coordinates used by its sample function. With fixed constants chosen in the order below, each active row output, score computed before incoming multiplication, support witness, and assembled correction has root error at most $(C_b/2)2^{-p}$ after compression and range clamping. Each final clipped vector has the same bound in its ordinary vertex metric.*

Proof. To apply Proposition 9.2, use the deterministic-mean and absolute-noise bounds of Lemma 8.6. The first target is nonnegative with bounded deterministic row mass before division by a , and its second target is bounded by a fixed multiple of ζ and vanishes below the preparatory support threshold. The buffered row functions and their witnesses have fixed Lipschitz constants for their s th roots and fixed support bounds. These verify its remaining hypotheses. The compression proposition shows that all their errors are a fixed multiple of $c_* 2^{-p}$ when its convergent series of floors, capacities, and noise weights are chosen. This includes the quantities before incoming multiplication required by the incoming comparison. Exact endpoint grouping gives common increment arrays for the clamped and unclamped calculations. Lemma 9.3 therefore preserves these row-function values, including scores and witnesses, so the same estimates hold after clamping.

Lemma 6.6 then charges the incoming gates by those same pre-output norms. Its separate column-vector calls have a fixed positive budget slack and decrease task rank. Equation (8.9) pays their broad-domain factor, and increasing A_* makes their remaining coefficient as small as required. A final vector needs no compression; its clipping is nonexpansive and the bounds at $d_j = 1$ apply directly.

Choose the fixed table range/support bound C_b first. Choose c_0 and K_{prec} to retain the margin in Equation (8.10) and fix B_1 . The compression proof then chooses its floor exponent c_1 , capacity exponent c_2 , and required noise decay K_n . Choose task separation and walk length sufficiently large to supply the needed small c_* . The total coefficient can thus be made at most $C_b/2$. Inactive tasks retain their separate trivial C_b bound. The remaining half of the active-task error bound is reserved for the coupled fingerprint error. $\square$

The reserve applies to a local exact-grouping calculation with the stated averaging property. The actual algorithm will also use short endpoint keys. In Section 10 we define an exact-grouping comparison within each budget block, verify that it has this averaging property, and bound its difference from the algorithm.

9.6 Deterministic comparisons for the compression bound

Call group j an overflow if its u_0 exceeds ρ_j . Before the first overflow every compression change has magnitude at most $6\rho_i$ and occurs only at input size below $3\rho_i$. Because successive slacks halve, choosing c_1 large gives, uniformly in j ,

$$6 \sum_{i < j} \rho_i \leq \rho_j/100, \quad 6 \sum_i \rho_i \leq \eta/100. \quad (9.10)$$

The same estimates cover dummy defaults. Before the first overflow, $u_0 = \rho_i$ at each earlier group i , so the threshold formula gives $z_i \leq 3\rho_i$, where z_i is the default after group i . In particular, the preceding default is at most $3\rho_{j-1} < \rho_j$, by Equation (9.10). Thus the old default cannot itself trigger the first overflow.

Lemma 9.5 (Earlier compression error and stationary reference). *Suppose group j is the first overflow in the unclamped calculation, and put $L_{j-1} = T_{j-1} - S_{j-1}$. Then $\|L_{j-1}\|_\infty \leq 6 \sum_{i < j} \rho_i$, and for every endpoint with $d_y \geq \eta$,*

$$\max_v |L_{j-1}^\nu(y)| \leq C \sum_{i < j} \rho_i e_i(y). \quad (9.11)$$

Apply the remaining compression maps to the starting array $(d, v) + L_{j-1}$ with the same preceding default and zero further increments. All outputs $H \in \mathcal{H}$ remain unchanged in this calculation, and their root discrepancy from $H(d, v; y)$ is at most $C \sum_{i < j} \rho_i e_i$, uniformly in H, y .

Proof. The uniform earlier compression error bound follows by adding the changes described above. If a change occurs at group i at a position with $d_y \geq \eta$, then $|w_y^1| \leq 3\rho_i$ and the accumulated compression error is much smaller than ρ_i . Thus

$$\frac{|S_i^1(y) - d_y|}{1 + d_y} \geq \frac{d_y - 3\rho_i - 6 \sum_{k < i} \rho_k}{1 + d_y} \geq c_\eta > 0.$$

The change of size at most $6\rho_i$ is consequently bounded by $C\rho_i e_i(y)$. Adding such changes proves Equation (9.11).

In the reference array negative first magnitudes and all second magnitudes are at most η , by the choices of ζ and the floors. Let q be its initial K th first-coordinate rank. The default is nonnegative, so $q \geq 0$. If $q > \eta$, then $n_i \geq K$ implies that the n_i th size rank is at most q . The floor and preceding default are also at most q , so $u_0 \leq q$ and $z' \leq q$. Entries at most q stay at most q . Entries at least q are unchanged if $q \geq 3u_0$; otherwise $z' = q$, and they stay at least q . The K th rank is therefore unchanged. Negative first magnitudes and second magnitudes do not grow, so this argument repeats at every subsequent group. If instead $q \leq \eta$, the size rank, u_0 , and new default are at most η , and the next K th rank is again at most η .

Consider an individual row function. In the first case its rank stays fixed. If its gate is not already zero, $q < g_H$, so every entry in its support has first coordinate greater than $e_H > 3q \geq 3u_0$ and both channels there are untouched. A position below entry support cannot cross into support because the default is at most $q < e_H$. In the second case the rank stays in the common full-gate region; the same identity argument applies to every position in entry support. Thus each reference output is preserved.

Finally, positions with $d_y < \eta$ stay below every entry threshold and cannot change a rank within its sensitivity region. Perturbations at all other positions satisfy Equation (9.11). Clip first entries and ranks to the fixed sensitivity ranges and use Lipschitz continuity of the s th roots of the row functions. This proves the last assertion. $\square$

If no overflow ever occurs, the same proof, together with the final prefix error, gives the bound

$$\sup_{H,y} |H(T_{m_*}; y)^{1/s} - H(d, v; y)^{1/s}| \leq C \left(e_{m_*} + \sum_i \rho_i e_i \right). \quad (9.12)$$

If the expression on the right is not small, bounded output ranges give the assertion after enlarging C . Otherwise low-target positions remain below sensitivity. At other positions clipped first entry discrepancies are bounded by a constant times their relative error, and second entry discrepancies by their absolute error. Adding the earlier compression error estimate and applying the Lipschitz bounds for the s th roots proves the claim.

Lemma 9.6 (Stability after the first overflow). *At a first overflow in group j , compare the final unclamped outputs with the zero-increment reference calculation of Lemma 9.5. Their discrepancy satisfies*

$$\sup_{H \in \mathcal{H}, y} |H(T_{m_*}; y)^{1/s} - H((d, v) + L_{j-1}; y)^{1/s}| \leq C \sum_{i \geq j} C_1^{m_* - i + 1} e_i, \quad (9.13)$$

The reference outputs are preserved by the remaining updates. The constants are independent of the largest true first entry.

Proof. The factors $1 + d_y$ can turn small relative errors into large absolute errors. We control them using the row gate when many entries are large, and saturation when fewer than K are large. Write E for the sum in Equation (9.13) without the initial constant. If E exceeds a sufficiently small fixed constant, bounded output ranges suffice. Otherwise all clipped errors involved are strictly below 1. Hence

$$|S_i^1(y) - d_y| \leq (1 + d_y)e_i, \quad |S_i^2(y) - v_y| \leq e_i,$$

and later increments have first magnitude at most $(1 + d_y)(e_i + e_{i-1})$ and second magnitude at most $e_i + e_{i-1}$. The input difference at group j is precisely $S_j - (d, v)$; both calculations include the same earlier compression error.

Choose G larger than every gate close threshold. A baseline at least G closes all gates, since the dummy entries force every first-coordinate rank to be at least the baseline. Closure is permanent under the unclamped update. Choose L_0 much larger than G , all sensitivity thresholds, and 1. If at least K true first entries exceed L_0 , the reference gates are closed. In the perturbed calculation either some $u_0 \geq G$, which forces permanent closure, or every $u_0 < G$. In the latter case those K coordinates stay larger than half their true value: their initial error and the total later additive variation are at most $C(1 + d_y) \sum_{i \geq j} e_i$, and their first value is then larger than $3G$. Inductively each such position lies in the compression identity region. These K positive entries close all gates as well.

It remains to consider the case of fewer than K true entries above L_0 . Lemma 9.5 bounds every reference rank and threshold by $2L_0$. Fix L_1 much larger than L_0 and all upper saturation thresholds. At the at most $K - 1$ positions where $d_y > L_1$, replace the first input at group j , in both calculations, by one common positive sentinel larger than L_1 , and give that coordinate zero later first-channel increments. Do not modify second channels. The reference ranks, thresholds, and outputs are unchanged: these coordinates stay above the relevant ranks and in all upper saturation regions. The sentinels are comparison devices in this proof. They need not be identified or stored by the compression algorithm. Fewer than K such coordinates occur, so replacing them by common large values removes their possibly large absolute differences while preserving every rank that lies in a sensitivity range.

All remaining first-coordinate input errors and increments now have absolute bounds by $C(1 + L_1)e_i$ and $C(1 + L_1)(e_i + e_{i-1})$, respectively. Lemma 9.1, iterated from j to the last group, therefore

bounds every array and threshold discrepancy by CE , on taking C_1 larger than the fixed update constant. If E is sufficiently small, all these thresholds remain below $2L_0 + 1$. The final root-output discrepancy is at most CE .

To remove the modification in the perturbed calculation, argue inductively. Every original coordinate with $d_y > L_1$ remains above $d_y/2$, because its total raw additive variation is at most $C(1 + d_y) \sum e_i$. It is thus above the bounded thresholds and in the identity and saturation regions. Replacing fewer than K such positive coordinates by sentinels cannot change either a K th first rank or an n_i th size rank whose sentinel-computed value is at most $2L_0 + 1$: both the original and modified values are above that bound, and $n_i \geq K$. The defaults and thresholds therefore agree, the compression is identity in both channels at these positions, and all other coordinates agree. The relevant output behavior is the same. This completes the induction and the proof. $\square$

9.7 Charging overflow events

The deterministic comparisons leave two cases to integrate: no overflow, and the first overflow. To handle the latter, we separate a large retained set of true heavy endpoints from overflow caused by errors at light endpoints. For this subsection let $\mathcal{O}_j$ denote the event that group j is the first overflow. These events are pairwise disjoint. The heavy-count event defined next depends only on the true row and its retained endpoints; it does not itself assert that any overflow occurs.

At first overflow in group j , the previous default is smaller than ρ_j . Thus at least n_j retained real positions have input size strictly greater than ρ_j . Define the true heavy-count event

$$\Theta_j = \{\#\{y \text{ retained} : d_y > \rho_j/4\} \geq n_j/2\}.$$

The dummy positions do not contribute to this count, since their input is the preceding default. On $\mathcal{O}_j \cap \Theta_j^c$, at least $n_j/2$ of the overflowing positions have $d_y \leq \rho_j/4$. Their second targets vanish, because the floors lie below the h -threshold. Equation (9.10) then implies $e_j(y) \geq c\rho_j$ at all these positions. Consequently the indicator of this first overflow outside Θ_j is bounded by

$$\frac{C}{n_j \rho_j^s} \sum_y e_j(y)^s. \quad (9.14)$$

A bounded row-output error may be charged to this indicator. On $\mathcal{O}_j \cap \Theta_j$, we instead use the stability estimate and the conditional rarity of Θ_j at each endpoint.

Lemma 9.7 (Conditional rarity of a true heavy count). *Under the deterministic target and mean-error hypotheses of Section 9.1, the heavy-count event Θ_j satisfies, for every specified row and endpoint,*

$$\Pr[\Theta_j \mid X_u(x) = Y_a(y) = 1] \leq \frac{C}{n_j \rho_j}. \quad (9.15)$$

The same bound holds with $Y_t(y) = 1$ in place of $Y_a(y) = 1$. For every prefix mean-error row supremum β_i ,

$$\|\mathbf{1}_{\Theta_j} \beta_i\|_{L_s(\mu)} \leq C c_* 2^{-p+B_1 d_i} (n_j \rho_j)^{-1/s}. \quad (9.16)$$

Proof. By Equation (9.1), the deterministic heavy set

$$A_j(x) = \{y : d_y > \rho_j/4\} \quad \text{satisfies} \quad |A_j(x)| \leq \frac{4C_{\text{mass}}}{a\rho_j}.$$

Conditional on one specified endpoint hit, each other endpoint in this set is retained with probability at most Ca , by Lemma 4.1. This remains true if the specified hit uses the narrower rate. The forced endpoint contributes at most 1, so the expected retained heavy count is at most $1 + Ca|A_j(x)| \leq C'/\rho_j$. The independent start condition has no effect. Markov's inequality at the threshold $n_j/2$ proves Equation (9.15). This calculation uses the conditional pair bound and linearity of expectation; it permits dependence among the heavy-endpoint hits.

For the first-channel mean error at a fixed row and endpoint, write its deterministic magnitude as $b_i(y)$. Then

$$\mathbf{1}_{\Theta_j} \sup_{y: Y_a(y)=1} b_i(y)^s \leq \sum_y \mathbf{1}_{\Theta_j} Y_a(y) b_i(y)^s.$$

Condition on the start and that endpoint before taking expectations. Equation (9.15) bounds the resulting expectation by $C/(n_j \rho_j)$ times the endpoint-integrated mean-error power. Indeed the factor $b_i(y)^s$ is fixed under this conditioning, so

$$\mathbb{E}[\mathbf{1}_{\Theta_j} X_u(x) Y_a(y) b_i(y)^s] \leq \frac{C}{n_j \rho_j} \mathbb{E}[X_u(x) Y_a(y) b_i(y)^s].$$

The second-channel mean error has a deterministic magnitude multiplied by $Y_t(y)$; repeat the same argument using the narrower conditional bound. Taking roots and using Equation (9.4) proves Equation (9.16). The conditioning step uses the deterministic mean-error magnitude at the specified endpoint. Prefix noise can depend on the heavy-count event, so no analogous conditional estimate is asserted for it. $\square$

Proof of Proposition 9.2. Equation (9.2) converts every row sup bound above into a bound for its summed output power with a fixed factor. In fact the support of a discrepancy lies in the union of the true and computed supports, whose total size over $\mathcal{H}$ is at most $2C_{\text{supp}}$. Thus this step introduces no factor from the ambient number of endpoints. The earlier compression error contributions have norm at most

$$Cc_* 2^{-p} \sum_i \rho_i (2^{B_1 d_i} + 2^{-K_n d_i}).$$

On the event of no overflow, Equation (9.12) adds the final-prefix cost, at most $Cc_* 2^{-p} (2^{B_1} + 2^{-K_n})$. By Equation (9.14) and the disjointness of the possible first-overflow events, the total error power from the events $\mathcal{O}_j \cap \Theta_j^c$ is at most

$$Cc_*^s 2^{-sp} \sum_j (n_j \rho_j^s)^{-1} (2^{sB_1 d_j} + 2^{-sK_n d_j}). \quad (9.17)$$

For a first overflow on Θ_j at j , use Lemma 9.6 and split $e_i \leq \beta_i + \gamma_i$. For the mean part, enlarge the actual first-overflow event to Θ_j , apply Equation (9.16), and use Minkowski's inequality. Its total norm is bounded by

$$Cc_* 2^{-p} \sum_j (n_j \rho_j)^{-1/s} \sum_{i \geq j} C_1^{m_* - i + 1} 2^{B_1 d_i}. \quad (9.18)$$

For the noise part, use the uniqueness of the first-overflow group on each row. Its tail noise sum is bounded pointwise by the single full sum $\sum_i C_1^{m_* - i + 1} \gamma_i$, regardless of dependence between noise and overflow. More explicitly,

$$\sum_j \mathbf{1}_{\mathcal{O}_j} \sum_{i \geq j} C_1^{m_* - i + 1} \gamma_i \leq \sum_i C_1^{m_* - i + 1} \gamma_i,$$

because at most one indicator on the left is nonzero. The absolute prefix-noise bound therefore gives norm at most

$$C_{c_*} 2^{-p} \sum_i C_1^{m_*-i+1} 2^{-K_n d_i}, \quad (9.19)$$

without an extra sum over j .

Choose $c_1 > B_1$ large enough for the floor conditions. Choose $c_2 > sc_1 + sB_1$ with a fixed positive margin and also large enough that $n_j \geq K$. Equation (9.17) then converges uniformly. For its mean term the exponential factor is exactly $2^{-(c_2-sc_1-sB_1)d_j}$. For Equation (9.18), its inner sum is at most $C(d_j+1)^A 2^{B_1 d_j}$ for a fixed A , by Equation (9.7) and dyadic grouping; the stronger choice of c_2 just made ensures convergence of the outer sum, whose exponential factor is $2^{-((c_2-c_1)/s-B_1)d_j}$. Taking K_n sufficiently large handles Equation (9.19). The earlier compression error sum converges by $c_1 > B_1$. All constants are fixed before choosing the arbitrarily small prefix coefficient c_* , giving Equation (9.8) and the asserted spare coefficient. $\square$

10 Adaptive fingerprints within budget blocks

We now replace endpoint comparisons at small budget slacks by short keys. The endpoint universe has size $2^{O(B)}$, whereas the key range has size B . These comparisons change the computed arrays. To control the change, we compare the algorithm with a calculation that uses exact equality within one budget block and the actual algorithms below that block.

This comparison has two roles. Its conditional averages give the deterministic means needed by the compression reserve, and its candidate endpoints form a set independent of the current block's key. We charge actual paths leaving that set to errors in lower calls, then bound collisions inside it. The resulting coupled induction proves the error bound for the actual estimator.

We use the root error metric $\mathfrak{d}_{u,a}$ for bounded nonnegative row outputs, its ordinary version $\|\cdot\|_{u,a;s}$ for raw arrays, and the ordinary vertex metric for vectors. For a row quantity $Z(x, \sigma)$, the vertex norm from Equation (6.20) is

$$\|Z\|_{u;s}^s = \frac{1}{N} \mathbb{E}_\sigma \sum_x \frac{X_u(x)}{u} |Z(x, \sigma)|^s. \quad (10.1)$$

The value of s remains 8. Supremums over endpoints below are over actual ID positions, with the appropriate retained-domain masks; virtual default positions are never counted with the multiplicity of the ambient universe.

10.1 Uniform salts and their access rule

The fixed-pair estimate combines the root bound for a univariate polynomial with affine hash maps as in Carter and Wegman [4, Lemma 6 and Proposition 7, p. 149]. Carter and Wegman sample a nonzero slope uniformly. Here zero slopes are allowed, the parameters come from reduced fair-bit strings, and the collision bound uses this law and the specified range partition. Adaptive fingerprints also require the block comparison below.

Lemma 10.1 (Short fingerprints). *Let endpoint IDs have injective binary encodings of one common length $L \leq c_{\text{ID}} B$. A uniform algorithm using $O(\log B)$ independent fair bits produces a map $k : \{\text{endpoint IDs}\} \rightarrow \{1, \dots, B\}$ for which every pair of distinct fixed IDs satisfies*

$$\Pr[k(v) = k(w)] \leq C/B \leq CB^{-1/2}. \quad (10.2)$$

Given access to an endpoint's encoding, the map is evaluable using $O(\log B)$ additional arithmetic scratch. The constants may depend on the fixed encoding-length bound c_{ID} .

Proof. Pad all encodings to the same length and put $P_v(T) = \sum_{i=0}^{L-1} v_i T^i$. Choose a prime $B^3 \leq q \leq 2B^3$, increasing the fixed lower bound on B if necessary. Thus $L < q$. Let $\ell = \lceil \log_2 q \rceil$. Split a string of 3ℓ fair bits into independent integers R, A, C in $\{0, \dots, 2^\ell - 1\}$, and reduce them modulo q to obtain r, α, β . Each residue has probability at most $2/q$. Let $\tau : \mathbb{F}_q \rightarrow \{1, \dots, B\}$ partition the standard residues into consecutive sets of sizes differing by at most one, and set

$$k(v) = \tau(\alpha P_v(r) + \beta).$$

All computations are in $\mathbb{F}_q$ before applying τ .

The nonzero polynomial $P_v - P_w$ has degree at most $L - 1$, so it has at most $L - 1$ roots. This root bound follows by repeatedly factoring $T - r_0$ at a root; it applies here because the coefficient difference 1 or -1 is nonzero in $\mathbb{F}_q$. Consequently

$$\Pr[P_v(r) = P_w(r)] \leq 2(L - 1)/q.$$

For distinct evaluations z, z' , the map $(\alpha, \beta) \mapsto (\alpha z + \beta, \alpha z' + \beta)$ is a bijection of $\mathbb{F}_q^2$. Under uniform parameters its two outputs are uniform and independent. The probability that their τ -values agree is

$$q^{-2} \sum_{j=1}^B |\tau^{-1}(j)|^2 \leq \frac{\lceil q/B \rceil}{q} \leq \frac{1}{B} + \frac{1}{q}.$$

The actual joint law of (α, β) is pointwise at most four times the uniform law and is independent of r . Adding the two cases gives $2(L - 1)/q + 4/B + 4/q = O(1/B)$. A polynomial evaluation by Horner's rule and the residue arithmetic use $O(\log q) = O(\log B)$ bits. The prime can be chosen uniformly by searching and trial division in that space. The existence of a prime in the indicated interval is the same elementary prime-interval fact used in choosing the environment field. $\square$

Fix a small positive rational ϵ , to be chosen last. This coefficient controls the fingerprint block width. For integer block indices $b \geq 0$, write

$$h_* = \lfloor \epsilon \log_2 B \rfloor, \quad M_b = b h_*, \quad \mathcal{I}_b = \{M_b, \dots, M_b + h_* - 1\}. \quad (10.3)$$

Enlarge the fixed minimum B so $h_* \geq 1$ and $h_* \geq (\epsilon/2) \log_2 B$. Here M_b is the block's lower budget boundary. Give each block meeting $\{0, \dots, M_{\text{top}}\}$ an independent salt from Lemma 10.1. Pad each salt to a multiple of the fixed integer H by unused independent bits. Since $M_{\text{top}} = O(B)$, the total number of bits is

$$O((1 + B/h_*)(\log B + H)) = O(B). \quad (10.4)$$

Here and throughout, constants may depend on the fixed ϵ and H . We enumerate the original fair-bit strings when enumerating environments; reducing chunks modulo q does not change that sampling convention.

The following access rule is part of the algorithm. A task whose budget lies in $\mathcal{I}_b$ may inspect salts with indices at most b . Its conditional walks vary all earlier salts as free bits and freeze the salt of block b and every later salt. The restriction applies to structural choices, port orders, validity tests, and predecessor tests as well as to numerical values. The controller construction below implements this rule for every access operation, including inverse queries. It is not enough to impose it only on the final output values.

This rule is stated in the coordinates of the task's *effective environment*. Write σ for that environment when analyzing a task. A fixed ancestor variant may mix its free salt coordinates before supplying σ to a descendant; the descendant's table depends on the resulting environment, not on the

word that produced it. Thus the rule does not forbid reading a later-indexed bit of the original master environment while replaying that word. It requires the descendant’s values, port orders, and access operations to be functions only of its permitted effective coordinates. Lemma 12.12 implements this distinction. Each fixed variant is measure preserving by Lemma 7.3, so the integrated estimates may change variables to its effective environment. We do not assume that an adaptively selected variant leaves a uniform conditional law. At a block- b task, its own variants act only on effective salts earlier than b and freeze its effective block- b salt and all later ones.

10.2 The update rule and the comparison calculation

We now complete the recursive definition of the actual task; inactive tasks still return zero outputs and empty access classes. For an active row task, the additive schedule of Section 8 supplies candidate increments from smaller-budget outputs. Process the groups in decreasing slack. In each group, distribute increments by the rule below, then use the compression, range clamps, and selected list rule of Section 9. After the last group, form the row functions before incoming multiplication and precursor classes from the completed arrays, then assemble any incoming gates and column multipliers as in Section 6. The separate column-vector calls have smaller budget. Vector tasks use the same schedule to sum their scalar terms, with their prescribed range clamps and final clipping; they require no endpoint grouping or row-list compression.

The row-group distribution rule is as follows. For a dyadic group with slack $d > 2h_*$, compare endpoint IDs exactly. For a group with $d \leq 2h_*$, use the key from the task’s block as follows. First discard invalid candidates, including invalid guessed wraps, individually. Sum all new candidate increments having the same key. Add a bucket’s sum to every old representative with that key. If the bucket has no old representative, introduce exactly its first valid new candidate, with its previous value equal to the old default, and add the bucket sum there. Do not combine or discard old representatives in this key step. Apply the same full-array compression, clamping, and selected list rule as in the preceding section.

Distinct actual representatives remain distinct. Indeed two introduced representatives have different keys, and an introduced representative’s key differs from every old key. Increments at one and the same actual endpoint are always combined, even when other endpoints collide with it. Stored values are never summed by key. The two channels use the same keys and the same representative selection for the entire group.

At budget $M \in \mathcal{I}_b$, the comparison calculation replaces key comparisons by exact ID comparisons throughout that block, including every same-block child. Children below M_b use the actual algorithms with exactly the same environments in both calculations. This replacement applies to the entire numerical and structural calculation: candidate validity, representative selection, compression, access classes, and their port orders. Descriptor enumeration uses fixed index orders. Each finite-accuracy test for a selected list or access class retains its prescribed accuracy level and tie convention, independently of any requested output accuracy. All these operations use the corresponding comparison operands. Thus the comparison calculation specifies its access choices as well as its represented arrays.

Denote the actual and comparison versions by superscripts ac and cmp . Both still perform compression and clamping. The comparison calculation is a finite analytical recursion; its exact comparisons are not asserted to satisfy the eventual space bound.

Clamping is part of both definitions. The comparison error reserve uses Lemma 9.3 for exact-ID increments, which are independent of the stored exception list. That lemma preserves the relevant row functions and witnesses; it need not preserve the list or port presentation of an unclamped calculation. The coupling below compares the two clamped calculations directly. Its propagation

bound uses the Lipschitz compression map and the nonexpansiveness of each common clamp, without introducing an unclamped actual calculation.

Lemma 10.2 (Independence of the comparison calculation). *For a task of budget $M \in \mathcal{I}_b$, the comparison calculation defined above, including its numerical values, validity tests, representatives, and port orders, is independent of effective salt blocks $b, b+1, \dots$. Before the first group with $d \leq 2h_*$, its row state agrees with the actual row state; this common state is independent of block b and has a list of size $2^{O(h_*)}$.*

Every sample function in the comparison calculation satisfies the deterministic slice-mean hypothesis of Lemma 8.6: its averaging varies all coordinates on which it depends.

Proof. Induct on the budget and, within a task, on its prescribed sequence of operations. Lower-block algorithms ignore the salt of block b and later salts by the access rule; same-block comparison children ignore them by induction. Candidate tests use these children, exact endpoint grouping precedes compression, and the compressed row determines its access classes and scores computed before incoming multiplication. Incoming ranks use those completed scores. Separate column-vector queries likewise use a smaller-budget comparison child in this block or an actual child below it. All statements here concern the current effective environment: each same-block transformation preserves its current and later salts, and each lower-block transformation acts on a subset of its earlier salts. Each numerical recipe uses the same fixed operand order, and each structural test uses its fixed accuracy level and tie rule. Its result therefore depends only on earlier comparison operands, including their finite numerical approximations. The fixed index orders then give salt-independent representatives and ports.

In particular, exact full-array addition, compression, and the prescribed row functions are salt independent. Reordering paths or choosing a different representative of the same endpoint would not change an exact sum; retaining an extra default-valued position would not change the represented array. The fixed conventions above also remove these presentation choices from the structural calculation.

For a group with $d > 2h_*$, every child budget is at most $M - d < M_b$. Both runs use the same lower-block children and exact comparisons there. All their list operations, including their deterministic tie conventions, ignore the present salt. Thus their boundary states agree. The last larger dyadic slack, if there is one, is at most $4h_*$, so its selected list has at most $2^{O(h_*)}$ positions. If there is no larger group, the boundary list is empty.

Finally, the comparison estimand depends only on sampling channels and earlier salts. These are precisely the variables averaged by its conditional walk, subject to its stated vertex conditions. Same-block salts are irrelevant, and lower-block salts are free variables of this walk. Projection onto the uniform measure of the slice therefore gives a deterministic quantity at its indicated pair or vertex, with the prescribed retained-domain masks. In particular, no random dependence on a frozen salt remains in this mean. For the second channel, its deterministic mean magnitude is extended by its prescribed narrower endpoint mask, as required in Section 9. The salt rule is required of all access operations, including inverse queries and port orders; Lemma 12.12 supplies its implementation. Lemmas 7.1 and 7.3 therefore apply to this function. This verifies the deterministic-mean hypothesis of Proposition 9.4; it does not assert independence between errors of different children. $\square$

10.3 The estimator bound and its deduction

The exact comparison has the error reserve of Proposition 9.4. The remaining input is the following bound on its difference from the actual calculation. The positive constants D_1, D_2 are fixed in its proof in Section 10.6, after the crude comparison constants and before the block-width coefficient ϵ .

Proposition 10.3 (Uniform block coupling). *For the actual and comparison calculations defined above, the construction constants can be fixed so that every admissible task of budget $M \in \mathcal{I}_b$ and precision p has discrepancy at most*

$$2^{-p-D_1h_*+D_2(M-M_b)} \quad (10.5)$$

in each prescribed output metric. This includes the root metrics for row outputs, scores computed before incoming multiplication, and support witnesses, and the ordinary metrics for vectors.

The coupling bound and exact-grouping error reserve give the estimator guarantee used in Section 3.

Theorem 10.4 (Error bounds for the actual estimators). *Fix the constants as in Proposition 10.3. For every admissible task tuple and nonnegative integer budget, let p be its precision from Equation (8.8). Every true nonnegative row output H and its actual estimate $\bar{H}$ satisfy*

$$\mathfrak{d}_{u,a}(\bar{H}, H) \leq C_b 2^{-p}.$$

The same bound holds for scores computed before incoming multiplication and support witnesses. For each clipped vector estimate the error is at most $C_b 2^{-p}$ in the ordinary vertex norm. In particular, every reward query obeys

$$\frac{1}{N} \mathbb{E}_\sigma \sum_{x \in V_l} \frac{X_u(x)}{u} |\bar{W}_l(x) - W_l(x)|^s \leq C_b^s 2^{-sp}. \quad (10.6)$$

The expectations use the common environment law and the task's specified masks and suffixes. The constants are uniform in the admissible parameters, stage, input, and B .

Proof. With $D_1 > D_2 + 1$ and $0 \leq M - M_b < h_*$, the relative coupling coefficient satisfies

$$2^{-D_1h_*+D_2(M-M_b)} \leq 2^{-(D_1-D_2)h_*} \leq C_b/2,$$

where the last inequality uses the fixed minimum-size choice of B . For an active task, add this bound to the comparison error $(C_b/2)2^{-p}$ from Proposition 9.4. The result is $C_b 2^{-p}$. Inactive tasks use the fixed bounded ranges and supports. The induction in Proposition 10.3 includes both kinds of tasks and every prescribed output. Applying its ordinary vector statement to W_l and taking the s -th power gives Equation (10.6). $\square$

It remains to prove the coupling estimate. A salt-independent set of comparison endpoints provides the collision bound. Paths leaving that set are charged to child witness errors; paths within it are compared by the bucket update. The following two subsections give these charges, and Section 10.6 sums them with the budget decrease. The target and coupling bounds are proved simultaneously there, so the preceding deduction does not assume an already closed target-error induction.

10.4 Candidate endpoint sets and paths leaving them

To control adaptive candidates, we first bound a set of endpoints determined by the comparison calculation. An actual path leaving this set will be charged to a child witness discrepancy.

We recall the relevant interfaces from Lemma 6.5. Every exposed actual path port has a broader witness whose sth root is Lipschitz and whose value is one at that port. Each witness has bounded row support. The actual correction ports, including exposed ports of weight zero, have incoming multiplicity at most K_c . Tables contain one entry per pair of actual IDs. After fixing a copy part, the child tuple is common over the whole table; in particular, a right child reached from a midpoint does not depend on the discarded old start. All these statements hold for arbitrary computed raw arrays, not just for accurate ones.

The incoming cap applies to the entire exposed correction relation, including its zero-weight entries. The broader auxiliary score class needs only a row bound: it is queried for scores and witnesses, whereas the left factor of a two-factor path is a capped correction table. This distinction permits the source sum below to be removed even when an offending candidate has zero numerical weight.

Let $\mathcal{G}_b$ be the sigma-field generated by all coordinates of the task's effective environment except the salt of block b . Fix a row x and condition on $\mathcal{G}_b$ with $X_u(x) = 1$. Take its boundary list from Lemma 10.2, including every stored exact-default slot. For every small group and every immediate descriptor variant, evaluate the comparison factors and their witnesses in the environment obtained by that fixed variant. This transformation leaves the present and later salts frozen. Include the endpoints in each direct witness support and, for a two-factor path, the endpoints in the right witness supports at all midpoints in the left witness support. Include local base endpoints when applicable. Let $\mathcal{E}_x$ be the union of these endpoints and the boundary list. Set $\mathcal{E}_x = \emptyset$ when $X_u(x) = 0$. Unused or invalid comparison descriptors may enlarge this set. Actual candidates must still satisfy their wrap tests and all transformed source, midpoint, and terminal memberships; every error estimate below retains the memberships needed for its child metrics.

Lemma 10.5 (Size and independence of the candidate set). *For the comparison endpoint set $\mathcal{E}_x$ defined above, $|\mathcal{E}_x| \leq 2^{C_E h_*}$ for a fixed C_E , and $\mathcal{E}_x$ is measurable with respect to $\mathcal{G}_b$. Every comparison nondefault representative in the small groups belongs to $\mathcal{E}_x$. Conditional on $\mathcal{G}_b$,*

$$\Pr[\exists v \neq w \in \mathcal{E}_x : k(v) = k(w) \mid \mathcal{G}_b] \leq C 2^{2C_E h_*} B^{-1/2}. \quad (10.7)$$

Proof. Each fixed variant is independent of the present salt and leaves it unchanged. Lemma 10.2 therefore makes its comparison witness supports $\mathcal{G}_b$ -measurable. We use these supports as sets of actual IDs, not the order of a port enumeration. At a group of slack d there are $2^{O(d)}$ immediate descriptors, including variants, difference parts, and copy choices. The number of endpoints contributed by a direct witness is bounded. A two-factor path contributes a bounded number of left witness midpoints and, for each, a bounded number of right witness endpoints. Summing over the dyadic groups with $d \leq 2h_*$ and adding the boundary list proves the size bound. No incoming bound on a comparison witness is used here.

Ports of the comparison calculation lie in the supports of its witnesses. Apart from the boundary representatives, exact updates can therefore create nondefault values only at the included endpoints. Compression maps all untouched default positions to the new default. Any extra exact-default slots in a selected list may be omitted when describing its full array. This proves the representative claim. Equation (10.7) is now the pair union bound from Lemma 10.1, conditional on the fixed set $\mathcal{E}_x$. $\square$

Let $\mathcal{F}_x$ be the event that some valid actual candidate in the small groups has endpoint outside $\mathcal{E}_x$, whether or not its increment is zero. As representatives are only carried forward or chosen among

new candidates, all actual representatives stay in $\mathcal{E}_x$ outside this event. This includes default-valued representatives: they start in the full boundary list or inherit a candidate endpoint. Dummy copies used to compute ranks are not new real representatives. Thus omitting an exact-default comparison slot when describing its full array does not omit an actual address from the collision analysis.

Lemma 10.6 (Charging a candidate outside the comparison set). *Suppose every relevant actual-comparison child discrepancy at budget m is at most $2^{-p_m}\lambda_m$ in its prescribed metric, including the support-witness discrepancies, where p_m is that child's precision and $\lambda_m \geq 0$. The part of any bounded final discrepancy between roots of row outputs before incoming multiplication on $\mathcal{F}_x$ has norm at most*

$$2^{-p} \sum_{\substack{d \leq 2h_* \\ m \in \mathcal{L}_b, m \leq M-d}} 2^{Cd} \lambda_m. \quad (10.8)$$

Here d ranges over the small dyadic group slacks, and the sum includes the dependency types in each group. Descriptor multiplicities are absorbed in C ; equivalently one may use each group's maximum child coefficient and enlarge C .

Proof. An actual endpoint outside $\mathcal{E}_x$ has an offending path factor. For a direct path its actual port is outside the comparison witness support. For a two-factor path, either its actual left port is outside its comparison witness support, or its midpoint lies in that support and its actual right port is outside the corresponding comparison right witness support. At the offending port the actual witness is one and the comparison witness is zero. Thus the indicator of this event is at most the s -th power of their root discrepancy there. Union bounding over candidate paths gives a sum of such nonnegative powers. The property that the witness equals one on every exposed port holds also for exposed zero-weight ports; no lower bound on the candidate's increment is being assumed.

Each output under consideration is bounded and has bounded row support. The union of the two output supports is still bounded, by Equation (9.2). Hence its entire row error power on $\mathcal{F}_x$ is at most a fixed multiple of $\mathbf{1}_{\mathcal{F}_x}$. We charge this row-failure indicator, rather than the size of the offending increment. There is no endpoint sum left on the output side to pay for.

The right-factor charge is the case where the old start must be summed out. Fix a guessed variant and a two-factor path $x \rightarrow z \rightarrow y$. Change variables by its measure-preserving transformation, as in Lemma 7.3; in the following display all memberships, ports, and witness values are evaluated in that transformed environment. Write W_R^{ac} and W_R^{cmp} for the actual and comparison right witnesses, and let $e_R(z, y) = (W_R^{\text{ac}}(z, y))^{1/s} - (W_R^{\text{cmp}}(z, y))^{1/s}$. Let a' be their common endpoint rate, and write $\mathcal{P}_L^{\text{ac}}$ for the actual left exposed-port relation. The candidate's validity retains $X_u(x)$, $Z_f(z)$, and the right child's endpoint membership $Y_{a'}(y)$. Dropping only surplus guards, the right-port contribution is at most a constant times

$$\begin{aligned} & \frac{1}{N} \mathbb{E} \sum_x \frac{X_u(x)}{u} \sum_{z: (x, z) \in \mathcal{P}_L^{\text{ac}}} Z_f(z) \sum_{y: Y_{a'}(y)=1} |e_R(z, y)|^s \\ & \leq \frac{K_c}{Nu} \mathbb{E} \sum_z Z_f(z) \sum_{y: Y_{a'}(y)=1} |e_R(z, y)|^s = K_c \frac{f}{u} \mathfrak{d}_{f, a'}(W_R^{\text{ac}}, W_R^{\text{cmp}})^s. \end{aligned} \quad (10.9)$$

The middle inequality uses the incoming cap of the *actual* left port relation. It does not use a comparison incoming cap. The right task's tuple is independent of x , so the same discrepancy is being summed when the old start is removed. This condition is necessary for the displayed equality with the right metric. For the fixed variant and copy part, the right tuple has only the prescribed task parameters and transformed environment. The midpoint z is its vertex argument, and the old

source x is not a parameter. Its final gate and column queries use that same effective tuple, rather than the ancestral descriptor that happened to discover the entry.

For a direct or left offending port, sum the child's witness discrepancy over its own row and endpoint domain; the bounded number of possible right ports incurs only a fixed factor. Projected paths are partitioned by the fixed copy symbols, with constant multiplicity. Normalization and preparation dependencies make the same direct witness comparison. If a candidate arose from a narrower terminal slice, its validity mask may be dropped in this row-event estimate. Thus no factor π_a/π_t is incurred. In contrast, the transformed source hit cannot be omitted before the change of variables and the incoming-cap sum; it has been retained in Equation (10.9).

At slack d , scale ratios, variants, and immediate path counts cost at most $2^{O(d)}$, as in Lemma 8.8. The possible reset factor in Equation (10.9) is absorbed by the precision gain. Admissible row tasks have $u \leq f$, as specified in Section 6. Equation (8.9) gives, after the scale-dependent $O(d)$ terms,

$$\begin{aligned} p_m &\geq p - O(d) + \frac{1}{2} \log_2(f/u), \\ (f/u)^{1/s} 2^{-p_m} &\leq 2^{-p+O(d)} (f/u)^{1/8-1/2} \leq 2^{-p+O(d)}. \end{aligned}$$

Take s -th roots and use Minkowski to sum the charges. A child below M_b is identical in the two versions and contributes zero. All remaining children satisfy $m \leq M - d$ and belong to $\mathcal{I}_b$, which proves Equation (10.8). $\square$

10.5 Bucket algebra and the collision charge

The bound for $\mathcal{F}_x$ leaves rows whose candidates stay in $\mathcal{E}_x$. On these rows we separate child discrepancies from collisions among the comparison endpoints. We compare full arrays with their defaults, so different selected list presentations require no extra error term. The following observation explains the key update rule.

Lemma 10.7 (Grouping increments by their keys). *Fix a small group on $\mathcal{F}_x^c$. Let $\Delta_{\text{ac}}(y)$ and $\Delta_{\text{cmp}}(y)$ be its sums by actual ID before key grouping, using the actual and comparison children, respectively. If it has at most $k_d \leq 2^{C_d}$ immediate descriptors and $\mathcal{C}_x$ is the event of a key collision between distinct IDs in $\mathcal{E}_x$, then the sup-norm difference between its actual distributed increment and comparison increment is at most*

$$(k_d + 1) \left(\sup_y |\Delta_{\text{ac}}(y) - \Delta_{\text{cmp}}(y)| + \mathbf{1}_{\mathcal{C}_x} \sup_y |\Delta_{\text{cmp}}(y)| \right). \quad (10.10)$$

This bound holds for both raw channels. The prior full-array discrepancy enters the resulting pre-compression bound with coefficient one.

Proof. When all keys on $\mathcal{E}_x$ are distinct, the rule gives exactly the actual true-ID increment: each old occurrence is updated at its own ID, and each genuinely new ID has its own bucket. Thus the discrepancy is simply $\Delta_{\text{ac}} - \Delta_{\text{cmp}}$.

When there is a collision, each updated representative receives a sum of increments at at most k_d distinct actual addresses. Its magnitude is at most $k_d \sup_y |\Delta_{\text{ac}}(y)|$. At positions not updated it is zero. At either type of position subtracting the comparison increment costs at most a further $\sup_y |\Delta_{\text{cmp}}(y)|$. Apply

$$\sup_y |\Delta_{\text{ac}}(y)| \leq \sup_y |\Delta_{\text{ac}}(y) - \Delta_{\text{cmp}}(y)| + \sup_y |\Delta_{\text{cmp}}(y)|$$

to obtain Equation (10.10). This also covers an increment duplicated at several old representatives and a missing increment at an unchosen new representative.

At no point are old values combined or moved to a different ID. An unlisted old position has exactly the old default, and a newly selected representative starts with that value. Therefore the full-array comparison before compression is the old discrepancy plus the increment discrepancy just bounded. There is no term involving the height of an old value multiplied by a collision indicator. $\square$

In applying this lemma to a group, combine contributions at equal actual endpoints *within each paired difference* before measuring its comparison height. At a fixed endpoint and original generator word, the outer slice and wrap tests are the same for budgets m and $m - 1$. Their constituent port sets may differ, but each set contains all nonzero entries and enumerates each actual ID pair once; an absent entry contributes zero. Their true-ID sums are therefore exactly the two terms of the paired difference. Subtracting the same stochastic true-table term from both sides, as in Lemma 8.8, preserves its cancellation. One can then sum the resulting sup bounds over difference layers. This remains legitimate if a first-bucket representative is chosen across several layers: the proof of Lemma 10.7 applies to their total increment, and the triangle inequality bounds the total by the sum of the paired bounds. A descriptor with zero increment contributes only to k_d in this bound. It may affect representative selection, which is already covered by Lemma 10.7 and the event $\mathcal{F}_x$.

Lemma 10.8 (Direct cost of collisions). *The direct collision term in a final error between roots of row outputs before incoming multiplication has norm at most*

$$2^{-p} 2^{Ch_*} B^{-1/(2s)}, \quad (10.11)$$

where C is fixed before the coupling constants D_1, D_2 and block-width coefficient ϵ are chosen. On $\mathcal{F}_x^c$, every remaining error term is bounded by the relevant child actual-comparison discrepancy norms, amplified by at most 2^{C^d} at slack d .

Proof. For each small dyadic group, let $\Delta_d^{\text{cmp}, \nu}(x, y)$ be its complete comparison true-ID increment in channel $\nu \in \{1, 2\}$, including any exact base term assigned to that group. Take an absent second channel to be zero, and choose a deterministic descriptor bound $k_d \leq 2^{C^d}$. By Lemma 8.8, the comparison paired increments at a group of slack d obey their crude absolute estimates in the retained endpoint metric. These estimates hold for each fixed valid variant after its measure-preserving change of variables. Counting variants and summing difference layers by the triangle inequality costs $2^{O(d)}$, without using walk contraction. Within each paired layer the true-ID cancellation is performed before this inequality is applied. If a local unpaired base term occurs in a small group, its first-group slack satisfies $M = O(d)$, so the same crude bound holds in its normalized raw units. Since $\sup_y |Z_y|^s \leq \sum_y |Z_y|^s$ over actual ID positions, the row norm of $\max_\nu \sup_y |\Delta_d^{\text{cmp}, \nu}(x, y)|$ is at most $2^{-p+O(d)}$.

Lemmas 9.1 and 9.3, together with Equation (9.7), propagate an input discrepancy at slack d to the final array with a factor polynomial in $d + 1$. At each step the discrepancy is measured between the clamped full arrays, including their defaults. The next bucket increment is compared by Lemma 10.7; compression then applies its uniform Lipschitz bound and the common clamp cannot enlarge the discrepancy. This recurrence does not require their later bucket distributions or stored lists to agree. Row functions with Lipschitz sth roots and bounded final support then give the same type of bound for the output metric. Choose fixed C_P, A_P so that $C_P(d + 1)^{A_P}$ bounds

this propagation. For a retained row define

$$H_x = C_P \sum_{\substack{d \leq 2h_* \\ d \text{ a group slack}}} (d+1)^{A_P} (k_d+1) \max_{\nu} \sup_y |\Delta_d^{\text{cmp}, \nu}(x, y)|.$$

Set $H_x = 0$ when $X_u(x) = 0$. This quantity bounds the propagated terms involving comparison increments in Equation (10.10). Write $H = (H_x)_x$. The preceding estimates give

$$\|H\|_{u;s} \leq 2^{-p+C_0 h_*}.$$

All increments defining H_x come from the comparison calculation, and its other factors are deterministic. Thus H_x and $\mathcal{E}_x$ are $\mathcal{G}_b$ -measurable. Conditioning on this sigma-field gives

$$\mathbb{E}[\mathbf{1}_{C_x} H_x^s \mid \mathcal{G}_b] = H_x^s \Pr(C_x \mid \mathcal{G}_b) \leq C 2^{2C_E h_*} B^{-1/2} H_x^s.$$

The row membership $X_u(x)$ is also $\mathcal{G}_b$ -measurable. Write $(\mathbf{1}_C H)_x = \mathbf{1}_{C_x} H_x$. Integrating the last inequality in the row measure therefore yields

$$\|\mathbf{1}_C H\|_{u;s}^s \leq C 2^{2C_E h_*} B^{-1/2} \|H\|_{u;s}^s.$$

Taking roots proves Equation (10.11) after enlarging C . We have not conditioned on $\mathcal{F}_x^c$ in this independence calculation: the actual error there is bounded by $\mathbf{1}_{C_x} H_x$, which is then integrated on the whole space. Conditioning on the salt-dependent stay-inside event is unnecessary and could destroy the stated independence.

For the first term of Equation (10.10), split products using bounded-factor differences, actual or comparison row caps, and the actual or comparison incoming cap of the represented two-way factor, exactly as in Lemma 8.8. These are comparisons of unaliased sums using the respective children; they make no assumption about key success or independence of child errors. Fixed valid variants, scale costs, and counts are all at most $2^{O(d)}$, with the possible source-release factor absorbed by the same precision gain as in Equation (10.9). Compression propagation is again absorbed in 2^{C_d} . This proves the remaining assertion. $\square$

10.6 Closing the coupled induction

Proof of Proposition 10.3. The induction has two sources of gain. A call to a smaller budget reduces its allowed coupling coefficient geometrically; a collision among comparison endpoints pays a power of B . We first keep these terms separate, then choose the budget decay and block width in that order. For this block put $\lambda_r = 2^{-D_1 h_* + D_2(r - M_b)}$ for each integer $r \in \mathcal{I}_b$. Perform strong induction on the statistical budget, simultaneously for the actual target estimates, the comparison target estimates needed at that budget, and Equation (10.5). An inactive task returns the same zero outputs and empty paths in both versions, so its coupling error is zero. Lower-block children are identical in a block coupling and already have their actual target bounds. Same-block comparison children have the comparison target bounds at smaller budgets. Therefore Lemma 10.2, the prefix estimate, and the compression analysis give the comparison active target bounds with the spare constants in Proposition 9.4.

For the coupling, Lemmas 10.6–10.8 bound the quantities before incoming multiplication. Incoming gates are then compared by their rank and Lipschitz bounds for the s th roots. In their multiplier effect, restrict to the union of the final supports of the two tables; these have bounded incoming multiplicity. Thus Lemma 6.6 costs only a fixed factor on errors before incoming

multiplication. Its separately queried column vectors have a positive constant budget drop and their f/u metric factor is covered by their precision gain. Vector row sums have no endpoint grouping, but their same-block children may differ. The same crude individual-variant comparisons and nonexpansive clamps bound those differences. Detector rank comparisons are already part of the assembly estimate. No simultaneous key-success event over an incoming search is required.

Collect the finite number of output types and absorb their fixed assembly factors into one fixed exponent C . For a child budget $m \in \mathcal{I}_b$ used at group slack d , we have

$$\lambda_m = \lambda_M 2^{-D_2(M-m)} \leq \lambda_M 2^{-D_2 d}. \quad (10.12)$$

For a separate post-assembly query with constant positive budget drop, use $d = 1$ in this inequality. Replacing the dyadic group sum by the larger sum over positive integers, all coupled child charges are at most

$$2^{-p} \lambda_M \sum_{d=1}^{2h_*} 2^{-(D_2-C)d}. \quad (10.13)$$

Counts of budget layers and dependency types are included in 2^{Cd} . Groups with larger slack have identical children and contribute zero. Together with Equation (10.11), this is the complete coupling recurrence.

Choose $D_2 > C$ so large that the infinite geometric sum in Equation (10.13) is at most $1/4$. Then choose $D_1 > D_2 + 1$. Finally choose rational $\epsilon > 0$ so small that

$$\frac{1}{2s\epsilon} > C + D_1 + 2. \quad (10.14)$$

Since $h_* \leq \epsilon \log_2 B$, the direct relative charge satisfies

$$2^{Ch_*} B^{-1/(2s)} \leq 2^{-(1/(2s\epsilon)-C)h_*} \leq \frac{1}{4} 2^{-D_1 h_*} \leq \frac{1}{4} \lambda_M,$$

after increasing the fixed minimum B to absorb any fixed coefficient. The child and direct charges fit within Equation (10.5), even with room for fixed norm conventions already included in C .

Moreover $0 \leq M - M_b < h_*$ implies

$$\lambda_M \leq 2^{-(D_1-D_2)h_*}.$$

This is uniformly as small as required after the same minimum-size padding. The triangle inequality therefore adds the coupling bound to the spare comparison active bound, for example $C_b 2^{-p}/2$, without exceeding $C_b 2^{-p}$. Inactive target bounds are the previously fixed bounded-range bounds. This completes all three inductions.

The order of choices is consequential. Fix the analytical hierarchy, table thresholds, and trivial-case bound first. Choose the precision and cutoff constants for the mean comparisons; then the compression floors, capacities, and walk lengths so the comparison sums converge with their spare coefficients. Fix clamping ranges and the crude exponents in the fixed-variant comparisons next. These choices determine C . Only after them choose D_2, D_1, ϵ as above. Thus reducing ϵ to obtain the direct collision gain does not alter an earlier comparison-coefficient requirement.

Later choices of the common stage-size bound, top-budget coefficient, and environment-field exponent do not change this crude exponent C . Immediate descriptor counts depend on relative slack, fixed support bounds and fixed walk length. The sampler gap is uniform in both field size and the length of its single free-bit component. Likewise, an enlargement of the fixed identifier-length coefficient only changes the polynomial-evaluation collision term in Lemma 10.1 to at most $2c_{\text{ID}}/B^2$; increasing the fixed minimum $B \geq c_{\text{ID}}$ bounds it by $2/B$. These enlargements can therefore be absorbed after D_2, D_1, ϵ have been fixed. $\square$

11 Finite numerical queries with bounded suspended storage

The preceding sections define finite estimator tables and bound their statistical errors. We now compute digits of those estimator values. For a fixed table tuple, including its statistical budget, the structural conventions are fixed and each entry has one exact mathematical value. Increasing the numerical clock requests a later digit of that same value. Each request is a finite deterministic computation.

A query returns one bounded digit. During its evaluation, the space needed by a suspended caller must be bounded by the size of its structural descriptors and a constant times the decrease from its output clock to the requested operand clock. An operand clock may exceed the output clock by an amount proportional to descriptor size; the structural term also pays for this overshoot. These differences will telescope when arithmetic queries are nested inside graph procedures. Thus the relevant question is which data remain live at each operand request.

11.1 Streams, records, and uniformity

We use integer clocks $t, j \geq 0$. Known binary scales are separated from numerical values, so a normalized value belongs to $[-1, 1]$. The fixed digit alphabet is

$$\mathcal{A} = \{-6, -5, \dots, 5, 6\}.$$

Every digit is encoded by a fixed number of bits. An operand stream $(x_j)_{j \geq 0}$ denotes the exact real number

$$x = \sum_{j \geq 0} x_j 2^{-j}, \quad x_j \in \mathcal{A}. \quad (11.1)$$

In particular, for its truncation $x^{[K]} = \sum_{j=0}^K x_j 2^{-j}$,

$$|x - x^{[K]}| \leq 6 \cdot 2^{-K}. \quad (11.2)$$

The representation is redundant: different streams can denote the same real number. This lets consecutive finite approximations define an exact stream even when their truncations differ. Signed-digit arithmetic has a long history [2]; the radix, alphabet, and residue-based digit rule below are the choices used here.

An arithmetic root has public parameters, its clock, and a description of its operand interfaces. We require these parameters to be reconstructible from the outer controller records without operand or graph queries. They are fixed inputs to each numerical computation; their full binary encodings are not copied at every gate. An operand label specifies its interface and descriptor; it never specifies an unknown vertex address from a previous cursor position.

The following lemma reduces each digit query to a fixed number of residue bits.

Lemma 11.1 (Exact streams from approximation residues). *Let $|z| \leq 1$ and suppose deterministic integers $N_t(z)$ are defined for every integer $t \geq 0$ with*

$$|N_t(z) - 2^t z| < 2. \quad (11.3)$$

Then

$$z_0 = N_0(z), \quad z_t = N_t(z) - 2N_{t-1}(z) \quad (t \geq 1) \quad (11.4)$$

is an exact $\mathcal{A}$ -stream for z . For any fixed integer $q \geq 4$, its place- t digit is determined by $N_t(z) \bmod 2^q$ and, when $t > 0$, $N_{t-1}(z) \bmod 2^q$.

Proof. Writing $e_t = N_t - 2^t z$ gives $z_t = e_t - 2e_{t-1}$ for $t > 0$, hence $|z_t| < 6$; the weaker alphabet bound $|z_t| \leq 6$ is convenient throughout. At place zero, $|N_0| < 3$, so $|N_0| \leq 2$. The partial sum through place T is $2^{-T}N_T$, which tends to z by Equation (11.3). Finally, reduction modulo 2^q is injective on the possible signed digits because $2^q > 12$. Thus the signed difference can be recovered from the two residues without forming the full integers N_t . Concretely, for $t > 0$ compute $(N_t \bmod 2^q) - 2(N_{t-1} \bmod 2^q)$ modulo 2^q , then use the fixed finite table which sends each occurring residue to its unique representative in $\mathcal{A}$. At $t = 0$, decode the residue of N_0 in the same way. This final decoding has constant storage and introduces no comparison of unknown exact real numbers. $\square$

The integer N_t is always defined before taking its residue. Different places may use different truncations or finite circuits. Only determinism and Equation (11.3) are required; consistency between the finite approximations is supplied by Equation (11.4). In particular, when a digit at t uses N_{t-1} , the latter is evaluated by its own place- $(t-1)$ definition.

We compute the required residues by finite Boolean circuits. To bound what survives an operand query, we evaluate those circuits from paths of child choices and regenerate their absolute addresses as needed. Here $d \geq 1$ measures the size of the immediate structural descriptors; in the application it is the current group's slack.

The local record of a suspended numerical query contains every local bit that must be retained across an operand query: its binary gate path, return modes, pending bounded results, stored descriptor choices, and any explicitly specified retained word. The input and master choice, the common vertex cursor, and one reusable scratch area are shared facilities. Temporary gate addresses, digit places and scan counters may occupy scratch while they are being computed. Any such data needed after a recursive graph query must either be in the charged local record or be reconstructible from the root parameters and that record. An operand-dependent word retained across the query is charged even if it was originally formed in scratch.

In particular, reconstructing an absolute digit place may use a binary counter in scratch, while the suspended record stores only the choices needed to recover that place. The next lemma makes this distinction precise for the finite circuits used below.

Lemma 11.2 (Evaluation from regenerated gate addresses). *Let a finite bounded-fan-in Boolean circuit have a specified output gate. Suppose a path of child choices from that gate, together with the public root parameters, determines each gate's type and predecessors, or its input operand label and digit place. Assume these descriptions can be regenerated in $O(B + t + d)$ scratch without operand or graph queries. Conditional gates may first evaluate their selector and then evaluate only the selected branch.*

The output can be evaluated depth first with $O(1)$ retained bits per gate on the active path. Above an operand call, suspended storage is at most a constant times this path length plus the explicitly retained root-local record; no absolute gate address need be saved at each gate.

Proof. At a gate, save its next-child or return mode and the already computed Boolean input, if any. At a conditional gate also save the selected branch bit. The bounded fan-in makes this a constant-size record. To locate the next child, scan the saved path from its root and regenerate successive gate descriptions in scratch. The final description determines the next child choice. On return, its Boolean result is combined with the saved bit and the local record is popped. All temporary addresses and scan counters are discarded before an operand query; a later scan recreates them from the same root and saved choices. Circuit acyclicity gives termination. This is an evaluation of the circuit unfolded into a formula; repeated gates are recomputed, so no array of previously computed values is stored. $\square$

11.2 The numerical interface

We state the composition result with its syntactic hypotheses. They concern finite arithmetic recipes and do not assert any property of the underlying graph operations.

Application to graph procedures requires the following additional checks. The semantic descriptor domains and structural validity conventions are fixed independently of the requested numerical clock; selected operands already have defined digit interfaces; reciprocal arguments lie in the stipulated positive intervals; and root parameters remain recoverable after lower calls. Clock-dependent truncations and comparison circuits may change N_t , but not its target recipe value. Lemmas 12.11 and 12.12 verify these conditions for the estimator procedures. An incoming rank over the whole vertex universe has no prescribed $2^{O(d)}$ descriptor bound and uses the separate construction of Lemma 12.7.

Pair averages, maxima, minima, fixed shifts, multiplication, and normalized reciprocals on fixed positive intervals have operand-clock overshoot $O(1)$ and suspended bound $O(1) + b(t - j)$ (Lemmas 11.5, 11.6, and 11.7). For ranks, bounded sums, and known scales on a $2^{O(d)}$ indexed domain, the bounds are $j \leq t + cd$ and $ad + b(t - j)$, with intrinsic space $O(d + t)$ (Lemmas 11.8 and 11.9). The fixed-precision structural tests of Lemma 11.10 have clock zero and request only $O(d)$ operand places. Each numerical primitive supplies approximations satisfying (11.3). Multiplication groups terms by their smaller digit index, and a reciprocal releases its early operand word before any late operand query; these choices control the storage retained across operand calls.

Theorem 11.3 (Uniform numerical interface). *Fix the digit alphabet, positive reciprocal intervals, and constants bounding the following recipe class. A recipe at slack $d \geq 1$ has a fixed number of stages built from bounded arithmetic operations, fixed piecewise formulas using maxima, minima and clipping, the reciprocals of Lemma 11.7, indexed ranks as in Lemma 11.8, and sums and known scales as in Lemma 11.9. The stage and descriptor descriptions are generated from the root parameters by index computations using $O(B + d + t)$ scratch and no operand queries. Count recipe dependency depth at the primitive interfaces: each listed arithmetic primitive, indexed rank, or balanced sum or average is one node, with edges to its operands. This depth is bounded independently of d . The internal comparison, counting, and selection circuits of an indexed rank and the pair-average tree of a balanced sum or average do not add nodes to this count; they use the constructions of Lemmas 11.8 and 11.9, respectively. Each indexed domain has size $2^{O(d)}$ and descriptors of length $O(d)$. Discrete domain and validity decisions are structural leaves or buffered tests with their own fixed precision conventions. Public coefficient leaves have exact normalized digit procedures using $O(B + d + t)$ shared scratch and making no operand or recursive graph queries.*

For every recipe in this class, its normalized value has one deterministic exact $\mathcal{A}$ -stream. For each finite integer clock $t \geq 0$, the digit procedure has uniform constants a, b, c such that every requested operand digit j satisfies

$$j \leq t + cd, \quad \text{total suspended local binary record length} \leq ad + b(t - j). \quad (11.5)$$

The bounds also hold for a requested code bit. A structural leaf is requested at clock zero and has suspended local length at most $ad + bt$. The maximum intrinsic local record length is $O(d + t)$.

All gate, digit and descriptor addresses can be regenerated from root parameters and the saved local records using $O(B + d + t)$ shared scratch and no recursive graph calls. When $d, t = O(B)$ this is $O(B)$ scratch. If a public rational coefficient has $O(B)$ -bit numerator and denominator that can be generated in $O(B)$ scratch, its queried digits in this clock range also require only that shared $O(B)$ scratch.

Proof. The primitive constructions in the following subsections define deterministic integers N_t . Lemma 11.1 turns them into exact streams. When composing primitives, their operands are these exact streams; numerical approximation errors are not accumulated along the recipe as if they were statistical errors. Instead, each primitive's deterministic N_t is an approximation to its own exact mathematical value. The construction uses only residues of N_t and N_{t-1} and a constant number of output code bits, which adds fixed overshoot and fixed local overhead.

For the record bound, choose one common positive clock coefficient b for the finite primitive and stage types. In any bound $L \leq a_0 d + b_0(t - j)$ with $j - t \leq c_0 d$, replacing b_0 by $b \geq b_0$ is justified by replacing a_0 by $a_0 + (b - b_0)c_0$. Thus the same b can be used everywhere. The bounded interface depth leaves only a bounded number of primitive costs on each composition path. An indexed rank has $O(d)$ overhead by Lemma 11.8; within a balanced sum or average, the elementary overheads total $O(d)$ by Lemma 11.9. The other primitive costs are also $O(d)$. Along any path the intermediate clock differences telescope, giving Equation (11.5) and total overshoot $O(d)$. The same sum, terminated at an internal arithmetic operation rather than an operand call, gives the intrinsic bound explicitly as follows. If that operation has current clock u , its own intrinsic record has length at most $C(d + u)$, while the outer saved portion has length at most $Ad + b(t - u)$. Choose the common b also to satisfy $b \geq C$. Their sum is then at most $(A + C)d + bt$, because $u \geq 0$. Increasing the stage constants as above makes this same choice of b valid in all local bounds. The early-word case is already included in Lemma 11.7: those retained bits are part of the local record and are paid for at the actual original-input clock by Equation (11.17).

Structural validity branches are finite branches in this circuit organization, with length at most its full intrinsic bound. They ask the structural operand at clock zero. Buffered tests are separate recipes with structural starting clock zero and precision fixed by Lemma 11.10; they are not altered when a surrounding value is requested more accurately. Thus no discontinuous comparison of exact reals at unbounded precision is part of the interface.

All circuit constructions below use padded domains and index rules. A saved gate choice selects an operand, a subtree, a bit of a known-width word, or one of the finite primitive modes. Scanning these choices from the root recovers the current descriptor and digit places by integer arithmetic. The reciprocal's extra records additionally specify its early-prefix formation state. None of these computations refers to an ancestor cursor address or requires an operand value to discover the next gate. Their integers have $O(B + d + t)$ bits; repeated scans reuse one scratch area. Lemma 11.2 therefore applies at every Boolean portion of the recipe.

Finally, for a public coefficient P/Q , $Q > 0$, producing a place- r approximation uses integer division of $2^r P$ by Q , with signed floors when needed. The inputs and intermediate remainders have $O(B + r)$ bits. Standard finite binary division and multiplication use that much space, and extracting the low residue or difference digit requires no larger word. All operand bits in this computation are public, so it makes no recursive graph call and can use shared scratch throughout. For $r = O(B)$ its scratch bound is $O(B)$. In contrast, any word whose bits depend on recursive operands and must survive a new operand request has already been charged to the local record. $\square$

Remark 11.4 (Use of clocks). Theorem 11.3 is a finite-request statement for every t . Its exact-stream conclusion follows by considering these finite requests separately, and does not require a machine to choose structural domains or port orders at infinite precision. For the actual top-level computation all reached clocks will be $O(B)$ by the allowance calculation in the next section. Known rate and field coefficients have $O(B)$ -bit descriptions, and a fixed rational generator probability raised to a walk length $O(d)$ has numerator and denominator of $O(d)$ bits; these are instances of the public-coefficient provision above.

11.3 Truncations, comparisons, and binary arithmetic

Lemma 11.5 (Finite comparisons and bounded elementary operations). *Pair averages, maxima, minima, and fixed binary shifts of normalized streams admit deterministic approximation integers satisfying Equation (11.3), after the specified fixed range normalization. For any fixed residue width q , there are constants C_0, C_1 such that a place- t residue query requests operand digits only at places $j \leq t + C_0$, with suspended local record length at most $C_0 + C_1(t - j)$ and intrinsic local space $O(t + 1)$.*

These bounds persist, with enlarged constants, under fixed finite compositions of the operations. Fixed Boolean operations on digit codes, including sign decoding and conditional code-bit selection, add only constant local overhead.

Proof. First consider comparing two truncations through $K = t + h$, where h is a sufficiently large fixed integer. Their difference digits have magnitude at most 12. Scanning from place zero to K , update a signed prefix by $p \leftarrow 2p + \delta$. While $|p| \leq 12$, retain p exactly. Once $p > 12$ or $p < -12$, retain only the corresponding absorbing sign state. This sign is correct for every finite continuation: the remaining suffix, in the current scale, has magnitude strictly less than 12. The state space is finite, so each transition has a fixed bounded-fan-in Boolean circuit. The sign at the end determines the exact order of the two finite truncations. The path from that sign bit to an input at place j has length $O(K - j + 1) = O(t - j + 1)$.

Let v_t be the average, maximum, or minimum of the two truncations, and put $N_t = \lfloor 2^t v_t \rfloor$. Each operation is 1-Lipschitz for the sup norm on its operands. Consequently $|v_t - v| \leq 6 \cdot 2^{-K}$, where v is the corresponding exact output. Choosing h so that $6 \cdot 2^{-h} < 1/2$ gives $|N_t - 2^t v| < 3/2$.

To obtain $N_t \bmod 2^q$, only a fixed number of low places of the selected truncation are needed. More explicitly, at scale 2^K , terms of input place $j \leq t - q$ are multiples of 2^{q+h} . Modulo this modulus the remaining $q + h$ places suffice. For an average with integer input prefixes A_K, B_K at scale 2^K , use $N_t = \lfloor (A_K + B_K)/2^{h+1} \rfloor$. Its residue needs modulus 2^{q+h+1} and one additional retained place: only places $j \leq t - q - 1$ can be omitted. The finite number of remaining digits can be added and shifted by a constant-size Boolean circuit. A maximum or minimum first uses the comparison bit to select these residue bits. Its long comparison branch has the path bound proved above, and its residue branch reaches only places within a fixed distance of t . A fixed shift changes these places by a fixed amount. Fixed normalization factors are handled in this way whenever an intermediate range exceeds $[-1, 1]$.

The topology of the comparison chain and the low-residue circuits depends only on the root clock, the fixed margins, and the gate path. Regenerating a place by replaying that path needs no saved binary place counter at each transition. Lemma 11.2 therefore gives the suspended-space bounds. Their longest paths have length $O(t + 1)$.

For composition, use one common coefficient of $(t - j)$ for the finite primitive list. Increasing such a coefficient by Δ requires increasing its constant term by at most ΔC_0 , because the operand overshoot is at most C_0 . With these common coefficients, the clock differences telescope along the composed path. The finite number of constant terms remains a constant. $\square$

We will also use the following explicit Boolean arithmetic facts. A carry bit in binary addition is described by a generate/propagate pair (g, p) . Composition of adjacent intervals is the associative operation

$$(g_2, p_2) \circ (g_1, p_1) = (g_2 \vee (p_2 \wedge g_1), p_2 \wedge p_1).$$

Balanced trees of these pairs give every carry in $O(\log(W + 1))$ depth for W -bit words. A balanced

tree adding R such words modulo 2^W has depth

$$O(\log(R+1)\log(W+1)). \quad (11.6)$$

The trees can be padded by zero words to powers of two, so their gate addresses are computable by ordinary index arithmetic. For counts of R Boolean bits one can do better: constant-depth full adders replace three same-width rows by two rows of their sum, with carries placed in the next column. In $O(\log(R+1))$ rounds only two rows remain. The word width is $O(\log(R+1))$; a final carry computation gives total depth $O(\log(R+1))$. These constructions use bounded fan-in throughout.

11.4 Multiplication by minimum-index bands

Grouping products by their smaller input index leaves a gap between the output clock and the requested input clocks; that gap pays for the retained circuit path within each band.

Lemma 11.6 (Multiplication). *For exact normalized $\mathcal{A}$ -streams representing $x, y \in [-1, 1]$, the product xy admits approximation integers and residue procedures with all bounds in Lemma 11.5. Thus a place- t query has constant operand-clock overshoot, suspended storage at most $C_0 + C_1(t - j)$ at a place- j operand query, and intrinsic space $O(t + 1)$.*

Proof. Fix q and write $A = 6$. Partition ordered pairs (i, j) of nonnegative places into a zero band, where $\min(i, j) = 0$, and positive bands, with the zero band denoted $\mathcal{P}_0 = \{(i, j) : i, j \geq 0, \min(i, j) = 0\}$:

$$\mathcal{P}_D = \{(i, j) : D \leq \min(i, j) < 2D\}, \quad D = 1, 2, 4, \dots$$

Set $h_0 = h_o$ for the zero band and $h_D = h_o + \lfloor D/4 \rfloor$ for positive bands. The fixed integer h_o will be chosen below. Retain in band D only pairs with $i + j \leq t + h_D$, and let

$$S_D = \sum_{\substack{(i,j) \in \mathcal{P}_D \\ i+j \leq t+h_D}} x_i y_j 2^{t+h_D-i-j}. \quad (11.7)$$

For the zero band the same formula uses its defining pair set. These are mathematical integers; no bound on their full binary length is assumed in the residue computation.

At any fixed diagonal $i + j = s$, a positive band contains at most $2D$ ordered pairs. Indeed the smaller place has D possibilities, and an off-diagonal pair has two orientations. The zero band has at most two pairs. Thus, in units of 2^{-t} , the total omitted product terms have absolute sum at most

$$E_{\text{omit}} \leq 2A^2 \sum_{D \in \{0, 1, 2, 4, \dots\}} (D+1) 2^{-h_D}. \quad (11.8)$$

The geometric summation of the diagonal weights gives this bound also when the truncation threshold lies below the first possible diagonal. Both the series in Equation (11.8) and $\sum_D 2^{-h_D}$ converge. Choose h_o sufficiently large that their displayed combined error bound, including coefficient $2A^2$, is less than $1/2$.

There are only finitely many nonempty retained bands. For a positive one, $2D \leq t + h_D$, whence

$$D \leq \frac{4}{7}(t + h_o). \quad (11.9)$$

The left side of the nonemptiness condition grows monotonically faster than its right side along the dyadic bands. We may therefore stop at the last band satisfying this inequality, retaining empty bands within the finite prefix if necessary.

The band widths have different truncation scales 2^{t+h_D} . Before adding a more distant band's tail to the current band, we must express that tail at the current scale. This is the purpose of the signed floor shift in the following recursion. Starting with the most distant retained band, define integers recursively by

$$F_D = S_D + \left\lfloor 2^{h_D-h_{D^+}} F_{D^+} \right\rfloor, \quad N_t = \left\lfloor 2^{-h_0} F_0 \right\rfloor, \quad (11.10)$$

where D^+ is the next band, and the last missing tail is zero. One internal floor, divided back to scale 2^t , changes the sum by less than 2^{-h_D} . Induction down the bands consequently shows that the difference between $2^{-h_0} F_0$ and the sum of all retained terms in scale- t units is at most $\sum_D 2^{-h_D}$. Equation (11.8) and the choice of h_0 make the error before the final floor less than $1/2$. The final floor costs less than one more unit. Therefore Equation (11.3) holds for $z = xy$.

We next implement Equation (11.10) using residues. At band D use modulus 2^{q+h_D} . An integer term in Equation (11.7) with $i + j \leq t - q$ is a multiple of this modulus and can be omitted from the circuit. It remains present in the mathematical integer S_D . Only the diagonals

$$t - q < i + j \leq t + h_D \quad (11.11)$$

need to be processed, giving $O((D+1)(h_D + q + 1)) = O((D+1)^2)$ candidate pairs.

Signed floor shifts are well defined on the required residues. For $h' \geq h$ and $a - b = k2^{q+h'}$,

$$\left\lfloor \frac{a}{2^{h'-h}} \right\rfloor - \left\lfloor \frac{b}{2^{h'-h}} \right\rfloor = k2^{q+h}. \quad (11.12)$$

This identity holds for all signed integers a, b . Thus a next-band residue modulo $2^{q+h'}$ supplies the correctly floor-shifted tail modulo 2^{q+h} . Apply the same observation to the final shift by h_0 to obtain $N_t \bmod 2^q$. In the finite binary circuit, this shift just discards the lowest $h' - h$ bits of the canonical $(q + h')$ -bit residue. The displayed identity proves that this is valid also when the underlying integer is negative.

The finite gate structure can be indexed without storing an absolute operand place at each gate. For a positive band enumerate triples (r, s, o) , where $D \leq r < 2D$, $t - q < s \leq t + h_D$, and $o \in \{0, 1\}$. The triple supplies $(r, s - r)$ if $s - r \geq r$ and $o = 0$; it supplies $(s - r, r)$ if $s - r > r$ and $o = 1$; otherwise it supplies a zero term. This enumerates the relevant pairs exactly once. For the zero band use $r = 0$ with the same orientation rule. Negative candidate places are invalid and supply zero. The offsets $r - D$ and $s - (t - q + 1)$ have $O(\log(D + 2))$ bits. They and o are obtained from paths in padded binary selection trees. The full places are then computed from the root clock in scratch.

Each nonzero term is a constant-bit signed product $x_i y_j$, shifted into a word of width $W_D = q + h_D = O(D + 1)$ and interpreted modulo 2^{W_D} . A bit of this word is a fixed Boolean function of the two bounded digit codes, the bit index and the known shift; negative products use their ordinary residue, equivalently sign extension before reduction. Add these words and the shifted tail using Equation (11.6). A band contributes depth $O(\log^2(D + 2))$. Bit selection for the shifted tail is wiring and does not require evaluating a full tail word.

The sum of these depths through the bands up to D is $O(\log^3(D + 2))$, hence at most $C(D + 1)$ for a fixed C . For an original input digit in a positive band, its partner place is at least D . Equation (11.11) gives

$$j \leq t + h_D - D \leq t + h_0 - \frac{3}{4}D. \quad (11.13)$$

Equivalently, $t - j \geq \frac{3}{4}D - h_o$, so the full path cost through these bands satisfies

$$C(D + 1) \leq C \left(1 + \frac{4}{3}h_o \right) + \frac{4C}{3}(t - j).$$

This gives $C_0 + C_1(t - j)$ with fixed constants. The zero band has constant depth and only constant overshoot. All its gate descriptions, term coordinates, bit positions and tail shifts are computable by replaying paths and doing integer index arithmetic from t and q . Lemma 11.2 converts these depth bounds to suspended storage. Finally, Equation (11.9) bounds every intrinsic path and local word width by $O(t + 1)$. $\square$

11.5 Reciprocals and the early-word convention

A prefix through roughly half the output clock makes the remainder in a first-order reciprocal expansion quadratically small. Releasing that prefix before later input queries keeps its long word from surviving a call with little remaining clock gap.

Lemma 11.7 (Reciprocal on a positive compact interval). *Fix $0 < a \leq b$. On inputs $x \in [a, b]$, the reciprocal $1/x$, after a fixed known normalization, admits the approximation, operand-clock, suspended-record, and intrinsic-space bounds of Lemma 11.5, with constants depending only on a, b . An $O(t + 1)$ -bit local word is retained across input queries only at places $j \leq \lfloor t/2 \rfloor + O(1)$.*

Proof. The input can first be normalized by a fixed binary scale. It is enough to prove the result with a fixed digit alphabet and fixed positive interval; all constants below may depend on a, b . Choose a fixed power of two C large enough that the desired output $z = 1/(Cx)$ lies in $[0, 1]$ and that the coefficients below have absolute value at most one. Put

$$k = \lfloor t/2 \rfloor + c, \quad x^0 = x^{[k]}, \quad e = x - x^0,$$

where c is a sufficiently large fixed integer. By Equation (11.2), $|e| \leq 6 \cdot 2^{-k}$. Choosing c first ensures $x^0 \in [a/2, 2b]$ and $|e| \leq 1$ at every t . Increasing C if necessary gives bounded coefficients

$$\alpha = \frac{1}{Cx^0}, \quad \beta = -\frac{1}{C(x^0)^2}.$$

The exact algebraic identity

$$\frac{1}{Cx} = \alpha + \beta e + \frac{e^2}{C(x^0)^2 x} \tag{11.14}$$

shows that the linear expression $T_t = \alpha + \beta e$ differs from z by at most a fixed constant times 2^{-2k} . Choose c large enough that

$$2^t |T_t - z| \leq \frac{1}{4} \quad \text{for every } t \geq 0. \tag{11.15}$$

Clipping T_t to $[-1, 1]$ cannot increase this error; denote the clipped expression by $\hat{T}_t$.

For this fixed original t , the coefficients α, β are exact rational numbers. Their digits are produced by integer arithmetic, as described below. The suffix e has the exact stream with digit zero through k and digit x_j thereafter. Thus Lemmas 11.5 and 11.6 provide exact streams for $\hat{T}_t$, using only a fixed number of normalized operations. Explicitly, form the product βe , then the pair average $u = (\alpha + \beta e)/2$, clip u to $[-1/2, 1/2]$ by a maximum and a minimum, and finally apply the fixed shift by 2. Every exported intermediate value is normalized, and the final value is $\hat{T}_t$. Forming this expression introduces no additional reciprocal operation.

Let $P_s(\widehat{T}_t)$ be the integer approximation furnished by those operations at place $s = t + 3$, and define

$$N_t(z) = \left\lfloor P_{t+3}(\widehat{T}_t)/8 \right\rfloor. \quad (11.16)$$

The cutoff k and the coefficients remain those of the original t , not those of s . Equations (11.3) and (11.15) give

$$|N_t(z) - 2^t z| < 1 + \frac{2}{8} + \frac{1}{4} = \frac{3}{2} < 2.$$

To obtain this integer modulo 2^q , ask for the approximation P_{t+3} modulo 2^{q+3} and use Equation (11.12).

It remains to justify storage in the coefficient computations. The signed numerator of x^0 in units 2^{-k} has $O(k + 1)$ bits. Form it by querying places $0, \dots, k$ and updating the integer prefix. The numerator, a place counter, and all data that must survive the next query are part of the local record. To return a coefficient digit of place r , numerator squaring and integer division require $O(k + r + 1)$ bits. Here $r \leq t + O(1)$, because the enclosing expression has fixed depth and constant overshoot. After the input prefix is obtained, these integer computations make no further operand queries. Standard binary long multiplication and division, performed by bounded loops with their partial products or remainder, use only this linear number of bits. For $r > 0$, computing floors of the coefficient scaled at r and $r - 1$ gives its exact difference digit; at $r = 0$ use the place-zero floor alone, as in Lemma 11.1. Clipping is unnecessary because the coefficients were normalized in advance.

Every original-input query during this formation has $j \leq k \leq t/2 + O(1)$. Hence

$$t + 1 \leq C_2 + C_3(t - j) \quad (11.17)$$

for fixed constants, at all such queries. The enclosing gate path also has length $O(t + 1)$, so the complete suspended local record satisfies the required bound there. Equivalently, the coefficient branch is charged against the original reciprocal clock t , not against its possibly small coefficient digit place r .

On a suffix branch with $j > k$, the coefficient word is absent. The Boolean evaluation asks for a coefficient digit at a separate leaf, returns its bounded code bits, and discards the coefficient word. If the other child of a gate subsequently asks for a late suffix digit, only the bounded pending bits and gate choices are retained. The path bound from the preceding arithmetic lemmas therefore gives $O(1) + c(t - j)$ also on this branch. Repeated coefficient requests rebuild the word. The parameter k and the choice of coefficient are regenerated from the original t and the saved gate path; no earlier cursor address is required. All intrinsic records have $O(t + 1)$ bits. $\square$

A reciprocal coefficient request must first query the operand-dependent prefix x^0 , so it cannot use the public-coefficient interface. It forms that charged early word, computes in scratch, returns one bounded digit, and releases the word before a suffix branch runs. The original clock t remains reconstructible. For the early queries, $j \leq \lfloor t/2 \rfloor + c$ gives the explicit storage payment $t + 1 \leq 2(t - j) + 2c + 1$.

11.6 Indexed ranks, sums, scales, and structural leaves

Lemma 11.8 (Indexed finite ranks). *Fix $\kappa > 0$ and $d \geq 1$. Let a semantic list have public length $n \leq 2^{\kappa d}$, normalized entries, and descriptors of at most κd bits. A structural validity predicate may assign prescribed constant dummy values to absent descriptors. For an $O(d)$ -bit rank index in $\{1, \dots, n\}$, its order statistic has deterministic approximation integers N_t satisfying Equation (11.3); an out-of-range request returns its prescribed normalized constant.*

There are constants a, b, c , depending only on the descriptor bounds, such that every operand digit request at place j obeys

$$j \leq t + cd, \quad L \leq ad + b(t - j), \quad (11.18)$$

where L is the total suspended local record length. Structural validity calls use clock zero and suspended length $O(d + t)$. The intrinsic local space is also $O(d + t)$.

Proof. For $1 \leq k \leq n$, the k th largest of the semantic n -entry list is its $(n - k + 1)$ st ascending order statistic, before any extra padding. Pad to a fixed descriptor domain whose cardinality is a power of two, adding entries of value $+1$. These entries do not change any of the first n ascending order statistics. An empty list or an out-of-range request returns its specified default immediately. Replace invalid original descriptors by the prescribed dummies. An operand access evaluates validity first and enters the actual value branch only when valid; an invalid access returns the dummy's digit. Thus a dummy does not require a query to an undefined value. For each valid numerical value use its truncation through $K = t + h$ for fixed large h . Order these finite rational values, breaking ties by descriptor index. Compare every pair using the signed comparison of Lemma 11.5. For each descriptor count how many descriptors precede it, then compare that count to the requested rank minus one. Exactly one descriptor is selected. Select its low-residue bits using Boolean AND and OR trees and finally floor to output scale 2^t . The selected descriptor is internal to this residue computation. It may change with t ; the exact order statistic being approximated is fixed by the semantic list and the requested index.

If all coordinates of two finite lists differ by at most η , their corresponding order statistics differ by at most η . Indeed, at least the specified number of the perturbed entries are no larger than the original order statistic plus η ; interchanging the lists gives the reverse inequality. Here $\eta = 6 \cdot 2^{-K}$. Thus the selected truncation, followed by the final floor, satisfies Equation (11.3) for sufficiently large fixed h . Descriptor ties affect neither this estimate nor the uniqueness of the finite selection. This argument perturbs the entire padded exact list. Some truncated original entries may exceed $+1$ slightly, but their error, and hence the error of the selected padded order statistic, is still at most η . The padding argument requires no separate claim that every finite truncation remains in $[-1, 1]$.

Each count has $O(d)$ bits. Carry-save counting as described above, the comparison to the requested rank, and selection among $2^{O(d)}$ descriptors add $O(d)$ Boolean depth above the pairwise comparison bits and selected residue bits. A place- j input to a pairwise comparison consequently has path length $O(d) + O(t - j)$; a residue input is within a fixed distance of t and has path length $O(d)$. The paths to validity predicates have length at most the whole circuit depth $O(d + t)$, and the predicates themselves are separate structural leaves of clock zero.

Each gate is specified by its pair of descriptor indices, its counting-tree or selection-tree position, and a comparator or bit position. The paths determine these indices in $O(d + t)$ scratch, using only the fixed domain and root parameters. No numerical comparison is needed to determine the circuit topology. Apply Lemma 11.2. The constants in Equation (11.18) can absorb the fixed digit overshoot because $d \geq 1$. $\square$

For the compression ranks, copies of a previously computed baseline are ordinary indexed operands: their digit interfaces query the baseline's exact stream one bounded digit at a time. A slot that chooses between an entry and the baseline first evaluates its structural flag and then queries the selected, already-defined stream. Treating this as a defined operand interface leaves no absent value for the rank procedure to evaluate. The required baseline copies belong to the semantic list; the extra power-of-two padding in the proof is then supplied by the constant $+1$. The structural choice adds only a fixed conditional branch, so the same digit-clock and suspended-record bounds apply.

A rank over the full incoming vertex universe need not have a $2^{O(d)}$ descriptor domain. That operation uses the terminal configuration procedure in the next section.

Lemma 11.9 (Bounded sums and known scales). *Fix $\kappa > 0$ and $d \geq 1$. Suppose an immediate domain has at most $2^{\kappa d}$ descriptors, all required binary scale exponents have magnitude at most κd , and all intermediate range bounds are at most $2^{\kappa d}$. Balanced sums and averages, with their prescribed known rescalings and bounded clipping, satisfy Equation (11.18) and use intrinsic local space $O(d + t)$. An empty sum is zero; averages are over public nonempty domains.*

Proof. Choose one common known power-of-two range bound for the immediate terms and divide each term by it, so every term lies in $[-1, 1]$. Pad their number to a power of two 2^r , $r = O(d)$, and use a complete binary tree of pair averages. Every internal value stays normalized. Along one path there are r elementary operations. Choose common coefficients in their bounds, as in the last paragraph of the proof of Lemma 11.5. The constant terms sum to $O(r)$ and the differences of successive digit clocks telescope. Thus the local length is $O(d) + b(t - j)$ and the total overshoot is $O(d)$. Restore the common normalization scale only after the tree of pair averages. For a sum, combine this factor with the known padding count 2^r in one binary shift of magnitude $O(d)$. For an average over an original domain of size m , combine the scale restoration with multiplication of the padded average by the public factor $2^r/m$. With minimal padding the latter factor is in $[1, 2)$, so it requires only one normalized public multiplication and a fixed shift in addition to the common scale restoration.

More explicitly, for a shift by 2^v , truncate the operand at $K = \max(0, t + v + h)$. If $t + v + h \geq 0$, its scaled truncation error in scale- t units is at most $6 \cdot 2^{-h}$. If the scaled output is known to be normalized, floor in the desired output scale. Otherwise, when a bounded clip is required, apply that clip to the scaled truncation before flooring. For the normalized clip to $[-1, 1]$, for example, define the approximation using $\min(1, \max(-1, 2^v x^{[K]}))$; its error cannot increase. This shift-and-clip procedure does not first export an unbounded intermediate as a normalized stream. The low residue requires the constant number of places near $t + v$, and a comparison needed for clipping has a constant-state chain through K . Comparisons to the scaled clipping endpoints are equivalently comparisons of $x^{[K]}$ to the known endpoints divided by 2^v ; their finite constants and shifted place data are public. When $v \geq 0$ these endpoints belong to $[-1, 1]$; when $v < 0$ the normalized operand is already in the clip interval after scaling, so no clipping comparison is needed. Its path cost is $O(1) + O(\max(0, t + v - j))$, which is at most $O(d) + O(t - j)$ on an actual operand query. If $t + v + h < 0$, zero is already a sufficiently accurate approximation after increasing h , and no operand query is needed. A fixed number of such rescalings and clips preserves the bounds by telescoping. Known non-power-of-two coefficients can instead be handled as public normalized streams and bounded multiplication. $\square$

Lemma 11.10 (Finite buffered tests). *Fix $\kappa > 0$. Let $d \geq 1$, $|s|, |u| \leq 2^{\kappa d}$, and $u \geq \rho \geq 2^{-\kappa d}$. A deterministic finite test can accept every $s > 2u$ while accepting only $s > u$. It uses normalized digit places $O(d)$ and intrinsic and suspended local space $O(d)$. Its precision and tie convention are fixed by these data, independently of later numerical output requests.*

Proof. Compute finite rational approximations $\tilde{s}, \tilde{u}$ with individual errors at most $\rho/16$. The known range and lower bound on ρ make $O(d)$ normalized digits sufficient. Accept exactly when $\tilde{s} > \frac{3}{2}\tilde{u}$, using a fixed convention at equality. If $s > 2u$, the left side minus the right side is greater than $u/2 - 5\rho/32 > 0$. If $s \leq u$, it is at most $-u/2 + 5\rho/32 < 0$. The finite comparison and all previous arithmetic have the bounds already proved. The test chooses its precision from d and ρ , not from an enclosing numerical clock. With its structural starting clock defined as zero, its operand requests have $j = O(d)$; increasing a in Equation (11.18) pays also for the term $-bj$. $\square$

12 Uniform controller implementation

We now implement the table and vector queries defined in the preceding sections. Theorem 11.3 supplies their numerical operations. We must arrange the recursive graph accesses so that the records retained by all callers fit in logarithmic space at the same time.

We keep one vertex identifier, called the cursor. Consider a product term $\mathcal{F}(x, z)\mathcal{B}_b(z, y)$ from Equation (8.2). For a selected valid product descriptor, a query begins with the cursor at x . The correction port moves it to z and supplies a bounded inverse port. The right factor can then be queried at z , using its own common table tuple; that tuple contains no copy of x . Returning along the inverse port restores x . Arithmetic requests the two factor digits separately, so it need not retain a complete left-factor value while computing a right-factor digit. If the operation instead asks to reach y , the right factor's terminal motion is the last recursive delegation; only local label completion follows it. These are the paths that Lemma 8.5 requires us to realize.

Incoming access reverses a different object: a finite program whose canonical initial states describe the forward row ports. We make the program deterministic with halting success states. The component of a specified success state is then a tree directed toward that state; its canonical initial states are precisely the valid row ports ending there. A contour of that tree enumerates them and returns to the success state without saving a second copy of its vertex address. This gives both capped incoming ports and incoming score ranks. To use the contour we must define program states and their predecessor queries carefully: working states of a completed lower query are not themselves part of the calling program graph.

The storage bound follows the same distinction. A caller retains its own descriptor choices and numerical path while a lower procedure runs. A slack- d layer pays $O(d)$ structural bits; a query from digit place t to place j pays for its numerical records through the bound $O(d) + c_{\text{num}}(t - j)$ of Theorem 11.3, for a fixed coefficient c_{num} . We assign a decreasing allowance to the finite operation order within a prefix, the move to its preceding prefix of slack $2d$, and the move to a child of budget at most $M - d$. The resulting differences telescope over the physical stack. The proof below establishes these record bounds on the actual recipes before imposing guards on artificial program states.

12.1 Parameters and interfaces

Fix the input and a master environment. The latter includes the rank matrices and all the fingerprint salts. A *table tuple* consists of the estimator type and stage, its statistical budget, scale and usage rates, channel roles and suffixes, and its effective environment. The tuple specifies an entire table, rather than an individual row or entry. Usage rates belong to this tuple; the separate integer clock t specifies a requested numerical digit. A generator and wrap word is a call-record descriptor used to derive a child's effective environment. It is not an additional semantic parameter of the child table: after this derivation, the child's values, forward port orders and guards depend on the resulting effective tuple, not on the ancestral word which produced it. Distinct walk descriptors are still separate summands with their prescribed weights, even if they induce the same environment transformation. The variants of Lemma 7.3 can be generated without their conditioning vertices. A vector tuple is defined in the same way, omitting the unneeded terminal parameters. All admissible absolute tuples have $O(B)$ bits.

The mutable cursor records a vertex and its stage interpretation. Other shared facilities are the input, the top root parameters, the master environment, a catalytic bit vector of length B , and one reusable scratch area. Its size is $O(B)$ in the actual computation, where the top numerical clock is $O(B)$; for a general top clock t it may use $O(B + M_{\text{top}} + t + 1)$ bits. Scratch may be used to perform arbitrary local integer computations in this space. It has no content which must survive a recursive graph operation. Every such operation starts its scratch calculations canonically. Before every input access, its requested position is regenerated in shared scratch. The physical input head stays between the real endmarkers; off-data simulated positions use the base symbol routine from Lemma 2.1. These rules also apply during inaccurate calculations and guarded artificial predecessor probes. The requested coordinate and scan counters coexist only within this pure routine and are discarded before a recursive graph operation. It is harmless to copy an address temporarily inside a scratch calculation, but such a copy is discarded before a recursive graph operation begins.

The finite interfaces to be constructed are as follows. A port is an index in a specified row class; the class may have holes.

1. A *restoring test* returns one bit. A *quantity query* returns one bounded signed digit of a specified value. Both restore the cursor and the catalytic vector.
2. A *terminal motion* starts at a row and a port and finishes at that entry's endpoint. No return address is provided. An *excursion* makes the same visit, performs a prescribed local action there, and returns. The action may use scratch and the catalyst but makes no recursive graph query.
3. A *paired motion* maps a valid row port to its endpoint and a bounded inverse port. Its inverse has the same table tuple. On invalid ports the restoring existence test returns false; an exported motion may instead return failure without moving the cursor.

At the final row interface all port alphabets are bounded by constants. At a prefix of slack d , indices and immediate descriptors have $O(d)$ bits. These two statements must be distinguished: an entire terminal path represented by one prefix port need not have a description of length $O(d)$.

A port code may be in range and still name a hole in its class. Its existence test returns false, its quantity query returns the zero digit, and its motion or excursion fails with cursor and catalyst unchanged. Rank operands retain their separately prescribed baseline or padding values: an absent real port is not a virtual default coordinate.

Theorem 12.1 (Realization of the estimator interfaces). *Fix the construction constants and an admissible top budget $M_{\text{top}} = O(B)$. Every actual row, vector, and prefix task reachable from it has a uniform deterministic implementation of the restoring-query and cursor-motion interfaces above. For every master environment, including those with inaccurate estimates:*

1. *Tests and numerical digit queries terminate with their defined values and restore the cursor and catalyst. Their results are independent of incoming scratch and catalyst contents. Structural ports and orders do not depend on an enclosing numerical clock.*
2. *Terminal motions reach their specified endpoints, and excursions restore the cursor. Capped correction motions are partial paired maps on the same table tuple, with bounded inverse ports and the stated failure behavior on invalid ports. Incoming score queries return the stipulated zero-padded ranks.*
3. *For a finite root clock $t \geq 0$, the total retained call records and all descendant clocks are $O(M_{\text{top}} + t + 1)$. Shared facilities occupy $O(B + M_{\text{top}} + t + 1)$ space. Thus a root query with $t = O(B)$ uses total space $O(B)$.*

The implementations retain no second vertex address across recursive graph operations.

The proof also constructs every lower operation instance admitted by the finite parameter ranges and permitted parent-to-child recipes. This includes calls made while testing possible predecessor states, whether or not those states occur in a designated execution. The larger construction is needed to compute incoming accesses by inverse traversal.

12.2 Program states and record bounds

A terminal program has a fixed root tuple and root clock. Its initial configuration is the cursor x , a distinguished initiation tag, and the original row port p in a separate recoverable field. Distinct ports give distinct initiation states. In a scoring program the success local state contains a quantized score; otherwise it has only a success tag. The starting row and port are not root parameters. The program universe thus includes all rows and all ports of this same table. Similarly, a scoring universe at a fixed precision includes all possible quantized scores.

There are two different forms of invocation inside this program. A restoring bit query or a lower paired motion is represented by one transition in the terminal program graph. Its temporary working configurations are excluded from that graph. They are computed by the lower procedure when the transition is probed. A terminal delegation with no bounded inverse port is instead *explicit*: it pushes the lower terminal controller into the local state. Thus the vertices of the program graph comprise the cursor and a finite chain of explicit controllers, but not the working states of transition queries implemented by subroutines. This distinction will also be used when graphs are traversed inversely.

Definition 12.2 (Records and parameter reconstruction). A local record is a delimited binary word with a tag from a fixed finite alphabet. It may contain immediate descriptor choices, bounded port or copy symbols, offsets, Boolean evaluation paths and pending bits, and the early numerical words permitted by Theorem 11.3. Head positions in such words are marked locally. A root tuple, a root operation and its numerical clock, together with the active records, determine every descendant tuple and clock by a scratch computation. This computation does not call a graph procedure and does not use a cursor that has been left behind. Thus the parameters of every descendant can be reconstructed from these records.

The records use delimited encodings, with marked positions rather than an independent absolute stack pointer at every level. Self-delimiting encodings enlarge a nonempty word of length r by

$O(r + 1)$ bits. Constant tags also identify a root boundary, the active candidate copy during a neighbor test, and the owner of a pending endpoint action. These tags permit a scratch scan to select the appropriate active view of the records. An inactive copy is not part of a descendant's program state or call recipe.

A constant tag marks each root boundary. Replay up to that tag recovers a nested query's logical root parameters and hence its mathematical program graph. The temporary absolute tuple is discarded before another graph query, so nested roots require no retained $O(B)$ -bit parameter word at each level. Lemma 12.12 supplies the replay procedure.

Definition 12.3 (Allowance and local storage bounds). A raw controller is the caller's own selection, arithmetic, scoring or terminal-path record, excluding the working states of a query it calls. In a terminal program graph an explicit chain contains only its terminal-path controllers and, in a scoring program, its score suffix or completed score word. The interiors of cap traversals, inverse score ranks, numerical digit queries and paired motions are subroutine queries; they are never explicitly pushed working graphs in this chain.

A raw controller has an integer allowance $w \geq 1$ and a numerical clock $t \geq 0$. Fixed raw constants $a_0, b_0, a' > 0$ give the following bound at a call to a lower query or an explicitly pushed raw terminal controller at (w', j) :

$$w' < w, \quad j - t \leq a'(w - w'), \quad \ell \leq a_0(w - w') + b_0(t - j). \quad (12.1)$$

Here ℓ counts only the suspended caller record, including its delimiters, return modes and pending data. It does not count the callee's physical working space. Without a pending call its own raw records have length at most $a_0w + b_0t$. A structural operation has a canonical starting clock zero; numerical queries used by its finite-margin tests are included in this bound.

We will prove this bound for the actual recipes below, before using guards to restrict artificial program states. In particular, guard insertion will not be used as a substitute for proving that a designated execution fits its bound.

We use the following finite instruction types.

1. *Local edit.* Read a bounded number of marked local symbols, change a bounded number of symbols and a finite mode, and advance the marked local heads by one position. A push opens a delimiter and writes a tag in a bounded number of steps. A pop is performed only after the child word has been erased down to that tag, also by local steps.
2. *Pure or restoring branch.* Compute one bit, either in scratch without a graph call or by a lower restoring test or digit query, and change a bounded local symbol or mode. A bounded signed digit is handled as a fixed number of bits. A longer selection result is written one bit at a time, with its index stored in the local record.
3. *Paired cursor step.* With its motion tuple and short token already selected, apply one lower partial paired map, or one elementary base or label map. Its output token and finite mode are the only simultaneous local changes. The linear-size motion tuple is regenerated from records; it is not written as an opaque payload.
4. *Explicit terminal call.* Create a lower controller with a reconstructible call recipe. Ordinary instructions then execute that controller. On its return, local erasure and the prescribed bounded label completion are performed by the preceding instruction types.

An endpoint action in an excursion may be a pure computation reading or toggling the catalyst. Such actions are not individual edges of a terminal program graph: that graph uses the complete restoring, catalyst-independent predicate which contains the excursion. Its other pure branches

depend only on initialized scratch, the input, cursor and permitted tuple data. A value test is never combined with a cursor-changing step. Conditional validity can be a separate restoring branch; a partial paired step may simply have no edge on an invalid domain.

Every record position erased by a local edit is marked before the edit, or is a known end of a delimited word. In particular, an instruction cannot erase an arbitrarily chosen position and then forget its index. Counters, carries and loop heads are manipulated by these local steps. Pure computations and restoring tests can be recomputed to select one output bit; their scratch states and physical input heads are not included in the states whose predecessors are enumerated.

Predecessor probes can reach artificial local states with unverified stored flags, so every lower query validates its arguments at entry. An argument is *range-valid* when its tuple, operation type, cursor type, port code and clock lie in the permitted finite ranges. A range-valid port can still name a hole: every exported existence predicate therefore makes a total restoring test of its earlier defining masks. Every port-dependent quantity query freshly tests presence at structural clock zero before entering its value calculation. If presence fails, the query returns the zero digit of the zero stream. A forward motion or excursion instead returns failure, with cursor and catalyst unchanged and without performing an endpoint action. A paired inverse uses its target-side incoming-count and inverse-rank validation. Excursion validity is checked before building its chain of terminal-path records, not anew at the remote endpoint or during rollback. An endpoint comparison returns false if either freshly tested descriptor is invalid. Wrong types or out-of-range codes are rejected by a pure check before any lower call.

Baseline queries and rank-padding inputs retain their separately specified values: an absent port is not a virtual default coordinate. In particular, incoming score ranks are padded by zero, whereas a compressed-array rank uses copies of its baseline. Every subroutine query tag invokes the canonical entry of one of these total procedures. No permitted call recipe targets an internal continuation after its presence test. A historical validity flag in a caller is never accepted as a certificate of presence. Lemma 12.11 proves that the fresh tests use earlier operations and fit the allowance bounds.

12.3 How the implementation bounds combine

Lemma 12.11 bounds each actual recipe: at slack d , an operand request from clock t to j retains at most $c_{\text{str}}d + c_{\text{num}}(t - j)$ bits and has $j - t \leq c_{\text{dig}}d$. Its remaining dependencies go to the preceding prefix of slack $2d$ or to budget at most $M - d$. Lemma 12.12 shows that their parameters and permitted salts are recoverable from these records, including in predecessor probes. The guarded graph lemma below then enlarges the raw bound once to account for the fixed number of copied local states.

Here is why these local conclusions imply a simultaneous space bound. Choose an integer S larger than the number of operation strata, $K_0 > S + 1$, and $K_1 > 2(K_0 + S + 1)$. At budget M , slack d , and stratum $s_0 \in \{0, \dots, S\}$ assign $w = K_1(1 + M) - (K_0 + s_0)d$. Every same-slack, preceding-prefix, or smaller-budget dependency then decreases w by a fixed positive multiple of d . The recipe bounds consequently have the form (12.1); the graph lemma gives enlarged constants A, D for which each physical segment costs at most $A(w - w') + D(t - j)$. Along a chain (w_i, t_i) the sum, including the bottom operation, is

$$\sum_{i < r} [A(w_i - w_{i+1}) + D(t_i - t_{i+1})] + Aw_r + Dt_r + O(1) = Aw_0 + Dt_0 + O(1).$$

Also $t_i + a'w_i$ cannot increase, so the same initial bound controls every descendant clock and the shared scratch. This is the space deduction in Section 12.8.

Totality and the meanings of incoming operations require a separate step. On the guarded graph, Lemma 12.5 enumerates the canonical initiations reaching one success halt. Lemmas 12.6 and 12.7 turn that enumeration into the capped paired map and zero-padded incoming rank. The final proof constructs these interfaces in increasing allowance, so each graph probe uses already total lower procedures. None of these conclusions uses accuracy of an estimator value.

Lemma 12.4 (Guarded local graph). *Fix a terminal recipe of the specified instruction types, with parameter-reconstructible records satisfying Equation (12.1) on designated executions. Assume its lower transition procedures are already implemented, total on admissible arguments, and safe on invalid ports. Their bits, flags, cursor outputs, and port tokens must be independent of incoming scratch and catalyst contents, and they must restore the catalyst.*

This recipe admits a finite guarded program graph of outdegree at most one, uniform in its root tuple, with these properties:

1. *Its designated executions are exactly the prescribed terminal executions. Successes are halts, with tags distinct from initiation tags.*
2. *Predecessors, ordered incidences, and paired edge motions are computable with one cursor and a bounded set of case labels. Every lower query made during a probe is preceded by its local bound check.*
3. *If a root at (w, t) probes a lower query at (w', j) , all live local probe records, including the fixed number of saved or edited controller copies, occupy at most*

$$A(w - w') + D(t - j) \tag{12.2}$$

bits for fixed enlarged A, D . Shared facilities and the outside stack are not copied.

For a fixed logical root tuple, the graph and guards are independent of the originating row, outside stack, and catalyst contents.

Independence from the outer stack is understood after fixing the logical root tuple: two invocations that regenerate the same tuple define the same graph. Their physical replay histories may differ. The hypothesis on designated executions is proved for the concrete recipes in Lemma 12.11; the guards below extend query safety to artificial states reached while enumerating predecessors.

Proof. Use as potential states the cursor and delimited words having the specified finite tags. Check absolute parameter ranges, descriptor ranges, stage and suffix types, instruction modes, and the allowed parent-to-child recipes. Check the intrinsic word bounds of Definition 12.3. For each explicit chain also check the local bounds along that chain. Before any subroutine query, check the bound from every ancestor of its local explicit chain to that query. All absolute parameters, clocks and lengths needed for these checks are regenerated or counted in scratch. No graph query is needed for a guard. At a state with an unavailable instruction omit that edge.

The designated paths pass the guards by hypothesis: their boundary bounds add along an explicit chain and telescope. Enlarging the coefficients once covers fixed instruction tags and bounded intermediate edits. The resulting graph is finite since its local words have $O(w + t)$ bits and its cursor has $O(B)$ bits. It is deterministic since every branch bit and partial paired map is deterministic. Guards may create additional dead ends, but they cannot create a new path from a canonical initiation to a success state.

For a local edit, predecessor candidates are obtained by guessing the bounded overwritten symbols, preceding finite mode, and bounded head moves, then testing the instruction forward. Marked positions and incremental erasure make the number of cases bounded. For a branch, perform the same fresh restoring query at the unchanged cursor and test the guessed preceding symbols.

Incorrect historical branch bits do not resume the interior of a lower procedure: the query is always started again on its regenerated arguments. The guards are checked before the query, including for a candidate predecessor.

For a cursor step, first construct the possible preceding local record by a bounded inverse edit, guessing the bounded preceding token if needed. Before any motion, replay that candidate's active records from the fixed root tuple. Check the instruction and stratum, permitted child kind, budget, scale and suffix ranges, salt permission, numerical clock and every ancestor-to-query raw length bound. These allocation and recipe checks do not use the predecessor cursor. The lower motion tuple is then fixed by the candidate record and the present output token. Apply the partial inverse at the present cursor and output token. If it succeeds, the returned token and preceding local symbols specify the candidate predecessor. Static checks and token comparisons can be made there. A guessed preceding port is compared with the actual port returned by the inverse. Any remaining cursor-specific type or domain check is pure at that cursor; a domain condition requiring another restoring query belongs to a separate branch edge. Thus it does not conceal a second high-cost query inside this paired step. To restore the probe, apply the paired forward map with that same tuple and the actual returned token. No saved copy of the cursor is used. Conversely, an outgoing paired probe checks the proposed target guards after its motion; if a guard fails, or if the probe is only a restoring incidence test, its inverse with the actual output token restores the starting cursor. Both directions have been allowed the same lower-operation cost. No unrelated value query is part of this cursor step.

Label an edge by its finite instruction case and bounded erased symbols. Its forward instruction determines a canonical reverse case code. An inverse probe accepts only that code. This removes any duplicate descriptions of the same predecessor edge without comparing two retained addresses. The finite case codes give a fixed local order and a bounded incidence token; a forward edge knows its reverse token.

For the space assertion, first ignore saved probe copies. Adding the boundary inequalities along the current explicit chain gives the bound $a_0(w - w') + b_0(t - j)$. A current state and a candidate predecessor differ in length by a constant: this follows from the bounded edit used to *construct* every candidate, before it is known to be an actual predecessor. Consequently retaining or editing both costs only a fixed multiple of that bound, with a constant adjustment absorbed by the strict allowance decrease. More explicitly, write $\Delta w = w - w' > 0$ and $\Delta t = t - j \geq -a'\Delta w$. For a fixed copy factor k and additive bounded metadata cost h , choose

$$D \geq kb_0, \quad A \geq ka_0 + (D - kb_0)a' + h. \quad (12.3)$$

Then $k(a_0\Delta w + b_0\Delta t) + h \leq A\Delta w + D\Delta t$. This also handles a negative clock decrement. These coefficients are uniform over all finite instruction cases.

The constants a_0, b_0 describe the raw skeleton uniformly over the whole recipe family. They have not been replaced by the physical probe constants A, D in a lower skeleton's guards. Fix the maximum finite copy factor k and metadata cost h and choose A, D once. A lower compiled query contributes a new physical stack segment when it runs; its contour state and saved copies are not words in the calling graph. Thus this estimate copies raw records, not another A, D -sized physical segment.

Only the local explicit chain is saved or edited. A nested query has its own root boundary and appends its local portion below the existing stack; its program graph excludes the caller's saved copies. A tag selects the candidate chain for parameter replay, ignoring an inactive saved copy. Therefore a later neighbor probe does not copy the older outside portions again. Finally, all guards and forward case codes use just the fixed tuple and its permitted records. Imposing the same

salt-visibility convention as the underlying recipe preserves this independence even at synthetic states. $\square$

12.4 Contours and incoming operations

The traversal composes cyclic rotation of incidences at a vertex with reversal across an edge. This composition has one cycle per tree component by Cook and McKenzie [7, Proposition 1, p. 387]. We add a sentinel incidence locally to include an isolated root. For space-preserving reversible simulation, see Lange, McKenzie and Tapp [9]. The contour and incoming operations also require the separate allowance and guard bounds to control their local records.

Lemma 12.5 (Inverse contour). *In a finite deterministic program graph supplied by Lemma 12.4, the undirected component of a specified canonical success halt is a tree directed toward that halt. Mark its root sentinel and one distinguished occurrence of each canonical initiation on its contour. Successive and preceding marks can be reached by mutually inverse operations that retain only the current program state and bounded incidence control, without a saved root address or contour-length counter.*

Proof. Following the unique successor from any node in the undirected component reaches its sink. Indeed a directed cycle in a graph of outdegree at most one cannot be joined by a path to a sink: at the first connection the unique outgoing edge is already used toward the cycle. Equivalently, on an undirected cycle all its vertices must use their outgoing edge on that cycle, so no sink lies in its component. The same argument excludes two sinks in one component. Thus the component is a finite tree with the given unique sink.

Take the incidences of its edges, with an additional sentinel incidence at the root. Edge flip exchanges the two incidences of an edge and fixes the sentinel. At each vertex cyclically rotate its incidences in the fixed local order. Writing R for local rotation and F for edge flip, one step is $F \circ R$, and its inverse is $R^{-1} \circ F$ because $F^2 = I$. This order agrees with Cook and McKenzie's left-to-right permutation product convention. These mutually inverse operations give one contour cycle on the incidences. This follows by induction on the number of edges: inserting a leaf inserts its two incidences into the old cycle. The isolated-root case consists only of the sentinel.

At a canonical initiation mark its unique successor incidence; mark the sentinel at the root. Each initiation is therefore counted once. Iterating the contour successor to the next mark always terminates, since the finite contour contains the root mark. Iterating its inverse to the preceding mark is the inverse operation. A success tag and the sentinel token identify the root in this component; its address never has to be compared with an earlier address. Constant-degree adjacency and edge flip are provided by Lemma 12.4. $\square$

Lemma 12.6 (Capped incoming ports). *Fix a row class with a constant port alphabet and distinct entries in each row. Suppose its valid row ports reach their specified endpoints by a deterministic terminal program satisfying Lemma 12.4. For every fixed positive integer K_c , the subrelation consisting of entries whose endpoint has at most K_c incoming class entries admits restoring existence tests and mutually inverse paired motions. The inverse-port alphabet is $\{1, \dots, K_c\}$, and both directions use the same table tuple. These operations hold no additional endpoint address or path lacking a bounded inverse port across lower calls.*

Proof. Give invalid row ports a distinct failure execution. The canonical initiations in a success-root component then correspond exactly to valid class entries at that endpoint. Class validity is tested before using a forward initiation as a contour starting point.

Start at a valid initiation mark. Go forward toward the sentinel, counting other initiation marks, stopping either at the sentinel or after K_c other marks. Store the number of mark jumps, at most $K_c + 1$, and restore the start by that many inverse jumps. Do the same in the backward direction and restore again. Reaching K_c other marks certifies excess. Otherwise, let a and b be the numbers of other initiations before the sentinel in the forward and backward directions. The total is $1 + a + b$; accept exactly when this is at most K_c . The initiation's rank after the sentinel in forward order is $r = b + 1$. A forward paired motion goes to the sentinel in $a + 1$ mark jumps and returns the inverse port r .

From a canonical endpoint halt, count forward marks until returning to the sentinel or encountering $K_c + 1$ initiations. In the latter case restore the sentinel by the bounded number of inverse jumps and reject. In the former case the total $n \leq K_c$ is known. For an inverse port $r \in \{1, \dots, n\}$, go forward by r mark jumps to its initiation and return the row port stored in that initiation state. Invalid ranks are rejected at the sentinel. A halt with no preimages has $n = 0$ and is harmless. Wrong vertex types can be rejected before creating a halt state.

Every unsuccessful inverse request therefore leaves its endpoint unchanged: an excessive incoming count is undone before failure, and an invalid rank is detected at the recovered sentinel. For a successful request the returned row port is the one in the reached initiation, not a guessed port. This is the token used by a later forward restoration in Lemma 12.4.

The forward map sends initiation number r after the sentinel to that sentinel with token r ; the inverse follows exactly r marks from the sentinel to that initiation. These are inverse maps by the common contour order. Existence tests make the corresponding visits and reverse them, so they restore the cursor. All counts, ranks and incidence tokens have a fixed alphabet. The terminal program graph has the common table tuple as its root; neither direction fixes the starting row or port as an external parameter. Consequently the two directions use exactly the same graph. Restoring lower queries restore the catalyst, and the contour itself does not otherwise change it. $\square$

The incoming scores need not have bounded support in a column, so a bounded incoming-port list does not suffice to compute their rank. We instead give the terminal program a success state containing both the endpoint and a rounded score. A contour at that halt counts the entries with that rounded score; successive contours visit the scores in decreasing order. The endpoint and score belong to the current program state. During a tour they may be replaced by an earlier state of the score computation, and the contour recovers them when it returns to its unique halt. Thus neither is retained separately while an earlier state requests a score digit. The next proof combines this observation with an assembly order that stores only a short suffix before a late digit query.

Lemma 12.7 (Incoming score rank). *Fix a bounded nonnegative score table and a forward row class of constant size containing every positive score, with distinct entries in each row. Suppose its score-digit queries, made before multiplication by an incoming gate, are restoring and its terminal program satisfies Lemmas 12.4 and 12.5. At any endpoint, every fixed positive-integer incoming rank k , padded by zeros, has exact redundant digits computed by a restoring procedure.*

At approximation clock t , a place- j score query has $j \leq t + c$ and retains $O(1 + (t - j)_+)$ local bits in addition to its structural allowance costs. A full $O(t + 1)$ -bit score word is retained across structural calls only.

Proof. Normalize the score range to $[0, 1]$, using a fixed public factor if necessary. At approximation place t , choose a fixed extra margin c . For the digit truncation $v^{[t+c]}$ define the integer

$$q_t(v) = \text{clip}_{[0, 2^t]} \left(\left\lfloor 2^t v^{[t+c]} \right\rfloor \right). \quad (12.4)$$

The digit tail has absolute value at most a fixed constant times 2^{-t-c} . Choose c so that this is less than 2^{-t} . Clipping is nonexpansive and fixes $2^t v$, so $|q_t(v) - 2^t v| < 2$. The same margin gives $q_t(0) = 0$: a truncation of the zero value has scaled magnitude less than one, so flooring and nonnegative clipping give zero. Since the forward class contains every positive score, enumerating its entries and padding by zero therefore gives the rank of the whole quantized score column.

Use a forward terminal program which first tests the entire class and membership domain at its canonical initiation, sending invalid inputs to failure. Its success *configuration* contains the endpoint in the cursor and $q_t(v)$ in its local state. Before making its final structural motion, assemble the score digits from least to most significant. Put $K = t + c$. Just before requesting place j , the stored suffix is the integer

$$S_{>j} = \sum_{r=j+1}^K v_r 2^{K-r}, \quad |S_{>j}| \leq 6(2^{K-j} - 1).$$

After the query, add $v_j 2^{K-j}$ and proceed to $j - 1$. Thus its signed storage, including a carry and a self-delimiting offset from K , is $O(1 + K - j) = O(1 + (t - j)_+)$ with the fixed margin c absorbed in the constant. This is why the full score word is absent at a late-place query. After assembly, division by 2^c , flooring, clipping and loop setup are pure local computations. The complete word is then held only through structural motion and local clearing to success. Thus all its graph-query boundaries satisfy the claimed record bound. The success word is part of the graph state, not a parameter of the graph. At this fixed t the graph includes every allowed quantized value.

At the requested endpoint, initialize the success state at $q = 2^t$. Tour its component, counting canonical initiations and saturating the cumulative count at the requested fixed rank. Complete the tour even if the count has reached that rank. The sentinel recovers both the endpoint and q , since its success halt is the component's unique sink. Only there may the procedure read q as its answer or, if the count is still too small, decrement it by pure local steps and tour the next success component. At zero, apply the stipulated zero padding. The saturated count and control bits have constant size.

The logical root fixes t and the common table tuple; the loop value q belongs to the current program state. During an inverse step to an earlier partial-suffix state, that suffix replaces the score word. Neither the final q nor the endpoint is saved alongside it: the completed contour recovers them before the next decrement. All these success components belong to the same program universe.

This computes the integer order statistic N_t of the values $q_t(v)$. A uniform perturbation of a finite list changes any order statistic by at most the same perturbation, including after zero padding. Thus N_t has error less than 2 as an approximation to 2^t times the desired rank. Keep only the fixed number of residue bits needed to form its redundant digit, clear the other local data without a graph query, and return at the unchanged endpoint. Computing N_{t-1} uses its own quantization convention; the difference-digit construction needs only the bounded errors, not nested quantizations.

During a tour the live program state may be a partial suffix state preceding a high-place query. It then contains that short suffix, rather than a saved copy of the final score. Lemma 12.4 therefore also gives the required bound for inverse neighbor queries. All root components are finite, including isolated success states, and the decreasing finite value loop terminates. Fresh lower procedures and the contour preserve the catalyst. $\square$

12.5 Local storage of the estimator operations

The graph construction requires bounds on the syntax, storage, and dependencies of each actual recipe. These bounds hold even when the estimated entries are inaccurate.

Lemma 12.8 (Ordered terminal paths and excursions). *Every required terminal path without a bounded inverse port admits the following order: restoring selection and validity tests; local label maps and any paired left-factor move; restoring queries at that midpoint; at most one final terminal delegation; and local label completion. For any prescribed pure local endpoint action, this path admits a restoring excursion with parameter-reconstructible records and the same allowance and clock bounds up to fixed factors. Its terminal-path records are not retained over a further high-cost endpoint query.*

Proof. For a direct summand the path is a local base path, a projected child path, or a child row port. A projection stores its bounded symbol before removing it and appends the prescribed destination symbol after the child terminal motion. For a product the left factor is a correction with paired access. Its inverse token is stored at the midpoint; right-factor validity and weight queries are restoring there. The right terminal path is then the last delegation. An old prefix port delegates to that preceding prefix; after its return it requires no new graph query. These are exactly the path cases of Lemma 8.5.

There are two compiled continuations of this same terminal syntax. A one-way continuation clears its path records after performing its prescribed local completions. A visiting continuation retains the immediate choices and inverse tokens on its explicit terminal spine. After the local endpoint action it undoes the bounded completions, undoes the last delegated spine, and then uses the paired inverse token of an earlier motion. Each deeper record is cleared before invoking an earlier inverse map. For a projection, the saved symbol and prescribed append symbol give its local inverse. The identical lower paired tuple and actual returned token give the inverse of a correction move.

This is an induction on terminal delegations, not an attempt to invert an arbitrary terminal macro in one step. A frame stores only its own immediate descriptor, bounded choices and phase. Its returned child spine is used for local completion and the endpoint action, which make no graph calls. When a further graph query is made during rollback, the deeper spine has already been discarded. Thus the suspended records are those at the corresponding forward boundary, enlarged by bounded phase and inverse-token data. The choices can be re-created or kept on their own frame; no concatenated description of all old prefixes is passed to a new tight query.

In a restoring use, intrinsic constituent validity is checked before the visit. A product can test its right port by crossing the paired left edge and returning; this test does not use the right terminal path. Outer slice and wrap tests then use a valid intrinsic excursion. Hence none of these validity checks recursively invokes the very bucket or selected list membership it helps to define. The endpoint action's owner is marked in the outer frame, so its tuple can be regenerated by a scratch scan even while a descendant path is retained. The action's result does not alter the selected paths or their undo. $\square$

12.6 Catalytic key equality

The key is short, but retaining a $\log B$ -bit key or index at each tight recursive frame would still be too costly. The following exact identity gives a comparison with only a constant number of endpoint excursions. Its use in the space accounting uses the ordered excursions of Lemma 12.8 within the totality induction of Section 12.8. The cancellation of arbitrary initial contents also appears in transparent programs for catalytic computation [3, Section 3]. Here the shared bit vector is included in the ordinary work-space bound.

Lemma 12.9 (Restoring equality test). *Let $V \in \{0,1\}^B$ have arbitrary initial contents. Suppose two deterministically specified endpoint excursions restore the cursor and invoke only nested procedures that restore V . Their endpoint selection, validity answers, and forward and return paths are independent of the incoming V . At an endpoint, the prescribed local action may read or toggle V and update a local result bit; this action makes no recursive graph query, and its accumulated result does not affect selection, validation, or the forward and return paths. Then the key-equality bit $\mathbf{1}_{\{k(v)=k(w)\}}$ can be xored into that result bit while restoring V , using four endpoint excursions and only constant local data between them.*

Proof. Starting with result bit r_0 , perform the following four excursions: flip $V_{k(v)}$; xor $V_{k(w)}$ into the result; flip $V_{k(v)}$ again; xor $V_{k(w)}$ into the result again. The key is evaluated only at the endpoint currently visited. If V^0 is the incoming vector, the first read equals $V_{k(w)}^0 \oplus \mathbf{1}_{k(v)=k(w)}$ and the second equals $V_{k(w)}^0$. Thus the final result is $r_0 \oplus \mathbf{1}_{k(v)=k(w)}$ and both flips restore V^0 .

For nesting, use induction on the well-founded procedure order. A nested call sees some arbitrary incoming vector, restores precisely that vector, and returns a result independent of it. Hence the four outer endpoint paths and their keys are the same on repeated visits, even between the two outer flips. The terminal evaluation, read, or flip is a local operation with no further structural call. Only a phase and a result bit must survive across the excursions; a key index is recomputed at each visit in shared scratch. The result bit being accumulated is not used to choose either endpoint path. These facts give the induction invariant and the claimed local storage bound. $\square$

For $d > 2h_*$, exact bit-by-bit ID comparison is retained. A bit position then takes $O(\log B) = O(d)$ suspended bits, with constants allowed to depend on the already fixed ϵ . For small slack, Lemma 12.9 avoids that per-frame index. The shared vector itself occupies B bits and, by the lemma, may be reused by all nested comparisons. This algebra establishes restoration; the controller and traversal arguments supply the deterministic excursion and call-order interfaces needed to apply it.

The endpoint action evaluates the key with the invoking comparison's effective salt, even when the excursion visits a lower-budget table. It makes no further access query and does not choose or alter the excursion. Thus the lower path remains independent of later salts. Lemmas 12.10 and 12.12 implement this separation for all restoring accesses, including inverse queries.

12.7 Endpoint comparisons and local storage bounds

Lemma 12.10 (Implemented endpoint comparison). *At slack d , comparison of two valid candidate or old-list endpoints is a restoring, catalyst-independent test using $O(d)$ suspended structural bits in addition to its lower visit costs. It saves no endpoint address. The comparison is exact when $d > 2h_*$ and uses the designated block fingerprint otherwise; this choice is independent of the surrounding numerical clock.*

Proof. The two descriptors occupy $O(d)$ bits. By Lemma 12.8, each endpoint can be visited and restored for a local bit or hash action. If $d > 2h_*$, compare the $O(B)$ identifier bits in order, retaining only the bit-position counter and bounded flags. Its counter uses $O(\log B) = O(d)$ bits, since $h_* = \lfloor \epsilon \log_2 B \rfloor$ with a fixed positive ϵ and sufficient minimum padding.

Otherwise use the four visits in Lemma 12.9: flip the first key's bit, xor the second key's bit into a result, and repeat. The final result is key equality for every incoming catalyst and that vector is restored. Key evaluation and the indicated vector-bit action are pure endpoint operations in scratch. No key index is saved across another visit. Nested structural procedures already restore the catalyst and give results independent of its contents, by their lower operation order. The outer descriptors and all path choices ignore the accumulated result bit. The two repeated visits therefore use identical paths. This proves the same independence and restoration for the completed comparison, supplying the induction hypothesis at the next layer. The comparison mode is determined by M, d and the public block boundaries, not by the requested numerical place. $\square$

For the block comparison in Section 10, the comparison calculation replaces designated key comparisons by exact endpoint equality. Endpoint comparison is a restoring bit query represented by a single transition in the terminal program graph; its internal states are excluded. The exact comparison can therefore use a longer bit-position counter without enlarging the guarded raw graph states. It still returns its bit and restores the cursor and catalyst. Only the actual algorithm must satisfy the tight physical-space bound; the comparison calculation supplies comparison values and paths for the error analysis.

Lemma 12.11 (Local storage of the estimator operations). *The actual estimator operations have parameter-reconstructible records and a fixed finite dependency order of operation types. At prefix slack d , each structural layer uses $O(d)$ control bits. A numerical layer of clock t calling an operand at j satisfies*

$$L \leq c_{\text{str}}d + c_{\text{num}}(t - j), \quad j - t \leq c_{\text{dig}}d, \quad (12.5)$$

where L counts its suspended record and the coefficients are fixed. Calls outside the fixed operation types of that prefix go to the preceding prefix of slack $2d$ or to an estimator of budget at most $M - d$. Final row and incoming wrappers add only a fixed number of types at $d = 1$. All designated terminal executions satisfy these bounds before graph guards are imposed.

Proof. Table 2 summarizes the records to be charged. We verify the recipes and their dependencies in the order in which they are constructed. All constants depend only on the fixed construction constants and the base machine.

We first construct the immediate paths and their values.

By Lemma 8.5, a group has at most $2^{O(d)}$ immediate descriptors: term and channel part, scale and budget offsets, a variant word and wrap choices, bounded copy symbols, and a bounded number of factor ports. All factor budgets are at most $M - d$. A descriptor is stored literally in $O(d)$ bits; it never stores a discovered endpoint. Index and range tests are pure computations. Constituent existence tests call only these lower factors. Products cross a paired left edge for right-factor tests or digits, retaining a bounded inverse port; they return after each such query. Factor digits can therefore be requested separately by an arithmetic evaluation path. No assembled left weight remains over a late right-factor digit. Source vector factors are queried at the current source, with the prescribed local projection if needed. Their numerical coefficients and the factor products use a fixed arithmetic recipe.

For terminal motion the right path without a bounded inverse port, or the direct child path, is the last delegation. This is the ordered path of Lemma 12.8. To avoid a validity cycle distinguish

Operation	Data retained across a lower call	Local space bound
Structural tests and list selection	Immediate descriptors, loop indices, counts and flags	$O(d)$
Numerical recipes	Gate paths, pending bits and permitted early operand words	$c_{\text{str}}d + c_{\text{num}}(t - j)$
Ordered terminal paths	One frame’s choices, phase and bounded inverse tokens	$O(d)$ per immediate frame
Incoming score ranks	A partial score suffix; the full word only over structural calls	$O(1 + (t - j)_+)$ plus structural costs
Graph transition probes	A fixed number of local raw-state copies and bounded incidence data	$A(w - w') + D(t - j)$

Table 2: Suspended records at a lower call. The bounds exclude shared facilities and the callee’s physical state. Terminal-path frames are charged separately along their explicit chain; graph probes copy only that local raw chain.

intrinsic constituent existence, intrinsic motion or excursion, and outer validity. Outer validity first checks the intrinsic constituents, then uses an intrinsic excursion for its own hit and wrap conditions. A subsequent outer-valid motion may repeat that validity check and then use the intrinsic motion. Each of these is a separate stratum; none tests the updated bucket or selected list. Invalid descriptors return failure or a zero value without relying on a motion’s promised inverse.

All rate factors multiplying these terms have logarithmic range $O(d)$: a variable bin has $|\log_2(b/a)| = O(d)$, while f/π_f is bounded. Resetting a source rate to the public stage rate does not multiply the numerical term by f/u ; that ratio was a norm cost. It therefore causes neither a large numerical shift nor a new long choice field. The fixed number of bounded-factor products, coefficients and $O(d)$ scale shifts obey Theorem 11.3.

With these paths defined, we form the candidate domain and its bucket sums.

The descriptor pool is the concatenation of the preceding list positions and this group’s unmerged paths, with invalid slots masked. A new descriptor is a representative exactly when it is the first valid new descriptor with its key and no old valid position has that key. Old valid positions remain separate. To decide these flags, enumerate the $2^{O(d)}$ descriptors, keeping only the current indices, bounded flags and $O(d)$ -bit counts. Each test uses lower validity and endpoint comparison operations. The comparisons use only those unmerged paths or the preceding prefix’s already-defined positions. In particular they never inspect the new value of the bucket being defined.

An intermediate entry is the old entry at an old representative, or the previous baseline at a new representative, plus the sum of all valid matching increments in each channel. That sum has a fixed $O(d)$ -bit descriptor domain and boundedly many $2^{O(d)}$ scale factors. It is evaluated by the balanced sum construction of Theorem 11.3, with the preceding domain tests as structural leaves. No high-precision sequential accumulator is held while requesting term digits. The bucket rule preserves distinct actual representatives, including when two old representatives have the same key. Thus it gives exactly the intermediate array used in Lemma 10.7.

We next compress the intermediate arrays and select their exceptions.

The intermediate domain and its values are now earlier inputs. Add enough copies of the default for the size and first-signal ranks. A finite domain of $2^{O(d)}$ indices suffices: the preceding list has size $2^{O(2d)}$ and the new group has $2^{O(d)}$ descriptors. The default is included as often as needed to represent the unlimited padding in the prescribed ranks. The rank, max, min, absolute-value

and clipping operations of Lemma 9.1 are the finite arithmetic primitives. The ranges are $2^{O(d)}$ by Lemma 9.3; their known normalizing shifts have magnitude $O(d)$.

Here the current group has slack d , floor $2^{-c_1 d}$ and capacity $2^{c_2 d}$, so $u_0 \geq 2^{-c_1 d}$. A canonical finite approximation can test a threshold strictly between u_0 and $2u_0$: with error a sufficiently small fixed fraction of $2^{-c_1 d}$, accept every size greater than $2u_0$ and accept only sizes greater than u_0 . The required numerical clock, after normalizing the range, is $O(d)$. Fix this clock and tie convention once for that structural test. Consequently every nondefault compressed position is included, and fewer than $2^{c_2 d}$ positions are selected, as in Lemma 9.1. Numerical equalities at the boundary need not be decided. No default can expose a positive row score: its rank dummies close the gate first, by the margins of Lemma 6.4.

Remap a requested list ordinal to its selected intermediate descriptor by enumerating the pool and counting the preceding accepted flags. The loop stores $O(d)$ -bit indices and a count. Its only calls are the already-defined intermediate flags and finite-margin selection tests. Compressed entry digits then query that descriptor's intermediate value and the common compression thresholds; baseline digits use those thresholds directly. A terminal list motion selects its descriptor before delegating to an old prefix path or a new valid path. A descriptor from many prefixes ago is therefore represented by a chain of delegations, not retained as one long word above a tight child.

Every value or motion query must first establish that its port is present.

Ordinal lookup returns a presence bit and a selected descriptor only when its finite enumeration reaches the requested ordinal. It starts with a false presence bit and returns false if enumeration ends first; the false branch supplies no descriptor. Every compressed-entry digit query and list motion branches on this fresh result before using a descriptor. The preceding-prefix lookup has the same convention. A hole denotes no real representative. Rank operands retain their prescribed structural selection of an entry, baseline, or padding value; they do not infer that value from the zero stream exported at an absent real port.

The final precursor A and auxiliary class A_I use the bounded final list-position alphabet, allowing holes. Their presence tests use list presence, role-membership masks, and the fixed-precision raw-entry and raw-rank tests of Lemma 6.4. Source membership is checked locally; endpoint membership uses the earlier list or intrinsic excursion with a pure membership action. It does not use the row-class motion whose validity is being defined. These tests precede evaluation of the score and use neither its incoming gate nor its cap. Every pre-score digit invocation repeats the corresponding class test and returns zero on failure. Other uncapped row-value queries use their support-covering class in the same way. Each such class contains the entire nonzero support, so the extension agrees with the mathematical table. Existence predicates evaluate their earlier masks directly; they do not ask for their own existence. Analytical witness functions without an exported port interface retain their stated full-array values and are not restricted by this convention.

Immediate-path values and visits first use their earlier constituent and, where appropriate, outer-validity tests. Endpoint comparisons likewise test their two earlier descriptors before making visits. A capped correction-value query stops with zero if its forward paired move fails; only a successful move permits a gate or column-value query at the endpoint. The paired inverse keeps its target-side validator from Lemma 12.6. Incoming ranks and vector queries return zero outside their prescribed vertex type or membership domain. These checks are repeated at canonical entry, including when the caller is a synthetic post-initiation state with a fabricated validity flag. An excursion checks before constructing its spine; its endpoint action and rollback retain the order of Lemma 12.8.

Place these entry wrappers a fixed number of strata above their validity predicates and value or motion cores. A wrapper at clock t calls its structural test at clock zero and retains $O(d)$ descriptor and control bits; its boundary is therefore bounded by $O(d) + bt$. When it subsequently calls a

numerical core at clock j , the test has finished and only bounded status and descriptor data remain, so the bound $ad + b(t - j)$ persists after enlarging a . The predicates use the fixed structural precisions already proved. They do not call the score, gate, cap, new bucket or new selected list whose value is being checked. The finite stratum list is enlarged before choosing the allowance constants. Recipe guards permit subroutine calls only to total entry wrappers, so a synthetic state cannot select a continuation that skips validation. No semantic predicate is evaluated by a guard itself.

These wrappers preserve the represented relations, values and outgoing endpoints. Their insertion may change the contour ordering of incoming ports; both directions use the same updated graph, so pairing remains consistent. The tests use the same effective tuples and fixed structural clocks, and restore the catalyst on both outcomes. They require no accuracy promise and add no retained endpoint address.

The dependencies just described have a finite order within each group.

For clarity, a permissible ordering, from an exported operation down to its dependencies, is:

1. compressed entries, baseline and list-port operations;
2. ordinal remapping, compression and exception selection;
3. size and first-signal ranks of intermediate entries;
4. intermediate domain flags and entry or bucket-sum values;
5. first-representative and key-match predicates;
6. endpoint comparisons of outer-valid paths;
7. outer-valid visits, then outer path validity;
8. intrinsic visits, constituent tests, factor digits and ordered motions, subdivided according to the preceding distinction.

The wording of an interface name does not force a call at the same stratum. For example, the “visit” used to verify outer validity is the intrinsic visit, while the visit used by key comparison is an outer-valid visit. Arrange these finite subtypes in their indicated dependency order: outer-valid visits use outer validity; outer validity uses intrinsic visits; key comparisons use outer-valid visits; and bucket predicates use those comparisons. Arithmetic on intermediate values uses only their domain flags and lower matching predicates. This refines the displayed list to a finite acyclic list. Repeated enumeration over descriptors stays in a loop of its layer, rather than creating one recursive frame per descriptor. Between these layers Theorem 11.3 gives (12.5); structural loops add $O(d)$ bits. All remaining path calls go to the preceding prefix or to the lower-budget factors already described.

At the final group, we construct the row classes and values before incoming multiplication.

At the final slack $d = 1$, the list size, port alphabet and all threshold gaps are constants. The classes and witnesses in Lemma 6.5 are defined by canonical finite-margin entry and rank tests on the final raw arrays. They can be enumerated without testing positivity of a computed score. Their tests and the digits of scores before incoming multiplication query only the final buffer and a fixed arithmetic recipe. The same raw-array tuple is used for every row. A normalized bin entry uses its second raw argument, divided by the fixed scaling constant and clipped before the reciprocal. It does not start a new preparation experiment at its endpoint reached without a bounded inverse port.

The broader auxiliary class used to score an incoming gate depends only on the corresponding row calculation before incoming multiplication. Its validity and score queries do not depend on that incoming gate or cap. The precursor A for a correction has the same property. Its canonical terminal program begins with a restoring check of the full precursor class and membership domain, and sends every invalid initiation to failure. Thus the initiation test makes Lemma 12.6 count

exactly the valid entries, independently of any earlier caller flag. The auxiliary scoring program has the same initial check. Its score digits are computed before incoming multiplication, so Lemma 12.7 invokes only lower forward operations. By Lemma 6.5, discarding a precursor with more than K_c incoming entries preserves its value.

The final operations add incoming access and vector queries.

Place the capped-pair and inverse-score wrappers above their respective terminal programs, and those programs above their pre-class, selection and pre-score queries. They are separate terminal programs: the gate score program does not call this task's cap pairing, and the cap program does not call its gate rank. A final correction-value query first completes the paired move. At the endpoint its motion-return data consist only of the bounded inverse port while it queries a gate-rank digit or a lower-budget column-vector digit, and then returns. This can be done separately for every arithmetic operand bit. Only numerical path bits and immediate descriptors coexist with a late query. The factor r_i at a projected source is handled in the same way at that source.

A detector trial is an inverse-score wrapper of its lower-budget input row task. Reward terms query source factors locally and cross paired corrections before midpoint vector queries. Vector prefix sums use the simpler balanced arithmetic form of the same dyadic schedule. These are all the vector and incoming uses in the additive formulas. There are only a fixed number of final wrapper types. Their short counts and return ports cost a fixed allowance gap. In an inverse score program the saved numerical suffix satisfies the clock bound in Lemma 12.7; its full word is held only over structural motion. The cap wrapper's counts have fixed size.

We finish by checking uniform generation and the retained record lengths.

Each enumerated object above has a literal $O(d)$ -bit index or is a Boolean path in one of the stated arithmetic recipes. Uniform index-generation uses finite integer loops on these indices, including rank comparison pairs and balanced-tree nodes. Constants, field/rate coefficients and walk probabilities have $O(B)$ -bit rational descriptions. Their requested digits use integer arithmetic in scratch and make no recursive graph call. In particular, a walk word has length $O(d)$ and a fixed rational generator probability, so its probability denominator has $O(d)$ bits. Rate and field expressions involve fixed powers of $O(B)$ -bit integers. The arithmetic theorem's uniformity and public-coefficient hypotheses are thus satisfied.

The finite arithmetic list covers the formulas used here explicitly. A fixed polynomial switch is evaluated on an affine-rescaled clipped argument; positive parts and absolute values use maxima, and a variable clip is a minimum followed by a maximum. The normalizing reciprocals have their arguments clipped to $[1/2, 1]$. The eighth roots used in the error estimates express regularity of the row functions and are not additional numerical operations requested by the controller. Ranks inside a prefix use the finite descriptor construction above; the unbounded incoming classes use Lemma 12.7. This accounts for the arithmetic primitives without adding a real-number comparison or an incoming-rank oracle.

Structural selection words and loop counters have $O(d)$ bits. The only longer local numerical words are the early operand records of Theorem 11.3 and the just-described score suffix; they obey their explicit clock-decrement bounds. Late operand calls hold only numerical paths, pending bounded results, immediate descriptors and short inverse tokens. A fixed number of arithmetic and structural layers enlarges the constants in (12.5); its clock differences add. Ordered terminal delegations add their own immediate records, with the same estimate at each boundary, including on rollback as proved in Lemma 12.8. This verifies the record bound on the designated executions, independently of any graph guards. $\square$

Lemma 12.12 (Parameter replay and salt visibility). *The records constructed in Lemma 12.11 determine every descendant tuple and clock by the scratch replay of Definition 12.2. As functions of their own effective environments, actual procedures in budget block b use no salt block later than b . This restriction holds for values, structural orders, graph universes, guards, and predecessor probes. Every re-query of an entry, and both directions of a paired motion, use that entry's same effective table tuple.*

Proof. Start with the fixed top parameters and scan the active records. An estimator edge specifies a task type, a stage change prescribed by the formula, a budget offset, a bin-scale offset if present, a bounded copy choice and channel-role recipe. These determine the next absolute stage, rates and suffixes. A projection appends the stored symbol before the inherited suffix. A fresh midpoint or vector role uses the specified public stage rate and its prescribed suffix; it does not carry the old source. These are the common-tuple recipes in Lemma 6.5 and Lemma 8.5.

An averaging edge also carries its generator and wrap word. Apply these fixed transformations in order to the current environment. Their descriptors are independent of the indicated endpoint; membership and wrap consistency are tested separately on visits to that vertex. Thus no discarded conditioning address is needed to regenerate the environment. For an old-prefix edge the tuple stays that of the same row calculation. At final assembly a gate or column multiplier uses this common tuple, rather than the internal variant which happened to lead to a selected representative. An action-owner tag allows an endpoint-local calculation to replay just to that ancestor tuple.

Numerical paths specify arithmetic operands, gates, digit places and the early-prefix modes of a reciprocal. Their finite index rules are those of Theorem 11.3; an absolute place is recovered from the root clock and the path or offset, not saved afresh per frame. Descriptor selections which used the cursor were stored explicitly as their bounded choices before movement. No later tuple computation needs to repeat such a selection at an unknown old cursor.

At every step the tuple and the temporary integers have $O(B)$ bits in the actual parameter and clock ranges. Keep a current tuple in scratch and overwrite it as each record is processed. Repeating the scan reconstructs an ancestor or a selected candidate chain. Delimiters and marked active-copy tags specify which chain is used. This costs one scratch facility, including its scanning counters, rather than one absolute pointer, input-head position or tuple per frame.

A pure endpoint action belongs to its outer invoking predicate and may read that owner's permitted salt. It is excluded from the lower program graph and cannot affect the lower path or rollback. Nested lower queries still restore their arbitrary incoming catalyst contents.

Finally, the only salt read by a new group comparison is its task's own block salt. Descendants do not increase their budget. Walks alter only the already designated earlier free salt blocks and leave the present and later ones frozen. All descriptor and instruction orders are fixed index orders; all range and length checks depend only on public bounds and accessible tuple fields. Give each recipe the syntactic permission to address only those salt blocks at most its budget block in its effective environment. This is a direct range check on a requested salt index, not a condition on the salt's value. A synthetic record cannot change that permission, choose a higher-budget recipe, or bypass the fixed parent-to-child rules. The same replay and checks are used before predecessor queries. Consequently the entire lower-block structure is independent of later salts. The comparison calculation in Lemma 10.2 uses the same fixed descriptor orders and structural clocks, with its stipulated replacements of comparison predicates. The error proof uses the salt independence of the actual lower algorithms and of the comparison recipes. $\square$

Remark 12.13 (Effective coordinates and physical replay). The salt assertion in Lemma 12.12 is a statement about a task as a function of its effective environment. For fixed other task parameters, write this function schematically as $F_b(\eta)$. Its values, access orders and guarded program graph use only the sampling channels and the permitted salt coordinates of η . An ancestor can invoke it at $\eta = \varphi(\sigma)$, where σ is the master environment and φ is a recorded environment transformation. Computing the permitted coordinates of $\varphi(\sigma)$ can read physical master bits whose original block indices exceed b . Those reads derive the effective argument; they do not give F_b a new semantic parameter or permission to inspect later coordinates of that argument.

For a comparison calculation in the current block, the current task's own walks vary the earlier effective salt coordinates and freeze its current and later ones. Exact comparisons within the current block omit the current key, and lower-block procedures obey the local permission rule just proved. This is the coordinate convention used for the slice means in Lemma 10.2. A callback owned by a higher task remains the separate pure endpoint action described above; it cannot enter the lower graph, change its selected path or affect rollback through catalytic contents.

12.8 The uniform space theorem

Proof of Theorem 12.1. Two linear quantities organize the final accounting. For positive physical-space coefficients A, D , chosen below, define

$$\Phi(w, t) = Aw + Dt, \quad \Psi(w, t) = t + a'w, \quad (12.6)$$

where a' is the clock-overshoot coefficient in Definition 12.3. The decrease of Φ will pay for a suspended physical segment, including its local probe copies. The nonincrease of Ψ will bound every descendant numerical clock and hence the shared arithmetic scratch.

Let S exceed the finite number of operation strata in Lemma 12.11, including the subdivision of path validation and all final wrappers. Choose fixed integers $K_0 > S + 1$ and $K_1 > 2(K_0 + S + 1)$, increasing them as needed for the local constants. Assign to an operation at budget M , prefix slack d and stratum $s_0 \in \{0, \dots, S\}$ the allowance

$$w = K_1(1 + M) - (K_0 + s_0)d. \quad (12.7)$$

Final wrappers use $d = 1$, with increasing s_0 down their dependencies. For a nontrivial prefix, $1 \leq d \leq M$; trivial tasks use fixed top strata and terminate locally.

A same-slack descent lowers w by at least d . A preceding-prefix descent to $2d$, even with a reset to an earlier stratum, lowers it by at least $(K_0 - S)d > d$. A statistical descent to $m \leq M - d$ lowers it by at least $(K_1 - K_0 - S)d$, since the child allowance is at most $K_1(1 + m)$ minus a nonnegative stratum term. Hence every required boundary has $w - w' \geq cd$ for a fixed $c > 0$, and all allowances remain positive and $O(1 + M)$. The estimates (12.5) and the ordered-path record bounds give Definition 12.3, with common coefficients. When a common clock coefficient is increased, compensate the allowance coefficient using the bounded clock overshoot, just as in (12.3). Add a fixed allowance gap for a traversal wrapper's bounded incidence, count and status data.

There are two distinct estimates in this induction. First, Lemma 12.11 bounds the raw caller words with the same a_0, b_0 at every budget; a lower subroutine operation contributes its call recipe, but none of its internal working states, to that raw record bound. Second, Lemma 12.4 converts a copied raw portion to its physical bound with one fixed pair A, D . The induction below appends a lower operation's physical portion; it never substitutes that physical portion into an upper raw graph state. The raw record bound therefore does not depend recursively on the already enlarged physical constants.

Construct the procedures in increasing allowance. The induction covers all recipe instances admitted by the finite range and parent-to-child checks: both designated calls and calls made when probing candidate predecessor records. It does not require such a candidate record to have occurred on a designated execution. Each subroutine call still enters the lower procedure at its canonical entry, where its semantic presence tests are performed. At the bottom are local base ports, rank and salt evaluation, rational arithmetic without graph operands, and trivial zero tasks. Suppose lower operations are total, have their claimed cursor and catalyst behavior, and obey the local bounds. The finite restoring and numerical recipes of Lemma 12.11, using Theorem 11.3, then compute exactly their specified bit or digit. Their finite loops and lower calls terminate. Endpoint visits use Lemma 12.8, and completed endpoint comparisons use Lemma 12.10. The latter restores arbitrary catalyst contents and has an independent answer, preserving the induction hypothesis. The other recipes make no direct catalyst-dependent choice.

The canonical terminal programs have finite loops and only lower terminal delegations or subroutine calls, so their prescribed executions terminate. Their class checks send invalid canonical initiations to failure. The bounds already proved show that all their designated boundaries fit the local bounds. By Lemma 12.12, their parameters and static guards are uniform and independent of a discarded cursor. We may therefore apply Lemma 12.4. Guards remove no designated execution. At synthetic states they ensure that every probed query has an admissible lower tuple and available space; fresh safe lower queries do not rely on the truth of an earlier stored flag.

The finite graph contours of Lemma 12.5 now implement the higher cap and inverse-rank wrappers in Lemmas 12.6 and 12.7. They terminate on success-root components even if other graph components contain cycles or guard-created dead ends. They count only the canonical initial executions of these programs, so their paired maps and ranks are the mathematical ones specified by the tables. By the common root tuple and salt convention the same maps are obtained in either direction and on every re-query. This completes the well-founded construction. A subroutine query in this induction denotes its already constructed finite procedure; recursively substituting these procedures gives an ordinary deterministic machine.

The allowance is a runtime parameter of this fixed finite collection of instruction types, rather than an index for input-dependent program text. Operation tags select the finite recipe family; loop counters, gate paths and reconstructible descriptors select its current instance. Thus the induction constructs one uniform interpreter for these instructions. It requires neither a separately compiled machine for each budget nor stored transition graphs for the contours.

It remains to add all simultaneously live data. By Lemma 12.4, a traversal root at (w_i, t_i) with a pending query at (w_{i+1}, t_{i+1}) contributes at most

$$A(w_i - w_{i+1}) + D(t_i - t_{i+1}) = \Phi(w_i, t_i) - \Phi(w_{i+1}, t_{i+1}) \quad (12.8)$$

local bits, including its fixed number of neighbor-state copies. Ordinary restoring and numerical calls have the same bound after one enlargement of A, D . Inverse traversal copies only its local explicit chain, not older stack segments or the internal configurations of a transition query. Choose A, D also to dominate the bottom operation's intrinsic linear bound. If the active chain has boundaries $0, \dots, r$, with boundary zero at the top, put $\Phi_i = \Phi(w_i, t_i)$. The total local space is at most

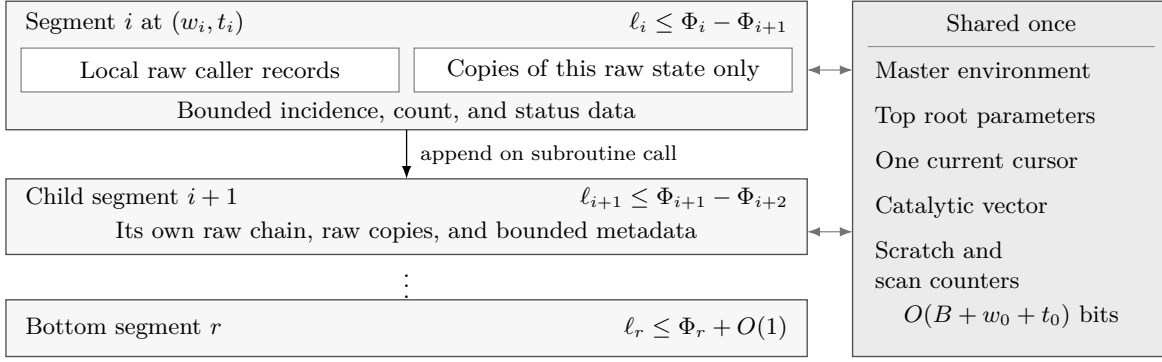
$$\sum_{i=0}^{r-1} (\Phi_i - \Phi_{i+1}) + \Phi_r + O(1) = \Phi_0 + O(1) = Aw_{\text{top}} + Dt_{\text{top}} + O(1).$$

Figure 1 shows which records enter each summand.

The clock inequality in (12.1) gives

$$\Psi(w_{i+1}, t_{i+1}) \leq \Psi(w_i, t_i). \quad (12.9)$$

Physical stack segments (earlier segments omitted)



$$\sum_{i=0}^r \ell_i \leq \sum_{i=0}^{r-1} (\Phi_i - \Phi_{i+1}) + \Phi_r + O(1) = \Phi_0 + O(1)$$

Figure 1: Physical stack space, with ℓ_i the length of segment i and $\Phi_i = \Phi(w_i, t_i)$. Child-subroutine work is appended outside raw copies. The record and local graph bounds, with enlarged A, D , include copies and metadata. Shared facilities are counted separately. The displayed copies occur at traversal calls; older physical segments are not copied into the child. Ordinary restoring and numerical segments obey the same difference bound.

Since $t_i \leq \Psi(w_i, t_i)$, all clocks are $O(t_{\text{top}} + w_{\text{top}})$. This also bounds the public scratch calculations in the arithmetic theorem. An uncharged full coefficient or cursor copy cannot persist there: any operand-dependent early word crossing a query was included in its record bound, while pure public computations have no recursive query. The master environment, cursor, catalyst, scratch and their global scan counters are single facilities. Their total space is $O(B + t_{\text{top}} + w_{\text{top}})$.

For exact-stream semantics the induction is valid for each finite requested clock, allowing scratch to scale with that clock. Every structural domain uses its fixed finite-margin clock, so its definition does not change in this argument. In particular the integer N_t of a score rank or arithmetic operation is always evaluated by its own place convention, even when used in a request for the next digit. The actual application has $M_{\text{top}}, t_{\text{top}} = O(B)$, and hence all these facilities and active records use $O(B)$ bits. $\square$

13 Finite parameter choices and certified accepting computations

The constants and finite ranges required in Section 3 admit the common order below. Fixed-precision approximation also yields accepting runs when the acceptance probability has a specified inverse-polynomial lower bound.

13.1 Choosing constants and finite ranges

The construction has the following order of choices.

1. Fix $H, h_0, s = 8$ and the fixed number of independent channels. The finite-action gap is uniform in every later modulus and bit length.
2. Choose the correction thresholds and cutoff, then the iteration count k_* from the factorial bound and its resulting uniform magnitude constants. These estimates hold for every column multiplier in $[0, 1]$.

3. Choose D_0 for the density reduction, the routing thresholds, and the finite ladders of entry and gate thresholds. The row-gate rank K can then be chosen from the row-mass bounds alone.
4. Choose the detector rank J large enough for the incidence bound on extra copies. Choose R_* large enough for overload detection, and finally the incoming rank K_c large enough for correction normalization. The earlier row-gate choice does not depend on K_c .
5. Fix the bounded-output constant, the scale cutoff and precision coefficient, and the prefix mean exponent. Choose the buffer floor and capacity exponents and its noise coefficient so that all buffer sums converge. Choose task separation and walk length coefficients sufficiently large for the small constants in the comparison estimation induction.
6. Fix the crude range and propagation constants, then the block gain coefficients D_2, D_1 and a sufficiently small rational block width coefficient ϵ . These choices close the coupled fingerprint induction of Theorem 10.4.
7. Choose the fixed operation-stratum and allowance coefficients and the guard constants in Theorem 12.1. Their finite enlargement does not alter the analytical tables or the distribution of the environment.
8. Choose the linear top-budget coefficient, the field-size exponent, and the minimum padding of B .

Each step requires only finitely many strict inequalities between fixed constants. In particular, the row rank is fixed before the incoming rank, the block constants follow the crude error constants, and the field is chosen after the reachable resource ranges. There is no dependence of an earlier sampler gap on that last choice.

The coefficient ϵ in this order is the fixed fingerprint block-width coefficient. Once it has been chosen, increasing the fixed linear top budget increases the number of salt blocks but leaves the sampler's gap uniform: all earlier salt bits form one free component in Lemma 7.1. The storage estimate (10.4) therefore remains $O(B)$.

For clarity, one may bound the stage index by L , the inherited suffix length by the stages remaining above a task, and logarithms of reciprocal requested rates by

$$HL + c(M_{\text{top}} - M) \tag{13.1}$$

with a sufficiently large fixed c . A child at slack d can extend the variable scale range by at most Cd for a fixed C and has budget $m \leq M - d$. If a parent's reciprocal-rate logarithm is at most $HL + c(M_{\text{top}} - M)$, choose $c \geq C$ to obtain

$$HL + c(M_{\text{top}} - M) + Cd \leq HL + c(M_{\text{top}} - m).$$

Stage-rate resets are at most HL , and taking the smaller of two rates takes the maximum of their reciprocal logarithms. Thus both resets and usage-rate minima preserve the same invariant. The finitely many local task operations use the stated widths and fixed subranks. Thus Equation (13.1) is closed under all estimator recipes. The controller guards allow these designated executions.

There is consequently a fixed C_P such that a prime

$$2^{C_P B} \leq P \leq 2^{C_P B+1}$$

is larger than all identifiers, makes the radix invertible, and gives $P2^{-k} \geq 1$ for every rate ever requested. Bertrand's postulate provides such a prime; testing consecutive integers by trial division finds one using $O(B)$ space. The search range and running time are $2^{O(B)}$. Field arithmetic and fixed-dimensional rank tests also use $O(B)$ space.

13.2 Certified accepting computations

We return to fixed inverse-polynomial accuracy to construct an accepting computation when the machine's acceptance probability has an inverse-polynomial lower bound. For a transducer, the conclusion concerns outputs certified by an accepting run: its output tape must be write-only, so past emitted symbols do not influence later transitions.

Corollary 13.1 (Certified accepting computations). *Fix a randomized logarithmic-space machine $\mathcal{M}$ with a fixed polynomial bound $T(n) \geq 1$ on every run, and fix an integer $c \geq 0$. A uniform deterministic logarithmic-space transducer, running in polynomial time, outputs an accepting computation of $\mathcal{M}$ on every input x for which*

$$p_{\mathcal{M}}(x) \geq (n + 2)^{-c}.$$

On other inputs it outputs either an accepting computation or a failure symbol. If $\mathcal{M}$ has a one-way write-only output tape and every accepting run emits a valid witness for a specified relation, the deterministic transducer may output that witness with the same guarantee.

Proof. Give halted configurations constant self-transitions through time $T(n)$. A configuration and its elapsed time use $O(\log(n + 2))$ bits. There is a fixed randomized logarithmic-space continuation machine which, on a valid encoding of (x, v, t) , starts from configuration v and simulates the remaining $T(n) - t$ steps with fresh coins. It checks the fixed work-space box and time bound, rejecting malformed encodings and any attempted violation. Its acceptance probability on the reachable configurations is the continuation value $p_t(v)$, and it has a polynomial worst-case time bound in its input length.

Write $\gamma = (n + 2)^{-c}$. Choose a fixed exponent d large enough that Theorem 3.1, applied to the continuation machine, approximates every queried value within $\delta \leq \gamma/(8T(n))$. Such a fixed exponent exists because T is a fixed polynomial and the continuation input contains x . The extra fields have logarithmic length. We supply them virtually from saved configuration and time registers, so the approximation and those registers together use logarithmic space.

Starting at the initial configuration, at each time estimate the continuation value of every positive-probability successor and choose one with maximum estimated value, breaking ties by a fixed port order. Since $p_t(v)$ is the weighted mean of the successor values, the chosen successor v' satisfies

$$p_{t+1}(v') \geq \max_w p_{t+1}(w) - 2\delta \geq p_t(v) - 2\delta.$$

On an input satisfying the stated lower bound, after $T(n)$ choices the terminal continuation value is at least $\gamma - 2T(n)\delta \geq 3\gamma/4 > 0$. A terminal continuation value is zero or one, so this computation accepts.

First run this construction without writing output. If its terminal state rejects, output the failure symbol. Otherwise repeat the same deterministic choices and stream the configurations through the original first halting state, or the symbols emitted by the simulated transducer. The artificial self-transitions after halting are omitted from the output. This second pass follows the same accepting computation. Consequently no rejecting computation is output, even outside the probability promise.

At any time the current, candidate, and best successor configurations, the time counter, a constant number of dyadic scores, the virtual input address, and the approximation workspace occupy a constant number of $O(\log(n + 2))$ -bit records. The finite successor lists have constant size, and there are only $T(n)$ steps in each pass, so the total running time is polynomial. Each approximation has a deterministic guarantee at its queried configuration; no simultaneous accuracy event for all adaptive queries is needed. $\square$

14 The effective compiler

The probability-approximation theorem fixes the source machine. We now construct one effective map from source descriptions and supplied resource bounds to deterministic simulations with explicit bounds.

Theorem 14.1 (Effective simulation with explicit bounds). *A terminating compiler takes a finite randomized machine description M and positive integers a, b and outputs a total deterministic machine D_M and positive integers K, H, c . On every input of length n , this machine uses at most $K \log(n+2)$ work bits and $H(n+2)^c$ elementary bit operations.*

If M halts within $(n+2)^a$ steps and uses at most $b \lceil \log(n+2) \rceil$ work bits on every length- n input and every random tape, and its acceptance probability is at most $1/3$ on no inputs and at least $2/3$ on yes inputs, then D_M decides the same language.

The compiler's own running time need only be finite, and its output exponent may depend on the source.

The compiler specializes one fixed deterministic program. A fixed randomized interpreter U enforces polynomial-time and logarithmic-space bounds on every input. Theorem 3.1 supplies a deterministic machine A that separates the inputs on which U accepts with probability at least $2/3$ from those on which it accepts with probability at most $1/3$. Both U and A are fixed before the compiler receives a source description M and bounds a, b .

When the source satisfies its supplied resource bounds, padding places each encoded computation within U 's caps without changing its acceptance probability. The generated machine runs A on that padded input, generating each requested symbol from the real input. We then compute the output bounds K, H, c from the generated finite transition table and the fixed library data. The compiler never has to select a new probability approximator or infer resource bounds from the source's behavior.

Pyne, Raz and Zhan [10, full version, Theorem 5.1] construct one deterministic algorithm that, on input $(1^n, \mathcal{B})$ with $n \geq 1$ and $\mathcal{B}$ an ordered branching program of length and width at most n , approximates its acceptance probability to error less than $1/n$. For every space-constructible $S(n) \geq \log n$, this single algorithm uses $O(S(n))$ space exactly when $\text{prBPL} \subseteq \text{SPACE}[O(S(n))]$. Our compiler uses one fixed separator library and computes explicit resource bounds for each source-specific specialization.

All quantities in this section are local to the compiler construction. In particular, P is a padding exponent, d_{code} is a program length, $B_n = \lceil \log_2(n+2) \rceil$, and N_{pad} is a virtual input length. They do not denote the statistical budgets, field modulus, or domain sizes used in the preceding sections. We use the usual counted-cell convention for work tapes: traversed writable cells count toward space, whether or not they contain a nonblank symbol. The finite control and the read-only input have their standard treatment.

14.1 Deduction from the compiler components

The total randomized interpreter U of Lemma 14.2 has a total deterministic separator A for acceptance probabilities at most $1/3$ and at least $2/3$, with the work-head bound (14.4) (Section 14.3). For source code z of length d_{code} and the explicitly computed exponent $P = 1000(d_{\text{code}}+2)^4(a+b+2)$, put $N_{\text{pad}} = 2^{PB_n}$. The virtual string y_x of that length has prefix $1^{d_{\text{code}}}0z1^n0E_{d_{\text{code}}}(x)$ and zero suffix, where $E_{d_{\text{code}}}$ uses fixed-width symbol codes. Under the source promises, $U(y_x)$ has exactly the acceptance probability of $M(x)$ (Section 14.4). The total finite machine $D(x) = A(y_x)$ generates each requested input bit (Section 14.5).

The positive integers K, H, c computed from D 's finite syntax in Section 14.6 give, simultaneously on every input and without the source promises,

$$\text{space}(D, x) \leq K \log_2(n + 2), \quad \text{time}_{\text{bit}}(D, x) \leq H(n + 2)^c.$$

The exact integer formulas are (14.11).

Proof of Theorem 14.1. The compiler returns the described machine D and the integers K, H, c in Equation (14.11). Its finite syntactic construction terminates. The totality of A and the finite callback loops prove that D halts on every input. Equations (14.12) and (14.14) give at most $K \log_2(n + 2)$ work bits and $H(n + 2)^c$ bit operations for this same machine, independently of the source promises. When the source satisfies those promises, Equation (14.9) and the fixed separator's threshold guarantee give the correct decision.

All claims include $n = 0$: then $B_n = 1$, the real-input scans are empty, the count $n + 1 = 1$ has one bit, and every displayed padding, strip, space, and time inequality remains valid. The library's grouping width already includes its finitely many small-length exceptions. All construction constants are fixed before the ordinary input is read.

The work and time internal to probability approximation—environment enumeration, numerical digits, recursive interfaces, recomputation, and precision management—belong to the single library machine supplied by Theorem 3.1. Its tapes remain counted throughout virtualization. The compiler adds only the exact input operations analyzed here, without an evaluation oracle or an external stack. Configuration counting gives polynomial time because this entire deterministic machine uses exact logarithmic space. $\square$

14.2 One fixed interpreter with enforced resource bounds

Fix a binary coding of ordinary finite transition lists. One concrete choice uses a fixed record grammar and the field encoding $w \mapsto 1^{|w|}0w$. Lists specify the finite state set, tape and alphabet declarations, initial and halting states, blanks and endmarkers, and transition records. A transition record lists its tested state, scanned symbols and coin pattern, followed by its next state, written symbols and local head movements. The grammar has fixed nesting depth; its fields are actual listed data, not recursively executable descriptions. State and symbol IDs are canonical binary integers used only as names. The first matching transition is used; a missing transition causes rejection. Wildcards for ignored coins can equivalently be expanded into a finite explicit list.

Write z for a program in this coding and $d_{\text{code}} = |z| \geq 1$. The coding permits at most d_{code} work tapes and a block of at most d_{code} fresh fair bits per simulated transition. Every ID has at most d_{code} bits. A harmless padding field can enlarge the program to meet these inequalities. All tape movements are local. Work coordinates are measured from their initial origins; the ordinary model has one input head and one head per work tape. For a source input x of length n , let $E_{d_{\text{code}}}(x)$ encode each of its symbols by its ID padded to exactly d_{code} bits.

The encoding preserves the supplied source bounds as follows.

An ordinary source machine M can be converted effectively to this format by renaming its states and symbols and listing its transitions. Its tapes are represented directly, so a source transition retains its work writes and head movements and counts as one interpreted transition. The conversion does not pass through a slower single-tape simulation.

Under the optional one-way read-only random-tape convention, the source may reuse its current random bit. Store a cached bit and a validity flag in finite control, initially invalid. At each transition obtain one fresh bit. Use it if the cache is invalid; otherwise use the cached bit and discard the

fresh one. Execute the source transition, then retain the used bit if the random head stayed and invalidate the cache if it advanced. This samples a newly visited random cell lazily on its next use. It preserves the joint distribution of the source run and needs no initialization transition. An initially halting state is recognized before the first transition. The same argument uses a fixed finite block for a convention allowing finitely many fresh bits per elementary transition, discarding unused bits. The finite cache changes neither counted work space nor the number of source transitions. Thus the supplied parameters a, b remain valid; only the static description length can increase.

Lemma 14.2 (A total resource-capped interpreter). *There is one randomized interpreter U that has polynomial worst-case time and $O(\log(m+2))$ counted work bits on every binary input of length m . Put*

$$\lambda_m = \lceil \log_2(m+2) \rceil, \quad R(m, d_{\text{code}}) = \left\lfloor \frac{\lambda_m}{100(d_{\text{code}} + 1)^3} \right\rfloor. \quad (14.1)$$

For an input beginning with

$$1^{d_{\text{code}}} 0 z 1^n 0 E_{d_{\text{code}}}(x), \quad (14.2)$$

where $|z| = d_{\text{code}}$ and $|x| = n$, it first checks $1 \leq d_{\text{code}} \leq \lambda_m$ and $R(m, d_{\text{code}}) \geq 1$. It simulates at most m transitions of the program z , restricting each counted work tape to coordinates $[-R, R]$ relative to its specified origin. It rejects malformed data, an out-of-range move, or a run that has not halted by that budget, and otherwise returns the encoded decision. The suffix after the complete prefix is ignored by parsing.

Proof. Count m before interpreting the prefix. Scan the unary code-length header with a bounded counter, rejecting if its length exceeds λ_m or the required delimiter is absent. Check the two eligibility conditions before copying z or allocating simulation storage. Then verify that the following d_{code} code bits, the unary data-length header, and the required data bits are present. The parsed n is at most m , and $d_{\text{code}}n \leq \lambda_m m$, so even size calculations preceding a failed length check use only $O(\lambda_m)$ bits. List counts, ID lengths, tape declarations, and movements are checked against the actual code and the stated bounds before any loops or allocations using them. There is no enumeration of an implicit domain of symbol or state names.

Store the encoded tapes sequentially on a fixed number of real work tapes of U . At most d_{code} strips, each with $2R + 1$ cells of d_{code} bits, require

$$d_{\text{code}}^2(2R + 1) = O(\lambda_m) \quad (14.3)$$

bits. Indeed, eligibility gives $d_{\text{code}}^3 \leq \lambda_m$ and $d_{\text{code}}^2 R \leq \lambda_m$. The copied program, current state, scanned and written symbol tuples, coin block, strip coordinates, step counter, and parsing scratch fit simultaneously in $O(\lambda_m)$ bits. A single simulated input-head coordinate has magnitude $O(m)$ by the transition cap and therefore also uses $O(\lambda_m)$ bits. Read the corresponding source symbol by rescanning the encoded data, supplying the appropriate markers or off-data blanks according to the encoded input-tape convention.

All routines are bounded scans and ordinary binary arithmetic. They iterate over actual code positions, input positions, a bounded tape/head list, strip positions, and bit positions, with a fixed nesting depth. Each range is polynomially bounded in $m + 2$; binary arithmetic is implemented by bit scans, not unit-cost operations on growing words. Matching a listed transition, fetching input symbols, updating the strips, and obtaining its finite coin block therefore take polynomial time in $m + 2$, with one fixed exponent determined by these routines. The same bounds hold when a check fails. There are at most m interpreted transitions. Checking the initial halting state and the state reached by the last permitted transition gives the asserted total behavior, including $m = 0$. The resulting fixed U has the required polynomial worst-case time on every random tape. $\square$

No bounded-error language promise is asserted for U on arbitrary inputs. Its acceptance probability may lie between $1/3$ and $2/3$. Probability approximation supplies a total separator even on these intermediate-probability inputs.

14.3 Fixing one deterministic library

Apply Theorem 3.1 to the fixed machine U and accuracy exponent 3, and compare its dyadic answer with $1/2$. The error on an input of length m is at most $(m+2)^{-3} \leq 1/8$. Hence the resulting deterministic machine accepts whenever $\Pr[U \text{ accepts}] \geq 2/3$ and rejects whenever that probability is at most $1/3$, since

$$\frac{1}{3} + \frac{1}{8} = \frac{11}{24} < \frac{1}{2} < \frac{13}{24} = \frac{2}{3} - \frac{1}{8}.$$

It halts on every input, including inputs between the two thresholds (Corollary 3.2). Its logarithmic space bound counts all simultaneously live work in the approximation algorithm. Store the $O(\log(m+2))$ -bit dyadic output in work space for the threshold test.

We fix this library once in the following convenient form. It is an ordinary finite-tape deterministic decision machine A with one input head confined to the endmarked input, initially at its left endmarker, such that every work head stays in

$$[-(\lambda_m + 1), \lambda_m + 1] \tag{14.4}$$

relative to its origin. To obtain this form, first consolidate any input heads by keeping their coordinates in logarithmic work space and scanning for each read. Even an off-data coordinate under an unbounded-input convention has logarithmic length because the original library has polynomial running time. A fixed change of initial control state handles its input-head start convention.

Next choose an integer w at least a coefficient bounding each counted work head's distance from its origin by $w\lambda_m$. Such one fixed w exists from the logarithmic bound, enlarged if necessary for the finitely many small lengths. Group consecutive work cells into blocks of width w , storing within-block offsets in finite control. The resulting finite alphabet and finite transition list realize Equation (14.4), also for two-sided tapes. All symbol widths will be charged in bits below; grouping only exposes a uniform address bound and gives no uncharged bit compression.

Fix A and its blocking width before any compiler input, using the existence guarantee of Theorem 3.1. The compiler's fixed data are the program grammar, A 's transition list with its blocked work alphabets and head conventions, and the finite callback templates below. The list includes the separator's construction constants, cutoffs, generators, and finite instruction routines.

The compiler computes its source-dependent integers and output program from these finite data and (M, a, b) by syntactic operations. Thus one fixed library gives one computable compiler for all source inputs; it need not discover a separator or a logarithmic-space coefficient from an arbitrary machine description. No constant is chosen as a noncomputable function of the source or its ordinary input.

14.4 Padding and exact preservation of acceptance probability

Given the source description and its supplied positive integers a, b , construct z as above and compute

$$\begin{aligned} P &= 1000(d_{\text{code}} + 2)^4(a + b + 2), \\ B_n &= \lceil \log_2(n + 2) \rceil, \quad N_{\text{pad}} = 2^{PB_n}. \end{aligned} \tag{14.5}$$

The generated machine will run A on a virtual binary input y_x of length N_{pad} whose prefix is Equation (14.2) and whose remaining bits are zero. Its prefix length is exactly

$$L_{\text{pre}} = (d_{\text{code}} + 1) + d_{\text{code}} + (n + 1) + d_{\text{code}}n = (d_{\text{code}} + 1)(n + 2). \quad (14.6)$$

Since $B_n \geq 1$, $2^{B_n} \geq n + 2$, and $2^{P-1} \geq d_{\text{code}} + 1$, this length is at most N_{pad} . The interpreter's length parameter is

$$\ell_{\text{pad}} := \lambda_{N_{\text{pad}}} = PB_n + 1. \quad (14.7)$$

All eligibility checks pass. In particular,

$$\frac{P}{100(d_{\text{code}} + 1)^3} = 10 \frac{(d_{\text{code}} + 2)^4}{(d_{\text{code}} + 1)^3} (a + b + 2) \geq 30(a + b + 2),$$

so

$$R(N_{\text{pad}}, d_{\text{code}}) \geq 30(a + b + 2)B_n - 1 \geq (a + b)B_n + 4. \quad (14.8)$$

Also $P > a$, whence $(n + 2)^a \leq 2^{aB_n} < N_{\text{pad}}$. Local motion from an initial work origin to a cell entails visiting the intervening counted cells. The source bound of bB_n work bits therefore places every source work cell in the allocated strip; the slack in Equation (14.8) also covers a final one-cell move. This uses the source's original cell costs, each at least one bit, and does not charge it for the larger interpreter IDs.

Couple the source and interpreter by giving them the same fresh coin block at each corresponding transition. Any extra bits in an interpreter block are ignored as specified by the source encoding. Induction on the transition count then preserves the encoded state, tape contents, head positions and decision; the time and strip inequalities above ensure that no cutoff interrupts this correspondence. For the optional cached random-tape convention, the cached bit is the same cell value in this coupling. Thus the two acceptance indicators agree on every coupled run, and

$$\Pr[U(y_x) \text{ accepts}] = \Pr[M(x) \text{ accepts}]. \quad (14.9)$$

There is no conditioning or additional approximation in this equality. The decision of $A(y_x)$ is therefore the decision required for the language recognized by M under the stated bounded-error promise. Although N_{pad} is given exponentially, it is at most $(n + 2)^{2P}$ for the fixed source. More importantly, it need never be written as an input string.

Remark 14.3 (Optional head conventions). The construction also applies when the fixed coding and U are chosen to allow at most d_{code} heads per tape, with the usual common initial origin. These conventions are fixed before choosing A . There are then at most d_{code}^2 work-head coordinates and scanned symbols. Their storage is bounded by

$$O(d_{\text{code}}^2 \log(2R + 1) + d_{\text{code}}^3) = O(\lambda_m).$$

For multiple input heads, or an optional entirely blank tape whose cells are not charged, use signed R -bit coordinates and reject on overflow. At most d_{code}^2 such coordinates still cost $O(d_{\text{code}}^2 R)$ bits. These optional conventions do not change the interpreter's bounds. The padded instances just constructed never overflow those coordinates.

Under the optional head conventions, local motion bounds an input or uncharged blank-tape coordinate by the source time, up to the fixed endmarker offset. Its signed encoding fits because $R \geq aB_n + 4$.

Figure 2 shows how the generated machine supplies the virtual input requested by A . The library's work tapes remain live while the callback scans the real input; the resource bound below counts the library and callback storage simultaneously.

Virtual input: $N_{\text{pad}} = 2^{PB_n}$ bits

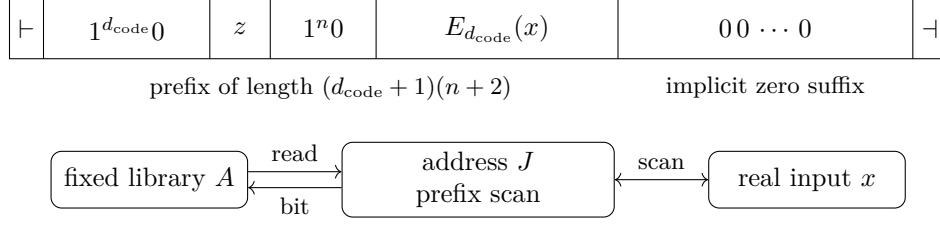


Figure 2: The fixed library reads the padded input through a callback. The callback regenerates prefix symbols from the real input and returns zero for an address in the padding interval. Here J is the virtual input address requested by A .

14.5 Compiling the virtual input operations

Inline the finite transition list of A , giving it its own work tapes. Write $D = D_M$ for the generated machine. It suspends its library control state in finite control during an input operation. Four auxiliary binary words suffice: the length count $n + 1$, the virtual right-endmarker address $N_{\text{pad}} + 1$, the virtual input-head position J , and a prefix position I . Each has a separate tape, with origin and termination marks and digits in low-end-first order.

Initialize the length count to one and increment it once per real input symbol. Its bit length is exactly

$$\text{bitlength}(n + 1) = \lfloor \log_2(n + 1) \rfloor + 1 = \lceil \log_2(n + 2) \rceil = B_n \quad (n \geq 0). \quad (14.10)$$

For every bit place in that count, write P zero bits on the bound tape, then append a one. This produces the low-end-first representation of $2^{PB_n} = N_{\text{pad}}$; increment it to obtain $N_{\text{pad}} + 1$. The repetition of P is hardwired finite control. Initially set $J = 0$, the library's left-endmarker position.

At a virtual read, first test $J = 0$ and $J = N_{\text{pad}} + 1$ and return the corresponding endmarker. Otherwise initialize $I = 1$ and regenerate the prefix in its natural order:

1. emit the fixed string $1^{d_{\text{code}}}0z$;
2. scan x , emitting one 1 per symbol, and then emit 0;
3. rewind and scan x again, emitting each fixed-width symbol code in $E_{d_{\text{code}}}(x)$.

Here “emit” means compare the current I with J , return the bit if they agree, and otherwise increment I and continue. If the scan finishes without a match, return zero. The strings z , the integer d_{code} , symbol codes and emission phases are fixed in the generated control. The real input head is rewound by scanning to its endmarker, without storing its position. In particular, $J = N_{\text{pad}} + 1$ is answered before regeneration, while a padding address such as $J = N_{\text{pad}}$ returns zero after the finite prefix scan. The largest prefix counter value is $L_{\text{pre}} + 1 \leq N_{\text{pad}} + 1$.

For explicit counter routines, an increment uses a finite carry flag. Compare two low-end-first words by scanning both, retaining only a finite comparison flag that is overwritten by each higher differing bit. After one terminator, treat its later bits as zero using a finite “ended” flag. Rewind each tape to its origin mark afterward. Zeroing and copying are likewise bounded bit scans. No growing arithmetic register beyond the named words is required. All words have at most $PB_n + 2$ bits; including delimiters and rewinds, their heads need visit only positions through $PB_n + 10$.

After the callback, perform the indicated library transition and update J in place for its input-head motion. The library's work heads and contents are untouched by the callback. Its state, the returned input symbol, and any step-local scanned tuple are fixed finite-control data. The callback

does not invoke A recursively or maintain a stack of callbacks. The real input head stays between its real endmarkers.

Every listed routine terminates. For example, a complete regeneration uses at most L_{pre} emissions, each with an $O(PB_n + 1)$ -bit comparison and increment, together with two real-input scans. Initialization makes n increments and writes $PB_n + 1$ bound bits. These observations give direct polynomial bounds for the auxiliary operations. The whole simulation halts because A is total. Its more uniform explicit time bound, including every such operation, is derived next from the complete configurations of this same machine.

All the above instructions can be generated as finite transition tables, unrolling fixed repetitions and strings if necessary. Formation of the source list, the integer P , and this transition table is a finite effective process. It makes no test of the source's semantic error or resource promises. Syntactically invalid compiler inputs may be assigned a fixed default output; correctness is required on the stipulated class. The generated simulation is total and obeys the bounds below even when the supplied source promises fail; those promises are used to establish that its decision agrees with the source.

14.6 Explicit simultaneous bounds

Use the finite-alphabet description just constructed to compute t , its number of work tapes; $g \geq 2$, the maximum of two and its work alphabet sizes, including all blank, origin and termination symbols; and q_D , its number of states, including callback phases. Let $u \geq 1$ dominate its explicit transition-table bit length, all alphabet sizes (including the input alphabet), the number of states, and the number of tapes. Padding this integer is effective. Return the positive integer bounds

$$\begin{aligned} \rho &= 100(P + 2), & S &= 1 + (t + 1)(2\rho + 1)(g + 2), \\ K &= 4S, & c &= 2S + 1, & H &= 2^{(u+100)^2}(q_D + 1). \end{aligned} \tag{14.11}$$

These depend on the fixed source and the fixed library, and are computed from finite syntax. They have no dependence on n or x .

Every work head stays in $[-\rho B_n, \rho B_n]$. For a library tape this follows from Equations (14.4) and (14.7), since $\ell_{\text{pad}} + 1 = PB_n + 2 \leq \rho B_n$. For an auxiliary tape it follows from $PB_n + 10 \leq \rho B_n$, using $B_n \geq 1$. Thus the library's live data and all callback data satisfy the same bound at the same time. Encoding a work alphabet of size g_i in $\max\{1, \lceil \log_2 g_i \rceil\} \leq g$ bits per symbol bounds their total data by

$$t(2\rho B_n + 1)g \leq SB_n. \tag{14.12}$$

Even twice this quantity satisfies the stronger logarithmic bound: with $L_n = \log_2(n + 2) \geq 1$ we have $B_n \leq 2L_n$ and

$$2SB_n \leq 4SL_n = KL_n.$$

For a fixed real input x , a complete configuration is specified by the state, the real input-head position, all work-head positions, and all work symbols in the stated intervals. Everything outside these intervals remains at its initial value. There are at most

$$q_D(n + 2)(2\rho B_n + 1)^t g^{t(2\rho B_n + 1)} \leq q_D(n + 2)2^{SB_n} \leq q_D(n + 2)^c \tag{14.13}$$

such configurations. For the first inequality use

$$2\rho B_n + 1 \leq (2\rho + 1)B_n, \quad \log_2 g \leq g, \quad \log_2(2\rho B_n + 1) \leq (2\rho + 1)B_n;$$

the second uses $B_n \leq 2 \log_2(n + 2)$ and $c = 2S + 1$. A deterministic halting computation cannot repeat a complete configuration. Therefore Equation (14.13) bounds its total transitions, including initialization, counter arithmetic, every regenerated input bit, and every library transition. The virtual head position is already in the counted word J ; it is not an additional uncounted configuration component.

The preceding count refers to ordinary finite-alphabet transitions. To charge them in bits, encode each work symbol by a fixed-width block, mapping blank to the all-zero block. Read and write within the current block, return to its first bit, and move to the first bit of the target block. Within-block phases and scanned finite tuples are stored in finite control. No per-block writable delimiter is needed, and these operations visit only blocks whose macro positions lie in the specified work intervals. Existing origin and termination symbols have already been included in g . Thus the block implementation obeys the bit-space estimate above; the factor two also allows the usual blank/bit tape convention. Input-symbol access has fixed width and is covered by the same finite description parameter u .

There are at most $u + 1$ scanned symbols of at most u bits each. An explicit sequential comparison against at most u transition rows, followed by the fixed-width reads, writes and moves, uses at most a fixed polynomial in $u + 1$ bit actions per transition. For example, the deliberately loose bound $100(u + 1)^8$ suffices for these scans and comparisons, with all temporary phase and symbol information in finite control. It is dominated, for $u \geq 1$, by

$$100(u + 1)^8 \leq 2^{(u+100)^2}.$$

The binary microcode may have more states than q_D , but no new configuration count is needed: multiply the number of original transitions by this per-transition bit cost. Equations (14.11) and (14.13) give the simultaneous bound

$$\text{time}_{\text{bit}}(D, x) \leq H(n + 2)^c. \quad (14.14)$$

No growing-word arithmetic is charged as a single elementary bit action.

References

- [1] AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. In *61st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 1295–1306. IEEE, 2020. Full version: arXiv:1912.04524v3, March 11, 2022.
- [2] Algirdas Avizienis. On a flexible implementation of digital computer arithmetic. In C. M. Popplewell, editor, *Information Processing 1962*, pages 664–670. North-Holland, 1963.
- [3] Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Computing with a full memory: Catalytic space. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 857–866. Association for Computing Machinery, 2014. Full version: ECCC TR14-053, revision 1.
- [4] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.
- [5] Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error reduction for weighted PRGs against read once branching programs. In *36th Computational Complexity*

- Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- [6] Gil Cohen, Dean Doron, Ori Sberlo, and Amnon Ta-Shma. Approximating iterated multiplication of stochastic matrices in small space. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 35–45. Association for Computing Machinery, 2023. Full version: ECCC TR22-149 (2022), published November 12, 2022.
 - [7] Stephen A. Cook and Pierre McKenzie. Problems complete for deterministic logarithmic space. *Journal of Algorithms*, 8(3):385–394, 1987.
 - [8] William M. Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics*, pages 28:1–28:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
 - [9] Klaus-Jörn Lange, Pierre McKenzie, and Alain Tapp. Reversible space equals deterministic space. *Journal of Computer and System Sciences*, 60(2):354–367, 2000.
 - [10] Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. In *64th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 989–1007. IEEE, 2023. Full version: arXiv:2303.16413v1, March 29, 2023.
 - [11] Marcel Riesz. Sur les maxima des formes bilinéaires et sur les fonctionnelles linéaires. *Acta Mathematica*, 49:465–497, 1927.
 - [12] Yehuda Shalom. Bounded generation and Kazhdan’s property (T). *Publications Mathématiques de l’IHÉS*, 90:145–168, 1999.