

Beyond Width-3: Better Hitting Set Generators for Width-4 Read-Once Branching Programs

Gonen Krak[†]

Abstract

We construct an explicit hitting set generator (HSG) for ordered read-once branching programs of width 4. For every length n and every $\varepsilon \in (0, 1)$, every width-4 length- n program that accepts more than an ε fraction of its inputs accepts at least one output of the generator, and the seed length is

$$O((\log n)^{3/2} \cdot \sqrt{\log \log n} \cdot (1 + \log(1/\varepsilon))).$$

For constant ε this is $O(\log^{3/2} n \cdot \sqrt{\log \log n})$. More generally, it is $o(\log^2 n)$ whenever $\varepsilon \geq 2^{-o(\sqrt{\log n / \log \log n})}$. This is the first explicit generator for width 4 that improves on the $O(\log^2 n)$ seed length of Nisan's generator (Combinatorica 1992).

Our construction first reduces the hitting problem for width-4 programs to the hitting problem for programs with three live states and one rejecting state, following Doron and Hoza (RANDOM 2025). It then simplifies these programs by rounds of pseudorandom restrictions, and hits the simplified programs with a small-bias string that is modified in a small number of positions.

[†]Tel Aviv University. gonenok10@gmail.com

Contents

1	Introduction	3
1.1	General ROBPs	3
1.2	Regular and permutation programs	4
1.3	Programs of very small width	4
1.4	Our result	5
2	Proof overview	5
3	Preliminaries and definitions	14
3.1	Notation and program conventions	14
3.2	Fourier analysis and approximation	15
3.3	Pseudorandom sources	16
3.4	A one-step restriction source	17
4	Reduction to programs with absorbing rejection	19
5	Collisions and the total repair bound	21
6	Repair cost under uniform input	26
6.1	The checkpoint class and the repair theorem	26
6.2	Proof overview	27
6.3	Charging repairs to suffix density	29
6.4	Allowed states and the uniform repair	32
6.5	Composing the uniform repairs	35
7	Repair charges as state visits	36
7.1	Proof overview	36
7.2	Constructing the structured local repairs	45
7.3	Proof of the threshold selection lemma	50
7.4	The gain estimate under uniform input	54
7.5	Global structured repair and uniform cost	56
8	Repair cost under small bias	57
8.1	Covering state visits by tests with two live states	57
8.2	Acceptance functions as parity constraints	65
8.3	Grouping parity constraints into a chain	68
8.4	Fourier majorants for products of acceptance functions	71
8.5	Transferring the cost bound	75
8.6	From local repairs to a hitting set	76
9	Restrictions to $\mathcal{K}_{3,\ell}$	77
9.1	The restriction source and checkpoints	79
9.2	A window condition for bounded collisions	80
9.3	A dense residual with checkpoints	81
10	Assembly and seed length	83
10.1	Completing the generator and balancing the seed costs	83
10.2	Constant-density assembly	84
10.3	Error reduction	85
	References	86

1 Introduction

A central question in complexity theory is whether randomness adds power to space-bounded computation, that is, whether $L = BPL$, and in its one-sided form whether $L = RL$. The standard route to both questions goes through generators for read-once branching programs. On a fixed input, a randomized algorithm that uses space S and n random bits is modeled by a branching program of width $2^{O(S)}$ and length n that reads the random bits.

Definition 1.1 (ROBP). Let $w, n \in \mathbb{N}$. A width- w length- n read-once branching program is a directed graph whose vertices are arranged in layers $V_0, V_1, \dots, V_n$ with $|V_i| \leq w$. Every vertex in $V_0 \cup \dots \cup V_{n-1}$ has two outgoing edges into the next layer, labeled 0 and 1. There is a start vertex $v_0 \in V_0$, and every vertex in V_n is labeled accept or reject. On input $x \in \{0, 1\}^n$ the program starts at v_0 and in step i follows the edge labeled x_i . It computes the function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ with $f(x) = 1$ exactly when the final vertex is accepting.

Throughout, programs read their input in the standard order $x_1, \dots, x_n$, and $\mathbb{E}[f]$ denotes the expectation of $f(x)$ for a uniform $x \in \{0, 1\}^n$.

Definition 1.2 (PRG). Let $\mathcal{F}$ be a class of functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$. A function $G: \{0, 1\}^s \rightarrow \{0, 1\}^n$ is an ε -PRG for $\mathcal{F}$ if for every $f \in \mathcal{F}$

$$\left| \mathbb{E}_{x \in \{0, 1\}^n}[f(x)] - \mathbb{E}_{y \in \{0, 1\}^s}[f(G(y))] \right| \leq \varepsilon.$$

In words, replacing a uniform input by the output of G on a uniform seed changes the acceptance probability of every $f \in \mathcal{F}$ by at most ε . The number s is the seed length.

Definition 1.3 (HSG). A function $G: \{0, 1\}^s \rightarrow \{0, 1\}^n$ is an ε -HSG for $\mathcal{F}$ if for every $f \in \mathcal{F}$

$$\mathbb{E}[f] > \varepsilon \implies f(G(y)) = 1 \text{ for some } y \in \{0, 1\}^s.$$

In words, every function in the class that accepts more than an ε fraction of the inputs accepts at least one output of G .

A generator is explicit if its output can be computed in time polynomial in n and $1/\varepsilon$. For the space-bounded derandomization implications below, one additionally requires computation in space $O(s)$. The probabilistic method gives a non-explicit ε -PRG for width- w length- n ROBPs with seed length

$$s = O(\log(wn/\varepsilon)),$$

which is optimal up to a constant factor for polynomial width and constant error. A PRG with this seed length for $w = \text{poly}(n)$ that is computable in space $O(s)$ would give $L = BPL$, and an HSG with the same space guarantee would give $L = RL$.

1.1 General ROBPs

In his seminal work, Nisan [Nis92] constructed an explicit ε -PRG for width- w length- n ROBPs with seed length

$$O(\log(wn/\varepsilon) \cdot \log n).$$

Impagliazzo, Nisan and Wigderson [INW94] obtained the same seed length by a different construction. For $w = \text{poly}(n)$ and constant ε the seed length is $O(\log^2 n)$, a factor of $\log n$ above the optimum. No better PRG is known for polynomial width.

For hitting, Hoza and Zuckerman [HZ20] constructed an explicit ε -HSG with seed length

$$O(\log(wn) \cdot \log n + \log(1/\varepsilon)).$$

This bound separates the dependence on ε from the dependence on n , but for $w = \text{poly}(n)$ its first term is still $\log^2 n$.

The two notions are related as follows. Every ε -PRG is an ε -HSG: if $\mathbb{E}[f] > \varepsilon$, then the expectation of $f(G(y))$ over a uniform seed y is positive, so some seed is accepted. The converse fails, and the direct simulation using an HSG only derandomizes one-sided error. Cheng and Hoza [CH22, Theorem 2.1] showed that hitting sets nevertheless suffice for two-sided error in the white-box setting: given the description of a width- w length- n ROBP f , one can deterministically estimate $\mathbb{E}[f]$ to additive error ε using a $1/2$ -hitting set for ROBPs of width $O(w^3 n^2 \log(wn)/\varepsilon^2)$ and length $O(w^3 n^4 \log(wn)/\varepsilon^2)$. The simulation uses the space needed to compute the hitting set plus $O(\log(wn/\varepsilon))$. In particular, HSGs for polynomial-width ROBPs with seed length $O(\log n)$ that are computable in space $O(\log n)$ would imply $\mathsf{L} = \mathsf{BPL}$. This makes HSGs a target of independent interest. An intermediate notion is the weighted PRG (WPRG) of Braverman, Cohen and Garg [BCG20], in which the outputs of the generator carry real weights that may be negative.

1.2 Regular and permutation programs

Because improving Nisan's generator for general ROBPs has proved very difficult, programs with additional structure have been studied.

Definition 1.4. A ROBP with layers $V_0, \dots, V_n$ is regular if every vertex in $V_1 \cup \dots \cup V_n$ has exactly two incoming edges. It is a permutation program if, in addition, the two incoming edges of every such vertex carry different labels.

Braverman, Rao, Raz and Yehudayoff [BRRY14] showed that the INW generator with suitable parameters ε -fools regular width- w length- n ROBPs with seed length

$$O((\log w + \log \log n + \log(1/\varepsilon)) \cdot \log n).$$

For constant w and ε this is $O(\log n \cdot \log \log n)$. The same paper gives a hitting set for regular programs: the strings with at most $w - 1$ ones hit every regular width- w program that accepts at least one input. For constant w this is a hitting set of size $\text{poly}(n)$ that does not depend on ε .

For permutation programs of constant width, PRGs with seed length $O(\log n \cdot \log(1/\varepsilon))$ are known [KNP11, De11, Ste12, CDG26]. At error $\varepsilon = 1/n$ this is again $\log^2 n$. Doron and Hoza [DH25] showed that this low-error regime is tied to width 4. Let $c \in [1, 2)$ be a constant. If there is an explicit $(1/n)$ -PRG for width-6 length- n permutation ROBPs with seed length $\tilde{O}(\log^c n)$, then there is an explicit ε -HSG for width-4 length- n ROBPs with seed length

$$\tilde{O}(\log^c n + \log n \cdot \log(1/\varepsilon)).$$

This result is conditional. No PRG as in the hypothesis is known for any $c < 2$.

1.3 Programs of very small width

A second line of work removes the structural assumption and restricts the width instead.

For width 2, small-bias distributions fool every program [SZ95, BDVY13], which gives an explicit ε -PRG with the optimal seed length $O(\log(n/\varepsilon))$.

For width 3, the first results were hitting sets. Šíma and Žák [SZ11, SZ21] gave a polynomial-time construction of a hitting set for width-3 programs with acceptance probability above $5/6$. Gopalan, Meka, Reingold, Trevisan and Vadhan [GMR⁺12] constructed an ε -HSG for all width-3 ROBPs with seed length $\tilde{O}(\log(n/\varepsilon))$, using iterated mild pseudorandom restrictions. Meka, Reingold and Tal [MRT19] obtained the first PRG for width 3 that improves on Nisan's, with seed length $\tilde{O}(\log n \cdot \log(1/\varepsilon))$.

Recently, Krak [Krak26] constructed an explicit ε -HSG for width-3 length- n ROBPs with seed length

$$O(\log(n/\varepsilon)).$$

This is the first construction for width 3 whose seed length is optimal in both n and ε . The hitting problem for width 3 is therefore settled up to the constant.

For width 4 the situation was very different. Before this work, no explicit PRG, WPRG or HSG with seed length $o(\log^2 n)$ was known for width-4 ROBPs [DH25], so the bound of Nisan and INW was still the best available. The gap between width 3, where the HSG seed length is optimal, and width 4, where it is $\log^2 n$, makes width 4 the natural next case.

1.4 Our result

We give the first explicit generator for width-4 ROBPs with seed length below $\log^2 n$.

Theorem 1.5. *For every $n \geq 4$ and $\varepsilon \in (0, 1)$, there is an explicit ε -HSG for width-4 length- n ROBPs with seed length*

$$O((\log n)^{3/2} \cdot \sqrt{\log \log n} \cdot (1 + \log(1/\varepsilon))). \quad (1.1)$$

For constant ε the seed length is $O(\log^{3/2} n \cdot \sqrt{\log \log n})$, compared with $O(\log^2 n)$ from [Nis92, INW94] and from [HZ20]. More generally, Theorem 1.5 improves on $\log^2 n$ whenever $\varepsilon \geq 2^{-o(\sqrt{\log n / \log \log n})}$.

2 Proof overview

Throughout the proof, we use the notation

$$\mu(F) = \Pr_{x \sim U}[F(x) = 1]$$

for the density of a Boolean function F , where U is uniform on its input coordinates.

We first construct one generator that hits every width-four program of density at least $1/4$. Doron–Hoza’s error reduction will then extend this generator to every density threshold $\varepsilon \in (0, 1]$. The constant-density construction has six steps.

1. **Reduction to a structured class.** We use Doron–Hoza’s reduction to reduce the width-four hitting problem to hitting a more structured class of programs. These programs have three live states and one rejecting state in each layer. Rejection is absorbing: once the computation reaches a rejecting state, it remains in rejecting states in all later layers. The program accepts exactly when its computation avoids rejection.

The reduction combines candidate prefixes generated by an ordinary width-three HSG with continuations generated by an HSG for this structured class. Optimal width-three HSGs are known [Kra26], so our main task is to construct the generator for the continuations.

We apply this reduction at constant density: hitting the structured programs at threshold $1/8$ suffices to hit width-four programs at threshold $1/4$. Finally, Doron–Hoza’s error-reduction lemma extends the resulting generator to every density threshold ε , as explained in Section 10.

2. **Repairing programs with few colliding layers.** We call a layer *colliding* if one of its transition maps sends two distinct live states to the same live state in the next layer. Transitions into rejection do not count as collisions.

We then prove the following lemma:

Lemma 2.1 (repair with few colliding layers). There is an absolute constant C with the following property. Let P be a length- n program with three live states, absorbing rejection, density at least $\varepsilon \in (0, 1]$, and at most $c \geq 0$ colliding layers. There is a deterministic procedure

$$\text{repair}_P^{\text{Total}} : \{0, 1\}^n \rightarrow \{0, 1\}^n$$

such that, for every $x \in \{0, 1\}^n$, the word $z = \text{repair}_P^{\text{Total}}(x)$ satisfies

$$P(z) = 1, \quad d_H(x, z) \leq C(c + \log(1/\varepsilon)).$$

Here d_H denotes Hamming distance.

Section 5 constructs this procedure and proves the bound with an explicit constant. Thus, any starting input can be made accepting by changing a bounded number of coordinates.

This immediately gives a hitting set for programs with few colliding layers. Start with 0^n , which we call the *center*, and enumerate all strings with at most $C(c + \log(1/\varepsilon))$ ones. The repair lemma guarantees that this set contains an accepting input. Specifying the positions of the ones uses

$$O((c + \log(1/\varepsilon)) \log n)$$

seed bits.

A general program with three live states and absorbing rejection may have many colliding layers, so this seed bound can be large. However, the repair lemma gives a worst-case bound over all centers. It does not imply that every center requires that many changes.

3. **Bounding repair cost by gains in suffix density.** This observation leads us to ask whether the average distance to an accepting input can be much smaller than the bound that holds for every center. For a program P of positive density, define

$$d_H^P(x) = \min\{d_H(x, y) : y \in \{0, 1\}^n, P(y) = 1\}.$$

Thus, $d_H^P(x)$ is the minimum number of coordinates that must be changed to make x accepting. Our first goal is to show that

$$\mathbb{E}_{X \sim U}[d_H^P(X)]$$

is small for every program P in a suitable class, where U is the uniform distribution on the input coordinates.

We could then hope to obtain a comparably small expected distance under a single pseudorandom distribution D that can be sampled using few seed bits. If

$$\mathbb{E}_{X \sim D}[d_H^P(X)] \leq R$$

for every program in the class, then the support of D contains a center within distance R of an accepting input for each such program. This suggests a generator: sample a center $x \sim D$, and enumerate all strings within Hamming distance R of x . If D uses r seed bits and R is an integer, specifying the center and the positions to change uses $r + O(R \log n)$ seed bits. The same distribution and enumeration must work for every target program.

We carry out this approach by constructing a repair procedure $\text{Repair}^P : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and a cost procedure $\text{Cost}^P : \{0, 1\}^n \rightarrow \mathbb{N}_0$, where $\mathbb{N}_0$ denotes the nonnegative integers. For every candidate input x , the repaired input $y = \text{Repair}^P(x)$ must satisfy $P(y) = 1$ and

$$d_H^P(x) \leq d_H(x, \text{Repair}^P(x)) \leq \text{Cost}^P(x).$$

It therefore suffices to bound the expected repair cost. We first do so under uniform input. For a pseudorandom distribution, we later construct a second repair with charges that admit a suitable representation.

Our approach begins by identifying a structured class $\mathcal{K}_{3,\ell}$, indexed by an integer $1 \leq \ell \leq n$. We will obtain programs in this class by applying restrictions. A *restriction* ρ assigns values to some input coordinates and leaves the others free. The resulting function on the free coordinates is the *residual program*, denoted $P|_\rho$. Item 6 describes the structural conditions defining $\mathcal{K}_{3,\ell}$ and constructs the required restrictions. For every program $P \in \mathcal{K}_{3,\ell}$ with density $\mu(P) \geq 1/16$, we construct repair and cost procedures satisfying

$$\mathbb{E}_{X \sim U}[\text{Cost}^P(X)] \leq C\ell$$

for an absolute constant C . Here the program P is fixed, and the expectation is over the independent uniform coordinates of X . The pointwise distance bound then gives

$$\mathbb{E}_{X \sim U}[d_H^P(X)] \leq C\ell.$$

We obtain this expectation bound with a repair procedure that processes the candidate one segment at a time. To define the segments, fix in advance layers

$$0 = t_0 < t_1 < \dots < t_m = d,$$

called *checkpoints*. They depend only on the program, not on the candidate. Segment j consists of the transitions $t_j + 1, \dots, t_{j+1}$, and F_j denotes the free coordinates read there. For a state s at checkpoint t_j and a block $z \in \{0, 1\}^{F_j}$, write $P_{j,s}(z)$ for the state reached at checkpoint t_{j+1} when the original program starts at s and reads z ; this may be rejection. For a candidate x , write $x_j = x|_{F_j}$ for its block in segment j . Starting from

$$S_0(x) = s_0,$$

stage j replaces the block x_j by a repaired block y_j , chosen so that the computation from $S_j(x)$ crosses segment j without rejecting and reaches a suitable live state

$$S_{j+1}(x) = P_{j,S_j(x)}(y_j).$$

This state is the entrance of stage $j + 1$. Let $c_j(x)$ be an integer with

$$d_H(x_j, y_j) \leq c_j(x),$$

and set

$$\text{Cost}^P(x) = \sum_{j=0}^{m-1} c_j(x).$$

The blocks $y_0, \dots, y_{m-1}$ form an input y accepted by P . Since the segments are disjoint,

$$d_H(x, y) = \sum_{j=0}^{m-1} d_H(x_j, y_j) \leq \text{Cost}^P(x).$$

To analyze the expected cost, let $X \sim U$. The state $S_j(X)$ depends only on $X_0, \dots, X_{j-1}$, while the block X_j is uniform and independent of them. So the repair at stage j , its charge $c_j(X)$, and the next state $S_{j+1}(X)$ are determined by a fresh uniform block.

We measure progress by the suffix density of Definition 3.1. Along every repaired computation it starts at $\mu = \mu(P)$ and ends at 1, so its increments satisfy

$$\sum_{j=0}^{m-1} \left(\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}(x)) - \mu_{t_j}^{\text{suffix}}(S_j(x)) \right) = 1 - \mu$$

for every candidate x , however many segments there are. Without repair, a fresh uniform block leaves the expected suffix density unchanged:

$$\mathbb{E}_{Z \sim U_j} \left[\mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(Z)) \right] = \mu_{t_j}^{\text{suffix}}(s).$$

Every expected increase is therefore produced by the repair. We build each local repair so that, from every entrance state s the procedure can reach,

$$\mathbb{E}_{Z \sim U_j} [c_j] \leq B \left(\mathbb{E}_{Z \sim U_j} \left[\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}) \right] - \mu_{t_j}^{\text{suffix}}(s) \right),$$

where $B = O(\ell)$. Here c_j and S_{j+1} are computed from the entrance $S_j = s$ and the uniform block Z . Averaging over $S_j(X)$ and summing over j , the intermediate densities cancel, which gives

$$\mathbb{E}_{X \sim U} [\text{Cost}^P(X)] \leq B(1 - \mu) \leq B = O(\ell).$$

The gain-cost lemma (Lemma 6.3) proves this bound.

The structural conditions defining $\mathcal{K}_{3,\ell}$, together with the density lower bound, allow us to construct repairs with this local guarantee. In the next two steps, we construct a second repair, prove its own uniform expected-cost bound, and transfer that bound to small bias, obtaining the pseudorandom centers needed for the Hamming-ball enumeration. Item 6 then describes $\mathcal{K}_{3,\ell}$ and shows how to obtain a residual in this class with the required density.

Theorem 2.2 (uniform expected repair cost). There exist program classes $(\mathcal{K}_{3,\ell})_{\ell \geq 1}$, described in Item 6, and an absolute constant A with the following property. Let $n \geq 4$ and $1 \leq \ell \leq n$, and let $P \in \mathcal{K}_{3,\ell}$ have at most $n + 1$ transition positions, free coordinates $F \subseteq [n]$, and density $\mu(P) \geq 1/16$. There are deterministic procedures

$$\text{Repair}^P : \{0, 1\}^F \rightarrow \{0, 1\}^F, \quad \text{Cost}^P : \{0, 1\}^F \rightarrow \mathbb{N}_0$$

such that, for every $x \in \{0, 1\}^F$,

$$P(\text{Repair}^P(x)) = 1, \quad d_H(x, \text{Repair}^P(x)) \leq \text{Cost}^P(x).$$

Moreover, if U_F is the uniform distribution on $\{0, 1\}^F$, then

$$\mathbb{E}_{X \sim U_F} [\text{Cost}^P(X)] \leq A\ell.$$

The repair modifies only free coordinates; all fixed transition choices remain unchanged.

4. **Represent repair charges by state visits.** The uniform argument of Item 3 does not apply directly under small bias, because it uses the independence of successive uniform blocks. Item 5 will instead bound state-visit probabilities for three-state programs satisfying the checkpoint and collision conditions stated below. Here we construct a new repair procedure whose charge is a weighted sum of such events. The simpler repair of Item 3 demonstrates how to bound repair cost by gains in suffix density. For the small-bias proof, we replace that procedure while retaining its cost-gain argument. Concretely, we want an auxiliary three-state program $\hat{P}$ that reads the candidate x and never rejects, with state $\hat{S}_t(x)$ at layer t , such that

$$\text{Cost}(x) = \sum_a w_a \mathbf{1}[x \in \mathcal{E}_a].$$

Each $\mathcal{E}_a$ specifies a visit to a state at a layer, possibly together with a specified value of the next candidate bit.

The direct representation of the previous repair's charge does not satisfy these requirements. It runs the original interval to its end and checks the exit state. That computation has three live states and a

rejecting state, hence four states. Section 7 therefore constructs a new repair whose main branch makes its decisions bit by bit.

The new repair. Let $\mu = \mu(P)$. A checkpoint state is *allowed* if it is reachable from the original start state and has suffix density at least $\mu/2$. Write $\mathcal{A}_j$ for the allowed states at checkpoint t_j .

Fix an interval I_j . For a state u at a layer t of this interval, define

$$\sigma_t(u) = \Pr \left[\begin{array}{c} \text{the original computation from } u \text{ at layer } t \\ \text{ends in } \mathcal{A}_{j+1} \end{array} \right],$$

where the remaining free bits of I_j are uniform. These probabilities refer to the fixed interval I_j , and $\sigma_t(\perp) = 0$.

We choose a threshold

$$\theta_j \in \left(0, \frac{\mu}{12}\right].$$

From an entrance state s with $\sigma_{t_j}(s) \geq \theta_j$, the repair scans the block. At a free transition t , from a current state u with $\sigma_{t-1}(u) \geq \theta_j$, it uses the bit

$$\beta_{t,\theta_j}(u, b) = \begin{cases} b, & \sigma_t(T_t^b(u)) \geq \theta_j, \\ 1 - b, & \sigma_t(T_t^b(u)) < \theta_j, \end{cases}$$

where $T_t^b(u)$ is the original successor of u under bit b .

The averaging identity

$$\sigma_{t-1}(u) = \frac{\sigma_t(T_t^0(u)) + \sigma_t(T_t^1(u))}{2}$$

ensures that at least one successor remains at or above the threshold. Fixed transitions preserve σ and require no edit. The repaired computation therefore maintains $\sigma \geq \theta_j$. At the exit layer,

$$\sigma_{t_{j+1}}(v) = \mathbf{1}[v \in \mathcal{A}_{j+1}],$$

so it ends in an allowed exit state.

A suitable choice of θ_j confines every possible flip to a common set E_j with

$$|E_j| = O(\ell),$$

independently of the entrance state and candidate block. Entrance states with $\sigma_{t_j}(s) < \theta_j$ use a separate repair that changes at most $r = O(\ell)$ free coordinates. For such an entrance state, the allowed exit set is a singleton.

The auxiliary program. From an entrance state handled by scanning, $\hat{P}$ reads the original candidate and reproduces the repaired computation: when the repair flips a bit, $\hat{P}$ takes the corresponding transition of P . From a low entrance state, it need only reach the same exit state as the separate repair. Uniqueness of the allowed exit makes this possible by a fixed reset in the auxiliary program. The actual repair still uses the original transitions of P .

Thus, at every checkpoint, the auxiliary state agrees with the state reached by the repaired computation. Let $\hat{t}_j$ denote the auxiliary checkpoint corresponding to the original checkpoint t_j ; these indices may differ because of the resets.

There are two kinds of charge.

1. A **scanning trigger** is a triple $a = (t, u, b)$, where t is a free transition of I_j , satisfying

$$\sigma_{t-1}(u) \geq \theta_j, \quad \beta_{t,\theta_j}(u, b) \neq b.$$

Let $O_a(x)$ indicate that $\hat{P}$ visits the corresponding source node, and let $i(a)$ be the input coordinate read there. The trigger is activated when

$$O_a(x) \mathbf{1}[x_{i(a)} = b] = 1.$$

2. A **low entrance** is a pair $\gamma = (j, s)$ with

$$s \in \mathcal{A}_j, \quad \sigma_{t_j}(s) < \theta_j.$$

Its visit indicator is

$$O_\gamma(x) = \mathbf{1}[\widehat{S}_{t_j}(x) = s].$$

Let $\mathcal{C}$ and $\mathcal{B}$ index these triggers and low entrances, respectively. Writing $\boldsymbol{\theta} = (\theta_0, \dots, \theta_{m-1})$, define

$$\text{Cost}_{\boldsymbol{\theta}}^P(x) = \sum_{a \in \mathcal{C}} O_a(x) \mathbf{1}[x_{i(a)} = b(a)] + r \sum_{\gamma \in \mathcal{B}} O_\gamma(x).$$

From a high entrance state, the trigger count equals the number of actual flips. From a low entrance state, the allowance r covers all actual edits, while the trigger count may add a nonnegative overcharge.

What is proved. The construction gives $\widehat{P}$ the structure required by Item 5:

1. It has three states in every layer and never rejects.
2. It has $O(n)$ transition layers.
3. At most two states are reachable at each checkpoint.
4. Each interval has $O(\ell)$ colliding layers.

Moreover,

$$|\mathcal{C}| + |\mathcal{B}| = O(n), \quad r = O(\ell).$$

The repair produces an accepted input, and the charge covers its edits:

$$P(\text{Repair}_{\boldsymbol{\theta}}^P(x)) = 1, \quad d_H(x, \text{Repair}_{\boldsymbol{\theta}}^P(x)) \leq \text{Cost}_{\boldsymbol{\theta}}^P(x).$$

Both procedures are fixed from P , independently of the candidate distribution. We prove their local cost–gain estimate and apply the gain-cost lemma of Item 3, obtaining

$$\mathbb{E}_{X \sim U}[\text{Cost}_{\boldsymbol{\theta}}^P(X)] = O(\ell).$$

The new repair therefore retains the required uniform bound, while its charge has the representation needed for the small-bias transfer.

5. **Transfer the cost bound to small bias.** Item 4 constructs a repair whose charge is a weighted sum of state visits, sometimes accompanied by a condition on the next input bit. We now bound these events under a small-bias distribution.

Bounding state visits. A δ -biased distribution D satisfies

$$|\mathbb{E}_D[f] - \mathbb{E}_U[f]| \leq \delta \|f\|_F,$$

where $\|f\|_F$ is the sum of the absolute values of the Fourier coefficients of f .

For the auxiliary program $\widehat{P}$ of Item 4, write

$$O_{t,u}(x) = \mathbf{1}[\widehat{S}_t(x) = u].$$

Section 8 constructs a function $G_{t,u}$ with the following properties:

- (a) $O_{t,u}(x) \leq G_{t,u}(x)$ for every x .
- (b) $\mathbb{E}_U[G_{t,u}] \leq 4 \mathbb{E}_U[O_{t,u}] + \frac{1}{2n}$.
- (c) $\|G_{t,u}\|_F \leq n^{O(\ell)}$.

We obtain this upper bound from two results.

The cover. Let $b = O(\ell)$ bound the number of colliding layers in each interval of $\widehat{P}$. Proposition 8.2 gives the following statement.

Proposition 8.2 (cover by products of acceptance functions). For every layer t and state u , there are functions Q_a , indexed by a set $\mathcal{A}_{t,u}$ of size at most $t + 1$, satisfying:

- (a) Each Q_a is the constant 1 or a product

$$Q_a(x) = \prod_{i=1}^{k_a} q_{a,i}(x),$$

where each $q_{a,i}$ is the acceptance function of a program with two live states, absorbing rejection, and at most b colliding free transitions. Within each product, the programs read pairwise disjoint blocks of input bits.

- (b) Their sum bounds the visit indicator pointwise:

$$O_{t,u}(x) \leq \sum_{a \in \mathcal{A}_{t,u}} Q_a(x).$$

- (c) Their total uniform expectation satisfies

$$\sum_{a \in \mathcal{A}_{t,u}} \mathbb{E}_U[Q_a] \leq 4 \mathbb{E}_U[O_{t,u}].$$

Upper bounds for the products. We next apply the following lemma to each Q_a .

Lemma 8.4 (product majorant). Let

$$Q(x) = \prod_i q_i(x),$$

where each q_i is the acceptance function of a program with two live states, absorbing rejection, and collision bound $b \geq 1$. Suppose the programs read pairwise disjoint blocks of input bits. For every $0 < \eta < 1/4$, there is a nonnegative function G , on the same coordinates, satisfying:

- (a) $G(x) \geq Q(x)$ for every x .
(b) $\mathbb{E}_U[G] \leq \mathbb{E}_U[Q] + \eta$.
(c) $\|G\|_F \leq 2^{O(b \log(1/\eta))}$.

There are $O(n)$ products in the cover. Applying the lemma to each one with a sufficiently small inverse-polynomial error η , and summing the resulting functions G_a , gives

$$G_{t,u}(x) = \sum_{a \in \mathcal{A}_{t,u}} G_a(x).$$

The cover supplies the factor four in its uniform expectation. The approximation errors sum to at most $1/(2n)$, and the Fourier norms sum to $n^{O(\ell)}$. These are precisely properties (a)–(c) above.

For a sufficiently large absolute constant C , choosing $\delta \leq n^{-C\ell}$ makes the Fourier error at most $1/(2n)$. Consequently,

$$\Pr_D[\widehat{S}_t(X) = u] \leq 4 \Pr_U[\widehat{S}_t(X) = u] + \frac{1}{n}.$$

From visits to repair cost. The charge representation of Item 4 lets us apply the visit bound term by term. This transfers the expected charge from uniform input to D , with at most a factor-eight increase. The additive errors total $O(\ell)$, since there are $O(n)$ terms with weights $O(\ell)$. Using the uniform bound of Item 4, we obtain

$$\mathbb{E}_D[\text{Cost}_\theta^P(X)] \leq 8 \mathbb{E}_U[\text{Cost}_\theta^P(X)] + O(\ell) = O(\ell).$$

Theorem 2.3 (expected repair cost under small bias). There are absolute constants $C_{\text{bias}}, C_{\text{edit}}$ with the following property. For every n, ℓ, P, F as in Theorem 2.2, there are a choice of interval thresholds θ and deterministic procedures

$$\text{Repair}_{\theta}^P : \{0, 1\}^F \rightarrow \{0, 1\}^F, \quad \text{Cost}_{\theta}^P : \{0, 1\}^F \rightarrow \mathbb{N}_0,$$

chosen independently of D , such that

$$\mathbb{E}_{X \sim D} [\text{Cost}_{\theta}^P(X)] \leq C_{\text{edit}} \ell$$

for every δ -biased distribution D on $\{0, 1\}^F$ with $\delta \leq n^{-C_{\text{bias}} \ell}$. For every candidate $x \in \{0, 1\}^F$, they satisfy

$$P(\text{Repair}_{\theta}^P(x)) = 1, \quad d_H(x, \text{Repair}_{\theta}^P(x)) \leq \text{Cost}_{\theta}^P(x),$$

and all fixed transition choices remain unchanged.

6. **Obtaining the required structure by restrictions.** We now specify the program class $\mathcal{K}_{3,\ell}$ used in Items 3–5. It consists of programs with three live states and absorbing rejection that admit a decomposition with the following properties:

- At designated layers, called *checkpoints*, including the start and end, at most two live states are reachable from the start state as the input varies.
- The transitions between consecutive checkpoints form an *interval*. Each interval contains at most ℓ colliding layers.

Every layer contains three live states. At each checkpoint, at most two of them are reachable from the original start state; inside an interval, all three may be reachable. The number of intervals may be large. Membership in $\mathcal{K}_{3,\ell}$ specifies only these structural conditions. To apply the repair bounds from Items 3–5, we separately require density at least $1/16$.

To obtain this structure, we use restrictions. Write $\mathcal{R}_{n,\ell}$ for the family of restrictions we construct. For every program P with three live states, absorbing rejection, and density at least $1/8$, some $\rho \in \mathcal{R}_{n,\ell}$ produces a residual of density greater than $1/16$ that belongs to $\mathcal{K}_{3,\ell}$ after the final transition described in the theorem below. Thus $\mathcal{R}_{n,\ell}$ contains restrictions, while $\mathcal{K}_{3,\ell}$ contains programs with the stated checkpoint and collision conditions. The family $\mathcal{R}_{n,\ell}$ depends only on n and ℓ ; the successful restriction may depend on P .

We will choose ℓ to balance the seed cost of the restrictions against that of the subsequent repairs. The following theorem states the restriction guarantee.

Theorem 2.4 (restrictions to the checkpoint class). For every $n \geq 4$ and integer $1 \leq \ell \leq n$, there is an explicit family of restrictions $\mathcal{R}_{n,\ell}$, depending only on n and ℓ , whose members can be generated using

$$O\left(\left(\log n + \frac{(\log n)^2}{\ell}\right) \log \log n\right)$$

seed bits. The implied constant is absolute.

For every length- n program P with three live states, absorbing rejection, and density at least $1/8$, some $\rho \in \mathcal{R}_{n,\ell}$ produces a residual $Q = P|_{\rho}$ with the following properties:

- Density.** The density of Q is greater than $1/16$.
- Checkpoints.** After appending a final fixed transition that sends every live state to one accepting state and reads no input, Q has checkpoints at its start and end. Each checkpoint has at most two reachable live states.
- Collisions.** Every interval between consecutive checkpoints contains at most ℓ colliding layers. This bound includes the added final transition.

Thus, after the final transition is appended, $Q \in \mathcal{K}_{3,\ell}$.

The generator. We now describe the generator $G_\wedge$ for programs with three live states and an absorbing rejecting state, at density threshold $1/8$. Fix $n \geq 4$ and an integer $1 \leq \ell \leq n$. Choose sufficiently large absolute constants $C_{\text{bias}}, C_{\text{edit}}$, as supplied by Theorem 2.3, and set

$$\delta = n^{-C_{\text{bias}}\ell}, \quad k = \lceil C_{\text{edit}}\ell \rceil, \quad h = \lceil \log_2(n+1) \rceil.$$

Fix the following explicit maps, whose seeds are separate parts of the generator's seed:

- $\text{Res}_{n,\ell} : \{0,1\}^{s_\rho} \rightarrow \mathcal{R}_{n,\ell}$ generates the restrictions of Theorem 2.4.
- $D_{n,\delta} : \{0,1\}^{s_D} \rightarrow \{0,1\}^n$ is a δ -biased source from Theorem 3.3.

For an h -bit word e , let $\text{Dec}(e)$ be its binary value if that value is at most n , and zero otherwise.

The seed has the form

$$z = (u_\rho, u_D, e_1, \dots, e_k) \in \{0,1\}^{s_\rho} \times \{0,1\}^{s_D} \times (\{0,1\}^h)^k.$$

Its three logical parts have the following roles:

- **Restriction seed** u_ρ (s_ρ bits). It selects $\rho = \text{Res}_{n,\ell}(u_\rho)$, determining which coordinates remain free and the values assigned to all other coordinates.
- **Center seed** u_D (s_D bits). It selects $x = D_{n,\delta}(u_D)$. The bits of x on the free coordinates form the center; its bits on assigned coordinates are ignored.
- **Edit fields** $(e_1, \dots, e_k)$ (kh bits). Each h -bit field specifies a coordinate to flip. If $\text{Dec}(e_j) = 0$, no bit is changed; otherwise, the indicated bit is flipped only if that coordinate is free. Thus the edit fields specify at most k flips, while assigned bits retain their restriction values.

All fields range independently, so every generated restriction is combined with every center and every edit list.

Algorithm 1. Generator $G_\wedge$ for programs with absorbing rejection

Fixed data: n, ℓ, k, h and the maps $\text{Res}_{n,\ell}, D_{n,\delta}$ above.

Input: a seed $z = (u_\rho, u_D, e_1, \dots, e_k)$, with $|u_\rho| = s_\rho$, $|u_D| = s_D$, and $|e_j| = h$ for $1 \leq j \leq k$.

Output: an n -bit word $G_\wedge(z)$.

```

1   $\rho \leftarrow \text{Res}_{n,\ell}(u_\rho)$  and  $x \leftarrow D_{n,\delta}(u_D)$ .
2  For each  $i \in \{1, \dots, n\}$ , set  $y_i \leftarrow x_i$  if  $\rho_i = *$ , and  $y_i \leftarrow \rho_i$  otherwise.
3  for  $j = 1, \dots, k$  do
4     $i \leftarrow \text{Dec}(e_j)$ . If  $1 \leq i \leq n$  and  $\rho_i = *$ , set  $y_i \leftarrow 1 - y_i$ .
5  end for
6  return  $y$ .
```

Varying the edit fields enumerates every change to at most k free coordinates: list the changed positions once and use zero in the remaining fields. Repeated indices are also allowed, and assigned coordinates remain unchanged. By Theorems 2.2, 2.3, and 2.4, as z ranges over all seeds, $G_\wedge(z)$ hits every length- n program with three live states, absorbing rejection, and density at least $1/8$. The seed length is

$$s_\rho + s_D + kh = O\left(\left(\log n + \frac{(\log n)^2}{\ell}\right) \log \log n + \ell \log n\right).$$

Choosing $\ell = \lceil \sqrt{\log n \log \log n} \rceil$ gives seed length $O((\log n)^{3/2} \sqrt{\log \log n})$. Sections 10.1 and 10.2 explain the equivalent balance in terms of the number of restriction steps.

Sections 2–9 concern this constant-density construction. Section 10 applies error reduction and accounts for the full dependence on ε .

3 Preliminaries and definitions

Sections 2, 6, 7, and 8 introduce repairs, restrictions, checkpoints, and the state-visit estimates. We now record the remaining program conventions, Fourier facts, and pseudorandom sources used in the proof. Throughout Sections 3–10, assume $n \geq 4$; Section 10 handles smaller lengths.

3.1 Notation and program conventions

We write $[n] = \{1, \dots, n\}$, and $\mathbf{1}[E]$ for the indicator of an event E . The uniform distribution on the coordinates under consideration is U , and an expectation without a subscript is taken under U . The density of a program P is

$$\mu(P) = \mathbb{E}_{x \sim U}[P(x)].$$

A generator also denotes its output distribution on a uniform seed. Logarithms have base two, except for $\ln$. In the seed estimates we write $N = \log_2 n$ and $g = \log_2 N$.

Programs and states. We use the program model of Definition 4.1. The input is read in the order $x_1, \dots, x_n$; any other fixed known order is handled by relabeling. Transition t reads x_t and leads from layer L_{t-1} to layer L_t . For a bit b ,

$$T_t^b : L_{t-1} \rightarrow L_t$$

is the map applied when $x_t = b$, and we call it the branch labeled b . The state reached in L_t on input x is $S_t(x)$. It is defined by $S_0(x) = s_0$, the designated start state, and

$$S_t(x) = T_t^{x_t}(S_{t-1}(x)).$$

A state $u \in L_t$ is *forward-reachable* if $S_t(x) = u$ for some input x . A singleton *endpoint event* is an event of the form $\{x : S_t(x) = s\}$ for a state $s \in L_t$.

Restrictions and fixed layers. A restriction is a string $\rho \in \{0, 1, *\}^n$. Coordinate i is free if $\rho_i = *$, and otherwise it is fixed to the bit ρ_i . The free set of ρ is

$$M(\rho) = \{i \in [n] : \rho_i = *\}.$$

The residual program $P|_\rho$ retains the original transition positions. At a fixed position t it applies the map $T_t^{\rho_t}$ whatever the input is; such a layer is called *fixed*, or *deterministic*. The following conventions apply to residual programs.

1. As an ordinary ROBP layer, a fixed layer has the same map on both branches, so its input bit is ignored.
2. Additional fixed layers introduce no input coordinates.
3. Within an interval, x_t denotes the free bit read at transition position t , when one is read.

A *reset* is a map that sends all live states to one live state. Further definitions for restrictions are given in Sections 5 and 9.

Collisions and completions. We label the three live states of each layer by $[3] = \{1, 2, 3\}$ and the rejecting state by $\perp$. Colliding layers are counted as in Section 5. For a branch map T , let

$$A(T) = \{u \in [3] : T(u) \neq \perp\}$$

be the set of live states that T does not reject. At a noncolliding transition, T is injective on $A(T)$. A *completion* of such a map T is a permutation π of $[3]$ that agrees with T wherever T does not reject:

$$\pi(u) = T(u) \quad \text{for every } u \in A(T).$$

A completion exists because a partial injection from $[3]$ to $[3]$ extends to a permutation. It assigns live successors to the states that T rejects and preserves every nonrejecting transition.

We use the following terms for programs.

1. A program is *total on the live states* if none of its transitions rejects, that is, $A(T_t^b) = [3]$ for every t and b . We shorten this to *total* when the state set is clear. A total three-state program may have any specified set of accepting final states.
2. A *permutation program* is a total program in which every T_t^b is a permutation of $[3]$.
3. Programs with absorbing rejection are also called programs with *sudden death* [GMR⁺12, Section 6]; the two terms refer to the same model here.

Definition 3.1 (suffix density). Let P be a read-once branching program with layers $L_0, \dots, L_d$, start state s_0 , and specified accepting final states. Fixed transitions are allowed. For $0 \leq t \leq d$, let $F_{>t}$ be the set of free coordinates read after layer L_t . The *suffix density* of a state $u \in L_t$ is

$$\mu_t^{\text{suffix}}(u) = \Pr_{z \sim U_{F_{>t}}} [\text{the computation of } P \text{ from } u \text{ in } L_t \text{ accepts on } z],$$

where $U_{F_{>t}}$ is uniform on $\{0, 1\}^{F_{>t}}$.

The computation follows the original transition maps, with all fixed choices unchanged and no repairs, and the state u need not be reachable from s_0 . Three values follow directly from the definition.

1. $\mu_0^{\text{suffix}}(s_0) = \mu(P)$.
2. $\mu_d^{\text{suffix}}(u) = \mathbf{1}[u \text{ is accepting}]$ for every $u \in L_d$.
3. $\mu_t^{\text{suffix}}(\perp) = 0$ for every t , when $\perp$ is an absorbing rejecting state.

The subscript always denotes the layer index, so at a checkpoint t_j we write $\mu_{t_j}^{\text{suffix}}(u)$. For a singleton endpoint event $\{x : S_t(x) = s\}$, we apply the definition to the prefix program with layers $L_0, \dots, L_t$ whose only accepting final state is s .

Distance. For words x, y on the same coordinates, the Hamming distance is

$$d_H(x, y) = |\{i : x_i \neq y_i\}|.$$

For a nonempty set A of words, $d_H(x, A) = \min_{y \in A} d_H(x, y)$. In a residual program the words are assignments to the free coordinates, so these distances count only free coordinates.

3.2 Fourier analysis and approximation

For $S \subseteq [n]$, define the parity character $\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$. The Fourier coefficient of a function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ at S is $\hat{f}(S) = \mathbb{E}_U[f\chi_S]$. These characters form an orthonormal basis, giving the expansion

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x).$$

We use the Fourier ℓ_1 -norm

$$\|f\|_F = \sum_{S \subseteq [n]} |\hat{f}(S)|.$$

The sum includes the constant coefficient $\hat{f}(\emptyset) = \mathbb{E}_U f$. Parseval's identity gives $\sum_S \hat{f}(S)^2 = \mathbb{E}_U f^2$. The *Fourier mass at level k* is $\sum_{|S|=k} |\hat{f}(S)|$. These definitions also apply to functions on any subset of the input coordinates.

The norm satisfies $\|fh\|_F \leq \|f\|_F \|h\|_F$, with equality when f and h depend on disjoint coordinate sets. An invertible affine change of variables over $\mathbb{F}_2$ preserves uniform measure and permutes the characters up to

sign, so it preserves the Fourier norm. An *affine event* is the solution set of parity equations, possibly with nonzero right-hand sides. Its indicator has norm one when the event is nonempty, and zero otherwise. A *subcube* is the special case obtained by fixing input coordinates. In particular, every literal indicator has norm one.

We compare functions pointwise. A function G is a *majorant* of f if $G(x) \geq f(x)$ for every x . A majorant need not approximate the mean of f . Sections 8.1–8.4 prove the mean and Fourier norm bounds needed for the state-visit majorants.

3.3 Pseudorandom sources

Small-bias distributions supply the centers and both strings used by each restriction: one selects the free coordinates, and the other supplies the assigned bits. We first state the source and its Fourier guarantee, then explain why it is approximately uniform on small coordinate sets.

Definition 3.2 (small bias). A distribution D on $\{0, 1\}^n$ is δ -biased if the parity of every nonempty subset of coordinates is approximately uniform. More precisely, for every nonempty $S \subseteq [n]$,

$$|\mathbb{E}_{x \sim D} \chi_S(x)| \leq \delta.$$

For a δ -biased distribution, Fourier expansion gives

$$\begin{aligned} |\mathbb{E}_D f - \mathbb{E}_U f| &\leq \delta \sum_{\emptyset \neq S \subseteq [n]} |\hat{f}(S)| \\ &\leq \delta \|f\|_F. \end{aligned} \tag{3.1}$$

We use this estimate with Fourier majorants to bound state-visit probabilities under the center source.

Theorem 3.3 (Naor–Naor [NN93]). *For every $0 < \delta < 1$, there is an explicit δ -biased distribution on $\{0, 1\}^n$ that can be sampled using $O(\log n + \log(1/\delta))$ seed bits. Its output is computable in time polynomial in n and $\log(1/\delta)$.*

Total variation. For distributions A, B on a finite set Ω , define

$$d_{\text{TV}}(A, B) = \frac{1}{2} \sum_{x \in \Omega} |A(x) - B(x)|.$$

This is the largest absolute difference in the probability of an event. Equivalently, it is half the largest absolute expectation difference over functions $f : \Omega \rightarrow [-1, 1]$. Applying a deterministic map cannot increase total variation distance. For independent products, the distance is at most the sum of the distances between corresponding factors.

For the masks, we need approximate uniformity on every small set of coordinates. We use the bounded-test convention of Forbes–Kelley [FK18, Definition 3.4].

Definition 3.4 (almost k -wise independence). A distribution Y on $\{0, 1\}^n$ is γ -almost k -wise independent if every function $f : \{0, 1\}^n \rightarrow [-1, 1]$ depending on at most k coordinates satisfies

$$|\mathbb{E}_Y f - \mathbb{E}_U f| \leq \gamma.$$

When this difference is always zero, the distribution is k -wise independent.

Lemma 3.5 (local uniformity from small bias). *Let D be a δ -biased distribution on $\{0, 1\}^n$, and let $J \subseteq [n]$ have m coordinates. Write D_J for its marginal on J , and U_J for the uniform distribution there. Then*

$$d_{\text{TV}}(D_J, U_J) \leq \frac{\delta}{2} \sqrt{2^m - 1}. \tag{3.2}$$

In particular, D is γ -almost k -wise independent for $\gamma = 2^{k/2} \delta$.

Proof. For $z \in \{0, 1\}^J$, set $h(z) = 2^m \Pr[D_J = z]$. This is the density of D_J relative to U_J . Its constant Fourier coefficient is one, and every nonconstant coefficient has absolute value at most δ . Cauchy–Schwarz and Parseval therefore give

$$\begin{aligned} d_{\text{TV}}(D_J, U_J) &= \frac{1}{2} \mathbb{E}_{U_J} |h - 1| \\ &\leq \frac{1}{2} (\mathbb{E}_{U_J} (h - 1)^2)^{1/2} \\ &= \frac{1}{2} \left(\sum_{\emptyset \neq S \subseteq J} |\hat{h}(S)|^2 \right)^{1/2} \\ &\leq \frac{\delta}{2} \sqrt{2^m - 1}. \end{aligned}$$

If f takes values in $[-1, 1]$ and depends on at most k coordinates, its expectation changes by at most twice this total variation distance. Thus $|\mathbb{E}_D f - \mathbb{E}_U f| \leq 2^{k/2} \delta$, as required. $\square$

Consequently, the restriction masks require no separate generator. A sufficiently small-bias source provides the approximate independence used in the density analysis and the total-variation guarantee used for the checkpoint analysis.

The prefix source. The width-three HSG is already stated in Theorem 4.4. At density threshold $0 < \alpha \leq 1$, it uses $O(\log(n/\alpha))$ seed bits. In Corollary 4.5 we take $\alpha = 1/(16n)$, so this part of the construction costs $O(\log n)$ seed bits.

3.4 A one-step restriction source

Pseudorandom restrictions. We write a restriction of n coordinates as a string $\rho \in \{0, 1, *\}^n$, where $\rho_i = *$ means that coordinate i is free. For $J \subseteq [n]$, the string $\rho_J \in \{0, 1, *\}^J$ is the projection of ρ to the coordinates in J . Let $\mathcal{I}$ be the distribution on $\{0, 1, *\}$ that assigns probability $1/2$ to $*$ and probability $1/4$ to each of 0 and 1, and let $\mathcal{I}^J$ be the distribution on $\{0, 1, *\}^J$ with independent coordinates drawn from $\mathcal{I}$.

For strings $z, y \in \{0, 1\}^n$, define $\rho(z, y)$ by $\rho(z, y)_i = *$ when $y_i = 1$ and $\rho(z, y)_i = z_i$ when $y_i = 0$. Let $u \in \{0, 1\}^n$ and write $\oplus$ and $\wedge$ for coordinatewise XOR and AND. The string $z \oplus (y \wedge u)$ equals z_i on every fixed coordinate and $z_i \oplus u_i$ on every free coordinate. For a uniform U it is therefore a uniform completion of $\rho(z, y)$, and every program Q satisfies

$$\mathbb{E}_U [Q(z \oplus (y \wedge U))] = \mu(Q|_{\rho(z, y)}). \quad (3.3)$$

In words, the acceptance probability of Q on the completed string is the density of the restricted program.

Forbes and Kelley bound the change in acceptance probability when z and y are pseudorandom. We state their bound for width four, with the Fourier-growth bound of Chattopadhyay, Hatami, Reingold and Tal substituted for the level- k Fourier mass.

Theorem 3.6 ([FK18, Lemma 7.2] with [CHRT18, Theorem 2]). *There is an absolute constant $C_1 \geq 1$ such that the following holds. Let $n \geq 4$, $1 \leq k \leq n$ and $0 < \delta, \gamma < 1$. Let Z be δ -biased, let Y be γ -almost k -wise independent, and let U be uniform, all three independent. Then every width-four ROBP Q on n bits satisfies*

$$\left| \mathbb{E}[Q(Z \oplus (Y \wedge U))] - \mu(Q) \right| \leq C_1 n \left(\sqrt{\delta} (C_1 \log_2 n)^{4k} + 2^{-k/2} + \sqrt{\gamma} \right).$$

The constant C_1 absorbs the sum over Fourier levels $1, \dots, k$ and the passage from the matrix form of [FK18] to the acceptance probability. A program with three live states and absorbing rejection is a width-four ROBP once the rejecting state is counted, so the theorem applies to it.

The next lemma is the form in which we use this estimate. Its density guarantee is Theorem 3.6 with a choice of parameters. We add a local-uniformity guarantee, which the checkpoint analysis of Section 9 requires, and obtain both guarantees from the same pair of small-bias strings.

Lemma 3.7 (restriction source). *Let $n \geq 4$, $0 < \eta, \zeta < 1/4$ and $1 \leq k_0 \leq n$. There is an explicit distribution over restrictions ρ of n coordinates, with seed length*

$$O\left(\log \frac{n}{\eta} \log \log n + k_0 + \log \frac{1}{\zeta}\right), \quad (3.4)$$

and the following two properties.

1. **Density preservation.** *Every program Q with three live states and absorbing rejection satisfies $|\mathbb{E}_\rho[\mu(Q|\rho)] - \mu(Q)| \leq \eta$.*
2. **Local uniformity.** *Every $J \subseteq [n]$ with $|J| \leq k_0$ satisfies*

$$d_{\text{TV}}(\rho_J, \mathcal{I}^J) \leq \zeta.$$

In words, on any set of at most k_0 coordinates the restriction is ζ -close to one that independently leaves each coordinate free with probability $1/2$ and otherwise assigns it a uniform bit.

The same source parameters apply after projection to any subset of the original coordinates, including the free coordinates left by earlier restrictions.

Proof. Let

$$k = \left\lceil 2 \log_2 \frac{4C_1 n}{\eta} \right\rceil.$$

Assume first that $k \leq n$. Sample two independent δ -biased strings Z, Y and output $\rho = \rho(Z, Y)$, where $\delta = 2^{-j}$ and j is the smallest positive integer for which the following two conditions hold.

- (a) $\delta \leq (\eta/(4C_1 n))^2 (C_1 \log_2 n)^{-8k}$.
- (b) $\delta \leq \zeta 2^{-k_0/2}$.

The first condition gives density preservation and the second gives local uniformity.

For density preservation, Lemma 3.5 shows that Y is γ -almost k -wise independent with $\gamma = 2^{k/2}\delta$, and the first condition gives $\gamma < 1$. Taking the expectation of (3.3) over Z, Y identifies $\mathbb{E}_\rho[\mu(Q|\rho)]$ with $\mathbb{E}[Q(Z \oplus (Y \wedge U))]$, so Theorem 3.6 bounds the density error by the sum of three terms. The first condition on δ gives

$$C_1 n \sqrt{\delta} (C_1 \log_2 n)^{4k} \leq \frac{\eta}{4}.$$

The choice of k gives

$$C_1 n 2^{-k/2} \leq \frac{\eta}{4}.$$

The third term is $C_1 n \sqrt{\gamma} = C_1 n 2^{k/4} \sqrt{\delta}$, which is at most the first term because $2^{k/4} \leq (C_1 \log_2 n)^{4k}$. The density error is therefore at most $3\eta/4$.

For local uniformity, fix a set J of $m \leq k_0$ coordinates. By Lemma 3.5, each of Z_J and Y_J is within total variation distance $(\delta/2)\sqrt{2^m - 1}$ of a uniform string. The two strings are independent, so the pair (Z_J, Y_J) is within distance

$$\delta \sqrt{2^m - 1} \leq \delta 2^{k_0/2} \leq \zeta$$

of a pair of independent uniform strings. The restriction on J is a function of this pair, and applying a function does not increase total variation distance. For a uniform pair (Z_J, Y_J) , the string $\rho(Z, Y)_J$ is distributed as $\mathcal{I}^J$, so $d_{\text{TV}}(\rho_J, \mathcal{I}^J) \leq \zeta$.

For the seed length, the choice of the smallest j means that 2δ violates one of the two conditions, so δ is at least half of the smaller of the two right-hand sides. Since $k = O(\log(n/\eta))$, this gives

$$\log \frac{1}{\delta} = O\left(\log \frac{n}{\eta} \log \log n + k_0 + \log \frac{1}{\zeta}\right).$$

Theorem 3.3 samples each of Z, Y with $O(\log n + \log(1/\delta))$ seed bits, which gives (3.4). If $k > n$, take Z, Y uniform instead. Both properties then hold exactly, and the seed length $2n < 2k$ is within (3.4).

Finally, a program left by earlier restrictions is a width-four ROBP on the original n coordinates whose transitions at the fixed coordinates ignore the input bit. Theorem 3.6 applies to it, and the choices of ρ on the ignored coordinates do not affect $\mu(Q|_\rho)$. Property 2 already covers every set of at most k_0 coordinates. $\square$

4 Reduction to programs with absorbing rejection

Our goal in this section is to reduce the width-four hitting problem at density threshold $1/4$ to hitting programs with absorbing rejection at threshold $1/8$. We apply Doron–Hoza’s width reduction at these fixed thresholds, so this step has no dependence on the final parameter ε . After constructing the resulting constant-density width-four HSG, we apply their separate HSG error-reduction lemma in Section 10 to handle every $\varepsilon \in (0, 1]$.

The width reduction constructs a family $\mathcal{P}$ of candidate prefixes. The family depends only on n and has the following guarantee. For every width-four program B of density at least $1/4$, there are a prefix $p \in \mathcal{P}$, of some length b , and a program Q on the remaining $n - b$ coordinates such that:

1. **Restricted form.** Q has three live states and one absorbing rejecting state in each layer. It accepts exactly when its computation avoids the rejecting state in every layer.
2. **Preserved density.**

$$\Pr_{z \sim U_{n-b}} [Q(z) = 1] > \frac{1}{8}.$$

3. **Acceptance in the original program.** For every continuation z ,

$$Q(z) = 1 \implies B(p \circ z) = 1,$$

where $p \circ z$ denotes concatenation.

Doron–Hoza construct $\mathcal{P}$ using an HSG for ordinary width-three programs at density threshold $1/(16n)$. Our remaining task is to construct an HSG for programs of the form in item 1, at density threshold $1/8$. For the prefix p guaranteed above, this second HSG supplies a continuation accepted by Q . Item 3 then ensures that the concatenated input is accepted by B . We therefore obtain a width-four hitting set by combining all candidate prefixes with all generated continuations. We first define the class of programs Q , then state the results that implement this reduction.

Definition 4.1 (programs with three live states and absorbing rejection). A length- n program of this type has layers $L_0, \dots, L_n$. Each layer L_i consists of exactly three *live states* and one rejecting state:

$$L_i = L_i^{\text{live}} \cup \{\perp\}, \quad |L_i^{\text{live}}| = 3.$$

The live states need not be reachable from the designated start state. The program starts at a designated state in L_0^{live} . At step i , it reads x_i and applies the corresponding transition map $T_i^{x_i} : L_{i-1} \rightarrow L_i$. The state $\perp$ is absorbing, so $T_i^0(\perp) = T_i^1(\perp) = \perp$. Every state in L_n^{live} accepts, and $\perp$ rejects.

Programs with fewer live states can be represented in this form by adding unreachable live states. Leave all original transitions unchanged and send every outgoing transition from an added state to rejection. This preserves the accepted function, its density, and the set of colliding layers.

Thus, the total width is four, and an input accepts precisely when its computation never reaches $\perp$. These programs are equivalent to Doron–Hoza’s width-three $\wedge$ -programs [DH25, Definition 1.9]. A $\wedge$ -program has an ordinary branching-program transition structure. Each edge also has an accepting or rejecting designation, separate from its input label 0 or 1. The program accepts if every traversed edge is designated accepting; there is no final-state acceptance condition. The equivalence preserves the accepted inputs:

- From a width-three $\wedge$ -program, add an absorbing state $\perp$ and redirect each rejecting edge to it. Keep the accepting edges and make every final live state accepting.
- In the reverse direction, remove $\perp$. Replace each edge into $\perp$ by a rejecting edge to an arbitrary live state in the next layer. Designate all other edges accepting. The destination of a rejecting edge cannot affect whether all traversed edges are accepting.

A $\vee$ -program uses the same edge designations but accepts if at least one traversed edge is designated accepting. Doron–Hoza use this model for the prefix task. We import their two reduction lemmas, specialized to width three.

Lemma 4.2 (hitting $\vee$ -programs [DH25, Lemma 4.3]). *For every length- n , width-three $\vee$ -program F , there is an ordinary length- n , width-three ROBP G such that $G(x) \leq F(x)$ for every x and $\mu(G) \geq \mu(F)/n$. Consequently, a hitting set for ordinary width-three ROBPs of density at least η/n also hits every width-three $\vee$ -program of density at least η .*

To state the second lemma, let $x_{i:j}$ denote coordinates $i, \dots, j$ of a string x , with an empty string when $i > j$. Write $\circ$ for concatenation.

Lemma 4.3 (reduction from width four [DH25, Lemma 4.5]). *Let $\eta_\vee, \eta_\wedge > 0$. Suppose $H_\vee, H_\wedge \subseteq \{0, 1\}^n$ satisfy:*

1. $H_\vee$ hits every length- n , width-three $\vee$ -program of density greater than $\eta_\vee$.
2. $H_\wedge$ hits every length- n , width-three $\wedge$ -program of density greater than $\eta_\wedge$.

Then

$$H_4 = \{0^a \circ x_{a+1:b} \circ y_{b+1:n} : 0 \leq a \leq b \leq n, x \in H_\vee, y \in H_\wedge\}$$

hits every ordinary length- n , width-four ROBP B satisfying

$$\mu(B) > \eta_\vee + \eta_\wedge.$$

Moreover,

$$|H_4| \leq (n+1)^2 |H_\vee| |H_\wedge|.$$

By Lemma 4.2, an ordinary width-three HSG can supply $H_\vee$. We use Krak’s optimal construction.

Theorem 4.4 (width-three hitting [Krak26, Theorem 1.1]). *For every $n \geq 2$ and $0 < \alpha \leq 1$, there is an explicit HSG for ordered width-three ROBPs of density at least α , with seed length*

$$O\left(\log \frac{n}{\alpha}\right).$$

The implied constant is absolute.

Corollary 4.5 (constant-density reduction). *If programs with three live states and absorbing rejection at density threshold $1/8$ have an HSG with seed length $r_\wedge(n)$, then ordered width-four ROBPs of length n and density at least $1/4$ have an HSG with seed length*

$$r_\wedge(n) + O(\log n). \tag{4.1}$$

The implied constant is absolute.

Proof. Let $G_\wedge$ be the assumed HSG, and let $H_\wedge$ be its output set. Use Theorem 4.4 at threshold $\alpha = 1/(16n)$ to obtain a generator G_3 , with output set H_3 and seed length r_3 . Lemma 4.2 implies that H_3 hits width-three $\vee$ -programs of density at least $1/16$. The equivalence following Definition 4.1 implies that $H_\wedge$ hits width-three $\wedge$ -programs of density at least $1/8$.

The full generator uses a seed for each of G_3 and $G_\wedge$, together with two strings of length $\lceil \log_2(n+1) \rceil$. Decode the latter as integers a, b ; if they do not satisfy $0 \leq a \leq b \leq n$, replace both by zero. For source seeds u, v , define

$$G_0(a, b, u, v) = 0^a \circ G_3(u)_{a+1:b} \circ G_\wedge(v)_{b+1:n}.$$

Every valid pair a, b occurs, so the output set is exactly the set in Lemma 4.3, taking $H_V = H_3$ and thresholds $\eta_V = 1/16$ and $\eta_\wedge = 1/8$. Every target B of density at least $1/4$ satisfies $\mu(B) \geq 1/4 > \eta_V + \eta_\wedge$. Lemma 4.3 therefore supplies an output of G_0 accepted by B .

The seed length of G_0 is

$$r_\wedge(n) + r_3 + 2\lceil \log_2(n+1) \rceil.$$

Theorem 4.4 gives

$$r_3 = O\left(\log \frac{n}{1/(16n)}\right) = O(\log n).$$

This proves (4.1). When a suffix program uses only coordinates $b+1, \dots, n$, we regard it as an n -bit program that ignores the preceding coordinates. The splice therefore keeps all input coordinates in their original positions. The construction is explicit whenever $G_\wedge$ is explicit. $\square$

Corollary 4.5 leaves one task: construct an HSG for programs with three live states and absorbing rejection at density threshold $1/8$.

5 Collisions and the total repair bound

In this section we bound the distance from an arbitrary input to an accepting input in terms of the total number of colliding layers. We therefore conclude that if this distance is small, every input is a useful center: it can be made accepting by changing few coordinates. This elementary repair argument will also supply a local repair in Section 6.4.

We allow *fixed layers*, each of which applies a single transition map and reads no input bit. Such layers arise when an input coordinate is assigned a value. A free layer instead reads the next free bit and applies one of two transition maps. Repairs change only free input coordinates; they leave all fixed transition choices unchanged.

A layer is *colliding* if an available transition map sends two distinct live states to the same live state in the next layer. At a free layer, either of its two maps can make the layer colliding. At a fixed layer, only its assigned map is considered. We count each colliding layer once, and transitions into rejection do not count as collisions. Write c for the total number of colliding layers.

Before any collision or possible rejection, each available map sends the reachable live states injectively to live states. Their number therefore cannot decrease. With three live states, it can grow at most twice. This observation will let us reach any state in such a segment with at most two changes. We end a repair stage at the next collision or possible rejection, then restart from the state reached by the repaired prefix.

Lemma 5.1 (total repair bound). *Let P have three live states, absorbing rejection, n free input coordinates, and at most c colliding layers. Suppose its density is $\mu = \mu(P) > 0$. Fixed layers are allowed. There is a deterministic procedure $\text{repair}_P^{\text{Total}} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that, for every x , the output $y = \text{repair}_P^{\text{Total}}(x)$ satisfies $P(y) = 1$ and*

$$d_H(x, y) \leq 3c + C_{\text{rad}} \log_2(1/\mu), \quad (5.1)$$

where $C_{\text{rad}} = 3/\log_2(8/7)$. In particular, the number of changed coordinates is $O(c + \log(1/\mu))$.

Proof. Write $L_0, \dots, L_m$ for the state layers. Step i is the transition from layer L_{i-1} to layer L_i . At a free step, the next input bit selects one of the two transition maps. At a fixed step, the computation applies its single map without reading an input bit. Let $f(i)$ be the number of free bits read while moving from L_0 to L_i , that is, during steps $1, \dots, i$.

Write $P_{s,t}(u; z)$ for the state reached from $u \in L_s$ after steps $s + 1, \dots, t$, using $z \in \{0, 1\}^{f(t)-f(s)}$ on their free coordinates. Fixed transitions remain unchanged. The corresponding block of x is $x_{f(s)+1:f(t)}$, possibly empty. We use the suffix densities μ_i^{suffix} from Definition 3.1. Here U_k denotes the uniform distribution on $\{0, 1\}^k$.

The procedure maintains a word y and an edited prefix ending at state u in layer L_s . One iteration is one repair stage. Resolve $\arg \max$ ties by a fixed order on states, and $\arg \min$ ties lexicographically.

Algorithm 2. $\text{repair}_P^{\text{Total}}(x)$

Fixed data: the program P , its start state u_0 , and the suffix densities μ_i^{suffix} .

Input: a candidate word $x \in \{0, 1\}^n$.

Output: an accepting word $y \in \{0, 1\}^n$.

```

1   $(y, s, u) \leftarrow (x, 0, u_0)$ .
2  while  $\mu_s^{\text{suffix}}(u) < 1$  do
3    Let  $t$  be the least  $j > s$  such that step  $j$  is colliding or some continuation from  $u$  rejects at step  $j$ .
4     $V \leftarrow \{P_{s,t}(u; z) : z \in \{0, 1\}^{f(t)-f(s)}\} \setminus \{\perp\}$ .
5     $v \leftarrow \arg \max_{w \in V} \mu_t^{\text{suffix}}(w)$ .
6     $z \leftarrow \arg \min_{z: P_{s,t}(u; z) = v} d_H(z, x_{f(s)+1:f(t)})$ .
7     $y_{f(s)+1:f(t)} \leftarrow z$ .
8     $(s, u) \leftarrow (t, v)$ .
9  end while
10 return  $y$ .
```

How one iteration works. The goal of one iteration is to extend the repaired prefix, using at most three additional bit changes, to a live state whose remaining density is at least as large as before. The iteration need not produce a complete accepting input; it must leave us in a suitable state from which to continue the repair.

Fix the original candidate input x . The procedure maintains an edited word y . At the beginning of an iteration, the prefix already processed reaches a live state u in layer L_s . All later coordinates of y still agree with x , and the remaining density $\mu_s^{\text{suffix}}(u)$ is positive. If $\mu_s^{\text{suffix}}(u) = 1$, every continuation from u accepts, so we keep the remaining bits and stop. Otherwise, we choose the endpoint $t > s$ of this iteration: the first colliding step or the first step at which some continuation from u can reject.

The iteration modifies only the free coordinates read between layers L_s and L_t . The objects V , v , and z specify which states this block can reach, which state we want to reach, and which bits we use to reach it.

V is the set of available live endpoints. Starting from u , consider every assignment to the free bits read in steps $s + 1, \dots, t$. Collect all live states that these assignments can reach in layer L_t . This is V . Thus, $w \in V$ means that some choice of this input block takes the computation from u to w without rejection. The set V depends on the program and the current state u ; its definition considers all possible assignments to the block.

v is the endpoint selected for the repaired computation. Among the states in V , we choose one with the largest remaining density:

$$v \in \arg \max_{w \in V} \mu_t^{\text{suffix}}(w).$$

Choosing v this way makes the remaining computation as likely to accept as possible among the available endpoints. The averaging argument below gives

$$\mu_t^{\text{suffix}}(v) \geq \mu_s^{\text{suffix}}(u).$$

If rejection is possible at step t , the stronger bound is

$$\mu_t^{\text{suffix}}(v) \geq \frac{8}{7} \mu_s^{\text{suffix}}(u).$$

z is the replacement input block that reaches v . After selecting the target state v , we choose how to reach it. Among all block assignments that take u to v , choose one closest to the original block:

$$z \in \arg \min_{b: P_{s,t}(u;b)=v} d_H(b, x_{f(s)+1:f(t)}).$$

Fixed transitions remain unchanged. The reachability argument below proves that some such block requires at most three changes. Because z minimizes the number of changes, it also satisfies this bound.

Finally, we replace the corresponding block of y by z . The extended prefix now reaches v in layer L_t , and all later bits still agree with x . We update (s, u) to (t, v) and begin the next iteration from that state.

We now verify these claims and bound the total number of changes.

1. Reaching a state before step t . Consider an iteration with $0 < \mu_s^{\text{suffix}}(u) < 1$. The stopping step t exists because some continuation from u rejects. Before t , no collision or rejection is possible. For $s \leq i < t$, let R_i be the set of states reachable from u in layer L_i . At a free step $j < t$, let T_j^b be the transition map for bit b . Both maps are injective on R_{j-1} , and

$$R_j = T_j^0(R_{j-1}) \cup T_j^1(R_{j-1}).$$

Each image therefore has size $|R_{j-1}|$. At a fixed step $j < t$, the single map is a bijection from R_{j-1} onto R_j . Consequently,

$$1 = |R_s| \leq |R_{s+1}| \leq \dots \leq |R_{t-1}| \leq 3.$$

Call j a *growth step* if $|R_j| > |R_{j-1}|$. There are at most two growth steps, both free. At any other free step, each of the two maps sends R_{j-1} onto all of R_j .

Fix all free bits in steps $s+1, \dots, t-1$ except those at growth steps. We claim that every $w \in R_{t-1}$ is still reachable by choosing the bits at the growth steps. Set $w_{t-1} = w$ and choose predecessors $w_{t-2}, \dots, w_s$ backward. At a free step without growth, take a predecessor under the already fixed bit; its map reaches every state of the next reachable set. At a growth step, take any predecessor and choose its transition bit. At a fixed layer, take the predecessor under its single map. The path ends at u , since $R_s = \{u\}$.

This proves both facts we need:

- Every $w \in R_{t-1}$ can be reached by changing at most two bits of x : keep all bits outside the growth steps equal to those of x .
- If the free bits read in steps $s+1, \dots, t-1$ are independent and uniform, every $w \in R_{t-1}$ is reached with probability at least $1/4$. Indeed, condition on all bits outside the growth steps. At least one assignment to the remaining at most two bits reaches w , so the conditional probability is at least $1/4$.

2. Completing one stage. Start from u and choose the free bits in steps $s+1, \dots, t$ independently and uniformly. Let r be the probability of rejection at step t , and let $\pi(w)$ be the probability of reaching a live state $w \in L_t$. No rejection is possible earlier, so $\sum_w \pi(w) = 1 - r$. The set V in line 4 is nonempty because $\mu_s^{\text{suffix}}(u) > 0$. Let v be the state selected in line 5. By conditioning on the state reached in layer L_t ,

$$\begin{aligned} \mu_s^{\text{suffix}}(u) &= \sum_{w: \pi(w) > 0} \pi(w) \mu_t^{\text{suffix}}(w) \\ &\leq \sum_{w: \pi(w) > 0} \pi(w) \mu_t^{\text{suffix}}(v) \\ &= \mu_t^{\text{suffix}}(v) \sum_{w: \pi(w) > 0} \pi(w) \\ &= (1 - r) \mu_t^{\text{suffix}}(v). \end{aligned}$$

There are two cases.

- *Rejection is possible at step t .* A state in R_{t-1} has a transition into rejection. It has probability at least $1/4$, and that transition is taken with probability at least $1/2$. Thus $r \geq 1/8$ and $\mu_t^{\text{suffix}}(v) \geq (8/7)\mu_s^{\text{suffix}}(u)$. At a fixed step the rejecting transition is certain, so the same bound holds.
- *Rejection is impossible at step t .* Then $r = 0$, so $\mu_t^{\text{suffix}}(v) \geq \mu_s^{\text{suffix}}(u)$. By the choice of t , this step is colliding.

To bound the minimum in line 6, take any transition into v from a state in R_{t-1} . Part 1 reaches that state with at most two edits. If step t is free, selecting the transition into v costs at most one more edit; if it is fixed, it costs none. This argument only needs one predecessor of v , so it also applies when step t is colliding. A block reaching v with at most three edits therefore exists, and the minimizing block in line 6 does at least as well.

3. Iterating and counting the edits. The loop maintains the following invariant: $y_{1:f(s)}$ reaches u , all later bits agree with x , and $\mu_s^{\text{suffix}}(u) \geq \mu > 0$. It holds initially. Parts 1–2 justify lines 3–6, and the updates in lines 7–8 preserve the invariant. Each iteration increases s , and every final live state has density one. Thus the loop stops with $\mu_s^{\text{suffix}}(u) = 1$, and the unchanged suffix accepts.

Let a be the number of stages at whose endpoint rejection is possible, and b the number of other stages. A stage that is both colliding and allows rejection is counted in a . The other stages end at distinct colliding layers, so $b \leq c$. Each of the a stages increases remaining density by a factor of at least $8/7$. The density starts at μ and never exceeds one, so $\mu(8/7)^a \leq 1$ and $a \leq \log_{8/7}(1/\mu)$. Each stage changes at most three bits, and the stages act on disjoint blocks. Therefore,

$$d_H(x, y) \leq 3(a + b) \leq 3c + 3\log_{8/7}(1/\mu) = 3c + C_{\text{rad}} \log_2(1/\mu). \quad \square$$

For $\mu(P) \geq \varepsilon$, the bound above is at most $C(c + \log(1/\varepsilon))$ for an absolute constant C , proving Lemma 2.1.

For the later local repair, acceptance alone is not enough: the exit must have sufficiently large suffix density. The same argument works when terminal states carry rewards.

Corollary 5.2 (repair toward a terminal reward). *Let P be as in Lemma 5.1, and give each final live state v a reward $\omega(v) \in [0, 1]$, with reward zero at rejection. Suppose the expected reward under uniform free bits is $a > 0$. There is a deterministic procedure $\text{repair}_{P, \omega}^{\text{Total}}$ that changes every input into one ending at a live state of reward at least a , using at most*

$$3c + C_{\text{rad}} \log_2(1/a) + 2 \quad (5.2)$$

coordinate changes. All fixed transition choices remain unchanged.

Proof. Use the notation of Lemma 5.1: $L_0, \dots, L_m$ are the state layers, $f(i)$ counts the free bits read through layer L_i , and $P_{s,t}(u; z)$ is the state reached from $u \in L_s$ using the block z . Put $\omega(\perp) = 0$, and define the expected remaining reward by

$$h_i(w) := \mathbb{E}_{Z \sim U_{n-f(i)}} [\omega(P_{i,m}(w; Z))],$$

where U_k is uniform on $\{0, 1\}^k$. Thus

$$0 \leq h_i(w) \leq 1, \quad h_0(u_0) = a, \quad h_m(v) = \omega(v).$$

Fix a candidate $x \in \{0, 1\}^n$. We adapt the stage construction of Lemma 5.1, maintaining a repaired prefix that reaches a live state $u \in L_s$, with

$$h_s(u) \geq a > 0,$$

while all unprocessed coordinates still agree with x . Initially, $s = 0$ and $u = u_0$.

If a collision or possible rejection remains, let $t > s$ be the first such step, exactly as in Lemma 5.1. Let V be the live states reachable from u at layer L_t . Since $h_s(u) > 0$, some continuation has positive terminal reward, so $V \neq \emptyset$. Choose

$$v \in \arg \max_{w \in V} h_t(w).$$

For a uniform block $Z \sim U_{f(t)-f(s)}$, write

$$r := \Pr_{Z \sim U_{f(t)-f(s)}} [P_{s,t}(u; Z) = \perp],$$

$$\pi(w) := \Pr_{Z \sim U_{f(t)-f(s)}} [P_{s,t}(u; Z) = w].$$

Conditioning on the state reached at layer L_t , and using the independence of the remaining uniform bits, gives

$$h_s(u) = \sum_{w \in V} \pi(w) h_t(w)$$

$$\leq h_t(v) \sum_{w \in V} \pi(w) = (1 - r) h_t(v).$$

In particular, $r < 1$ and $h_t(v) \geq h_s(u)$.

We now use two structural facts proved in Lemma 5.1. Every state in V can be reached by changing at most three bits of the current block. Moreover, if rejection is possible at step t , then $r \geq 1/8$. These facts depend only on the transition maps and the absence of earlier collisions or rejection, so they apply unchanged here. Consequently,

$$h_t(v) \geq \begin{cases} \frac{8}{7} h_s(u), & \text{if rejection is possible at step } t, \\ h_s(u), & \text{otherwise.} \end{cases}$$

Replace the current block by a closest block reaching v , and update (s, u) to (t, v) . This costs at most three edits and preserves the invariant. Each stage strictly increases s , so the procedure eventually reaches a point where no collision or possible rejection remains.

At that point, let V_{fin} be the set of final states reachable from u . Every such state is live. Choose

$$v_* \in \arg \max_{w \in V_{\text{fin}}} \omega(w).$$

The expected remaining reward is an average of these terminal rewards, hence

$$\omega(v_*) \geq \mathbb{E}_{Z \sim U_{n-f(s)}} [\omega(P_{s,m}(u; Z))] = h_s(u) \geq a.$$

The remaining segment has no collisions or possible rejection and therefore has at most two growth steps. The backward-tracing argument in Lemma 5.1, applied through the final layer, reaches v_* by changing only the bits at those growth steps. Thus this final completion costs at most two additional edits.

To count the total cost, let N_{rej} be the number of stages whose endpoint permits rejection, including stages that are also colliding. Every other stage ends at a distinct colliding layer, so their number N_{col} is at most c . Expected reward never decreases, increases by a factor of at least $8/7$ at each rejection stage, and is always at most one. Therefore,

$$a(8/7)^{N_{\text{rej}}} \leq 1, \quad N_{\text{rej}} \leq \log_{8/7}(1/a).$$

The stages and final completion modify disjoint blocks, so the resulting word y satisfies

$$d_H(x, y) \leq 3(N_{\text{rej}} + N_{\text{col}}) + 2$$

$$\leq 3c + 3 \log_{8/7}(1/a) + 2$$

$$= 3c + C_{\text{rad}} \log_2(1/a) + 2.$$

Its final state is v_* , whose reward is at least a . All modifications concern free coordinates. Resolving state choices by a fixed order and block choices lexicographically makes the procedure deterministic. $\square$

From repairs to a hitting set. For a density threshold $\mu > 0$ and a total collision bound c , apply Lemma 5.1 to the center 0^n . Enumerate all words whose number of ones is at most $3c + C_{\text{rad}} \log_2(1/\mu)$. Every program in this class accepts one of them. An ordered list of changed coordinates, with a symbol for an unused list entry, encodes this enumeration using

$$O((c + \log(1/\mu)) \log n)$$

seed bits for $n \geq 2$. If the radius exceeds n , enumerate all inputs instead. At density one, the single center 0^n already suffices. The generator does not compute the program-dependent repair; it enumerates a set containing its output.

A general program can have a large total collision count c , making this seed bound too large. The lemma gives a worst-case bound for every center; particular centers may require far fewer changes. We next study programs with few collisions between designated layers having at most two reachable live states. For this class, Section 6 constructs Repair^P and Cost^P with small uniform expected cost, even when the total number of collisions remains large. Section 7 constructs a second pair suited to small bias. We then use restrictions to obtain this structure. The procedure $\text{repair}_P^{\text{Total}}$ supplies the elementary local bound proved here.

6 Repair cost under uniform input

The total repair bound grows linearly with the total number of colliding layers, so it is too weak for programs with many collisions. Instead, we look for a candidate input whose repair is cheap. Dividing the computation into m intervals with at most ℓ colliding layers each does not suffice by itself: repairing each interval separately at cost $O(\ell)$ gives a total of $O(m\ell)$, and m may be large. We therefore coordinate the repairs across intervals, so that their total expected cost under a uniform candidate is $O(\ell)$, independently of m .

This section proves that bound for a repair that may inspect the whole candidate block of each interval before changing it. Section 7 constructs a second repair, whose charges can also be bounded when the candidate is drawn from a small-bias distribution. Section 9 constructs restrictions that produce the required interval structure.

6.1 The checkpoint class and the repair theorem

Let P have three live states and absorbing rejection. Write $L_0, \dots, L_d$ for its state layers and F for its free input coordinates, in their reading order. Fixed transitions are allowed, as in Section 5. Define the set of live states reachable from the original start state by

$$R_t(P) = \{s \in L_t^{\text{live}} : \text{some assignment to the preceding free bits reaches } s\}.$$

Every layer contains three live states, but $R_t(P)$ contains only those reachable from the original start state. In particular, the checkpoint condition

$$|R_{t_j}(P)| \leq 2$$

restricts reachability; it does not remove the other live states or their transitions.

Rejected computations contribute no live state. Every surviving repaired prefix also reaches a state in $R_t(P)$, since it is an assignment to those same free coordinates.

Write $\text{Coll}(P) \subseteq \{1, \dots, d\}$ for the colliding transition positions, as defined in Section 5. Fixed colliding maps are included, each position is counted once, and transitions into rejection do not count as collisions.

For an integer $\ell \geq 1$, a program P belongs to $\mathcal{K}_{3,\ell}$ if and only if there exist indices

$$0 = t_0 < t_1 < \dots < t_m = d$$

satisfying both of the following conditions:

1. At most two live states are reachable at each checkpoint: $|R_{t_j}(P)| \leq 2$ for every $j = 0, \dots, m$.

2. Each interval between consecutive checkpoints has at most ℓ colliding layers:

$$|\text{Coll}(P) \cap \{t_j + 1, \dots, t_{j+1}\}| \leq \ell \quad \text{for every } j = 0, \dots, m-1.$$

The layers $L_{t_0}, \dots, L_{t_m}$ are called *checkpoints*. For a member of $\mathcal{K}_{3,\ell}$, fix one such sequence independently of the candidate input. The *interval* I_j consists of transitions $t_j + 1, \dots, t_{j+1}$, and F_j denotes the free coordinates read there. Thus the F_j 's partition F , and consecutive intervals share a checkpoint layer. To concatenate local repairs, the exit of one must be the entrance of the next. Membership specifies structure only; the density lower bound used below is a separate hypothesis.

The start is already a checkpoint. When the final layer has three reachable live states, we may append a fixed transition sending all live states to one accepting state and $\perp$ to $\perp$. It reads no input and preserves acceptance. It adds at most one colliding layer, which must be included in the last interval's bound; the parameter ℓ does not automatically remain unchanged. The restriction construction accounts for this final merge. The use of collisions and blocks with two-state boundaries follows MRT [MRT19, Section 2.2 and Definition 2.1]; here the boundary condition concerns live states.

Proposition 6.1 (uniform repair cost). *There is an absolute constant A such that, for every $P \in \mathcal{K}_{3,\ell}$ with density $\mu = \mu(P) \geq 1/16$, there are deterministic procedures*

$$\text{Repair}^P : \{0, 1\}^F \rightarrow \{0, 1\}^F, \quad \text{Cost}^P : \{0, 1\}^F \rightarrow \mathbb{N}_0$$

with the following guarantees. For every candidate x ,

$$P(\text{Repair}^P(x)) = 1, \quad d_H(x, \text{Repair}^P(x)) \leq \text{Cost}^P(x). \quad (6.1)$$

For independent uniform free bits X ,

$$\mathbb{E}_U \text{Cost}^P(X) \leq A\ell(1 - \mu) \leq A\ell. \quad (6.2)$$

All fixed transition choices remain unchanged.

The repair guarantee holds for every candidate. The small-cost guarantee is an expectation and does not assert that every candidate has a cheap repair. We will combine compatible local repairs using the gain-cost lemma (Lemma 6.3), which bounds their total expected cost.

6.2 Proof overview

For a program $P \in \mathcal{K}_{3,\ell}$ with density $\mu = \mu(P) \geq 1/16$ and checkpoints and intervals as above, we construct Repair^P one interval at a time. The main task is to coordinate these local repairs and bound their total expected cost independently of the number of intervals. Throughout this overview, U denotes the uniform distribution on $\{0, 1\}^F$, and U_j denotes the uniform distribution on $\Omega_j = \{0, 1\}^{F_j}$.

1. **Suffix density as a measure of progress.** Our proof measures progress at a checkpoint t_j using the suffix density $\mu_{t_j}^{\text{suffix}}(s)$ of its current state s , as defined in Definition 3.1.

Writing s_0 for the initial state, we have

$$\mu_0^{\text{suffix}}(s_0) = \mu, \quad \mu_{t_m}^{\text{suffix}}(v) = 1 \quad \text{for every final live state } v.$$

Thus, along any repaired accepting computation, the changes in suffix density between consecutive checkpoints sum to $1 - \mu$. Individual changes may be negative. Our argument will relate repair cost to *expected* increases in this quantity.

2. **The gain-cost lemma.** For each interval I_j , entrance state s , and candidate block $z \in \Omega_j$, our construction specifies a repaired block and a charge $c_{j,s}(z)$ that bounds the number of changed bits. The

original exit state is denoted by $P_{j,s}(z)$, and the repaired exit state by $E_{j,s}(z)$. The gain caused by the repair is

$$g_{j,s}(z) = \mu_{t_{j+1}}^{\text{suffix}}(E_{j,s}(z)) - \mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(z)).$$

Both computations start from the same state s ; the comparison is between repairing the block and leaving it unchanged.

We prove a gain-cost lemma (Lemma 6.3): if the local repairs satisfy

$$\mathbb{E}_Z[c_{j,s}(Z)] \leq B \mathbb{E}_Z[g_{j,s}(Z)],$$

where $Z \sim U_j$ and the same B works for every interval and every entrance state that preceding repairs can produce, then their composition has total expected cost at most $B(1 - \mu)$. We construct local repairs for which $B = O(\ell)$.

This inequality makes the repair's cost allowance proportional to the improvement it produces. A repair with a larger expected charge must produce a correspondingly larger expected increase in the acceptance probability.

3. Allowed checkpoint states. We require every repaired checkpoint state to belong to

$$\mathcal{A}_j = \left\{ s \in R_{t_j}(P) : \mu_{t_j}^{\text{suffix}}(s) \geq \frac{\mu}{2} \right\},$$

the set of live states at checkpoint t_j that are reachable from the start and from which the rest of the program accepts with probability at least $\mu/2$.

These sets are nonempty. Averaging over the bits before t_j gives

$$\mu = \sum_{s \in R_{t_j}(P)} \Pr_{X \sim U} \left[\begin{array}{c} \text{the original computation on } X \\ \text{reaches } s \text{ at } t_j \end{array} \right] \mu_{t_j}^{\text{suffix}}(s).$$

Since these probabilities sum to at most 1, some $s \in R_{t_j}(P)$ has $\mu_{t_j}^{\text{suffix}}(s) \geq \mu$, and this state belongs to $\mathcal{A}_j$.

We want each repair to produce a definite gain: whenever the unchanged block would end outside $\mathcal{A}_j$, redirecting it to an allowed exit state should increase suffix density by a fixed amount. The two-state checkpoint condition guarantees a gain of at least $\mu/2$. If the unchanged block rejects, its suffix density is 0, while every allowed state has suffix density at least $\mu/2$. If it ends at a live state outside $\mathcal{A}_j$, that state has suffix density below $\mu/2$. Since at most two live states are reachable at t_j , the only allowed state is then the one of suffix density at least μ found above, so the gain is again at least $\mu/2$.

It is therefore enough for the repair of I_j to end at some state of $\mathcal{A}_{j+1}$; the gain of at least $\mu/2$ holds whichever allowed state it reaches. This state then serves as an allowed entrance state for I_{j+1} , so the repairs of consecutive intervals can be chained. The chain starts at $s_0 \in \mathcal{A}_0$ and ends in $\mathcal{A}_m$, whose states all accept, so the repaired input is accepted.

4. The repair. The repair procedure we build changes a block only when it has to, and then uses a single repair of bounded cost. Given an allowed entrance state $s \in \mathcal{A}_j$ and a candidate block $z \in \Omega_j$, it works as follows.

1. It runs the original computation from s on the whole block and computes the exit state $P_{j,s}(z)$.
2. If $P_{j,s}(z) \in \mathcal{A}_{j+1}$, it leaves the block unchanged.
3. Otherwise, it applies the terminal-reward repair of Corollary 5.2, giving each exit state its suffix density as reward. The expected reward is $\mu_{t_j}^{\text{suffix}}(s) \geq \mu/2$. So the corollary reaches an exit state of reward at least $\mu/2$, which is therefore allowed, using at most $K = O(\ell + \log(1/\mu)) = O(\ell)$ changes to free coordinates.

We charge K when step 3 is used and 0 otherwise. On a block whose original exit state is allowed, both the charge and the gain are zero. On every other block, the charge is K and the gain is at least $\mu/2$. Hence, for $Z \sim U_j$,

$$\begin{aligned}\mathbb{E}[c_{j,s}(Z)] &= K \Pr[P_{j,s}(Z) \notin \mathcal{A}_{j+1}], \\ \mathbb{E}[g_{j,s}(Z)] &\geq \frac{\mu}{2} \Pr[P_{j,s}(Z) \notin \mathcal{A}_{j+1}],\end{aligned}$$

and the local estimate holds with $B = 2K/\mu = O(\ell)$.

5. **Combining the intervals.** The full repair processes the intervals in order. Interval I_j is repaired from the entrance state S_j actually reached by the earlier repairs, and Cost_P is the sum of the charges c_j .

The expected gain of interval j equals the expected increase in suffix density from S_j to S_{j+1} . Indeed, running a uniform block without repair leaves the expected suffix density unchanged:

$$\mathbb{E}_{Z \sim U_j} [\mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(Z))] = \mu_{t_j}^{\text{suffix}}(s).$$

Combined with the local estimate, averaged over the entrance state S_j , this gives

$$\mathbb{E}[c_j] \leq B \left(\mathbb{E}[\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1})] - \mathbb{E}[\mu_{t_j}^{\text{suffix}}(S_j)] \right).$$

Summing over j , the intermediate terms cancel. Only the initial density $\mu_{t_0}^{\text{suffix}}(s_0) = \mu$ and the final density 1 remain, since every state in $\mathcal{A}_m$ accepts. Therefore

$$\mathbb{E}_{X \sim U} [\text{Cost}_P(X)] \leq B(1 - \mu) = \frac{2K}{\mu}(1 - \mu) = O(\ell).$$

Section 6.3 proves the gain-cost lemma (Lemma 6.3). Section 6.4 proves the density gap and defines the uniform local repair. Section 6.5 composes the repairs and completes the uniform bound. Section 7 will construct a different repair with the additional structure needed for the small-bias argument.

6.3 Charging repairs to suffix density

Fix $P \in \mathcal{K}_{3,\ell}$ with checkpoints $0 = t_0 < t_1 < \dots < t_m = d$. Let I_j be the interval from t_j to t_{j+1} , and let F_j be its free coordinates. Write

$$\Omega_j = \{0, 1\}^{F_j}, \quad U_j = \text{Unif}(\Omega_j), \quad U = \bigotimes_{j=0}^{m-1} U_j.$$

A uniform candidate $X = (X_0, \dots, X_{m-1}) \sim U$ has independent blocks $X_j \sim U_j$. For a fixed candidate x , write $x_j = x|_{F_j}$, and put $\mu = \Pr_{X \sim U}[P(X) = 1]$.

This subsection proves the gain-cost lemma (Lemma 6.3): if, for every interval and allowed entrance state, the expected local charge is at most B times the expected local gain, then

$$\mathbb{E}_{X \sim U} [\text{Cost}(X)] \leq B(1 - \mu).$$

The lemma applies to any compatible local repair scheme on independent uniform input blocks. Section 6.4 will construct repairs satisfying its hypothesis with $B = O(\ell)$.

Suffix density and the unrepaired exit state. At checkpoint t_j , we use $\mu_{t_j}^{\text{suffix}}$ from Definition 3.1. In particular, $\mu_0^{\text{suffix}}(s_0) = \mu$, and $\mu_{t_m}^{\text{suffix}}(v) = 1$ for every final live state v , since no input remains and every such state accepts.

For $z \in \Omega_j$, let $P_{j,s}(z)$ be the state reached by the original interval from s on the unchanged block z , including the possibility of rejection. Set $P_{j,\perp}(z) = \perp$.

Lemma 6.2 (averaging suffix densities). *For every $j < m$ and every state s at checkpoint t_j , including $\perp$,*

$$\mathbb{E}_{Z \sim U_j} [\mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(Z))] = \mu_{t_j}^{\text{suffix}}(s).$$

Proof. A uniform suffix from t_j consists of a uniform block $Z \sim U_j$ followed by an independent uniform suffix from t_{j+1} . For each $z \in \Omega_j$, the acceptance probability on that remaining suffix is $\mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(z))$. Averaging over $Z \sim U_j$ therefore gives

$$\begin{aligned} \mu_{t_j}^{\text{suffix}}(s) &= \sum_{z \in \Omega_j} \Pr_{Z \sim U_j} [Z = z] \mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(z)) \\ &= \mathbb{E}_{Z \sim U_j} [\mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(Z))]. \end{aligned}$$

For $s = \perp$, both sides are zero. $\square$

Thus crossing an interval without repair leaves the expected suffix density unchanged; an expected increase can be attributed to the repair.

A local repair scheme. The framework below is independent of the choice of local repair. Its generic symbols **Repair**, **Cost**, $E_{j,s}$, $c_{j,s}$, and $g_{j,s}$ will be instantiated separately by the uniform construction in this section and the structured construction in Section 7. Let $\mathcal{A}_0, \dots, \mathcal{A}_m$ be sets of allowed live states at the respective checkpoints, with $s_0 \in \mathcal{A}_0$. Their specific choice is deferred to Section 6.4. A *local repair scheme* assigns to each $j < m$ and $s \in \mathcal{A}_j$ deterministic maps

$$\text{Repair}_{j,s} : \Omega_j \rightarrow \Omega_j, \quad c_{j,s} : \Omega_j \rightarrow \mathbb{N}_0,$$

fixed independently of the sampled candidate. Its actual repaired exit state is

$$E_{j,s}(z) = P_{j,s}(\text{Repair}_{j,s}(z)).$$

We require, for every $z \in \Omega_j$,

$$E_{j,s}(z) \in \mathcal{A}_{j+1}, \quad d_H(z, \text{Repair}_{j,s}(z)) \leq c_{j,s}(z).$$

The first condition allows the next local repair to start from the state actually reached by the current one. The second ensures that the charge covers the edits, while allowing it to overestimate their number.

Define the *comparative gain* by

$$g_{j,s}(z) = \mu_{t_{j+1}}^{\text{suffix}}(E_{j,s}(z)) - \mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(z)).$$

Both computations start from the same entrance state and use the same candidate block; one block is repaired and the other is unchanged. Their difference measures the change in suffix density caused by that repair.

The global repair. Given a scheme and a candidate x , successively define

$$S_0(x) = s_0, \quad S_{j+1}(x) = E_{j,S_j(x)}(x_j),$$

Thus $S_j(x)$ is the state reached at checkpoint t_j after repairing the preceding blocks; for $j < m$, it is the entrance state of interval I_j . We then define the repaired candidate and its charges by

$$\text{Repair}(x)|_{F_j} = \text{Repair}_{j,S_j(x)}(x_j), \quad c_j(x) = c_{j,S_j(x)}(x_j), \quad \text{Cost}(x) = \sum_{j=0}^{m-1} c_j(x).$$

Induction gives $S_j(x) \in \mathcal{A}_j$. The repaired blocks drive the original program through these live checkpoint states without rejection, so $P(\text{Repair}(x)) = 1$. Since the blocks use disjoint free coordinates,

$$d_H(x, \text{Repair}(x)) = \sum_{j=0}^{m-1} d_H(x_j, \text{Repair}_{j,S_j(x)}(x_j)) \leq \text{Cost}(x).$$

It therefore remains to bound the expected sum of the local charges.

Lemma 6.3 (gain-cost). *Suppose there is a constant $B \geq 0$ such that, for every interval $j < m$ and every allowed entrance state $s \in \mathcal{A}_j$,*

$$\mathbb{E}_{Z \sim U_j}[c_{j,s}(Z)] \leq B \mathbb{E}_{Z \sim U_j}[g_{j,s}(Z)].$$

Then

$$\mathbb{E}_{X \sim U}[\text{Cost}(X)] \leq B(1 - \mu).$$

The suffix density of the current state measures progress toward acceptance. It starts at μ and must end at 1, so the total progress is exactly $1 - \mu$, however many intervals there are. Left alone, a random interval makes no progress on average, so every expected unit of progress is produced by a repair. The lemma says: if each repair costs at most B edits per unit of expected progress it produces, the total expected cost is at most $B(1 - \mu)$.

Proof. The entrance state $S_j(X)$ depends only on $X_0, \dots, X_{j-1}$, so it is independent of the current block $X_j \sim U_j$. Splitting according to its value gives

$$\begin{aligned} \mathbb{E}_{X \sim U}[c_j(X)] &= \sum_{s \in \mathcal{A}_j} \Pr_{X \sim U}[S_j(X) = s] \mathbb{E}_{Z \sim U_j}[c_{j,s}(Z)], \\ \mathbb{E}_{X \sim U}[\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}(X))] &= \sum_{s \in \mathcal{A}_j} \Pr_{X \sim U}[S_j(X) = s] \mathbb{E}_{Z \sim U_j}[\mu_{t_{j+1}}^{\text{suffix}}(E_{j,s}(Z))]. \end{aligned}$$

These identities average the appropriate local repair over the entrance state produced by earlier repairs. For each fixed s , Lemma 6.2 gives

$$\mathbb{E}_{Z \sim U_j}[g_{j,s}(Z)] = \mathbb{E}_{Z \sim U_j}[\mu_{t_{j+1}}^{\text{suffix}}(E_{j,s}(Z))] - \mu_{t_j}^{\text{suffix}}(s).$$

Average the hypothesis of Lemma 6.3 over s with weights $\Pr_{X \sim U}[S_j(X) = s]$. The preceding identities yield

$$\begin{aligned} \mathbb{E}_{X \sim U}[c_j(X)] &= \sum_{s \in \mathcal{A}_j} \Pr_{X \sim U}[S_j(X) = s] \mathbb{E}_{Z \sim U_j}[c_{j,s}(Z)] \\ &\leq B \sum_{s \in \mathcal{A}_j} \Pr_{X \sim U}[S_j(X) = s] \mathbb{E}_{Z \sim U_j}[g_{j,s}(Z)] \\ &= B \sum_{s \in \mathcal{A}_j} \Pr_{X \sim U}[S_j(X) = s] \left(\mathbb{E}_{Z \sim U_j}[\mu_{t_{j+1}}^{\text{suffix}}(E_{j,s}(Z))] - \mu_{t_j}^{\text{suffix}}(s) \right) \\ &= B \left(\mathbb{E}_{X \sim U}[\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}(X))] - \mathbb{E}_{X \sim U}[\mu_{t_j}^{\text{suffix}}(S_j(X))] \right). \end{aligned}$$

The same state $S_{j+1}(X)$ is the repaired exit state of interval j and the entrance state of interval $j + 1$. Hence its expected density appears with opposite signs in the two consecutive bounds. Summing cancels all intermediate terms:

$$\begin{aligned} \mathbb{E}_{X \sim U}[\text{Cost}(X)] &\leq B \sum_{j=0}^{m-1} \left(\mathbb{E}_{X \sim U}[\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}(X))] - \mathbb{E}_{X \sim U}[\mu_{t_j}^{\text{suffix}}(S_j(X))] \right) \\ &= B \left(\mathbb{E}_{X \sim U}[\mu_{t_m}^{\text{suffix}}(S_m(X))] - \mu_0^{\text{suffix}}(s_0) \right) \\ &= B(1 - \mu). \end{aligned}$$

The final equality holds because every repaired computation ends at a live accepting state. Thus more intervals do not create a larger total available density increase. $\square$

The telescoping quantity and the potential. For every candidate x , the checkpoint densities start at μ and end at one. Their increments therefore satisfy

$$\sum_{j=0}^{m-1} \left(\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}(x)) - \mu_{t_j}^{\text{suffix}}(S_j(x)) \right) = 1 - \mu.$$

Individual increments may be negative. The comparative gain $g_{j,s}(z)$ is generally different from the increment $\mu_{t_{j+1}}^{\text{suffix}}(E_{j,s}(z)) - \mu_{t_j}^{\text{suffix}}(s)$ on a particular block; their expectations under $Z \sim U_j$ agree by Lemma 6.2. Thus it is the checkpoint-density increments that telescope pointwise. For the comparative gains, we obtain

$$\mathbb{E}_{X \sim U} \left[\sum_{j=0}^{m-1} g_{j,S_j(X)}(X_j) \right] = 1 - \mu.$$

The hypothesis of Lemma 6.3 only compares expected charge and expected gain, with B converting density increase into a bound on bit changes.

In potential form, set $\Phi_j(s) = B(1 - \mu_{t_j}^{\text{suffix}}(s))$. By Lemma 6.2, the hypothesis of Lemma 6.3 is equivalent to

$$\mathbb{E}_{Z \sim U_j} [c_{j,s}(Z) + \Phi_{j+1}(E_{j,s}(Z))] \leq \Phi_j(s).$$

For each fixed entrance state, the potential covers the expected current charge and the expected potential remaining at the exit state. This is the potential-descent formulation of [Krak26]; the gain-cost lemma obtains the global bound by averaging over entrance states and summing these local bounds.

Lemma 6.4 (local cost–gain ratio). *Fix $j < m$ and $s \in \mathcal{A}_j$. Suppose an event $\mathcal{D} \subseteq \Omega_j$ and constants $K, \gamma > 0$ satisfy*

$$\begin{aligned} 0 \leq c_{j,s}(z) \leq K \quad \text{for every } z \in \Omega_j, \quad c_{j,s}(z) = 0 \quad \text{for } z \notin \mathcal{D}, \\ \mathbb{E}_{Z \sim U_j} [g_{j,s}(Z)] \geq \gamma \Pr_{Z \sim U_j} [Z \in \mathcal{D}]. \end{aligned}$$

Then

$$\mathbb{E}_{Z \sim U_j} [c_{j,s}(Z)] \leq \frac{K}{\gamma} \mathbb{E}_{Z \sim U_j} [g_{j,s}(Z)].$$

Proof.

$$\begin{aligned} \mathbb{E}_{Z \sim U_j} [c_{j,s}(Z)] &= \sum_{z \in \Omega_j} \Pr_{Z \sim U_j} [Z = z] c_{j,s}(z) = \sum_{z \in \mathcal{D}} \Pr_{Z \sim U_j} [Z = z] c_{j,s}(z) \\ &\leq \sum_{z \in \mathcal{D}} \Pr_{Z \sim U_j} [Z = z] K = K \Pr_{Z \sim U_j} [Z \in \mathcal{D}] \\ &= \frac{K}{\gamma} \left(\gamma \Pr_{Z \sim U_j} [Z \in \mathcal{D}] \right) \leq \frac{K}{\gamma} \mathbb{E}_{Z \sim U_j} [g_{j,s}(Z)]. \quad \square \end{aligned}$$

By Lemma 6.4, common constants K, γ establish the hypothesis of Lemma 6.3 with $B = K/\gamma$. The same event controls the possible charge and the net expected gain, including any losses. Outside $\mathcal{D}$, the charge is zero, so the repair-distance guarantee forces $\text{Repair}_{j,s}(z) = z$. The exit states then agree and $g_{j,s}(z) = 0$. The event $\mathcal{D}$ need not coincide with the event that bits are changed: charges are allowed to overestimate actual edits.

6.4 Allowed states and the uniform repair

We first prove the following lemma, which identifies the allowed exit states and the density gain from reaching them.

Lemma 6.5 (allowed exit states and density gain). *Let $P \in \mathcal{K}_{3,\ell}$ have density $\mu \geq 1/16$. For each checkpoint t_k , define*

$$\mathcal{A}_k = \left\{ v \in R_{t_k}(P) : \mu_{t_k}^{\text{suffix}}(v) \geq \frac{\mu}{2} \right\}.$$

Then, for every interval I_j , the set $\mathcal{A}_{j+1}$ is nonempty. Moreover, for every allowed entrance state $s \in \mathcal{A}_j$, candidate block $z \in \Omega_j$, and allowed exit state $v \in \mathcal{A}_{j+1}$,

$$P_{j,s}(z) \notin \mathcal{A}_{j+1} \implies \mu_{t_{j+1}}^{\text{suffix}}(v) - \mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(z)) \geq \frac{\mu}{2}, \quad (6.3)$$

where $\mu_{t_{j+1}}^{\text{suffix}}(\perp) = 0$.

The lemma says two things about every interval. First, among the states the interval can end in, there is at least one allowed state: a state from which the rest of the program accepts with probability at least $\mu/2$. Second, suppose the unchanged block would end outside the allowed states, either by rejecting or at a state with suffix density below $\mu/2$. Then redirecting it to any allowed state increases the suffix density by at least $\mu/2$.

The allowed exit state of one interval is an allowed entrance state for the next, so repairs with these endpoints concatenate. For the uniform procedure, we invoke repair exactly when the original exit state is disallowed. The lemma then gives a gain of at least $\mu/2$, while the unchanged blocks have gain zero. This will relate repair cost to gain without any possible negative comparative gains.

Proof. We first show that every checkpoint t_k has a forward-reachable live state of suffix density at least μ . For a candidate x , define the states of the original computation recursively by

$$S_0^{\text{orig}}(x) = s_0, \quad S_{j+1}^{\text{orig}}(x) = P_{j, S_j^{\text{orig}}(x)}(x_j).$$

Thus $S_j^{\text{orig}}(x)$ is the state reached at checkpoint t_j by the original program on the unchanged candidate x .

Let $X = (X_0, \dots, X_{m-1}) \sim U$, as in Section 6.3. The state $S_j^{\text{orig}}(X)$ depends only on $X_0, \dots, X_{j-1}$, so it is independent of $X_j \sim U_j$. Averaging Lemma 6.2 over this entrance state therefore gives

$$\mathbb{E}_{X \sim U} [\mu_{t_{j+1}}^{\text{suffix}}(S_{j+1}^{\text{orig}}(X))] = \mathbb{E}_{X \sim U} [\mu_{t_j}^{\text{suffix}}(S_j^{\text{orig}}(X))].$$

Iterating from $S_0^{\text{orig}}(X) = s_0$, we obtain, for every k ,

$$\mathbb{E}_{X \sim U} [\mu_{t_k}^{\text{suffix}}(S_k^{\text{orig}}(X))] = \mu_0^{\text{suffix}}(s_0) = \mu.$$

Rejected computations contribute zero to this expectation, and every live state reached at t_k belongs to $R_{t_k}(P)$. Since $\mu > 0$, this set is nonempty. Consequently,

$$\begin{aligned} \mu &= \sum_{v \in R_{t_k}(P)} \Pr_{X \sim U} [S_k^{\text{orig}}(X) = v] \mu_{t_k}^{\text{suffix}}(v) \\ &\leq \max_{v \in R_{t_k}(P)} \mu_{t_k}^{\text{suffix}}(v), \end{aligned}$$

where the inequality uses that the displayed probabilities sum to at most one. This proves the preliminary claim.

Now fix $j < m$, and write

$$R^{\text{next}} = R_{t_{j+1}}(P).$$

The preliminary claim supplies $e \in R^{\text{next}}$ with $\mu_{t_{j+1}}^{\text{suffix}}(e) \geq \mu$. Thus $e \in \mathcal{A}_{j+1}$, proving that $\mathcal{A}_{j+1} \neq \emptyset$.

Fix $s \in \mathcal{A}_j$ and $z \in \Omega_j$ such that $w := P_{j,s}(z) \notin \mathcal{A}_{j+1}$. If $w = \perp$, then every $v \in \mathcal{A}_{j+1}$ satisfies

$$\mu_{t_{j+1}}^{\text{suffix}}(v) - \mu_{t_{j+1}}^{\text{suffix}}(\perp) = \mu_{t_{j+1}}^{\text{suffix}}(v) \geq \frac{\mu}{2}.$$

Otherwise, w is a live state in R^{next} : an original prefix reaching s , followed by z , reaches w . Since $w \notin \mathcal{A}_{j+1}$,

$$\mu_{t_{j+1}}^{\text{suffix}}(w) < \frac{\mu}{2}.$$

The two-state checkpoint condition gives $|R^{\text{next}}| \leq 2$. Because $e \in \mathcal{A}_{j+1}$ and $w \notin \mathcal{A}_{j+1}$, it follows that

$$R^{\text{next}} = \{e, w\}, \quad \mathcal{A}_{j+1} = \{e\}.$$

Hence every $v \in \mathcal{A}_{j+1}$ equals e , and

$$\mu_{t_{j+1}}^{\text{suffix}}(v) - \mu_{t_{j+1}}^{\text{suffix}}(w) > \mu - \frac{\mu}{2} = \frac{\mu}{2}.$$

This proves the claimed gain. $\square$

We use these $\mathcal{A}_k$ as the allowed states of Section 6.3, and call them *high-density* states. In particular, $s_0 \in \mathcal{A}_0$, and $\mathcal{A}_m = R_{t_m}(P)$, since every final live state accepts with probability 1.

The interval program. Fix $j < m$ and an allowed entrance state $s \in \mathcal{A}_j$, and let c be a bound on the number of colliding layers in I_j . Let $\mathcal{I}_{j,s}$ be the program obtained from P by keeping only the transitions $t_j + 1, \dots, t_{j+1}$ and starting at s :

- its layers are the layers $t_j, \dots, t_{j+1}$ of P , its start state is s , and its final layer is layer t_{j+1} ;
- its free input coordinates are those of F_j , read in the same order as in P ;
- its transitions, free and fixed, are those of P .

Thus $\mathcal{I}_{j,s}$ has at most three live states in each layer and absorbing rejection. Its colliding layers are the colliding layers of P in I_j , so there are at most c of them. On a block $z \in \Omega_j$, its final state is $P_{j,s}(z)$.

Terminal rewards. Give each live state v of the final layer the reward

$$\omega_j(v) = \mu_{t_{j+1}}^{\text{suffix}}(v) \in [0, 1],$$

and give rejection reward 0. By the averaging identity for suffix densities, the expected reward under uniform free bits is

$$a_{j,s} = \mathbb{E}_{Z \sim U_j} [\omega_j(P_{j,s}(Z))] = \mu_{t_j}^{\text{suffix}}(s) \geq \frac{\mu}{2},$$

where the inequality holds because $s \in \mathcal{A}_j$.

The reward repair. Apply Corollary 5.2 to the program $\mathcal{I}_{j,s}$ with rewards ω_j and collision bound c . This is allowed, since $a_{j,s} > 0$. Fix its deterministic tie-breaking from the program alone. The corollary gives a map

$$R_{j,s}^{\text{reward}} : \Omega_j \rightarrow \Omega_j$$

such that, for every candidate block $z \in \Omega_j$:

1. the computation of $\mathcal{I}_{j,s}$ on $R_{j,s}^{\text{reward}}(z)$ ends at a live state v with $\omega_j(v) \geq a_{j,s}$;
2. the number of changed coordinates satisfies

$$d_H(z, R_{j,s}^{\text{reward}}(z)) \leq 3c + C_{\text{rad}} \log_2 \frac{1}{a_{j,s}} + 2 \leq r_{\text{reward}}(c, \mu),$$

where

$$r_{\text{reward}}(c, \mu) = 3c + \left\lceil C_{\text{rad}} \log_2 \frac{2}{\mu} \right\rceil + 2;$$

3. only coordinates in F_j change, so all fixed transitions remain unchanged.

The exit state is allowed. Let $v = P_{j,s}(R_{j,s}^{\text{reward}}(z))$. By item 1,

$$\mu_{t_{j+1}}^{\text{suffix}}(v) = \omega_j(v) \geq a_{j,s} \geq \frac{\mu}{2}.$$

In particular v is live, since rejection has reward 0. It is also forward-reachable: since $s \in R_{t_j}(P)$, concatenating an original prefix that reaches s with the block $R_{j,s}^{\text{reward}}(z)$ reaches v . Hence $v \in \mathcal{A}_{j+1}$.

Globally we take $c = \ell$. Then $r_{\text{reward}}(\ell, \mu) = \tilde{O}(\ell)$, uniformly for $\mu \geq 1/16$ and $\ell \geq 1$.

Definition 6.6 (uniform interval repair). For $s \in \mathcal{A}_j$ and $z \in \Omega_j$, define

$$\text{Repair}_{j,s}(z) = \begin{cases} z, & P_{j,s}(z) \in \mathcal{A}_{j+1}, \\ R_{j,s}^{\text{reward}}(z), & P_{j,s}(z) \notin \mathcal{A}_{j+1}. \end{cases} \quad (6.4)$$

For $s \in \mathcal{A}_j$ and $z \in \Omega_j$, let

$$E_{j,s}(z) = P_{j,s}(\text{Repair}_{j,s}(z))$$

be the exit state of the repaired block. Let $g_{j,s}(z)$ be its comparative gain, defined as in Section 6.3. We charge

$$c_{j,s}(z) = \begin{cases} 0, & P_{j,s}(z) \in \mathcal{A}_{j+1}, \\ r_{\text{reward}}(\ell, \mu), & P_{j,s}(z) \notin \mathcal{A}_{j+1}. \end{cases} \quad (6.5)$$

The procedure reads the whole block before changing anything, and reading changes no bit. All of its edits are made by the terminal-reward repair, at free coordinates. For every candidate block, the repaired exit state is allowed,

$$E_{j,s}(z) \in \mathcal{A}_{j+1},$$

and the charge covers the edits,

$$d_H(z, \text{Repair}_{j,s}(z)) \leq c_{j,s}(z).$$

To compare the charge with the gain, we consider two cases.

- **The original exit state is allowed.** The block is unchanged, so $c_{j,s}(z) = 0$ and $g_{j,s}(z) = 0$.
- **The original exit state is not allowed.** The charge is $r_{\text{reward}}(\ell, \mu)$, and Lemma 6.5 gives $g_{j,s}(z) \geq \mu/2$.

In both cases,

$$c_{j,s}(z) \leq \frac{2r_{\text{reward}}(\ell, \mu)}{\mu} g_{j,s}(z). \quad (6.6)$$

Thus the local cost–gain inequality holds for every block, not only on average. In particular, for a uniform block $Z \sim U_j$, the expected charge is

$$\mathbb{E}[c_{j,s}(Z)] = r_{\text{reward}}(\ell, \mu) \Pr[P_{j,s}(Z) \notin \mathcal{A}_{j+1}],$$

and the expected gain satisfies

$$\mathbb{E}[g_{j,s}(Z)] \geq \frac{\mu}{2} \Pr[P_{j,s}(Z) \notin \mathcal{A}_{j+1}].$$

6.5 Composing the uniform repairs

For a candidate $x = (x_0, \dots, x_{m-1})$, define the actual repaired checkpoint states and the global procedure by

$$\begin{aligned} S_0(x) &= s_0, \\ S_{j+1}(x) &= E_{j,S_j(x)}(x_j), \\ \text{Repair}^P(x)|_{F_j} &= \text{Repair}_{j,S_j(x)}(x_j), \\ \text{Cost}^P(x) &= \sum_{j=0}^{m-1} c_{j,S_j(x)}(x_j). \end{aligned} \quad (6.7)$$

The local endpoint guarantee gives $S_j(x) \in \mathcal{A}_j$ by induction. The final state therefore accepts, and the sum of the charges bounds the total number of edits.

Proof of Proposition 6.1. Each interval repair ends at an allowed exit state, and its charge bounds the number of changed bits. Combining these repairs gives an accepting input whose total number of changes is at most $\text{Cost}^P(X)$, proving (6.1).

The entrance state $S_j(X)$ depends only on the earlier blocks $X_0, \dots, X_{j-1}$. Under uniform input, the current block X_j is therefore independent of $S_j(X)$. This is the independence needed to apply the gain-cost lemma (Lemma 6.3); the repair may inspect all of X_j before changing it.

By (6.6), one common coefficient is $B = 2r_{\text{reward}}(\ell, \mu)/\mu = O(\ell)$. Hence

$$\mathbb{E}_U[\text{Cost}^P(X)] \leq \frac{2r_{\text{reward}}(\ell, \mu)}{\mu}(1 - \mu) \leq A\ell(1 - \mu)$$

for an absolute constant A , proving (6.2). Together with the definition of $\mathcal{K}_{3,\ell}$, this proves Theorem 2.2. $\square$

The argument proves that some uniform candidate has repair cost $O(\ell)$, independently of the number of intervals. To obtain a small universal family of candidates, Section 7 constructs a second repair, Repair_{θ}^P , whose cost has additional structure suited to small bias. It will prove a uniform bound for that second cost function; Section 8 then transfers its expectation to the pseudorandom source.

7 Repair charges as state visits

So far we have a repair whose cost is small for a *uniform* candidate:

$$\mathbb{E}_{X \sim U}[\text{Cost}^P(X)] = O(\ell).$$

We want to build a generator that produces its centers from a short seed. A uniform center on n coordinates would require n seed bits, so the generator we build will instead draw its center from a small-bias distribution D , which can be sampled with few seed bits. We therefore need a repair with the same bound under D . We will construct such a repair below, with cost denoted Cost_{θ}^P :

$$\mathbb{E}_{X \sim D}[\text{Cost}_{\theta}^P(X)] = O(\ell).$$

Then some center in the support of D has cost $O(\ell)$, so enumerating all changes to $O(\ell)$ coordinates around every center yields an accepted input.

7.1 Proof overview

Throughout this section, $n \geq 4$ and $P \in \mathcal{K}_{3,\ell}$ has density $\mu = \mu(P) \geq 1/16$, free coordinates F , and $O(n)$ transition layers. Smaller lengths are handled in Section 10. We retain the checkpoints and allowed states from Section 6 and use the suffix densities of Definition 3.1. All probabilities defining these quantities and the interval probabilities below use independent uniform free bits, even when the eventual candidate source is D .

Write $\theta = (\theta_0, \dots, \theta_{m-1})$ for the interval thresholds chosen below. The new repair and cost are denoted Repair_{θ}^P and Cost_{θ}^P . They are fixed from P , independently of the particular distribution D .

1. **What the small-bias transfer requires.** We cannot apply the state-visit bound below directly to Cost^P . Instead, we write the new cost as a weighted sum of simple events and transfer each event from U to D separately. The events we use are *state visits*, sometimes together with a specified value of the next input bit. Let Q be a program with $O(n)$ transition layers that reads the candidate x , and for a layer t and a state u let

$$O_{t,u}(x) = \mathbf{1}[Q \text{ on } x \text{ is at state } u \text{ at layer } t].$$

The probability of such a visit under D can be bounded by its probability under U , provided Q has the following simple form:

- it has three states in every layer and never rejects;
- at each checkpoint, at most two of its states are reachable;
- each interval contains only $O(\ell)$ colliding layers.

For such a program, Theorem 8.1 shows that a δ -biased distribution with $\delta \leq n^{-C\ell}$, for a sufficiently large absolute constant C , cannot make any visit much more likely than uniform input does:

$$\mathbb{E}_D[O_{t,u}] \leq 4\mathbb{E}_U[O_{t,u}] + \frac{1}{n}.$$

Only this upper bound is needed, since we want to show that the cost is small. The proof therefore has three steps.

- (i) Build a repair whose charge is a weighted sum of these visit events and events specifying a visit and the next bit.
- (ii) Bound the expected charge under U by the gain-cost lemma.
- (iii) Transfer each event from U to D .

What this requires of the repair. The repair of Definition 6.6 does not give the representation needed in the first step. It is charged exactly on the blocks whose unchanged computation from s misses $\mathcal{A}_{j+1}$:

$$r_{\text{reward}}(\ell, \mu) \cdot \mathbf{1}[P_{j,s}(z) \notin \mathcal{A}_{j+1}].$$

This happens when the original computation either ends at a live state outside $\mathcal{A}_{j+1}$ or rejects. For a fixed entrance state s , run the original interval from s , keeping the rejecting state $\perp$ as a fourth state. The charged event is exactly a visit, at the exit layer, to $\perp$ or to a disallowed live state:

$$\mathbf{1}[P_{j,s}(z) \notin \mathcal{A}_{j+1}] = O_{t_{j+1}, \perp}(z) + \sum_{v \in L_{t_{j+1}}^{\text{live}} \setminus \mathcal{A}_{j+1}} O_{t_{j+1}, v}(z).$$

Here $L_{t_{j+1}}^{\text{live}}$ is the set of three live states at the exit layer, and $O_{t,u}$ denotes the visit events of that four-state program. The bound on visit probabilities is proved only for three-state programs of the form above, so it does not apply to this representation of the charge.

The fourth state comes from how the repair decides, not from the repaired computation, which never rejects: a kept block ends at an allowed exit state, and a repaired block ends at an exit state of reward at least $\mu/2 > 0$. The repair acts exactly when the *unchanged* computation from s ends outside $\mathcal{A}_{j+1}$, possibly by rejecting, which is known only after the whole block has been inspected. Representing this decision requires running the unchanged computation to the end of the interval, including its rejecting state.

A repair that decides bit by bit. We now build a new interval repair procedure. Two different objects are involved, and it helps to separate them from the start.

- **The repair procedure.** Given an allowed entrance state $s \in \mathcal{A}_j$ and a candidate block $x \in \Omega_j$, it outputs a repaired block y . Its *repaired computation* is the computation of P started at s and run on y . It therefore uses the original transition maps T_t^0, T_t^1 of P ; the only thing the procedure chooses is which of the two transitions is taken at each free layer.
- **The program Q_j .** This is an analysis device, not part of the repair. It is a three-state read-once program that reads the *original* block x . On the states the repaired computation reaches, Q_j follows the same transitions, so it reproduces the repaired computation step by step. On the remaining states it may change the transitions of P , so that it is a genuine three-state program that never rejects.

The repair procedure. The procedure treats an interval differently depending on its entrance state s . If the original interval started at s reaches an allowed exit state with probability at least a threshold θ_j under

uniform free bits, it uses $\text{Repair-Interval}_{j,\theta_j}^{\text{high}}$. Otherwise, it uses a different procedure, denoted here by $\text{Repair-Interval}_j^{\text{low}}$. For this fixed interval and threshold, we abbreviate these names to $\text{Repair-Interval}^{\text{high}}$ and $\text{Repair-Interval}^{\text{low}}$.

Q_j reproduces step by step only the computation of $\text{Repair-Interval}^{\text{high}}$. From entrance states handled by $\text{Repair-Interval}^{\text{low}}$, it need only end at the same exit state as that procedure, since the next interval depends only on this state; Step 4 arranges this.

We first describe $\text{Repair-Interval}^{\text{high}}$, which scans the block and decides each bit as it is read. $\text{Repair-Interval}^{\text{low}}$ is described in Step 4 below.

Our goal is to build the repair procedure so that such a Q_j exists with three properties.

- (i) **It reproduces the repaired computation.** From an entrance state handled by scanning, Q_j passes through exactly the states that P visits on the repaired block. From every allowed entrance state, its exit state agrees with the repaired exit state.
- (ii) **It expresses the charges.** The charge is a weighted sum of visit indicators of Q_j , sometimes together with a specified value of the next input bit.
- (iii) **It has the form required by the visit bound.** Each Q_j has three states, never rejects, and has only $O(\ell)$ colliding layers. When the interval programs are composed from the original start state, at most two states are reachable at each checkpoint.

At each free transition t , $\text{Repair-Interval}^{\text{high}}$ uses the bit given by an *output function*

$$\beta_t(u, b) \in \{b, 1 - b\}$$

of the current state u of the repaired computation and the candidate bit b alone. Here β_t abbreviates β_{t,θ_j} . The repaired computation must end in an allowed exit state, so it can never reject. We will choose β_t so that every step of the repaired computation moves to a live state.

The program Q_j . Formally, Q_j is a read-once program on the free coordinates of I_j . On the original layers $t_j, \dots, t_{j+1}$, its three states are exactly the three live states of P . Write L_t^{live} for this set at layer t . A fixed reset may be appended at the exit, as explained in Step 4. Its free transition maps are

$$\delta_t^b : L_{t-1}^{\text{live}} \longrightarrow L_t^{\text{live}}, \quad b \in \{0, 1\}.$$

At a free transition t , from a state u reached by the scanning repair, on input bit b , it moves to

$$\delta_t^b(u) = T_t^{\beta_t(u, b)}(u).$$

At fixed transitions it uses the original maps on these states. Its transitions from the remaining states are specified below. Q_j does not depend on the entrance state: the same maps are used whatever state it starts from.

Fix an allowed entrance state s handled by $\text{Repair-Interval}^{\text{high}}$ and a candidate block $x \in \Omega_j$. Let

$$y = \text{Repair-Interval}^{\text{high}}(x, s)$$

be the repaired block, and let u_t be the state of P at layer t when started at s and run on y . Write x_t for the candidate bit read at free transition t , and $Q_j(s; x)_t$ for the state of Q_j at original layer t . Since Q_j reads x_t and applies the same transition as the repair, induction over the layers gives

$$Q_j(s; x)_t = u_t \quad \text{for } t_j \leq t \leq t_{j+1}.$$

In particular, at the original exit layer it reaches the repaired exit state:

$$Q_j(s; x)_{t_{j+1}} = E_{\theta_j, j, s}(x),$$

where $E_{\theta_j, j, s}(x)$ denotes the exit state selected by the new repair. The optional reset leaves this state unchanged. So Q_j answers the question “where is the repaired computation right now?” while reading only the unrepaired bits. From entrance states handled by $\text{Repair-Interval}^{\text{low}}$, Q_j need only end at the same exit state as that procedure, after the optional reset; Step 4 arranges this.

Charges as visits. For an entrance state handled by $\text{Repair-Interval}^{\text{high}}$, let $O_{t,u}(x)$ denote the visit indicators of Q_j started at that entrance state. A flip happens at transition t exactly when Q_j is at some state u at layer $t-1$, the next candidate bit is some b , and $\beta_t(u, b) \neq b$. Each flip is therefore detected by a visit of Q_j to a specified state, together with a specified value of the next input bit. Summing over all such prescribed scanning choices (t, u, b) , with t a free transition, counts the flips:

$$\#\{\text{flips in the high-threshold repair of } I_j\} = \sum_{t, u, b: \beta_t(u, b) \neq b} O_{t-1, u}(x) \cdot \mathbf{1}[x_t = b].$$

Step 4 adds a checkpoint visit charge to cover the low-threshold repair.

Few colliding layers. The visit bound also requires Q_j to have only $O(\ell)$ colliding layers, while the original interval has at most ℓ of them. Among the original transitions, a transition t can be colliding for Q_j in only two ways.

- **It is colliding for P .** There are at most ℓ such transitions in I_j .
- **A flip is possible at t .** Suppose the scanning repair keeps the candidate bit b at a state u but flips it at another state u' . Then δ_t^b sends u to $T_t^b(u)$ and u' to $T_t^{1-b}(u')$, and these two states may coincide:

$$T_t^b(u) = T_t^{1-b}(u').$$

Now consider a free transition t at which no scanning flip is possible and which is not colliding for P . Since no flip is possible, Q_j applies the original map T_t^b to every state reached by the scanning repair. Since that repair never rejects, every such state u satisfies

$$T_t^b(u) \neq \perp$$

for both bits b . For each bit b , let

$$D_t^b = \{u \in L_{t-1}^{\text{live}} : T_t^b(u) \neq \perp\}$$

be the set of live states at layer $t-1$ from which T_t^b does not reject. Since t is not colliding for P , the map T_t^b is injective on D_t^b .

Here Q_j departs from P . The remaining states are those from which T_t^b rejects. The scanning repair never reaches them at layer $t-1$, so Q_j 's transitions from these states can be chosen freely without changing any scanning repair computation. We send them bijectively onto the states at layer t that are not in $T_t^b(D_t^b)$. The resulting map δ_t^b agrees with T_t^b on D_t^b and is a permutation of the three live states, so t is not a colliding layer of Q_j . The same extension applies to the single map of a fixed noncolliding transition. At original colliding transitions and possible scanning-flip transitions, we complete any unprescribed map values to live states arbitrarily.

Hence the number of colliding layers of Q_j is at most ℓ plus the number of transitions at which a scanning flip is possible, plus at most one for the optional final reset. Since Q_j must be one program for every allowed entrance state and every candidate, we need a fixed set of $O(\ell)$ transitions, independent of both, outside which the high-threshold repair never flips a bit.

2. **Identify when a repair gains enough.** Besides being a sum of visit events, the charge must satisfy the cost–gain estimate of Section 6.2 under uniform input: its expected value must be at most $B = O(\ell)$ times the expected gain. Fix an interval I_j and an allowed entrance state $s \in \mathcal{A}_j$. For a candidate block z , the comparative gain $g_{\theta_j, j, s}(z)$ compares the repaired exit state with the original one, and there are two cases.

- **The original exit state is outside $\mathcal{A}_{j+1}$.** Moving to any allowed exit state gains at least $\mu/2$, by Lemma 6.5:

$$P_{j,s}(z) \notin \mathcal{A}_{j+1} \implies g_{\theta_j,j,s}(z) \geq \frac{\mu}{2}.$$

- **The original exit state is already allowed.** The repair may move to a different allowed exit state of smaller suffix density. Since suffix densities lie in $[0, 1]$, the loss is at most 1:

$$g_{\theta_j,j,s}(z) \geq -1.$$

Losses are possible because the scanning repair decides each bit as it is read. It may change a bit even though the remaining candidate bits would have led the unchanged computation to an allowed exit state.

To keep the losses small, the repair should start changing bits only after prefixes from which the unchanged computation is unlikely to reach $\mathcal{A}_{j+1}$. To make this precise, fix a transition t of I_j , and write a block $z \in \Omega_j$ as

$$z = (z_{\leq t}, z_{> t}),$$

where $z_{\leq t}$ assigns the free coordinates of I_j read up to transition t and $z_{> t}$ assigns the rest. For a prefix w , let

$$p(w) = \Pr_{Z \sim U_j} [P_{j,s}(Z) \in \mathcal{A}_{j+1} \mid Z_{\leq t} = w]$$

be the probability that the unchanged computation from s ends in $\mathcal{A}_{j+1}$ when the bits after w are uniform. Assume also that every repaired computation ends in $\mathcal{A}_{j+1}$:

$$E_{\theta_j,j,s}(z) \in \mathcal{A}_{j+1} \quad \text{for every } z \in \Omega_j.$$

Then, conditioned on the prefix w , the unchanged computation ends outside $\mathcal{A}_{j+1}$ with probability $1 - p(w)$. On each such block the repaired computation ends in $\mathcal{A}_{j+1}$, so the gain is at least $\mu/2$. On the remaining blocks the gain is at least -1 . Hence

$$\mathbb{E}_{Z \sim U_j} [g_{\theta_j,j,s}(Z) \mid Z_{\leq t} = w] \geq (1 - p(w)) \frac{\mu}{2} - p(w).$$

If $p(w) \leq \theta_j$ for a threshold θ_j that is small relative to μ , the gains outweigh the losses. For example, $\theta_j \leq \mu/12$ gives

$$(1 - p(w)) \frac{\mu}{2} - p(w) \geq \frac{3\mu}{8}.$$

This bound concerns the gain of the whole interval, measured at its exit state. It holds however many bits the repair changes after the prefix w . The next step introduces the quantity that measures $p(w)$ at every state and layer of the interval.

3. **Measure the chance of reaching an allowed exit, and repair by scanning.** The probability $p(w)$ of Step 2 depends on the prefix w only through the state that the unchanged computation reaches after reading it. We therefore define this probability for every state and layer of the interval.

For a state u at a layer t of I_j , its *interval acceptance probability* is

$$\sigma_t^{(j)}(u) = \Pr \left[\text{the original computation from } u \text{ at layer } t \text{ ends in } \mathcal{A}_{j+1} \right],$$

where the remaining free bits of I_j are uniform, and $\sigma_t^{(j)}(\perp) = 0$. If the unchanged computation from s reaches the state v at layer t after the prefix w , then

$$p(w) = \sigma_t^{(j)}(v).$$

At the exit layer, $\sigma^{(j)}$ is 1 on allowed states and 0 on all others:

$$\sigma_{t_{j+1}}^{(j)}(u) = \mathbf{1}[u \in \mathcal{A}_{j+1}].$$

Suffix density, in contrast, measures acceptance by the entire remaining program and gives each exit state its own suffix density. When the interval is fixed, we write σ_t .

The interval index matters at a shared checkpoint t_j . As the exit layer of I_{j-1} ,

$$\sigma_{t_j}^{(j-1)}(u) = \mathbf{1}[u \in \mathcal{A}_j],$$

while as the entrance layer of I_j ,

$$\sigma_{t_j}^{(j)}(u) = \Pr_{Z \sim U_j} [P_{j,u}(Z) \in \mathcal{A}_{j+1}].$$

Conditioning on the next bit gives the averaging identity. At a free transition t ,

$$\sigma_{t-1}(u) = \frac{1}{2} \left(\sigma_t(T_t^0(u)) + \sigma_t(T_t^1(u)) \right).$$

At a fixed transition,

$$\sigma_{t-1}(u) = \sigma_t(T_t(u)).$$

Choosing the branch. Every allowed entrance state s has $\sigma_{t_j}(s) > 0$, but this probability can be arbitrarily small. We fix a threshold $\theta_j \in (0, \mu/12]$ for the interval:

- if $\sigma_{t_j}(s) \geq \theta_j$, the repair uses $\text{Repair-Interval}_{j,\theta_j}^{\text{high}}$;
- if $\sigma_{t_j}(s) < \theta_j$, it uses $\text{Repair-Interval}_j^{\text{low}}$, described in Step 4.

The scanning repair. $\text{Repair-Interval}_{j,\theta_j}^{\text{high}}$ keeps the candidate bit whenever the next state still has interval acceptance probability at least θ_j , and flips it otherwise. At a free transition t , for a current state u with $\sigma_{t-1}(u) \geq \theta_j$, its output function is

$$\beta_{t,\theta_j}(u, b) = \begin{cases} b, & \sigma_t(T_t^b(u)) \geq \theta_j, \\ 1 - b, & \sigma_t(T_t^b(u)) < \theta_j. \end{cases}$$

A flip always leads back above the threshold. If $\sigma_{t-1}(u) \geq \theta_j$ and $\sigma_t(T_t^b(u)) < \theta_j$, the averaging identity gives

$$\sigma_t(T_t^{1-b}(u)) = 2\sigma_{t-1}(u) - \sigma_t(T_t^b(u)) > \theta_j.$$

Fixed transitions preserve σ without any edit. Starting from $\sigma_{t_j}(s) \geq \theta_j$, the repaired computation therefore stays at states with $\sigma \geq \theta_j$. In particular, it never rejects, since $\sigma(\perp) = 0$. At the exit layer, $\sigma \geq \theta_j > 0$ means $\sigma = 1$, so the repaired exit state lies in $\mathcal{A}_{j+1}$.

This rule also matches Step 2. The repaired and unchanged computations agree before the first transition at which the unchanged computation reaches a state with $\sigma < \theta_j$. The first flip occurs at this transition, and the candidate prefix w read through it satisfies $p(w) < \theta_j \leq \mu/12$. Thus the gain estimate of Step 2 applies.

Choosing the threshold. Fix a free transition t and a state u at layer $t-1$, and write

$$a_u = \sigma_t(T_t^0(u)), \quad b_u = \sigma_t(T_t^1(u))$$

for the chances of reaching an allowed exit along its two branches. A flip from u at t requires two conditions.

- (i) **The repaired computation must be at u .** It only visits states at or above the threshold, so u can be its current state at layer $t-1$ only if

$$\sigma_{t-1}(u) \geq \theta_j.$$

By the averaging identity, $\sigma_{t-1}(u) = \frac{1}{2}(a_u + b_u)$, so this condition reads

$$\theta_j \leq \frac{a_u + b_u}{2}.$$

- (ii) **Some candidate bit must lead below the threshold.** The repair flips bit b at u exactly when $\sigma_t(T_t^b(u)) < \theta_j$. A flip is possible for some candidate bit exactly when the worse of the two branches is below the threshold:

$$\min\{a_u, b_u\} < \theta_j.$$

Combining the threshold condition with the existence of a triggering bit, the local scanning decision permits a flip from u at transition t exactly when

$$\min\{a_u, b_u\} < \theta_j \leq \frac{a_u + b_u}{2},$$

that is, when θ_j lies in the window

$$W_{t,u} = \left(\min\{a_u, b_u\}, \frac{a_u + b_u}{2} \right].$$

We form these windows for all three live states, whether or not they arise in a particular repaired computation. The window lies between the worse branch and the average of the two branches, so its length is

$$\frac{a_u + b_u}{2} - \min\{a_u, b_u\} = \frac{|a_u - b_u|}{2}.$$

In particular, the window is empty when the two branches have equal chances, as at every fixed transition viewed as two identical branches. The three-state structure and the collision bound limit the total length of all windows. Averaging over $\theta \in (0, \mu/12]$ then yields a threshold θ_j for which all possible flips lie in a common set of $O(\ell)$ transitions, independent of the entrance state and the block. This supplies the collision bound required for Q_j . The window and potential argument is proved below.

4. **Repair the remaining entrance states and assemble the program.** It remains to repair from allowed entrance states s with

$$\sigma_{t_j}(s) < \theta_j,$$

and to combine both branches into one three-state program.

A low entrance has a unique allowed exit. Suppose first that every forward-reachable live exit state of I_j is allowed. Every live exit state reached from s is forward-reachable, so each outcome $P_{j,s}(Z)$ is either rejection or a state of $\mathcal{A}_{j+1}$. Conditioning on the exit state of the interval gives

$$\begin{aligned} \mu_{t_j}^{\text{suffix}}(s) &= \sum_{\substack{v \in \mathcal{A}_{j+1} \\ Z \sim U_j}} \Pr[P_{j,s}(Z) = v] \mu_{t_{j+1}}^{\text{suffix}}(v) \\ &\leq \sum_{v \in \mathcal{A}_{j+1}} \Pr_{Z \sim U_j}[P_{j,s}(Z) = v] \\ &= \Pr_{Z \sim U_j}[P_{j,s}(Z) \in \mathcal{A}_{j+1}] \\ &= \sigma_{t_j}(s). \end{aligned}$$

The equality holds because rejection contributes 0 and every other outcome lies in $\mathcal{A}_{j+1}$. The inequality uses $\mu_{t_{j+1}}^{\text{suffix}}(v) \leq 1$. Since s is a low entrance state,

$$\sigma_{t_j}(s) < \theta_j \leq \frac{\mu}{12},$$

so $\mu_{t_j}^{\text{suffix}}(s) < \mu/2$. This contradicts $s \in \mathcal{A}_j$. Hence some forward-reachable live exit state is disallowed. By the two-state checkpoint condition and Lemma 6.5, the allowed set then consists of a single state:

$$\mathcal{A}_{j+1} = \{e\}.$$

The low branch. For such s , let

$$\text{Repair-Interval}_j^{\text{low}}(z, s) = R_{j,s}^{\text{reward}}(z)$$

be the terminal-reward repair of Section 6.4. Its expected terminal reward is $\mu_{t_j}^{\text{suffix}}(s) \geq \mu/2$, so it reaches an exit state of suffix density at least $\mu/2$, which must be e . It changes at most

$$r_{\text{reward}}(\ell, \mu) = O(\ell)$$

free coordinates. Unlike the scanning branch, it may inspect the whole block.

The local repair. For $s \in \mathcal{A}_j$ and $z \in \Omega_j$, the two branches together define

$$\text{Repair}_{\theta_j, j, s}(z) = \begin{cases} \text{Repair-Interval}_{j, \theta_j}^{\text{high}}(z, s), & \sigma_{t_j}(s) \geq \theta_j, \\ \text{Repair-Interval}_j^{\text{low}}(z, s), & \sigma_{t_j}(s) < \theta_j. \end{cases}$$

This is the repair constructed in Section 7.2. Both branches use the original transitions of P and change only free coordinates. In both cases the repaired exit state $E_{\theta_j, j, s}(z)$ lies in $\mathcal{A}_{j+1}$.

Completing Q_j . On states reached from high entrance states, Q_j follows the scanning transitions, and its other transitions are completed as in Step 1. Its last layer depends on the exit structure of I_j .

- **Some forward-reachable live exit state is disallowed.** Then $\mathcal{A}_{j+1} = \{e\}$, so every repaired computation of I_j , from either branch, ends at e . From a high entrance state, Q_j already ends at e . From a low entrance state, however, Q_j does not follow the actual repair, so its last state may differ. To fix this, we add one fixed transition at the end of Q_j . It reads no input bit and sends every state to e :

$$u \mapsto e \quad \text{for every state } u.$$

The auxiliary checkpoint corresponding to t_{j+1} is placed after this fixed transition, so its state is always e . This *reset* exists only in Q_j . It changes nothing from high entrance states, and it adds at most one colliding layer to the interval.

- **Otherwise.** The original exit layer serves as the checkpoint of Q_j .

With this choice, Q_j started at any allowed entrance state s and run on x ends at the actual repaired exit state:

$$Q_j(s; x)_{\text{exit}} = E_{\theta_j, j, s}(x).$$

In the first case this holds by the reset. In the second case all entrance states are high, as shown above, and Q_j follows the scanning repair to its exit. In both cases the maps of Q_j are independent of the entrance state, so it never needs to know which entrance state it started from. The actual repair still uses the original transitions of P .

The charge. For $s \in \mathcal{A}_j$ and $x \in \Omega_j$, set

$$\begin{aligned} c_{\theta_j, j, s}(x) = & \sum_{\substack{t, u, b: \sigma_{t-1}(u) \geq \theta_j \\ \beta_{t, \theta_j}(u, b) \neq b}} O_{t-1, u}(x) \cdot \mathbf{1}[x_t = b] \\ & + r_{\text{reward}}(\ell, \mu) \cdot \mathbf{1}[\sigma_{t_j}(s) < \theta_j], \end{aligned}$$

where $O_{t,u}$ are the visit events of Q_j started at s . The sum ranges over free transitions t , live states u at layer $t - 1$, and bits $b \in \{0, 1\}$ satisfying the displayed conditions. Thus it counts only the designated scanning steps. The charge covers the edits:

$$d_H(x, \text{Repair}_{\theta_j, j, s}(x)) \leq c_{\theta_j, j, s}(x).$$

From a high entrance state, the first term is exactly the number of flips and the second term vanishes. From a low entrance state, the second term covers all actual edits, and the first term may overcount, which is harmless. Both terms have the form required by the visit bound. The first is a sum of visits followed by a specified bit, and the second is a weighted visit to the entrance state at the checkpoint.

The global program. Composing $Q_0, \dots, Q_{m-1}$ from s_0 gives a single program $\hat{P}$ reading the candidate x . Let $\hat{t}_j$ be its auxiliary checkpoint corresponding to the original checkpoint t_j , after any reset, and let $S_{\theta_j}(x)$ denote the actual repaired checkpoint state. Then

$$\hat{P}(x)_{\hat{t}_j} = S_{\theta_j}(x) \in \mathcal{A}_j.$$

It satisfies all conditions of the visit bound stated in Step 1.

- **States.** It has three states in every layer and never rejects.
- **Checkpoints.** At each checkpoint, its reachable states lie in $\mathcal{A}_j \subseteq R_{t_j}(P)$, so at most two of them are reachable.
- **Size.** It has $O(n)$ transition layers: those of P , plus at most one reset per interval.
- **Collisions.** Each interval has $O(\ell)$ colliding layers: those of P , the transitions where a flip is possible, and possibly the reset.

The total charge

$$\text{Cost}_{\theta}^P(x) = \sum_{j=0}^{m-1} c_{\theta_j, j, S_{\theta_j}(x)}(x_j)$$

is therefore a weighted sum of visit events of $\hat{P}$, each possibly multiplied by the indicator of one input bit.

5. **Bound the expected charge under uniform input.** Fix an interval I_j and an allowed entrance state $s \in \mathcal{A}_j$. The repair acts only after the unchanged computation becomes unlikely to reach an allowed exit. Let

$$\mathcal{D}_{j,s} = \left\{ z \in \Omega_j : \begin{array}{l} \text{the unchanged computation from } s \text{ on } z \\ \text{visits a state with } \sigma < \theta_j \end{array} \right\},$$

where the entrance state s itself counts as a visited state. In particular, $\mathcal{D}_{j,s} = \Omega_j$ when s is a low entrance state.

Outside $\mathcal{D}_{j,s}$, nothing happens. If $z \notin \mathcal{D}_{j,s}$, the unchanged computation stays at states with $\sigma \geq \theta_j$, so the scanning repair never flips a bit. The block is unchanged, and

$$c_{\theta_j, j, s}(z) = 0$$

together with

$$g_{\theta_j, j, s}(z) = 0.$$

On $\mathcal{D}_{j,s}$, the charge is bounded. The scanning contribution to the charge counts at most one repair step at each of the at most $L_{\text{repair}}(\ell, \mu)$ possible repair transitions of Step 3, and a low entrance state adds the allowance $r_{\text{reward}}(\ell, \mu)$. Hence

$$c_{\theta_j, j, s}(z) \leq K_{\text{loc}} = L_{\text{repair}}(\ell, \mu) + r_{\text{reward}}(\ell, \mu) = O(\ell).$$

Therefore

$$\mathbb{E}_{Z \sim U_j} [c_{\theta_j, j, s}(Z)] \leq K_{\text{loc}} \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j, s}].$$

On $\mathcal{D}_{j, s}$, the expected gain is large. Split $\mathcal{D}_{j, s}$ according to the first layer t at which the unchanged computation is at a state with $\sigma < \theta_j$, and the prefix $w = z_{\leq t}$ read up to that layer. These prefix events are disjoint and together form $\mathcal{D}_{j, s}$. For each of them,

$$p(w) = \sigma_t(v) < \theta_j \leq \frac{\mu}{12},$$

where v is the state reached after w . For a low entrance state, $t = t_j$ and w is empty. Every repaired computation ends in $\mathcal{A}_{j+1}$, so Step 2 applies to each prefix event:

$$\mathbb{E}_{Z \sim U_j} [g_{\theta_j, j, s}(Z) \mid Z_{\leq t} = w] \geq \frac{3\mu}{8}.$$

Summing over the disjoint prefix events, and using that the gain is 0 outside $\mathcal{D}_{j, s}$, gives

$$\mathbb{E}_{Z \sim U_j} [g_{\theta_j, j, s}(Z)] \geq \frac{3\mu}{8} \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j, s}].$$

Conclusion. Combining the two bounds, the local cost–gain estimate holds with

$$B = \frac{8K_{\text{loc}}}{3\mu} = O(\ell),$$

since $\mu \geq 1/16$:

$$\mathbb{E}_{Z \sim U_j} [c_{\theta_j, j, s}(Z)] \leq B \mathbb{E}_{Z \sim U_j} [g_{\theta_j, j, s}(Z)].$$

The accounting of Lemmas 6.3 and 6.4 then gives

$$\mathbb{E}_{X \sim U} [\text{Cost}_{\theta}^P(X)] \leq B(1 - \mu) = O(\ell).$$

This bounds the cost of the new repair, the one whose charge was written as a sum of visit events in Step 4. The earlier bound on Cost^P concerns a different repair and is not used here.

Section 7.2 constructs the local repairs and their auxiliary representation. Section 7.3 proves the threshold selection lemma. Sections 7.4 and 7.5 prove their uniform cost bound. The transfer to small bias is carried out in Section 8.

7.2 Constructing the structured local repairs

We construct the local repairs $\text{Repair}_{\theta_j, j, s}$, their charges $c_{\theta_j, j, s}$, and auxiliary interval programs Q_j , using the allowed states $\mathcal{A}_j$ of Section 6.4. The construction establishes the endpoint, distance, and representation guarantees in Lemma 7.4; Section 7.4 proves its expected-charge and gain bounds.

For the structured construction, let I_j have an integer collision bound $c \geq 1$; globally we take $c = \ell$, including for intervals without collisions.

Step 1: define the scanning repair and choose its threshold. We first construct the repair for entrance states whose probability of reaching $\mathcal{A}_{j+1}$ is at least a threshold $\theta_j > 0$.

Definition 7.1 (interval repair procedure). Fix an interval I_j , the set $\mathcal{A}_{j+1}$ of allowed exit states, and a threshold $\theta_j > 0$. An interval repair procedure for high-threshold entrances is a deterministic map

$$\text{Repair-Interval}_{j, \theta_j}^{\text{high}} : \Omega_j \times \left\{ s \in \mathcal{A}_j : \Pr_{Z \sim U_j} [P_{j, s}(Z) \in \mathcal{A}_{j+1}] \geq \theta_j \right\} \longrightarrow \Omega_j$$

such that, for every candidate block $z \in \Omega_j$ and allowed entrance state s satisfying this threshold condition, the repaired block

$$y = \text{Repair-Interval}_{j,\theta_j}^{\text{high}}(z, s)$$

satisfies

$$P_{j,s}(y) \in \mathcal{A}_{j+1}.$$

Both z and y assign only the free coordinates of I_j , so fixed transitions remain unchanged.

Definition 7.2 (interval acceptance probability). Regard $\mathcal{A}_{j+1}$ as the set of accepting exit states of I_j . For a state u in layer t , with $t_j \leq t \leq t_{j+1}$, let $\sigma_t(u)$ be the probability that the original computation from u in layer t is in $\mathcal{A}_{j+1}$ at layer t_{j+1} , when the remaining free bits of I_j are independent and uniform. Fixed transitions remain unchanged. Set $\sigma_t(\perp) = 0$. The notation refers to the fixed interval I_j . When the interval must be explicit, we write $\sigma_t^{(j)}(u)$; probabilities for different intervals need not agree at a shared checkpoint.

Transition t maps layer $t - 1$ to layer t , as in Section 3.1. At the exit layer, the definition gives

$$\sigma_{t_{j+1}}(u) = \begin{cases} 1, & u \in \mathcal{A}_{j+1}, \\ 0, & u \notin \mathcal{A}_{j+1}. \end{cases}$$

At a free transition t , conditioning on the uniform input bit gives the averaging identity

$$\sigma_{t-1}(u) = \frac{1}{2}(\sigma_t(T_t^0(u)) + \sigma_t(T_t^1(u))).$$

At a fixed transition t , the next state is determined, so $\sigma_{t-1}(u) = \sigma_t(T_t(u))$.

At a free transition t , define the *output function*

$$\beta_{t,\theta_j}(u, b) = \begin{cases} b, & \sigma_t(T_t^b(u)) \geq \theta_j, \\ 1 - b, & \sigma_t(T_t^b(u)) < \theta_j, \end{cases}$$

for states u satisfying $\sigma_{t-1}(u) \geq \theta_j$ and candidate bits $b \in \{0, 1\}$. A *repair step* occurs when $\beta_{t,\theta_j}(u, b) \neq b$. Algorithm 3 implements this function and follows the original maps at fixed transitions.

Algorithm 3. $\text{Repair-Interval}_{j,\theta_j}^{\text{high}}(z, s)$

Fixed data: the interval I_j , its transition maps, the probabilities σ_t , and a threshold $\theta_j > 0$.

Input: a candidate block $z \in \Omega_j$ and an allowed entrance state $s \in \mathcal{A}_j$ with $\sigma_{t_j}(s) \geq \theta_j$.

Output: a repaired block $y \in \Omega_j$ with $P_{j,s}(y) \in \mathcal{A}_{j+1}$.

```

1   $u \leftarrow s, \quad y \leftarrow z.$ 
2  for  $t = t_j + 1, \dots, t_{j+1}$  do
3    if transition  $t$  is free then
4      if  $\sigma_t(T_t^{z_t}(u)) < \theta_j$  then
5         $y_t \leftarrow 1 - z_t.$ 
6      end if
7       $u \leftarrow T_t^{y_t}(u).$ 
8    else
9       $u \leftarrow T_t(u).$ 
10   end if
11 end for
12 return  $y.$ 
```

The averaging identity ensures that Algorithm 3 maintains $\sigma \geq \theta_j$. If $\sigma_{t-1}(u) \geq \theta_j$ and the candidate bit b leads below the threshold, the other successor satisfies

$$\sigma_t(T_t^{1-b}(u)) = 2\sigma_{t-1}(u) - \sigma_t(T_t^b(u)) > \theta_j.$$

Fixed transitions preserve σ , so the invariant holds throughout and the computation never rejects. At the exit layer, σ is zero outside $\mathcal{A}_{j+1}$, so the repaired exit state is allowed. By induction, the repaired and unchanged computations agree until the unchanged computation first reaches a state with $\sigma < \theta_j$. This first departure cannot occur at a fixed transition; at that free transition the algorithm flips the candidate bit. If no departure occurs, the block remains unchanged. Each decision depends only on the current layer, current state, and candidate bit, through β_{t,θ_j} .

The following threshold selection lemma gives a single threshold and a common set of possible repair positions. We prove it in Section 7.3.

Lemma 7.3 (threshold selection). *Let the interval I_j contain at most $c \geq 1$ colliding transitions. There exist a threshold*

$$\theta_j \in \left(0, \frac{\mu}{12}\right]$$

and a set E_j of transitions of I_j , both depending only on the program, with the following properties.

1. **Few transitions.** *The set E_j is small:*

$$|E_j| \leq L_{\text{repair}}(c, \mu) = c + \frac{180(c+1)}{\mu}. \quad (7.1)$$

2. **Collisions are included.** *E_j contains every colliding transition of I_j .*

3. **No flips outside E_j .** *For every free transition $t \notin E_j$, every state u with $\sigma_{t-1}(u) \geq \theta_j$, and every bit b ,*

$$\beta_{t,\theta_j}(u, b) = b.$$

Consequently, let $s \in \mathcal{A}_j$ be an allowed entrance state with $\sigma_{t_j}(s) \geq \theta_j$, let $z \in \Omega_j$ be a candidate block, and let

$$y = \text{Repair-Interval}_{j,\theta_j}^{\text{high}}(z, s).$$

Then the following hold.

- **It ends in $\mathcal{A}_{j+1}$.** *The repaired computation stays at states with interval acceptance probability at least θ_j , so y is well defined, and it ends in $\mathcal{A}_{j+1}$.*
- **It changes few bits.** *The block y differs from z only at free transitions in E_j , so*

$$d_H(z, y) \leq L_{\text{repair}}(c, \mu).$$

- **Fixed transitions are unchanged.**

The threshold θ_j and the set E_j work simultaneously for all such s and z . Since $\mu \geq 1/16$, the bound $L_{\text{repair}}(c, \mu) = O(c)$ is uniform in μ .

Step 2: repair the block from each allowed entrance state. Fix, for the rest of the construction, the threshold θ_j and the set E_j given by Lemma 7.3. They depend only on the program. For $s \in \mathcal{A}_j$ and $z \in \Omega_j$, define the full local repair by

$$\text{Repair}_{\theta_j, j, s}(z) = \begin{cases} \text{Repair-Interval}_{j,\theta_j}^{\text{high}}(z, s), & \sigma_{t_j}(s) \geq \theta_j, \\ R_{j,s}^{\text{reward}}(z), & \sigma_{t_j}(s) < \theta_j, \end{cases}$$

where $R_{j,s}^{\text{reward}}$ is the terminal-reward repair of Section 6.4, with its deterministic tie-breaking fixed there. We also denote the second branch by $\text{Repair-Interval}_j^{\text{low}}(z, s)$.

For an entrance state with $\sigma_{t_j}(s) \geq \theta_j$, Lemma 7.3 applies to the first branch with this threshold. For every block $z \in \Omega_j$, it gives three facts:

- the repaired computation ends in $\mathcal{A}_{j+1}$;
- the repaired block differs from z only at free transitions in E_j , so

$$d_H(z, \text{Repair}_{\theta_j, j, s}(z)) \leq L_{\text{repair}}(c, \mu);$$

- fixed transitions are unchanged.

It remains to treat the low branch, and to build the auxiliary program. For both we need the following classification of exit states.

Let

$$R_{\text{next}} = R_{t_{j+1}}(P)$$

be the set of forward-reachable live states at the exit layer. By the proof of Lemma 6.5, some state of R_{next} has suffix density at least μ . Since $|R_{\text{next}}| \leq 2$, exactly one of the following holds:

- (A) every reachable live exit state is allowed, that is, $\mathcal{A}_{j+1} = R_{\text{next}}$;
- (B) $R_{\text{next}} = \{e, v_{\text{low}}\}$ and $\mathcal{A}_{j+1} = \{e\}$, where $\mu_{t_{j+1}}^{\text{suffix}}(e) \geq \mu$ and $\mu_{t_{j+1}}^{\text{suffix}}(v_{\text{low}}) < \mu/2$.

A low entrance state can occur only in case (B). Indeed, let $s \in \mathcal{A}_j$ with $\sigma_{t_j}(s) < \theta_j$, and suppose case (A) holds. Every live exit state reached from s is forward-reachable, so each outcome $P_{j,s}(Z)$ is either rejection or a state of $\mathcal{A}_{j+1}$. Conditioning on the exit state of the interval gives

$$\begin{aligned} \mu_{t_j}^{\text{suffix}}(s) &= \sum_{v \in \mathcal{A}_{j+1}} \Pr_{Z \sim U_j} [P_{j,s}(Z) = v] \mu_{t_{j+1}}^{\text{suffix}}(v) \\ &\leq \sum_{v \in \mathcal{A}_{j+1}} \Pr_{Z \sim U_j} [P_{j,s}(Z) = v] \\ &= \Pr_{Z \sim U_j} [P_{j,s}(Z) \in \mathcal{A}_{j+1}] \\ &= \sigma_{t_j}(s). \end{aligned}$$

The equality holds because rejection contributes 0 and every other outcome lies in $\mathcal{A}_{j+1}$, and the inequality uses $\mu_{t_{j+1}}^{\text{suffix}}(v) \leq 1$. Since $\sigma_{t_j}(s) < \theta_j \leq \mu/12$, this gives $\mu_{t_j}^{\text{suffix}}(s) < \mu/2$, contradicting $s \in \mathcal{A}_j$. Hence case (B) holds, and e is the only allowed exit state.

For a low entrance state s , the expected terminal reward of the original interval is $\mu_{t_j}^{\text{suffix}}(s) \geq \mu/2$, so the reward repair reaches an exit state of suffix density at least $\mu/2$, which by case (B) is e . By the bound of Section 6.4, it changes at most

$$r_{\text{reward}}(c, \mu) = O(c) \tag{7.2}$$

free coordinates, and it preserves all fixed transitions.

Thus, in both branches, the repaired exit state

$$E_{\theta_j, j, s}(z) = P_{j,s}(\text{Repair}_{\theta_j, j, s}(z))$$

belongs to $\mathcal{A}_{j+1}$. Its comparative gain is

$$g_{\theta_j, j, s}(z) = \mu_{t_{j+1}}^{\text{suffix}}(E_{\theta_j, j, s}(z)) - \mu_{t_{j+1}}^{\text{suffix}}(P_{j,s}(z)).$$

Step 3: represent the repaired computation by a three-state program. We construct one program Q_j on the interval I_j with the following properties. It has three states in every layer and never rejects. It uses the same transition maps whatever entrance state it starts from. From every allowed entrance state $s \in \mathcal{A}_j$, its exit state on a block z is the repaired exit state $E_{\theta_j, j, s}(z)$. From high entrance states, it moreover reproduces the repaired computation step by step.

The states of Q_j at each original layer are the three live states of P at that layer. Its transitions are defined in three stages.

First, we prescribe the transitions used by the repaired computation of Algorithm 3. At a free transition t , from a state u with $\sigma_{t-1}(u) \geq \theta_j$, on input bit b , set

$$\delta_t^b(u) = T_t^{\beta_{t,\theta_j}(u,b)}(u).$$

At a fixed transition, set $\delta_t(u) = T_t(u)$ for such states u . By Lemma 7.3, these successors have interval acceptance probability at least $\theta_j > 0$, so they are live.

Second, at every transition $t \notin E_j$ we extend these maps to permutations. For each bit b , let

$$D_t^b = \{u : T_t^b(u) \neq \perp\}$$

be the set of states from which the original map does not reject. Since t is not colliding, T_t^b is injective on D_t^b . We extend it to a permutation of the three live states by sending the states outside D_t^b bijectively onto the states not in $T_t^b(D_t^b)$. This extension agrees with the prescribed transitions. Indeed, at a free transition $t \notin E_j$, no flip is possible, so every state u with $\sigma_{t-1}(u) \geq \theta_j$ satisfies $\beta_{t,\theta_j}(u,b) = b$ and has live successors under both bits. Hence $u \in D_t^b$, and the extension sends u to $T_t^b(u)$, as prescribed. Fixed noncolliding transitions are treated in the same way, with $T_t^0 = T_t^1 = T_t$.

Third, at the transitions of E_j , the prescribed maps are completed arbitrarily to maps into the three live states. All completions are fixed deterministically from the program.

Finally, the last layer of Q_j depends on the exit structure of I_j . In case (B), we append one fixed transition that reads no input bit and sends every state to e :

$$u \mapsto e \quad \text{for every state } u,$$

and the checkpoint of Q_j is placed after this *reset*. In case (A), the original exit layer serves as the checkpoint.

The construction keeps the number of collisions small. Outside E_j every map of Q_j is a permutation, so collisions can occur only at transitions of E_j and at the reset. By Lemma 7.3, Q_j therefore has at most

$$L_{\text{repair}}(c, \mu) + 1$$

colliding layers.

It remains to check that Q_j ends at the repaired exit state. Let s be a high entrance state, $\sigma_{t_j}(s) \geq \theta_j$. By induction over the layers of I_j , Q_j started at s and run on z is at every layer in the same state as P run on $\text{Repair-Interval}_{j,\theta_j}^{\text{high}}(z, s)$: at each transition it applies the prescribed map to a state with $\sigma \geq \theta_j$. In case (B) this computation already ends at e , the only allowed exit state, so the reset leaves it there. Now let s be a low entrance state, $\sigma_{t_j}(s) < \theta_j$. Then case (B) holds, and the reset sends Q_j to e , which is also the exit state of the terminal-reward repair. In both cases,

$$Q_j(s; z)_{\text{exit}} = E_{\theta_j, j, s}(z).$$

The reset exists only in Q_j . The actual repair reaches its exit state through the original transitions of P .

Step 4: define the charges. Let $N_{j,s}(z)$ count designated repair steps along Q_j 's run from s on z : visits at transition t to a state u with $\sigma_{t-1}(u) \geq \theta_j$, when the candidate bit b satisfies $\sigma_t(T_t^b(u)) < \theta_j$. Arbitrary completions from states with $\sigma_{t-1}(u) < \theta_j$ and the final reset do not themselves count as repair steps. Set

$$c_{\theta_j, j, s}(z) = \begin{cases} N_{j,s}(z), & \sigma_{t_j}(s) \geq \theta_j, \\ N_{j,s}(z) + r_{\text{reward}}(c, \mu), & \sigma_{t_j}(s) < \theta_j, \end{cases} \quad (7.3)$$

$$K_{\text{loc}}(c, \mu) = L_{\text{repair}}(c, \mu) + r_{\text{reward}}(c, \mu) = O(c).$$

When $\sigma_{t_j}(s) \geq \theta_j$, $N_{j,s}(z)$ is exactly the number of bit changes made by $\text{Repair-Interval}_{j,\theta_j}^{\text{high}}(z, s)$. When $\sigma_{t_j}(s) < \theta_j$, the allowance $r_{\text{reward}}(c, \mu)$ covers the edits made by the terminal-reward repair; counting the

auxiliary repair steps as well may overestimate the cost, which is harmless. Each auxiliary run makes at most one designated repair step per transition of E_j . Therefore

$$d_H(z, \text{Repair}_{\theta_j, j, s}(z)) \leq c_{\theta_j, j, s}(z) \leq K_{\text{loc}}(c, \mu).$$

Each designated repair step is a source-state visit together with a specified value of the next free bit. The allowance for $\sigma_{t_j}(s) < \theta_j$ is a checkpoint state-visit charge. Section 7.5 composes these auxiliary intervals and charges into a single program.

7.3 Proof of the threshold selection lemma

Proof of Lemma 7.3. Transition indices range over $t_j < t \leq t_{j+1}$. Label the three live states of each layer by $[3] = \{1, 2, 3\}$, including states unreachable from the original start. Rejection is represented separately by $\perp$. Set $\sigma_t(\perp) = 0$, and write $T_t^0 = T_t^1 = T_t$ at a fixed transition. Let C be the set of colliding transitions of I_j , so $|C| \leq c$.

Plan. For each transition t and state u , part (a) defines an interval $W_{t,u} \subseteq (0, 1]$, called a *window*, such that a flip from u at t is possible exactly when

$$\theta_j \in W_{t,u}.$$

For $\theta > 0$, call a transition t a *repair layer* if $\theta \in W_{t,u}$ for some u , and let $N(\theta)$ count the repair layers outside C . These are the transitions outside C at which a flip is possible. We will take

$$E_j = C \cup \{t : \theta_j \in W_{t,u} \text{ for some } u\},$$

so that $|E_j| \leq c + N(\theta_j)$. The lemma therefore follows once we find a threshold $\theta_j \in (0, \mu/12]$ with

$$N(\theta_j) \leq \frac{180(c+1)}{\mu}.$$

To find it, let Θ be uniform on $(0, \mu/12]$. A transition $t \notin C$ is counted in $N(\Theta)$ exactly when at least one of its windows contains Θ , so its contribution is at most the number of its windows containing Θ :

$$N(\Theta) \leq \sum_{t \notin C} \sum_u \mathbf{1}[\Theta \in W_{t,u}].$$

Since Θ is uniform on $(0, \mu/12]$, each window satisfies

$$\Pr[\Theta \in W_{t,u}] = \frac{\text{len}(W_{t,u} \cap (0, \mu/12])}{\mu/12} \leq \frac{12}{\mu} \text{len}(W_{t,u}),$$

so taking expectations gives

$$\mathbb{E}[N(\Theta)] \leq \sum_{t \notin C} \sum_u \Pr[\Theta \in W_{t,u}] \leq \frac{12}{\mu} \sum_{t \notin C} \sum_u \text{len}(W_{t,u}).$$

For some $\theta_j \in (0, \mu/12]$, the value $N(\theta_j)$ is at most this expectation. So it suffices to bound the total window length:

$$\sum_{t \notin C} \sum_u \text{len}(W_{t,u}) \leq 15(c+1), \tag{\dagger}$$

which gives a threshold with $N(\theta_j) \leq \frac{12}{\mu} \cdot 15(c+1) = \frac{180(c+1)}{\mu}$.

To bound the total window length, we use a potential Φ of the three probabilities at a layer, defined in part (d) as a weighted combination of their sum and their range. Each probability lies in $[0, 1]$, so their sum lies in $[0, 3]$ and their range in $[0, 1]$, and the weights make

$$0 \leq \Phi \leq 15.$$

We show that every noncolliding transition satisfies

$$\Phi(\sigma_t) - \Phi(\sigma_{t-1}) \geq \sum_u \text{len}(W_{t,u}).$$

Summing these decreases over the noncolliding transitions telescopes. The total is at most the change of Φ over the whole interval, which is at most 15, plus the possible increases at the at most c colliding transitions, at most 15 each. This gives (†).

(a) The windows. We first describe exactly which thresholds allow a flip at a given state. For a transition t and a state $u \in [3]$, write

$$a_u = \sigma_t(T_t^0(u))$$

and

$$b_u = \sigma_t(T_t^1(u))$$

for the probabilities of reaching an allowed exit along the two branches. By the averaging identity, $\sigma_{t-1}(u) = \frac{1}{2}(a_u + b_u)$.

The output function can flip a candidate bit from u exactly when u satisfies the threshold condition $\sigma_{t-1}(u) \geq \theta$ and some branch leads below the threshold, meaning $\min\{a_u, b_u\} < \theta$. That is, θ lies in the window

$$W_{t,u} = \left(\min\{a_u, b_u\}, \frac{a_u + b_u}{2} \right],$$

whose length is

$$\text{len}(W_{t,u}) = \frac{1}{2}|a_u - b_u|.$$

Windows are defined for all states, whether or not a given repaired computation reaches them; this is what makes the final set independent of the entrance state and the block. At a fixed transition $a_u = b_u$, so every window is empty.

(b) The threshold and the set E_j . Assume (†), which is proved in parts (c)–(e). By the plan, some threshold $\theta_j \in (0, \mu/12]$ satisfies

$$N(\theta_j) \leq \frac{180(c+1)}{\mu},$$

and the set E_j defined there satisfies $|E_j| \leq c + 180(c+1)/\mu$. It contains every colliding transition. At a free transition outside E_j , no window contains θ_j , so $\beta_{t,\theta_j}(u, b) = b$ for every state u with $\sigma_{t-1}(u) \geq \theta_j$ and every bit b . Algorithm 3 keeps $\sigma \geq \theta_j$, so every bit it changes lies at a transition where a flip is possible, that is, in E_j , and it changes at most one bit per transition. Since the windows were defined for all states, this holds for every entrance state and block in its domain.

(c) Averaging pulls the values together. For $v \in [0, 1]^3$, write

$$\text{range}(v) = \max_{i \in [3]} v_i - \min_{i \in [3]} v_i,$$

and let $\sigma_t \in [0, 1]^3$ denote the vector of interval acceptance probabilities $(\sigma_t(1), \sigma_t(2), \sigma_t(3))$ at layer t .

We will compare the probability vectors σ_t and σ_{t-1} at a noncolliding transition t . By the averaging identity, each entry $\sigma_{t-1}(u)$ is the average of σ_t at the two successors of u . If no branch rejected, the two branch maps would be permutations π_0, π_1 of the three states, since t is noncolliding, and we would have

$$\sigma_{t-1}(u) = \frac{1}{2} \left(\sigma_t(\pi_0(u)) + \sigma_t(\pi_1(u)) \right).$$

In general some branches reject. Part (d) handles them by first computing this average as if no branch rejected, and then lowering the entries where a branch actually rejects. This part studies the first part, the averaging. Each new entry is the midpoint of two old entries, so the averaging pulls the three values closer together. The following claim quantifies this, and it is the only place where we use that there are three states.

Claim. Let $v \in [0, 1]^3$ and let π_0, π_1 be permutations of $[3]$. For each $u \in [3]$, let

$$q_u = \frac{1}{2}(v_{\pi_0(u)} + v_{\pi_1(u)})$$

be the average of the values at the two successors of u , and let

$$M = \max_u |v_{\pi_0(u)} - v_{\pi_1(u)}|$$

be the largest difference between the two values averaged at a single state. Then

$$\text{range}(q) \leq \text{range}(v) - \frac{M}{2}.$$

That is, averaging shrinks the spread of the values by at least half the largest difference it averages.

Proof of the claim. Reduction. Let $\tau = \pi_1 \circ \pi_0^{-1}$, a permutation of $[3]$, and substitute $w = \pi_0(u)$. As u ranges over $[3]$, so does w , and $\pi_1(u) = \tau(w)$. Hence the entries of q are, up to order, the numbers

$$p_w = \frac{1}{2}(v_w + v_{\tau(w)}), \quad w \in [3],$$

and

$$M = \max_{w \in [3]} d_w, \quad \text{where } d_w = |v_w - v_{\tau(w)}|.$$

The range depends only on the entries and not on their order, so $\text{range}(q) = \text{range}(p)$. It therefore suffices to show

$$\text{range}(p) \leq \text{range}(v) - \frac{M}{2}.$$

Each average stays away from the extremes. Let $x = \min_i v_i$ and $z = \max_i v_i$. Since p_w is the midpoint of v_w and $v_{\tau(w)}$,

$$p_w = \min\{v_w, v_{\tau(w)}\} + \frac{d_w}{2} \geq x + \frac{d_w}{2},$$

and

$$p_w = \max\{v_w, v_{\tau(w)}\} - \frac{d_w}{2} \leq z - \frac{d_w}{2}.$$

That is,

$$p_w \in \left[x + \frac{d_w}{2}, z - \frac{d_w}{2} \right]. \quad (*)$$

Case analysis. Every permutation of $[3]$ is the identity, a transposition, or a three-cycle.

- **τ is the identity.** Then $d_w = 0$ for all w , so $M = 0$, and $p = v$. The claim holds with equality.
- **τ exchanges two elements i, k and fixes m .** Then

$$p_i = p_k = \frac{1}{2}(v_i + v_k)$$

and $p_m = v_m$. Moreover $d_i = d_k = |v_i - v_k|$ and $d_m = 0$, so $M = |v_i - v_k|$. Write $h = \frac{1}{2}(v_i + v_k)$. By $(*)$,

$$x + \frac{M}{2} \leq h \leq z - \frac{M}{2},$$

and $x \leq v_m \leq z$. Hence

$$\text{range}(p) = |v_m - h| \leq \max\{z - h, h - x\} \leq z - x - \frac{M}{2}.$$

- **τ is a three-cycle.** As w ranges over $[3]$, the pairs $\{w, \tau(w)\}$ are the three distinct two-element subsets of $[3]$. So the entries of p are the three pairwise averages of the entries of v . Write the entries of v in increasing order as $x \leq y \leq z$. The averages satisfy

$$\frac{x+y}{2} \leq \frac{x+z}{2} \leq \frac{y+z}{2},$$

so

$$\text{range}(p) = \frac{y+z}{2} - \frac{x+y}{2} = \frac{z-x}{2}.$$

Since every pair is averaged, $M = z - x$, and therefore $\text{range}(p) = (z - x) - M/2$.

In all three cases $\text{range}(p) \leq (z - x) - M/2 = \text{range}(v) - M/2$. $\square$

(d) The potential pays for the windows. We now combine the total mass and the range into a potential that decreases, at every noncolliding transition, by at least the total length of that transition's windows. Averaging preserves the mass and contracts the range; deletion reduces the mass but may enlarge the range. The weights are chosen so that the loss of mass covers the possible gain in range. For $v \in [0, 1]^3$, set

$$\Phi(v) = 4 \sum_{i=1}^3 v_i + 3 \text{range}(v) \in [0, 15],$$

and write $\Delta_t = \Phi(\sigma_t) - \Phi(\sigma_{t-1})$. We show that every noncolliding transition satisfies

$$\Delta_t \geq \sum_{u \in [3]} \text{len}(W_{t,u}). \quad (\ddagger)$$

Fix $t \notin C$ and let $v = \sigma_t$. Each branch map is injective on the states from which it does not reject, so it extends to a permutation of $[3]$; choose such extensions π_0 and π_1 . The rejecting branches are accounted for by deleted amounts:

$$a_u = v_{\pi_0(u)} - \alpha_u$$

and

$$b_u = v_{\pi_1(u)} - \beta_u.$$

Here $\alpha_u = 0$ unless branch 0 rejects from u , in which case $\alpha_u = v_{\pi_0(u)}$, and similarly for β_u . With q and M as in the claim, the update splits into averaging followed by deletion:

$$\sigma_{t-1}(u) = q_u - \frac{1}{2}(\alpha_u + \beta_u),$$

and the total deleted mass is

$$D = \frac{1}{2} \sum_u (\alpha_u + \beta_u) \geq 0.$$

We now measure the effect of the two moves on each term of Φ , and compare with the windows.

- **Mass.** Averaging preserves the sum of the entries, and deletion removes D :

$$\sum_u \sigma_{t-1}(u) = \sum_i v_i - D.$$

- **Range.** Deletion lowers entries by a total of D , so the maximum does not rise and the minimum falls by at most D . With the claim,

$$\text{range}(\sigma_{t-1}) \leq \text{range}(v) - \frac{M}{2} + D.$$

- **Windows.** A window is long only if its two branches are far apart, which requires either a large averaged difference or a deletion. By the triangle inequality, $|a_u - b_u| \leq M + \alpha_u + \beta_u$. Summing over u ,

$$\sum_u \text{len}(W_{t,u}) = \frac{1}{2} \sum_u |a_u - b_u| \leq \frac{3}{2}M + D.$$

Combining the three bounds proves (‡):

$$\Delta_t \geq 4D + 3\left(\frac{M}{2} - D\right) = \frac{3}{2}M + D \geq \sum_u \text{len}(W_{t,u}).$$

(e) Summing over the interval. Finally, we add (‡) over all noncolliding transitions. The potential takes values in $[0, 15]$, so it changes by at most 15 over the whole interval and by at most 15 at each colliding transition. Hence

$$\begin{aligned} \sum_{t \notin C} \sum_{u \in [3]} \text{len}(W_{t,u}) &\leq \sum_{t \notin C} \Delta_t \\ &= \Phi(\sigma_{t_{j+1}}) - \Phi(\sigma_{t_j}) - \sum_{t \in C} \Delta_t \\ &\leq 15 + 15|C| \leq 15(c+1). \end{aligned}$$

This is (‡). The threshold invariant, the endpoint guarantee, and the preservation of fixed transitions were proved in Section 7.2. $\square$

7.4 The gain estimate under uniform input

We now prove the local bounds on the expected charge and gain for the construction of Section 7.2. Fix an interval I_j and an allowed entrance state s . The repair acts only on blocks whose unchanged computation from s becomes unlikely to reach an allowed exit. Let

$$\mathcal{D}_{j,s} = \left\{ z \in \Omega_j : \begin{array}{l} \text{the unchanged computation from } s \text{ on } z \text{ reaches a state } u \\ \text{at some layer } t \text{ with } \sigma_t(u) < \theta_j \end{array} \right\}.$$

The entrance layer t_j counts, and rejection has interval acceptance probability 0. In particular, $\mathcal{D}_{j,s} = \Omega_j$ when $\sigma_{t_j}(s) < \theta_j$. For $z \in \mathcal{D}_{j,s}$, the first such layer t is the *departure layer*, and the state u reached there is the *departure state*.

Lemma 7.4 (compatible local repair). *Let $c \geq 1$ be an integer, and let I_j have at most c colliding layers. The local repairs $\text{Repair}_{\theta_j, j, s}$ of Section 7.2 have the following properties for every allowed entrance state $s \in \mathcal{A}_j$ and every block $z \in \Omega_j$.*

1. *The repaired exit state is allowed:*

$$E_{\theta_j, j, s}(z) \in \mathcal{A}_{j+1}.$$

2. *The program Q_j of Step 3 has three states in every layer, never rejects, and uses the same maps for every entrance state. Its exit state from s on z is $E_{\theta_j, j, s}(z)$, and it has at most $L_{\text{repair}}(c, \mu) + 1$ colliding layers.*
3. *The charge $c_{\theta_j, j, s}(z)$ of (7.3) is a nonnegative integer. It is the number of visits of Q_j to specified states followed by specified input bits, plus $r_{\text{reward}}(c, \mu)$ when $\sigma_{t_j}(s) < \theta_j$. It covers the edits and is bounded:*

$$d_H(z, \text{Repair}_{\theta_j, j, s}(z)) \leq c_{\theta_j, j, s}(z) \leq K_{\text{loc}}(c, \mu).$$

4. *The charge vanishes off $\mathcal{D}_{j,s}$:*

$$c_{\theta_j, j, s}(z) = 0 \quad \text{if } z \notin \mathcal{D}_{j,s}.$$

5. *Under a uniform block, the expected charge is controlled by the probability of $\mathcal{D}_{j,s}$,*

$$\mathbb{E}_{Z \sim U_j} [c_{\theta_j, j, s}(Z)] \leq K_{\text{loc}}(c, \mu) \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}],$$

and so is the expected gain:

$$\mathbb{E}_{Z \sim U_j} [g_{\theta_j, j, s}(Z)] \geq \frac{3\mu}{8} \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}]. \quad (7.4)$$

6. *The actual repair leaves all fixed transitions unchanged.*

Proof. Items 1, 2, 3, and 6 were established in the construction of Section 7.2. It remains to prove item 4 and the two bounds of item 5. The charge bound follows from items 3 and 4 directly. For the gain bound, we first show that blocks in $\mathcal{D}_{j,s}$ rarely reach an allowed exit, and then compare gains and losses on $\mathcal{D}_{j,s}$.

The charge off $\mathcal{D}_{j,s}$. Let $z \notin \mathcal{D}_{j,s}$. Then the unchanged computation from s stays at states with interval acceptance probability at least θ_j , including at the entrance, so $\sigma_{t_j}(s) \geq \theta_j$ and the scanning branch is used. That branch never flips a bit, so

$$\text{Repair}_{\theta_j,j,s}(z) = z,$$

and its exit state is the original one:

$$E_{\theta_j,j,s}(z) = P_{j,s}(z).$$

Since Q_j reproduces the repaired computation, it visits no flip-triggering state with its triggering bit, and no reward-repair term is charged. Hence

$$c_{\theta_j,j,s}(z) = 0,$$

and also

$$g_{\theta_j,j,s}(z) = 0.$$

This proves item 4. Combined with $c_{\theta_j,j,s} \leq K_{\text{loc}}(c, \mu)$, it gives the expected-charge bound of item 5.

Blocks in $\mathcal{D}_{j,s}$ rarely reach an allowed exit. Partition $\mathcal{D}_{j,s}$ according to the departure layer t and the prefix $w = z_{\leq t}$ read up to that layer. The prefix determines the departure state u , and it also determines that no earlier departure occurred, so these prefix events are disjoint and together form $\mathcal{D}_{j,s}$. Conditioned on such a prefix, the unread free bits are still independent and uniform. By Definition 7.2, the conditional probability that the original exit state is allowed is therefore

$$\Pr_{Z \sim U_j} [P_{j,s}(Z) \in \mathcal{A}_{j+1} \mid Z_{\leq t} = w] = \sigma_t(u) < \theta_j.$$

Summing over the disjoint prefix events gives

$$\Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s} \text{ and } P_{j,s}(Z) \in \mathcal{A}_{j+1}] \leq \theta_j \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}]. \quad (7.5)$$

For a low entrance state, the departure layer is t_j and the prefix is empty, so the same argument gives $\Pr_{Z \sim U_j} [P_{j,s}(Z) \in \mathcal{A}_{j+1}] = \sigma_{t_j}(s) < \theta_j$. Since (7.5) bounds a joint probability, it also holds when $\Pr[Z \in \mathcal{D}_{j,s}] = 0$.

The gain. We now compare gains and losses on $\mathcal{D}_{j,s}$. On every block, the repaired exit state is allowed, by item 1. If the original exit state is not allowed, Lemma 6.5 gives

$$g_{\theta_j,j,s}(z) \geq \frac{\mu}{2}.$$

Otherwise, both suffix densities lie in $[0, 1]$, so

$$g_{\theta_j,j,s}(z) \geq -1.$$

Off $\mathcal{D}_{j,s}$, the gain is 0 by the argument for item 4. Writing p_{good} for the left-hand side of (7.5), we have

$$\Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s} \text{ and } P_{j,s}(Z) \notin \mathcal{A}_{j+1}] = \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}] - p_{\text{good}}.$$

Hence

$$\begin{aligned} \mathbb{E}_{Z \sim U_j} [g_{\theta_j,j,s}(Z)] &\geq \frac{\mu}{2} \left(\Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}] - p_{\text{good}} \right) - p_{\text{good}} \\ &= \frac{\mu}{2} \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}] - \left(1 + \frac{\mu}{2} \right) p_{\text{good}} \\ &\geq \left[\frac{\mu}{2} - \left(1 + \frac{\mu}{2} \right) \theta_j \right] \Pr_{Z \sim U_j} [Z \in \mathcal{D}_{j,s}], \end{aligned}$$

where the last step uses (7.5). Finally, $\theta_j \leq \mu/12$ and $\mu \leq 1$ give

$$\frac{\mu}{2} - \left(1 + \frac{\mu}{2}\right)\theta_j \geq \frac{\mu}{2} - \frac{3}{2} \cdot \frac{\mu}{12} = \frac{3\mu}{8},$$

which proves (7.4). $\square$

7.5 Global structured repair and uniform cost

With $\theta = (\theta_0, \dots, \theta_{m-1})$, compose the local repairs from Section 7.2 according to the actual entrance states:

$$\begin{aligned} S_{\theta,0}(x) &= s_0, \\ S_{\theta,j+1}(x) &= E_{\theta_j,j,S_{\theta,j}(x)}(x_j), \\ \text{Repair}_{\theta}^P(x)|_{F_j} &= \text{Repair}_{\theta_j,j,S_{\theta,j}(x)}(x_j), \\ \text{Cost}_{\theta}^P(x) &= \sum_{j=0}^{m-1} c_{\theta_j,j,S_{\theta,j}(x)}(x_j). \end{aligned}$$

All thresholds, transition completions, and tie-breaking choices depend only on the program and fixed parameters, independently of any distribution D . The superscript P specifies the program; the subscript θ records its chosen interval thresholds.

Proposition 7.5 (uniform cost of the structured repair). *For every $P \in \mathcal{K}_{3,\ell}$ with density $\mu \geq 1/16$, the procedures just defined satisfy, for every candidate x ,*

$$P(\text{Repair}_{\theta}^P(x)) = 1, \quad d_H(x, \text{Repair}_{\theta}^P(x)) \leq \text{Cost}_{\theta}^P(x). \quad (7.6)$$

For an absolute constant A' ,

$$\mathbb{E}_U[\text{Cost}_{\theta}^P(X)] \leq A'\ell(1 - \mu) \leq A'\ell. \quad (7.7)$$

Its selected checkpoint states are reproduced by a total three-state auxiliary program $\hat{P}$, with $O(n)$ transition layers, at most two reachable states at each checkpoint, and $O(\ell)$ collisions per interval. Its charges are the state-visit charges constructed in Step 4 of Section 7.2.

Proof. With $c = \ell$, the local maps and charges form a repair scheme in the sense of Section 6.3. Their endpoint and distance guarantees give (7.6), and all fixed transition choices remain unchanged. Applying Lemma 6.4 to the bounds of Lemma 7.4 gives the local expected cost–gain inequality with

$$B = \frac{K_{\text{loc}}(\ell, \mu)}{3\mu/8} = \frac{8K_{\text{loc}}(\ell, \mu)}{3\mu} = O(\ell).$$

The gain-cost lemma (Lemma 6.3) now gives (7.7).

Composing $Q_0, \dots, Q_{m-1}$ from s_0 gives a total three-state program $\hat{P}$. Let $\hat{t}_j$ be its checkpoint corresponding to t_j , placed after any reset from the preceding interval. The local endpoint agreement gives, by induction,

$$\hat{P}(x)_{\hat{t}_j} = S_{\theta,j}(x) \in \mathcal{A}_j.$$

Since $\mathcal{A}_j \subseteq R_{t_j}(P)$, at most two states are reachable there. Each auxiliary interval has at most $L_{\text{repair}}(\ell, \mu) + 1 = O(\ell)$ colliding layers. The program has $O(n)$ layers, since it adds at most one fixed reset per original interval. Finally, the local state visits and triggering bits in Step 4 are the corresponding visits and bits of $\hat{P}$, giving the claimed representation of the total charge. $\square$

8 Repair cost under small bias

Transferring the cost bound. Write $O_{t,u}(x)$ for the indicator of the event that $\hat{P}$ on input x is at state u at layer t . Here t indexes layers of $\hat{P}$, and at a free transition t , x_t denotes the candidate bit read there. By Step 4 of Section 7.1, the cost is

$$\begin{aligned} \text{Cost}_{\theta}^P(x) &= \sum_{(t,u,b) \in \mathcal{C}} O_{t-1,u}(x) \mathbf{1}[x_t = b] \\ &\quad + r_{\text{reward}}(\ell, \mu) \sum_{(\hat{t}_j, s) \in \mathcal{B}} O_{\hat{t}_j, s}(x), \end{aligned}$$

where $\mathcal{C}$ consists of the flip triggers of Step 3 of Section 7.1, placed at their corresponding transition positions in $\hat{P}$. At an original transition r of I_j , a trigger satisfies $\sigma_{r-1}(u) \geq \theta_j$ and $\beta_{r, \theta_j}(u, b) \neq b$. The set $\mathcal{B}$ consists of the pairs $(\hat{t}_j, s)$ with $s \in \mathcal{A}_j$ and $\sigma_{\hat{t}_j}(s) < \theta_j$, the low entrance states at the auxiliary checkpoints. Since $\hat{P}$ satisfies the conditions of the state-visit theorem, there is an absolute constant C such that every δ -biased D with $\delta \leq n^{-C\ell}$ satisfies

$$\mathbb{E}_D[O_{t,u}] \leq 4\mathbb{E}_U[O_{t,u}] + \frac{1}{n}.$$

For a flip trigger, we use only the pointwise bound by the visit under D , and the independence of the next bit under U :

$$\begin{aligned} \mathbb{E}_D[O_{t-1,u} \mathbf{1}[x_t = b]] &\leq \mathbb{E}_D[O_{t-1,u}] \\ &\leq 4\mathbb{E}_U[O_{t-1,u}] + \frac{1}{n} \\ &= 8\mathbb{E}_U[O_{t-1,u} \mathbf{1}[x_t = b]] + \frac{1}{n}. \end{aligned}$$

The charges for low entrance states are visit events, so the visit bound applies to them directly. There are $O(n)$ charged events, with weights at most $r_{\text{reward}}(\ell, \mu) = O(\ell)$, so the additive errors total $O(\ell)$. By linearity of expectation and Step 5 of Section 7.1,

$$\mathbb{E}_{X \sim D}[\text{Cost}_{\theta}^P(X)] \leq 8\mathbb{E}_{X \sim U}[\text{Cost}_{\theta}^P(X)] + O(\ell) = O(\ell).$$

The independence of unread bits and the gain argument are used only under uniform input.

Sections 8.1–8.4 prove the state-visit estimate, and Sections 8.5 and 8.6 transfer the cost and obtain the hitting set.

8.1 Covering state visits by tests with two live states

We bound the probability that the auxiliary program visits a specified state at any layer, including layers inside intervals. The argument first covers the visit event by products of tests with two live states, then constructs their Fourier majorants in Section 8.4. The Fourier facts are recorded in Section 3.2. The majorant construction uses the elementary fact that a sequence of permutation transitions on two states computes a parity of the input bits up to a constant.

Theorem 8.1 (state-visit probabilities). *Consider a total width-three program with $O(n)$ transition layers, two-state checkpoints, and at most $b \geq 1$ collisions per interval. Write $\hat{S}_t(x)$ for its state at layer t on input x . There is an absolute constant C such that every δ -biased distribution D on its input coordinates with $\delta \leq n^{-Cb}$ satisfies*

$$\Pr_D[\hat{S}_t = s] \leq 4 \Pr_U[\hat{S}_t = s] + \frac{1}{n} \tag{8.1}$$

for every layer t and state s . The layer need not be a checkpoint, and the same distribution works for all state visits.

The goal of this subsection is the following statement. Recall that $O_{t,s}(x) = \mathbf{1}[\widehat{S}_t(x) = s]$ indicates that the program P of Theorem 8.1 visits state s at layer t on input x . This is an event of a three-state program. The statement replaces it by products of acceptance functions of programs with two live states, at the price of a factor four in the expectation.

Definition (program with two live states). A *program with two live states* has a fixed start and maps from $\{0, 1\}$ into $\{0, 1, \perp\}$, where $\perp$ is absorbing rejection. At a free transition j , its own input bit x_j selects the map $f_j^{x_j}$; distinct free transitions read distinct coordinates. At a fixed transition it applies a single map f_j . At free transitions its state satisfies

$$z_j(x) = \begin{cases} f_j^{x_j}(z_{j-1}(x)), & z_{j-1}(x) \neq \perp, \\ \perp, & z_{j-1}(x) = \perp, \end{cases}$$

and the same formula with f_j applies at fixed transitions.

Its *acceptance function* indicates whether the program survives all m transitions:

$$q(x) = \mathbf{1}[z_m(x) \neq \perp].$$

A free transition j is *colliding* if one of its maps sends both live states to the same live state, that is, if $f_j^v(0) = f_j^v(1) \neq \perp$ for some $v \in \{0, 1\}$. The program has *collision bound* b if at most b free transitions are colliding. For these programs, fixed transitions, including resets, are unrestricted and are not counted. A prescribed accepting exit is imposed by a final fixed map that rejects the other live state.

If a layer has only one live node, the program has an unused second position whose outgoing transitions reject.

Proposition 8.2 (cover by products of acceptance functions). *Let t be a layer and s a state. There are functions $Q_a(x)$, $a \in \mathcal{A}_{t,s}$, with $|\mathcal{A}_{t,s}| \leq t + 1$, such that:*

1. *Each $Q_a(x)$ is the constant 1 or a product of acceptance functions of programs with two live states and collision bound b , which read pairwise disjoint blocks of input bits,*

$$Q_a(x) = q_{a,1}(x) \cdot q_{a,2}(x) \cdots q_{a,k_a}(x).$$

2. *The sum of the functions is a pointwise upper bound on the visit indicator,*

$$O_{t,s}(x) \leq \sum_{a \in \mathcal{A}_{t,s}} Q_a(x).$$

3. *Their total expectation under uniform input is at most four times the visit probability,*

$$\sum_{a \in \mathcal{A}_{t,s}} \mathbb{E}_U[Q_a(x)] \leq 4 \mathbb{E}_U[O_{t,s}(x)].$$

The proof has two parts. The cover lemma, Lemma 8.3, proves the statement using tests with two live states in place of the acceptance functions in item 1. Step 5 then shows that every test with two live states in this cover is a product of acceptance functions of programs with two live states and collision bound b on disjoint blocks, which gives item 1 as stated. Items 2 and 3 are not affected by this last step, since the functions $Q_a(x)$ stay the same.

The proposition is the input of Sections 8.2–8.4. Lemma 8.4 gives each product $Q_a(x)$ a pointwise upper bound of small Fourier norm; its proof in Section 8.4 completes the proof of Theorem 8.1.

Step 1: Path indicators and the role of the cover. Let P denote the total width-three program of Theorem 8.1. We view P as a layered graph. A *node* is a pair $v = (j, u)$ of a layer j and a state u , and v_0 is the start node at layer 0. For nodes v and v' , with v' at the same or a later layer, we use one indicator for all events of P :

$$O_{v \rightarrow v'}(x) = \mathbf{1} \left[\begin{array}{c} \text{the path of } P \text{ started at node } v \text{ and run on } x \\ \text{passes through node } v' \end{array} \right].$$

It depends only on the free input coordinates read after v up to v' . In particular, $O_{v \rightarrow v} = 1$. We write $p(v \rightarrow v') = \mathbb{E}_U [O_{v \rightarrow v'}(x)]$ for the probability of this event under uniform input.

A *target* is a node $v^* = (t, s)$, and the visit indicator of Theorem 8.1 is $O_{v_0 \rightarrow v^*}(x)$. This is an event of a program with three states. To explain the role of the cover, we first describe how it will be used in Section 8.4. For every target v^* , that subsection constructs a function $G_{v^*}(x)$, depending on the target and built from tests with two live states. It lies above the visit indicator, its expectation under U exceeds the visit probability by at most a factor four and a small additive term, and its Fourier norm is polynomial:

$$\begin{aligned} G_{v^*}(x) &\geq O_{v_0 \rightarrow v^*}(x), \\ \mathbb{E}_U G_{v^*} &\leq 4p(v_0 \rightarrow v^*) + \frac{1}{2n}, \\ \|G_{v^*}\|_F &\leq n^{C_0 b}. \end{aligned}$$

The constant C_0 is absolute and does not depend on the target. These three properties suffice. The first lets us replace the visit indicator by G_{v^*} under D , and the small-bias inequality then moves from D to U :

$$\mathbb{E}_D [O_{v_0 \rightarrow v^*}(x)] \leq \mathbb{E}_D G_{v^*} \leq \mathbb{E}_U G_{v^*} + \delta \|G_{v^*}\|_F.$$

For $\delta \leq n^{-Cb}$ with $C \geq C_0 + 2$ the last term is at most $1/(2n)$, for every target at once. Hence one distribution D satisfies (8.1) for all targets, although the functions G_{v^*} differ from target to target.

Step 2: Path decomposition and live and lost nodes. A path from v to v' passes through exactly one node w of any intermediate layer τ , so the indicator splits over the nodes of that layer:

$$O_{v \rightarrow v'}(x) = \sum_{w \text{ at layer } \tau} O_{v \rightarrow w}(x) \cdot O_{w \rightarrow v'}(x).$$

The two factors of each term read disjoint inputs, those up to layer τ and those after it. Under uniform input they are independent, and their expectations multiply:

$$\mathbb{E}_U [O_{v \rightarrow w}(x) \cdot O_{w \rightarrow v'}(x)] = p(v \rightarrow w) \cdot p(w \rightarrow v').$$

Fix a target $v^* = (t, s)$. For each layer $j \leq t$ we sort the nodes that P can reach from v_0 into those from which the target can still be reached, and those from which it cannot:

$$\begin{aligned} \text{Live}_j(v^*) &= \{w \text{ at layer } j : p(v_0 \rightarrow w) > 0 \text{ and } p(w \rightarrow v^*) > 0\}, \\ \text{Lost}_j(v^*) &= \{w \text{ at layer } j : p(v_0 \rightarrow w) > 0 \text{ and } p(w \rightarrow v^*) = 0\}. \end{aligned}$$

The path from v_0 reaches the target exactly when it stays on live nodes:

$$O_{v_0 \rightarrow v^*}(x) = \mathbf{1} \left[\begin{array}{c} \text{the path of } P \text{ started at } v_0 \text{ and run on } x \\ \text{passes only through live nodes up to layer } t \end{array} \right].$$

Two kinds of targets are trivial, and for them we define the majorant directly:

$$G_{v^*} = \begin{cases} 0 & \text{if } p(v_0 \rightarrow v^*) = 0, \\ 1 & \text{if } \text{Lost}_j(v^*) = \emptyset \text{ for all } j \leq t. \end{cases}$$

In the first case the visit indicator is identically zero, and in the second it is identically one. For every other target some layer contains a lost node, and we take the first such layer:

$$\tau(v^*) = \min\{j \leq t : \text{Lost}_j(v^*) \neq \emptyset\}.$$

Write $\tau = \tau(v^*)$. Since the target is reachable, v_0 is live and $1 \leq \tau \leq t$. For these targets, Steps 3–5 below construct the cover; Section 8.4 uses it to build G_{v^*} .

Step 3: From three states to two. Call a node w' at layer $j+1$ a *successor* of a node w at layer j if $p(w \rightarrow w') > 0$. A lost node stays lost, whatever input is read next:

$$w \in \text{Lost}_j(v^*) \text{ and } w' \text{ is a successor of } w \implies w' \in \text{Lost}_{j+1}(v^*).$$

Since P is total, every node has a successor, so a layer with a lost node is followed by a layer with a lost node. Hence every layer from τ to t contains a lost node, and of its three nodes at most two are live:

$$|\text{Live}_j(v^*)| \leq 2 \quad \text{for all } \tau \leq j \leq t.$$

For a node $w \in \text{Live}_\tau(v^*)$, the path from w reaches the target exactly when it stays on live nodes:

$$O_{w \rightarrow v^*}(x) = \mathbf{1} \left[\begin{array}{l} \text{the path of } P \text{ started at } w \text{ and run on } x \\ \text{passes only through live nodes up to layer } t \end{array} \right].$$

This is a test with two live states. It follows the at most two nodes of $\text{Live}_j(v^*)$ at each layer j and rejects when the path leaves them.

We now split the visit indicator at layer τ by the first rule. Nodes of layer τ that are not live contribute zero, so only live nodes remain:

$$O_{v_0 \rightarrow v^*}(x) = \sum_{w \in \text{Live}_\tau(v^*)} O_{v_0 \rightarrow w}(x) \cdot O_{w \rightarrow v^*}(x).$$

Each term is a product of two indicators of different kinds. The second, $O_{w \rightarrow v^*}(x)$, is a test with two live states, which is what G_{v^*} may be built from. The first, $O_{v_0 \rightarrow w}(x)$, is an event of the three-state program P , which we cannot use. Our aim is to remove the first indicator from every term, while keeping an upper bound on the visit indicator and losing at most a factor four in the expectation. This factor is the four allowed in (8.1).

The simplest way to remove an indicator is to replace it by 1. Since $O_{v_0 \rightarrow w}(x) \leq 1$, this can only increase the term, so it keeps an upper bound:

$$O_{v_0 \rightarrow w}(x) \cdot O_{w \rightarrow v^*}(x) \leq O_{w \rightarrow v^*}(x).$$

The price is paid in the expectation. By the second rule, the expectation of the term before the replacement is the product of the two probabilities:

$$\mathbb{E}_U[O_{v_0 \rightarrow w}(x) \cdot O_{w \rightarrow v^*}(x)] = p(v_0 \rightarrow w) \cdot p(w \rightarrow v^*).$$

By the definition of p , the expectation of the term after the replacement is the second probability alone:

$$\mathbb{E}_U[O_{w \rightarrow v^*}(x)] = p(w \rightarrow v^*).$$

Comparing the two, the replacement multiplies the expectation of the term by exactly $1/p(v_0 \rightarrow w)$:

$$\mathbb{E}_U [O_{w \rightarrow v^*}(x)] = \frac{1}{p(v_0 \rightarrow w)} \cdot \mathbb{E}_U [O_{v_0 \rightarrow w}(x) \cdot O_{w \rightarrow v^*}(x)].$$

Our budget for this multiplier is four. So the replacement is affordable whenever w is reached from v_0 with probability at least $1/4$, since then the expectation grows by at most a factor four:

$$p(v_0 \rightarrow w) \geq \frac{1}{4} \implies \mathbb{E}_U [O_{w \rightarrow v^*}(x)] \leq 4 \cdot \mathbb{E}_U [O_{v_0 \rightarrow w}(x) \cdot O_{w \rightarrow v^*}(x)].$$

We show that this condition holds for one of the live nodes of layer τ . We look at a single step of the path, from a reachable node u of layer $\tau - 1$ to layer τ . Three facts follow one from another:

$$u \text{ is live} \implies u \text{ has a live successor } w_u \implies p(u \rightarrow w_u) \geq \frac{1}{2}.$$

First, u is live because no node before layer τ is lost. Second, a live node can reach the target, and the first step of such a path leads to a live successor. Third, at a free transition the step from u is decided by a single uniform input bit, so every successor of u is taken with probability at least $1/2$; at a fixed transition its unique successor is taken with probability 1. The node w_u is one of the live nodes of layer τ , and all terms of the following sum are nonnegative, so the sum is at least its term for w_u :

$$\sum_{w \in \text{Live}_\tau(v^*)} p(u \rightarrow w) \geq p(u \rightarrow w_u) \geq \frac{1}{2} \quad \text{for every reachable node } u \text{ of layer } \tau - 1.$$

Now we average over the node u at layer $\tau - 1$, using the two rules and the previous bound:

$$\begin{aligned} \sum_{w \in \text{Live}_\tau(v^*)} p(v_0 \rightarrow w) &= \sum_{w \in \text{Live}_\tau(v^*)} \sum_{u \text{ at layer } \tau-1} p(v_0 \rightarrow u) \cdot p(u \rightarrow w) \\ &= \sum_{u \text{ at layer } \tau-1} p(v_0 \rightarrow u) \cdot \left(\sum_{w \in \text{Live}_\tau(v^*)} p(u \rightarrow w) \right) \\ &\geq \frac{1}{2} \sum_{u \text{ at layer } \tau-1} p(v_0 \rightarrow u) = \frac{1}{2}. \end{aligned}$$

So the path from v_0 is at a live node at layer τ with probability at least $1/2$. The sum on the left has at most two terms. Let w_+ be the node with the larger term, so that $p(v_0 \rightarrow w_+) \geq 1/4$, and let w_- be the other live node, if it exists. For w_+ we replace the first indicator by 1. For w_- the probability $p(v_0 \rightarrow w_-)$ may be very small, and the same replacement could cost far more than four, so there we keep the first indicator:

$$O_{v_0 \rightarrow v^*}(x) \leq O_{w_+ \rightarrow v^*}(x) + O_{v_0 \rightarrow w_-}(x) \cdot O_{w_- \rightarrow v^*}(x).$$

The second term is omitted when w_- does not exist. The first term is now a test with two live states, and its expectation is at most four times the expectation of the term it replaced:

$$p(w_+ \rightarrow v^*) \leq 4p(v_0 \rightarrow w_+) \cdot p(w_+ \rightarrow v^*).$$

In the second term one unwanted indicator remains, $O_{v_0 \rightarrow w_-}(x)$. It is the visit indicator of the earlier target w_- , an object of the same kind as the one we started with, so the same argument applies to it. Step 4 carries this out by induction on the layer of the target.

Step 4: The cover lemma.

Definition (test with two live states). An indicator $O_{w \rightarrow w'}(x)$ is a *test with two live states* if every layer from that of w to that of w' contains at most two nodes that lie on a path from w to w' . The test follows the original transitions while the computation remains among these nodes and rejects when it leaves them. The absorbing rejecting state $\perp$ is separate from the at most two live states.

Lemma 8.3 (cover by tests with two live states). *Let v^* be a target at layer t . There are functions $Q_a(x)$, $a \in \mathcal{A}_{v^*}$, with $|\mathcal{A}_{v^*}| \leq t + 1$, such that:*

1. *Each $Q_a(x)$ is the constant 1 or a product of tests with two live states along a chain of nodes $w_0, w_1, \dots, w_k = v^*$ at increasing layers,*

$$Q_a(x) = O_{w_0 \rightarrow w_1}(x) \cdot O_{w_1 \rightarrow w_2}(x) \cdots O_{w_{k-1} \rightarrow v^*}(x).$$

2. *The functions together lie above the visit indicator,*

$$O_{v_0 \rightarrow v^*}(x) \leq \sum_{a \in \mathcal{A}_{v^*}} Q_a(x).$$

3. *Their total expectation is at most four times the visit probability,*

$$\sum_{a \in \mathcal{A}_{v^*}} \mathbb{E}_U[Q_a(x)] \leq 4p(v_0 \rightarrow v^*). \quad (8.2)$$

Proof. We argue by induction on the layer t of the target, and assume the lemma for all targets at layers before t . We distinguish four cases. Only the last one uses the induction hypothesis.

Case 1: the target is unreachable, $p(v_0 \rightarrow v^*) = 0$. Then $O_{v_0 \rightarrow v^*}(x) = 0$ for every x , and the empty family satisfies both inequalities.

Case 2: no node is lost, $\text{Lost}_j(v^*) = \emptyset$ for all $j \leq t$. Then every path from v_0 passes through v^* , so $O_{v_0 \rightarrow v^*}(x) = 1$ for every x and $p(v_0 \rightarrow v^*) = 1$. The family consisting of the constant 1 satisfies both inequalities.

In the remaining cases the target is reachable and some node is lost. Let $\tau = \tau(v^*)$ be the first layer with a lost node. Three facts from Steps 2 and 3 are used. First, every layer from τ to t has at most two live nodes:

$$|\text{Live}_j(v^*)| \leq 2 \quad \text{for all } \tau \leq j \leq t.$$

Second, the visit indicator splits over the live nodes of layer τ :

$$O_{v_0 \rightarrow v^*}(x) = \sum_{w \in \text{Live}_\tau(v^*)} O_{v_0 \rightarrow w}(x) \cdot O_{w \rightarrow v^*}(x).$$

Third, the path from v_0 is at a live node at layer τ with probability at least $1/2$:

$$\sum_{w \in \text{Live}_\tau(v^*)} p(v_0 \rightarrow w) \geq \frac{1}{2}.$$

From the first fact, for every $w \in \text{Live}_\tau(v^*)$ the indicator $O_{w \rightarrow v^*}(x)$ is a test with two live states. Indeed, a node on a path from w to v^* is reachable from v_0 and can reach v^* , so it is live, and each layer from τ to t has at most two live nodes. Let w_+ be the live node of layer τ with the largest probability $p(v_0 \rightarrow w_+)$. Taking expectations in the split and applying the second rule expresses the visit probability through the live nodes:

$$p(v_0 \rightarrow v^*) = \sum_{w \in \text{Live}_\tau(v^*)} p(v_0 \rightarrow w) \cdot p(w \rightarrow v^*).$$

Case 3: layer τ has the single live node w_+ . The split has one term, and dropping its first factor gives an upper bound by a test with two live states:

$$O_{v_0 \rightarrow v^*}(x) = O_{v_0 \rightarrow w_+}(x) \cdot O_{w_+ \rightarrow v^*}(x) \leq O_{w_+ \rightarrow v^*}(x).$$

The family is the single function $O_{w_+ \rightarrow v^*}(x)$. If $\tau = t$, this is the constant 1, so no zero-length test is needed in the chain. By the third fact $p(v_0 \rightarrow w_+) \geq 1/2$, so its expectation is within the allowed factor:

$$p(w_+ \rightarrow v^*) \leq 2p(v_0 \rightarrow w_+) \cdot p(w_+ \rightarrow v^*) = 2p(v_0 \rightarrow v^*).$$

Case 4: layer τ has two live nodes w_+ and w_- . By the third fact $p(v_0 \rightarrow w_+) \geq 1/4$. Layer t has the single live node v^* , so $\tau < t$, and w_- is a target at an earlier layer. The induction hypothesis gives a family $Q'_a(x)$, $a \in \mathcal{A}_{w_-}$, of at most $\tau + 1$ functions, each the constant 1 or a product of tests with two live states along a chain ending at w_- , with

$$\begin{aligned} O_{v_0 \rightarrow w_-}(x) &\leq \sum_{a \in \mathcal{A}_{w_-}} Q'_a(x), \\ \sum_{a \in \mathcal{A}_{w_-}} \mathbb{E}_U[Q'_a(x)] &\leq 4p(v_0 \rightarrow w_-). \end{aligned}$$

In the split we drop the first factor of the term of w_+ and bound the first factor of the term of w_- by this family:

$$O_{v_0 \rightarrow v^*}(x) \leq O_{w_+ \rightarrow v^*}(x) + \sum_{a \in \mathcal{A}_{w_-}} Q'_a(x) \cdot O_{w_- \rightarrow v^*}(x).$$

The family for v^* consists of the functions on the right. The first is a test with two live states ending at v^* . Each product $Q'_a(x) \cdot O_{w_- \rightarrow v^*}(x)$ extends a chain ending at w_- by the test with two live states from w_- to v^* , so it is a product of tests with two live states along a chain ending at v^* . The family has at most $1 + (\tau + 1) \leq t + 1$ members.

For the expectation, $Q'_a(x)$ reads only inputs of layers up to τ and $O_{w_- \rightarrow v^*}(x)$ reads only inputs after layer τ . They are independent, so the expectation of the product factors:

$$\mathbb{E}_U[Q'_a(x) \cdot O_{w_- \rightarrow v^*}(x)] = \mathbb{E}_U[Q'_a(x)] \cdot p(w_- \rightarrow v^*).$$

Summing over the family, we use $p(v_0 \rightarrow w_+) \geq 1/4$ for the first function and the induction hypothesis for the others, and then the expression of the visit probability through the two live nodes:

$$\begin{aligned} p(w_+ \rightarrow v^*) + \sum_{a \in \mathcal{A}_{w_-}} \mathbb{E}_U[Q'_a(x)] \cdot p(w_- \rightarrow v^*) \\ \leq 4p(v_0 \rightarrow w_+) \cdot p(w_+ \rightarrow v^*) + 4p(v_0 \rightarrow w_-) \cdot p(w_- \rightarrow v^*) = 4p(v_0 \rightarrow v^*). \end{aligned}$$

This proves both inequalities for v^* and completes the induction. The factor four is spent once, on the likely node w_+ , and the induction hypothesis supplies the same factor for w_- , so it is not multiplied along the recursion. $\square$

Step 5: Proof of Proposition 8.2. Fix a layer t and a state s , and let $v^* = (t, s)$. The proposition and Lemma 8.3 are related by the dictionary

$$\begin{aligned} O_{t,s}(x) &= O_{v_0 \rightarrow v^*}(x), \\ \mathbb{E}_U[O_{t,s}(x)] &= p(v_0 \rightarrow v^*), \\ \mathcal{A}_{t,s} &= \mathcal{A}_{v^*}. \end{aligned}$$

With it, items 2 and 3 of the lemma are items 2 and 3 of the proposition, and $|\mathcal{A}_{t,s}| \leq t + 1$. For item 1, the lemma gives each nonconstant $Q_a(x)$ as a chain of tests with two live states:

$$Q_a(x) = \prod_{l=1}^k O_{w_{l-1} \rightarrow w_l}(x), \quad w_k = v^*.$$

By the proof of the lemma, every test of the chain starts at the first lost layer of its own endpoint:

$$w_{l-1} \in \text{Live}_{\tau(w_l)}(w_l) \quad \text{for every } l.$$

So it suffices to prove the following for one test.

Claim. Let v' be a target at layer t' for which $\tau' = \tau(v')$ is defined, and let $w \in \text{Live}_{\tau'}(v')$. Then $O_{w \rightarrow v'}(x)$ is a product of acceptance functions of programs with two live states and collision bound b , which read disjoint blocks of input bits.

Proof of the claim. Let $j_1 < \dots < j_r$ be the checkpoint layers of P strictly between τ' and t' . At a checkpoint at most two nodes are reachable, and from layer τ' on one of them is lost, so exactly one is live:

$$\text{Live}_{j_i}(v') = \{c_i\} \quad \text{for } i = 1, \dots, r. \quad (8.3)$$

Set $c_0 = w$ and $c_{r+1} = v'$. Splitting at layer j_i by the first rule, the only nonzero term is that of the single live node c_i :

$$O_{c_{i-1} \rightarrow v'}(x) = O_{c_{i-1} \rightarrow c_i}(x) \cdot O_{c_i \rightarrow v'}(x).$$

Applying this for $i = 1, \dots, r$ in turn writes the test as a product over the intervals of P :

$$O_{w \rightarrow v'}(x) = \prod_{i=0}^r q_i(x), \quad q_i(x) = O_{c_i \rightarrow c_{i+1}}(x).$$

Each $q_i(x)$ is the acceptance function of a program with two live states: its start is c_i , its live states at layer j are the nodes of $\text{Live}_j(v')$, and moving to a lost node is rejection. It reads only the inputs of one interval of P , the layers $j_i + 1, \dots, j_{i+1}$ (with $j_0 = \tau'$ and $j_{r+1} = t'$), so the blocks of different q_i are disjoint. A colliding transition of this program is a collision of P in that interval, which bounds their number:

$$\begin{aligned} \#\{\text{collisions in the program computing } q_i\} &\leq \#\{\text{collisions of } P \text{ in layers } j_i + 1, \dots, j_{i+1}\} \\ &\leq b. \end{aligned}$$

This proves the claim. □

Applying the claim to every test of the chain, with $v' = w_l$ and $w = w_{l-1}$, turns $Q_a(x)$ into a product of acceptance functions of programs with two live states and collision bound b :

$$Q_a(x) = \prod_{l=1}^k \prod_{i=0}^{r_l} q_{l,i}(x).$$

The tests of the chain read disjoint ranges of layers, from w_{l-1} to w_l , so all functions $q_{l,i}(x)$ depend on pairwise disjoint blocks. This is item 1. Together with items 2 and 3 of the lemma and the dictionary, we obtain

$$\begin{aligned} O_{t,s}(x) &\leq \sum_{a \in \mathcal{A}_{t,s}} Q_a(x), \\ \sum_{a \in \mathcal{A}_{t,s}} \mathbb{E}_U[Q_a(x)] &\leq 4 \mathbb{E}_U[O_{t,s}(x)], \end{aligned}$$

which proves the proposition. ■

8.2 Acceptance functions as parity constraints

The goal of this subsection is the following lemma. Recall from Section 8.1 that the *acceptance function* of a program with two live states indicates that the program does not reject on input x :

$$q(x) = \mathbf{1}[z_m(x) \neq \perp].$$

Proposition 8.2 bounds every visit indicator by a sum of products of such functions. The lemma gives each product a pointwise upper bound of small Fourier norm, and together they prove Theorem 8.1.

Lemma 8.4 (product majorant). *Let $Q(x) = \prod_i q_i(x)$, where each $q_i(x)$ is the acceptance function of a program with two live states and collision bound $b \geq 1$, and the programs read pairwise disjoint blocks of input bits. For every $0 < \eta < 1/4$ there is a nonnegative function $G(x)$, on the same coordinates, such that:*

1. $G(x) \geq Q(x)$ for every x .
2. $\mathbb{E}_U[G(x)] \leq \mathbb{E}_U[Q(x)] + \eta$.
3. Its Fourier norm satisfies

$$\|G\|_F \leq 2^{O(b \log(1/\eta))}. \quad (8.4)$$

All constants are absolute. The majorant need not be Boolean.

The idea. In a program with two live states, every layer has two live states, labelled 0 and 1. The program moves to a new layer at every transition, so it never returns to the same state. What we follow while the computation is live is the *state label*

$$z_j(x) \in \{0, 1\},$$

which says in which of the two live states of layer j the program is after transition j on input x . The label is one bit of memory that the program carries from layer to layer. We use the label $\perp$ after rejection, and write the transition maps on these labels.

The proof builds on parities, and they arise from the way permutations act on this label. Suppose both maps T_j^0 and T_j^1 of a free transition j permute $\{0, 1\}$. Every permutation of $\{0, 1\}$ adds a constant to the label, so there are constants $\kappa_j, a_j \in \{0, 1\}$ with

$$T_j^0(u) = u \oplus \kappa_j \quad \text{and} \quad T_j^1(u) = u \oplus \kappa_j \oplus a_j.$$

Hence the new label is the old label plus an affine function of the bit x_j :

$$z_j(x) = z_{j-1}(x) \oplus \kappa_j \oplus a_j x_j.$$

The constant a_j separates two cases:

1. If $a_j = 0$, the bit x_j has no influence on the label. The new label is determined by the old one: it is copied if $\kappa_j = 0$ and flipped if $\kappa_j = 1$.
2. If $a_j = 1$, the bit x_j is added to the label.

For a fixed permutation transition, set $a_j = 0$ and let κ_j record whether the map flips the label.

So in a program whose maps are permutations, the label after transition j is the start label plus the input bits of the transitions with $a_i = 1$, plus a constant. It is a parity of input bits up to a constant, and the indicator of a label has Fourier norm at most one:

$$z_j(x) = c_j \oplus \bigoplus_{i \in S_j} x_i \implies \|\mathbf{1}[z_j(x) = u]\|_F \leq 1.$$

Here $S_j = \{i \leq j : a_i = 1\}$ and $c_j = z_0 \oplus \kappa_1 \oplus \dots \oplus \kappa_j$.

The goal for one acceptance function. The following lemma expresses one acceptance function through parity constraints.

Lemma 8.5 (parity representation). *Let q be the acceptance function of a program with two live states, m transitions, and collision bound $b \geq 0$. Let H be the set of input coordinates read by its colliding free transitions, so $|H| \leq b$. For $\alpha \in \{0,1\}^H$, let q_α be the restriction of q obtained by fixing $x_H = \alpha$. Then:*

1. *The restrictions reconstruct q pointwise:*

$$q(x) = \sum_{\alpha \in \{0,1\}^H} \mathbf{1}[x_H = \alpha] q_\alpha(x).$$

2. *For each fixed α , let T_j be the map that the restricted program applies at transition j , that is, $T_j = T_j^{x_j}$ if the transition is free. There are affine parities $s_1(x), \dots, s_m(x)$, where $s_j(x)$ depends only on the free bits read before transition j , such that q_α is a product of m constraints:*

$$q_\alpha(x) = \prod_{j=1}^m (1 - E_j(x)).$$

Here $E_j(x)$ indicates that the map of transition j rejects from the live state with label $s_j(x)$:

$$E_j(x) = \mathbf{1}[T_j(s_j(x)) = \perp].$$

An affine parity is a parity plus a constant, possibly a constant function. The parities s_j may depend on α .

Such a form reduces the Fourier analysis of $q(x)$ to products of constraints of bounded norm. The program does not give it directly, because its label fails to be a parity for two reasons: a colliding free transition can merge the two live states depending on the input, and a rejecting transition gives the label $\perp$. The proof removes these two obstacles to establish items 1 and 2.

Proof. Let H be the set of input bits read by the colliding free transitions, so $|H| \leq b$. For a pattern $\alpha \in \{0,1\}^H$, let $x|_{H=\alpha}$ denote the input x with its bits in H replaced by α . The restriction of q to the pattern α is the function

$$q_\alpha(x) = q(x|_{H=\alpha}),$$

which no longer depends on the bits of H . Every input x agrees with exactly one pattern on H , namely $\alpha = x_H$, and for this pattern $x|_{H=\alpha} = x$. So exactly one pattern indicator in the following sum is one, and its term equals $q(x)$. This is item 1:

$$q(x) = \sum_{\alpha \in \{0,1\}^H} \mathbf{1}[x_H = \alpha] \cdot q_\alpha(x).$$

Second obstacle: rejecting transitions. Fix a pattern α , and consider the restricted program in which $x_H = \alpha$. Let $T_1, \dots, T_m$ be its transition maps. At a free transition j , the symbol T_j stands for the map $T_j^{x_j}$ selected by the input bit. Each map $T_j: \{0,1\} \rightarrow \{0,1,\perp\}$ is of one of three kinds:

1. a permutation: $T_j(0)$ and $T_j(1)$ are the two distinct labels 0 and 1;
2. a colliding map: $T_j(0) = T_j(1) \neq \perp$, which after fixing H occurs only at fixed transitions;
3. a rejecting map: $T_j(u) = \perp$ for at least one live label u .

Maps of the first two kinds keep the label an affine parity: a permutation changes it by an affine function, and a colliding map sets it to a constant. Only a rejecting map takes the label out of $\{0,1\}$. So we change only the rejecting maps, in a way that keeps the label in $\{0,1\}$, and we record each rejection as a separate constraint. For a rejecting map T_j , choose a permutation $\bar{T}_j$ of $\{0,1\}$ that agrees with T_j wherever T_j does not reject:

$$\bar{T}_j(u) = T_j(u) \quad \text{whenever } T_j(u) \neq \perp.$$

At a free transition, this choice is made separately for each branch, independently of the input. For the other maps put $\bar{T}_j = T_j$. The *completed program* has the same start label z_0 as the restricted program and the transition maps $\bar{T}_1, \dots, \bar{T}_m$. Write $z_j(x)$ for the label of the restricted program at layer j and $\bar{z}_j(x)$ for the label of the completed program at that layer. All maps $\bar{T}_j$ take values in $\{0, 1\}$, so the completed program is always in a live state:

$$\bar{z}_j(x) = \bar{T}_j(\bar{z}_{j-1}(x)) \in \{0, 1\}.$$

The label of the completed program before transition j is

$$s_j(x) = \bar{z}_{j-1}(x).$$

We call transition j *forbidden on input x* if the restricted map rejects from the state with this label, and write $E_j(x)$ for its indicator:

$$E_j(x) = \mathbf{1}[T_j(s_j(x)) = \perp].$$

Let $j^*(x)$ be the first forbidden transition on input x , with $j^*(x) = m + 1$ if there is none:

$$j^*(x) = \min\left(\{j \in \{1, \dots, m\} : E_j(x) = 1\} \cup \{m + 1\}\right).$$

The two programs have the same label at every layer before $j^*(x)$; if $j^*(x) \leq m$, the restricted program rejects exactly at that transition:

1. $z_j(x) = \bar{z}_j(x)$ for every $0 \leq j < j^*(x)$.
2. $z_j(x) = \perp$ for every $j^*(x) \leq j \leq m$.

Indeed, both computations start with label z_0 . If they agree before a transition $j < j^*(x)$, then $E_j(x) = 0$, so the restricted map does not reject at their common label. The completed map agrees with it there, and the labels after transition j also agree. Induction gives agreement before the first forbidden transition. If $j^*(x) \leq m$, the restricted map rejects at the common label before transition $j^*(x)$, and absorbing rejection keeps its label equal to $\perp$ thereafter.

From layer $j^*(x)$ onward, when it exists, the two programs have different labels, and the later indicators $E_j(x)$ describe the path of the completed program, which the restricted program does not follow. Their values do not matter. The restricted program accepts exactly when it never reaches $\perp$, that is, when there is no forbidden transition:

$$q_\alpha(x) = \mathbf{1}[j^*(x) = m + 1] = \prod_{j=1}^m (1 - E_j(x)).$$

If some transition is forbidden, the factor $1 - E_{j^*(x)}(x)$ is zero, so the product is zero whatever the later factors are. If none is forbidden, every factor is one.

It remains to verify that each s_j is an affine parity. At a free transition the two completed maps are permutations, so the completed label changes by an affine function of the current bit. A fixed permutation changes the label by a constant, and a fixed colliding map sets it to a constant. Induction therefore shows that $s_j(x) = \bar{z}_{j-1}(x)$ is an affine parity of the remaining free bits read before transition j . These parities and the maps T_j are determined by the restricted program and its fixed completions, so the functions E_j do not depend on the coordinates in H . This proves item 2 and completes the lemma. $\square$

The restriction formula in item 1 also controls the Fourier norm. The indicator $\mathbf{1}[x_H = \alpha]$ is a subcube indicator, so its Fourier norm is one. It depends only on the bits of H , while $q_\alpha(x)$ depends only on the other bits, so the norm of their product is the product of their norms. By the triangle inequality over the $2^{|H|} \leq 2^b$ patterns,

$$\|q\|_F \leq \sum_{\alpha \in \{0,1\}^H} \|q_\alpha\|_F \leq 2^b \cdot \max_{\alpha} \|q_\alpha\|_F.$$

The bound holds in the same way for any functions assembled pattern by pattern. If $h(x) = \sum_{\alpha} \mathbf{1}[x_H = \alpha] \cdot h_\alpha(x)$, where each $h_\alpha(x)$ does not depend on the bits of H , then $\|h\|_F \leq 2^b \cdot \max_{\alpha} \|h_\alpha\|_F$. This explains how the upper bounds $q_\alpha^+(x)$ constructed below are combined into an upper bound on $q(x)$.

Section 8.3 groups this representation into a chain of constraints. Section 8.4 uses the chain to prove Lemma 8.4 and complete the proof of Theorem 8.1.

8.3 Grouping parity constraints into a chain

This subsection proves Lemma 8.6, which groups the constraints of Lemma 8.5 into a chain of Boolean functions with bounded Fourier norm and independence between nonadjacent groups. Throughout the explanation, the restriction pattern α is fixed.

The difficulty. Lemma 8.5 writes the restricted acceptance function as a product of m constraints,

$$q_\alpha(x) = \prod_{j=1}^m (1 - E_j(x)),$$

but a product of many constraints need not have small Fourier norm. For example, the function $f(x) = \prod_{i=1}^m (x_{2i-1} \vee x_{2i})$ is such a product, and:

1. $\|f\|_F = (3/2)^m$.
2. $\mathbb{E}_U[f(x)] = (3/4)^m$.

So the norm of $q_\alpha(x)$ can be exponential in m , and we do not try to bound it. The example also suggests what to do instead: in this example, a long product has tiny expectation. We seek a factorization with this same property. Since deleting factors from a product of indicators gives a pointwise upper bound, we can then replace $q_\alpha(x)$ by a shorter product.

What the shortening needs. We look for a factorization into Boolean functions $g_0(x), \dots, g_K(x)$,

$$q_\alpha(x) = \prod_{k=0}^K g_k(x),$$

whose factors behave like independent trials:

1. Each factor has small Fourier norm, $\|g_k\|_F \leq 4$.
2. Each factor that can fail does so with constant probability: $\mathbb{E}_U[g_k(x)] \leq 3/4$ unless $g_k \equiv 1$.
3. Every collection of pairwise non-neighbouring factors is mutually independent under U : the factors $g_{k_1}, \dots, g_{k_r}$ are independent whenever $|k_i - k_h| \geq 2$ for all distinct i, h .

If at least t factors are not identically 1, where t is even, retain the first t such factors in their original order and denote their product by $q_\alpha^+(x)$. Then:

1. $q_\alpha^+(x) \geq q_\alpha(x)$, because factors were deleted.
2. $\|q_\alpha^+\|_F \leq 4^t$, by the first property.
3. $\mathbb{E}_U[q_\alpha^+(x)] \leq (3/4)^{t/2}$, by the second and third properties applied to every second kept factor.

Why single constraints do not qualify. The constraints $1 - E_j(x)$ themselves need not satisfy the third property. A free transition j rejects according to the state label before it and its own input bit. With $\mathcal{F}_j \subseteq \{0, 1\}^2$ the set of pairs (label, bit) at which it rejects,

$$E_j(x) = \mathbf{1}[(s_j(x), x_j) \in \mathcal{F}_j].$$

At a fixed transition, the rejection condition depends only on the label. Here $s_j(x)$ is the label of the completed program before transition j , as constructed in the proof of Lemma 8.5.

The input bits of different free transitions are independent, so dependence between constraints arises through their labels. And the labels can be strongly dependent. At a free transition they satisfy the recursion

$$s_{j+1}(x) = s_j(x) \oplus \kappa_j \oplus a_j x_j.$$

A fixed permutation changes the label by a constant, and a fixed colliding map resets it to a constant. Whenever $a_j = 0$ at a free transition, the label $s_{j+1}(x)$ is determined by $s_j(x)$. If every free transition from j to $l-1$ has this property and there is no reset between them, then $s_l(x) = s_j(x) \oplus \text{constant}$. Thus $E_j(x)$ and $E_l(x)$ can both depend on the same random label $s_j(x)$, however far apart j and l are.

The effect is that the expectations of such constraints need not multiply. Suppose r free transitions, indexed by $j_1, \dots, j_r$, preserve the same label $s(x)$ in the completed program. Suppose this label is uniform and independent of their input bits, as when an earlier input bit sets it, and each transition rejects exactly when the label is 1 and its own input bit is 1. Independent factors would give expectation $(3/4)^r$. But if $s(x) = 0$ none of them rejects, and if $s(x) = 1$ all r bits must be 0:

$$\mathbb{E}_U \left[\prod_{i=1}^r (1 - E_{j_i}(x)) \right] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2^{-r} \geq \frac{1}{2}.$$

So keeping more of these constraints does not make the expectation small.

The structure of the dependence. Let A be the set of free transitions with $a_j = 1$ and B the set of free transitions with $a_j = 0$, with input bits x_A and x_B . Only the bits of x_A enter the label recursion; fixed maps only flip or reset the label. Hence the whole vector of labels is a function of x_A , which we denote by Λ :

$$(s_1(x), \dots, s_m(x)) = \Lambda(x_A).$$

Fix the value of x_A . Then every label is fixed, each constraint with $j \in B$ becomes a function of its own bit x_j , and every other constraint is determined. The bits of x_B are independent, so given x_A the constraints of the transitions in B are independent: for every assignment $a \in \{0, 1\}^A$ and every $(e_j)_{j \in B} \in \{0, 1\}^B$,

$$\Pr_U [E_j(x) = e_j \text{ for all } j \in B \mid x_A = a] = \prod_{j \in B} \Pr_U [E_j(x) = e_j \mid x_A = a].$$

So all dependence between the constraints passes through the labels. We therefore organize the factorization according to those labels. The following lemma achieves this by grouping transitions into stages and making an invertible affine change of the remaining coordinates.

Lemma 8.6 (chain form). *Let q be the acceptance function of a program with two live states. Let H be the set of coordinates read by the program's colliding free transitions, and fix $\alpha \in \{0, 1\}^H$. Let q_α be the restriction of q to $x_H = \alpha$. After an invertible affine change of the remaining coordinates, there are Boolean functions $g_0, \dots, g_K$ with*

$$q_\alpha = \prod_{k=0}^K g_k$$

such that:

1. $\|g_k\|_F \leq 4$ for every k .
2. $\mathbb{E}_U[g_k] \leq 3/4$ unless g_k is identically one.
3. g_k and g_l depend on disjoint coordinates whenever $|k - l| \geq 2$.

Proof. We use the completed program constructed in the proof of Lemma 8.5 for the restriction q_α , and let m be its number of transitions. Its maps $\bar{T}_j$ take values in $\{0, 1\}$, and $s_j(x)$ denotes its state label before

transition j , so that $s_{j+1}(x) = \bar{T}_j(s_j(x))$. Let $\mathcal{F}_j$ be the set of pairs (label, bit) at which the restricted map of transition j rejects; at a fixed transition $\mathcal{F}_j$ is a set of labels. By item 2 of Lemma 8.5,

$$q_\alpha(x) = \prod_{j=1}^m (1 - E_j(x)),$$

where $E_j(x) = \mathbf{1}[(s_j(x), x_j) \in \mathcal{F}_j]$ at a free transition and $E_j(x) = \mathbf{1}[s_j(x) \in \mathcal{F}_j]$ at a fixed transition. The proof organizes these constraints according to the labels.

Two types of transitions. Every map of the completed program is a permutation or a fixed colliding map. At a free transition the two permutations give, with constants $\kappa_j, a_j \in \{0, 1\}$,

$$s_{j+1}(x) = s_j(x) \oplus \kappa_j \oplus a_j x_j.$$

We distinguish two types of transitions.

1. Transition j *carries* the label if it is a fixed permutation or a free transition with $a_j = 0$. In both cases, $s_{j+1}(x) = s_j(x) \oplus \kappa_j$ for every x .
2. Transition j is an *opener* otherwise. These are the *switches*, the free transitions with $a_j = 1$, and the fixed colliding maps, after which the label is a constant.

Stages. Let $j_1 < \dots < j_K$ be the openers, and put $j_0 = 0$ and $j_{K+1} = m + 1$. We partition the transitions $1, \dots, m$ into consecutive blocks, called stages, so that each opener is the first transition of its block:

$$I_0 = \{1, \dots, j_1 - 1\}, \quad I_k = \{j_k, \dots, j_{k+1} - 1\} \quad (1 \leq k \leq K).$$

When $K = 0$, this means $I_0 = \{1, \dots, m\}$. Let $y_k = s_{j_k+1}(x)$ be the label right after opener j_k for $1 \leq k \leq K$, and let $y_0 = s_1(x)$ be the start label. All transitions of I_0 carry the label; in I_k for $k \geq 1$, all transitions other than j_k carry it. So for each of these transitions the label is y_k up to a known constant c_j :

$$s_j(x) = y_k \oplus c_j \quad (j \in I_k, j \neq j_k).$$

The opener acts on the label of the previous stage, so $s_{j_k}(x) = y_{k-1} \oplus c_{j_k}$ for $k \geq 1$. Let w_k be the list of input bits of the free transitions $j \in I_k$ with $j \neq j_k$.

New coordinates. We want the labels of the stages to be independent variables. At a switch j_k the new label is the old one plus the input bit of the switch:

$$y_k = y_{k-1} \oplus c_{j_k} \oplus \kappa_{j_k} \oplus x_{j_k}.$$

We replace each bit x_{j_k} of a switch by the label y_k and keep all other free bits. Since y_{k-1} depends only on earlier switch bits, this substitution is triangular, hence an invertible affine map of the free coordinates. It preserves the uniform distribution and the Fourier norm. In the new coordinates:

1. The labels y_k at switches and all bits of the w_k are independent uniform bits.
2. The start label y_0 and the labels y_k at fixed colliding maps are constants.

Stage functions. Let g_k be the product of the constraints of stage k ,

$$g_k(x) = \prod_{j \in I_k} (1 - E_j(x)),$$

so that $q_\alpha = \prod_{k=0}^K g_k$. We express each constraint in the new coordinates.

1. An opener that is a fixed colliding map never rejects, so its constraint is identically one.
2. For an opener that is a switch, the label $s_{j_k}(x)$ is determined by y_{k-1} and the bit x_{j_k} by (y_{k-1}, y_k) . So its constraint is a function of (y_{k-1}, y_k) .

3. For any other $j \in I_k$, the label $s_j(x)$ is determined by y_k . So its constraint is a function of y_k and, if the transition is free, of its own bit x_j in w_k .

Hence the stage functions depend on the following variables:

$$\begin{aligned} g_0 &= g_0(y_0, w_0), \\ g_k &= g_k(y_{k-1}, y_k, w_k) \quad (1 \leq k \leq K). \end{aligned}$$

Item 1. For $k \geq 1$, fix values (u, v) of (y_{k-1}, y_k) . The constraint of the opener becomes a constant, and each other constraint of the stage becomes a constant or a condition on its own bit. So the product of the constraints becomes a conjunction of literals of w_k , possibly constant; denote it by $C_{u,v}(w_k)$. Then

$$g_k = \sum_{u,v \in \{0,1\}} \mathbf{1}[y_{k-1} = u] \cdot \mathbf{1}[y_k = v] \cdot C_{u,v}(w_k),$$

where fewer terms are nonzero if y_{k-1} or y_k is a constant. Each term is a subcube indicator in the new coordinates, possibly zero, so it has Fourier norm at most one. There are at most four terms, hence $\|g_k\|_F \leq 4$. For $k = 0$, the label y_0 is fixed, so g_0 itself is a conjunction of literals of w_0 , possibly constant, and has Fourier norm at most one.

Item 2. Suppose $g_k \neq 1$. Then some transition $j \in I_k$ has $E_j(x) = 1$ with positive probability. By the list above, the event $E_j(x) = 1$ is determined by at most two of the independent uniform bits: (y_{k-1}, y_k) for a switch, (y_k, x_j) for any other free transition, or y_k alone for a fixed transition, with constant labels omitted. So its probability is at least $1/4$. Since $g_k \leq 1 - E_j$,

$$\mathbb{E}_U[g_k] \leq 1 - \Pr_U[E_j(x) = 1] \leq \frac{3}{4}.$$

Item 3. For $k \geq 1$, the function g_k uses only the coordinates y_{k-1} , y_k and w_k , with constant labels omitted. The function g_0 uses only w_0 , since y_0 is constant. For $|k - l| \geq 2$, the two functions use disjoint label coordinates and disjoint lists w_k and w_l . Hence g_k and g_l depend on disjoint coordinates. In particular, any set of pairwise non-adjacent stage functions is mutually independent. $\square$

8.4 Fourier majorants for products of acceptance functions

We now prove Lemma 8.4 and Theorem 8.1, in three parts. First we use the chain form of Lemma 8.6 to approximate one acceptance function from both sides. Then we treat a product of many acceptance functions by three successive pointwise upper bounds. Finally we combine the result with the cover of Proposition 8.2.

Part 1: one program.

Lemma 8.7 (programs with few collisions). *Let $q(x)$ be the acceptance function of a program with two live states and collision bound $c \geq 0$.*

1. *If q is not identically one, it fails with probability bounded from below:*

$$\mathbb{E}_U[1 - q(x)] \geq 2^{-c-2}. \quad (8.5)$$

2. *For every $0 < \gamma < 1$ there are Boolean functions $q^-(x) \leq q(x) \leq q^+(x)$, on the same coordinates, such that*

$$\begin{aligned} \mathbb{E}_U[q^+(x) - q^-(x)] &\leq \gamma, \\ \|q^\pm\|_F &\leq 16 \cdot 2^c \gamma^{-10}. \end{aligned} \quad (8.6)$$

Proof. Let H be the set of coordinates read by the colliding free transitions, so $|H| \leq c$. For each pattern $\alpha \in \{0, 1\}^H$, let $q_\alpha(x)$ be the restriction of $q(x)$ to $x_H = \alpha$, and let $g_0, \dots, g_K$ be the stage functions that Lemma 8.6 gives for it, so that $q_\alpha = \prod_k g_k$.

Item 1. We look for one pattern and one stage that fails with constant probability. Since $q \not\equiv 1$, some pattern α has $q_\alpha \not\equiv 1$, and then some stage has $g_k \not\equiv 1$. As $q_\alpha \leq g_k$, the second property of the stage functions gives

$$\mathbb{E}_U[1 - q_\alpha(x)] \geq \mathbb{E}_U[1 - g_k] \geq \frac{1}{4}.$$

The pattern $x_H = \alpha$ has probability $2^{-|H|} \geq 2^{-c}$, and (8.5) follows.

Item 2. For each pattern we bound q_α between two functions of small norm, and then we recombine the patterns. Let

$$t = 2 \lceil \log_{4/3}(1/\gamma) \rceil,$$

and let $k_1 < \dots < k_s$ be the stages with $g_k \not\equiv 1$, so that $q_\alpha = \prod_{i=1}^s g_{k_i}$. There are two cases.

1. If $s \leq t$, the chain is short. We take $q_\alpha^+ = q_\alpha^- = q_\alpha$.
2. If $s > t$, we keep the first t stages for the upper function and take zero for the lower one, $q_\alpha^+ = \prod_{i=1}^t g_{k_i}$ and $q_\alpha^- = 0$.

We verify three facts about $q_\alpha^\pm$.

Order. In the first case there is nothing to show. In the second case q_α is q_α^+ times further stage functions, and each of them takes values in $\{0, 1\}$:

$$0 = q_\alpha^- \leq q_\alpha = q_\alpha^+ \cdot \prod_{i=t+1}^s g_{k_i} \leq q_\alpha^+.$$

Norm. Each of $q_\alpha^\pm$ is zero or a product of at most t stage functions, and the Fourier norm is submultiplicative. By the first property of the stage functions,

$$\|q_\alpha^\pm\|_F \leq \prod_{i=1}^{\min(s,t)} \|g_{k_i}\|_F \leq 4^t.$$

Gap. In the first case the gap $q_\alpha^+ - q_\alpha^-$ is zero. In the second case it equals q_α^+ . We drop the stage functions in even positions of the retained list, which can only increase the product:

$$q_\alpha^+ = \prod_{i=1}^t g_{k_i} \leq \prod_{\substack{1 \leq i \leq t \\ i \text{ odd}}} g_{k_i}.$$

The remaining stages $k_1, k_3, \dots, k_{t-1}$ satisfy $k_{i+2} \geq k_i + 2$, so they are pairwise non-adjacent and, by the third property, independent. The expectation of their product is therefore the product of their expectations, and the second property bounds each by $3/4$:

$$\mathbb{E}_U[q_\alpha^+] \leq \prod_{\substack{1 \leq i \leq t \\ i \text{ odd}}} \mathbb{E}_U[g_{k_i}] \leq \left(\frac{3}{4}\right)^{t/2} \leq \gamma.$$

So $\mathbb{E}_U[q_\alpha^+ - q_\alpha^-] \leq \gamma$ for every pattern. We now undo the affine change of coordinates, which preserves pointwise inequalities, expectations and Fourier norms. We regard $q_\alpha^\pm$ as functions of the original coordinates outside H , and recombine the patterns:

$$q^\pm(x) = \sum_{\alpha \in \{0,1\}^H} \mathbf{1}[x_H = \alpha] \cdot q_\alpha^\pm(x).$$

These functions are Boolean, because exactly one pattern indicator is nonzero on each input. They have the required properties:

1. $q^-(x) \leq q(x) \leq q^+(x)$, because exactly one pattern indicator is nonzero on each input.
2. $\mathbb{E}_U[q^+(x) - q^-(x)] \leq \gamma$, by averaging over the patterns.
3. $\|q^\pm\|_F \leq 2^{|H|} \cdot 4^t \leq 16 \cdot 2^c \gamma^{-10}$, by the norm bound for functions assembled pattern by pattern and by $4^t \leq 16\gamma^{-10}$, since $2 \log_{4/3} 4 < 10$. $\square$

Part 2: many programs.

Proof of Lemma 8.4. Remove the factors that are identically one. If none remain, take $G = 1$, and if some factor is identically zero, take $G = 0$. Otherwise let $p_i = \mathbb{E}_U[1 - q_i(x)]$ be the failure probability of factor i ; by (8.5), $p_i \geq 2^{-b-2}$. We use the parameters

$$L_* = \ln(16/\eta) \quad \text{and} \quad d = 2\lceil 4L_* \rceil.$$

Thus d is even, and $L_* > 4$ because $\eta < 1/4$. The proof passes from $Q(x)$ to $G(x)$ through two intermediate functions,

$$G(x) \geq F(x) \geq Q_*(x) \geq Q(x),$$

and each of the three replacements costs at most $\eta/16$ in expectation.

First replacement: keep a bounded number of factors. The later steps count terms, so we need a bound on the number of factors. Keep all factors if $\sum_i p_i < L_*$, and otherwise keep the shortest initial list $q_1, \dots, q_m$ whose failure probabilities sum to at least L_* . Let $Q_*(x) = \prod_{i \leq m} q_i(x)$ and $\lambda = \sum_{i \leq m} p_i$. Then:

1. $Q(x) \leq Q_*(x)$, because factors were dropped.
2. $\lambda \leq L_* + 1$: if all factors were kept because their total was below L_* , this is immediate; otherwise it follows from the shortest-list choice and $p_i \leq 1$.
3. $m \leq 2^{b+2}(L_* + 1)$, because each $p_i \geq 2^{-b-2}$.
4. $\mathbb{E}_U[Q_*(x)] \leq \mathbb{E}_U[Q(x)] + \eta/16$. If no factor was dropped the two are equal. Otherwise $\lambda \geq L_*$, and independence gives $\mathbb{E}_U[Q_*(x)] = \prod_{i \leq m} (1 - p_i) \leq e^{-\lambda} \leq \eta/16$.

Second replacement: inclusion-exclusion. We bound $Q_*(x)$ by an expression that involves only products of few factors. Let $Z(x) = \sum_{i \leq m} (1 - q_i(x))$ be the number of failing factors, so $Q_*(x) = \mathbf{1}[Z(x) = 0]$, and put

$$F(x) = \sum_{j=0}^d (-1)^j \binom{Z(x)}{j}. \tag{8.7}$$

For an integer $z \geq 1$, since d is even, Pascal's identity gives $\sum_{j=0}^d (-1)^j \binom{z}{j} = \binom{z-1}{d}$, which lies between 0 and $\binom{z}{d+1}$; for $z = 0$ the sum is one. So F is a pointwise upper bound on Q_* , with excess at most $\binom{Z}{d+1}$:

$$0 \leq F(x) - Q_*(x) \leq \binom{Z(x)}{d+1}.$$

The right side counts the sets of $d+1$ failing factors. The factors are independent, so its expectation is a sum of products of failure probabilities:

$$\begin{aligned} \mathbb{E}_U[F(x) - Q_*(x)] &\leq \sum_{|S|=d+1} \prod_{i \in S} p_i \leq \frac{\lambda^{d+1}}{(d+1)!} \\ &\leq \left(\frac{e\lambda}{d+1} \right)^{d+1} \leq \frac{\eta}{16}. \end{aligned} \tag{8.8}$$

The second inequality holds because each product with distinct indices occurs $(d+1)!$ times in the expansion of λ^{d+1} . The last one uses $\lambda \leq 2L_*$ and $d+1 \geq 8L_*$, which give $(e/4)^{8L_*} \leq e^{-L_*} = \eta/16$.

Next we write F in terms of products of factors. For $J \subseteq \{1, \dots, m\}$ let $Q_J(x) = \prod_{i \in J} q_i(x)$. Each binomial coefficient is a sum of products of failures, $\binom{Z}{j} = \sum_{|S|=j} \prod_{i \in S} (1 - q_i)$, and expanding each such product gives

$$F(x) = \sum_{j=0}^{\min(d,m)} \sum_{|S|=j} \sum_{J \subseteq S} (-1)^{j+|J|} Q_J(x). \quad (8.9)$$

Every term is a product of at most d factors. Let W be the number of signed terms, counted with repetitions, and let $M = \max(m, d)$. Then

$$W = \sum_{j=0}^{\min(d,m)} \binom{m}{j} 2^j \leq 2^d \sum_{j=0}^d \binom{M}{j} \leq \left(\frac{2eM}{d}\right)^d \leq 2^{(b+3)d}. \quad (8.10)$$

Here $1 \leq d \leq M$, so the binomial-sum bound applies. The last step uses $2e < 8$ and $M/d = \max\{1, m/d\} \leq 2^b$, since $b \geq 1$ and $m/d \leq 2^{b+2}(L_* + 1)/(8L_*) \leq 2^b$. Since $d = O(\log(1/\eta))$ and $b \geq 1$, this gives $\log W = O(b \log(1/\eta))$.

Third replacement: approximate each term as one program. It remains to replace each $Q_J(x)$ by a function of small Fourier norm. For this we show that $Q_J(x)$ is itself the acceptance function of a single program with few collisions. For $J = \emptyset$, the product is $Q_J = 1$, and we take $Q_J^- = Q_J^+ = 1$. Now let $J = \{i_1 < \dots < i_r\}$ with $1 \leq r \leq d$, let $\mathcal{P}_i$ be the program of q_i , and let $z_0^{(i)}$ be its start label. Define the program $\mathcal{P}_J$ as follows: run $\mathcal{P}_{i_1}, \dots, \mathcal{P}_{i_r}$ one after the other, and before each $\mathcal{P}_{i_\ell}$ with $\ell \geq 2$ insert one fixed transition that sends both live states to the start label of $\mathcal{P}_{i_\ell}$,

$$T(0) = T(1) = z_0^{(i_\ell)}.$$

Rejection stays absorbing. The programs read disjoint blocks, so $\mathcal{P}_J$ reads every input bit at most once. It reaches its end without rejecting exactly when each $\mathcal{P}_{i_\ell}$ does not reject on its own block, so its acceptance function is the product:

$$q_{\mathcal{P}_J}(x) = \prod_{i \in J} q_i(x) = Q_J(x).$$

The inserted transitions are fixed, and by the convention for programs with two live states in Section 8.1, fixed transitions are not counted in the collision bound. So the colliding free transitions of $\mathcal{P}_J$ are those of its parts:

$$\begin{aligned} \#\{\text{colliding free transitions of } \mathcal{P}_J\} &= \sum_{i \in J} \#\{\text{colliding free transitions of } \mathcal{P}_i\} \\ &\leq b|J| \leq bd. \end{aligned}$$

Hence item 2 of Lemma 8.7 applies to each Q_J as a whole, with $c = bd$ and

$$\gamma = \frac{\eta}{16W},$$

and gives Boolean functions $Q_J^-(x) \leq Q_J(x) \leq Q_J^+(x)$. Let $G(x)$ be the sum (8.9) with each positive term Q_J replaced by Q_J^+ and each negative term $-Q_J$ replaced by $-Q_J^-$. Then:

1. $G(x) \geq F(x)$, because every replacement increases the value.
2. $\mathbb{E}_U[G(x) - F(x)] \leq W\gamma = \eta/16$, because each of the W replacements costs at most γ .
3. $\|G\|_F \leq W \cdot 16 \cdot 2^{bd} \gamma^{-10}$, by the triangle inequality.

Conclusion. The chain $G \geq F \geq Q_* \geq Q \geq 0$ shows that G is a nonnegative pointwise upper bound on Q . The three costs add up to

$$\mathbb{E}_U[G(x)] \leq \mathbb{E}_U[Q(x)] + \frac{3\eta}{16} \leq \mathbb{E}_U[Q(x)] + \eta.$$

For the norm, substituting $\gamma = \eta/(16W)$ and using $d = O(\log(1/\eta))$ and (8.10),

$$\|G\|_F \leq W \cdot 16 \cdot 2^{bd} \left(\frac{16W}{\eta} \right)^{10} = 2^{O(b \log(1/\eta))}. \quad \square$$

Part 3: the theorem.

Proof of Theorem 8.1. Fix a target $v^* = (t, s)$. We write $G_{t,s}$ for its majorant, denoted G_{v^*} in Section 8.1. If the visit indicator $O_{t,s}(x)$ is identically zero or identically one, the functions 0 and 1 have the three properties stated in Step 1 of Section 8.1. Otherwise Proposition 8.2 gives a nonempty family of products $Q_a(x)$, $a \in \mathcal{A}_{t,s}$, with $|\mathcal{A}_{t,s}| \leq t + 1$. Apply Lemma 8.4 to each of them with

$$\eta = \frac{1}{2n|\mathcal{A}_{t,s}|},$$

which satisfies $0 < \eta < 1/4$ since $n \geq 4$, and let $G_{t,s}(x) = \sum_{a \in \mathcal{A}_{t,s}} G_a(x)$. We check the three properties.

1. $G_{t,s}(x) \geq \sum_a Q_a(x) \geq O_{t,s}(x)$, so $G_{t,s}$ is a pointwise upper bound on the visit indicator.
2. Its expectation is the sum of the expectations of the G_a , so the $|\mathcal{A}_{t,s}|$ additive errors add up to $1/(2n)$:

$$\mathbb{E}_U[G_{t,s}(x)] \leq 4 \mathbb{E}_U[O_{t,s}(x)] + |\mathcal{A}_{t,s}| \eta = 4 \mathbb{E}_U[O_{t,s}(x)] + \frac{1}{2n}.$$

3. Its Fourier norm is at most the sum of the Fourier norms of the G_a . Since $|\mathcal{A}_{t,s}| = O(n)$, we have $\log(1/\eta) = O(\log n)$, and for an absolute constant C_0 independent of the target,

$$\|G_{t,s}\|_F \leq |\mathcal{A}_{t,s}| \cdot 2^{O(b \log(1/\eta))} \leq n^{C_0 b}.$$

By (3.1), for $\delta \leq n^{-C_b}$ with $C \geq C_0 + 2$,

$$\begin{aligned} \mathbb{E}_D[O_{t,s}(x)] &\leq \mathbb{E}_D[G_{t,s}(x)] \leq \mathbb{E}_U[G_{t,s}(x)] + \delta \|G_{t,s}\|_F \\ &\leq 4 \mathbb{E}_U[O_{t,s}(x)] + \frac{1}{n}. \end{aligned}$$

All functions use only coordinates through layer t , and projecting D to these coordinates preserves small bias. So the same distribution satisfies the bound for every target. $\square$

8.5 Transferring the cost bound

Proof. We apply Theorem 8.1 to the auxiliary program $\hat{P}$ of Proposition 7.5, whose collision bound is $O(\ell)$. Choose an absolute constant C large enough that the visit bound of Theorem 8.1 holds for $\hat{P}$ whenever $\delta \leq n^{-C\ell}$, and put $r_{\text{loc}} = r_{\text{reward}}(\ell, \mu)$.

The cost is a sum of two kinds of terms. The first counts *repair triggers*, which specify a transition, a source state, and a candidate bit that the scanning repair would flip.

For every free transition t of an interval I_j , a triple

$$a = (t, u, b)$$

is a repair trigger if u is a live state at layer $t - 1$, $b \in \{0, 1\}$, and

$$\sigma_{t-1}(u) \geq \theta_j, \quad \beta_{t,\theta_j}(u, b) \neq b.$$

The first condition says that u satisfies the scanning repair's threshold condition. The second says that, from u , the repair flips the candidate bit b . Let $\mathcal{C}$ be the set of all these triples. These are the designated repair triggers from Step 4 of Section 7.2; the set depends only on the program and the chosen thresholds.

For a trigger $a = (t, u, b)$, let v_a be the corresponding source node in $\hat{P}$, and let $i(a)$ be the input coordinate read at transition t . The trigger is activated on a candidate x precisely when $\hat{P}$ visits v_a and the candidate bit $x_{i(a)}$ equals b . Define

$$\begin{aligned} O_a(x) &= O_{v_0 \rightarrow v_a}(x) = \mathbf{1}[\hat{P} \text{ on } x \text{ visits } v_a], \\ I_a(x) &= \mathbf{1}[x_{i(a)} = b]. \end{aligned}$$

Thus $O_a(x)I_a(x)$ indicates that trigger a is activated.

Let $\mathcal{B}$ index the allowed entrance states $s \in \mathcal{A}_j$ with $\sigma_{t_j}(s) < \theta_j$, and for $\gamma \in \mathcal{B}$ let $O_\gamma(x) = O_{v_0 \rightarrow v_\gamma}(x)$ indicate the visit to the corresponding checkpoint node v_γ . Then

$$\text{Cost}_\theta^P(x) = \sum_{a \in \mathcal{C}} O_a(x)I_a(x) + r_{\text{loc}} \sum_{\gamma \in \mathcal{B}} O_\gamma(x). \quad (8.11)$$

We bound each term under D . For a repair trigger, the triggering event is contained in the visit to its source node, Theorem 8.1 bounds the visit, and under uniform input the triggering bit is read after the visit and is independent of it:

$$\Pr_D[O_a = 1, I_a = 1] \leq \Pr_D[O_a = 1] \leq 4 \Pr_U[O_a = 1] + \frac{1}{n} = 8 \Pr_U[O_a = 1, I_a = 1] + \frac{1}{n}.$$

For a low entrance, Theorem 8.1 applies directly:

$$\Pr_D[O_\gamma = 1] \leq 4 \Pr_U[O_\gamma = 1] + \frac{1}{n}.$$

It remains to sum. There are $O(n)$ triggers and $O(n)$ low entrances, since each free transition has at most six state-bit pairs and each checkpoint has at most two allowed entrance states. As $r_{\text{loc}} = O(\ell)$, the additive errors $1/n$, weighted by 1 and by r_{loc} , sum to $O(\ell)$. By linearity of expectation,

$$\mathbb{E}_D[\text{Cost}_\theta^P(X)] \leq 8 \sum_{a \in \mathcal{C}} \Pr_U[O_a = 1, I_a = 1] + 4r_{\text{loc}} \sum_{\gamma \in \mathcal{B}} \Pr_U[O_\gamma = 1] + O(\ell).$$

All terms are nonnegative, so the right side is at most eight times the uniform cost plus $O(\ell)$, and Proposition 7.5 bounds the uniform cost:

$$\mathbb{E}_D[\text{Cost}_\theta^P(X)] \leq 8 \mathbb{E}_U[\text{Cost}_\theta^P(X)] + O(\ell) = O(\ell). \quad (8.12)$$

Independence of unread bits and the gain estimate are used only under uniform input. $\square$

8.6 From local repairs to a hitting set

The transferred cost bound yields an accepted input near some center in the support of a small-bias source.

Theorem 8.8 (checkpoint HSG). *There is an absolute constant C such that the following holds for every $P \in \mathcal{K}_{3,\ell}$ of density at least $1/16$ and every δ -biased distribution D with $\delta \leq n^{-C\ell}$. For X drawn from D , the fixed procedures above satisfy:*

1. $\mathbb{E}_D[\text{Cost}_\theta^P(X)] = O(\ell)$.
2. $\mathbb{E}_D[d_H(X, P^{-1}(1))] = O(\ell)$.

Consequently, this class has an explicit HSG with seed length $O(\ell \log n)$.

Proof. Item 1 is the expected-cost bound of Section 8.5. For item 2, Proposition 7.5 shows that the repair of a candidate is an accepted input and that it changes at most $\text{Cost}_\theta^P(x)$ coordinates. So for every candidate x ,

$$d_H(x, P^{-1}(1)) \leq d_H(x, \text{Repair}_\theta^P(x)) \leq \text{Cost}_\theta^P(x).$$

Taking expectations under D gives item 2. Taking $C_{\text{bias}} = C$ and C_{edit} large enough to dominate the absolute constant in (8.12) also proves Theorem 2.3 for these same procedures, independently of D .

A good center. Let D be a small-bias source with $\delta = n^{-C\ell}$. A source on all n coordinates may be projected to the free coordinates, since projections preserve small bias. By item 2 the expected distance to an accepted input is at most $C_{\text{edit}}\ell$, so some center c in the support of D is within the integer radius

$$r = \lceil C_{\text{edit}}\ell \rceil$$

of an accepted input.

The generator. A seed consists of a center seed σ and r indices $i_1, \dots, i_r \in \{0, 1, \dots, n\}$. Let $c(\sigma)$ be the center produced by the small-bias source, and let e_i denote the vector that flips coordinate i if i is a free coordinate, and the zero vector if $i = 0$ or i is a fixed coordinate. The output is the center with the indicated flips:

$$\text{Gen}(\sigma, i_1, \dots, i_r) = c(\sigma) \oplus e_{i_1} \oplus \dots \oplus e_{i_r}.$$

Every word within radius r of a center occurs as an output: flip its differing coordinates with distinct indices, and fill the unused positions with 0. In particular the accepted input near the good center is an output, so the generator hits P .

Seed length and explicitness. The seed has two parts:

1. The center seed uses $O(\log n + \log(1/\delta)) = O(\ell \log n)$ bits.
2. The r indices use $r \cdot \lceil \log_2(n+1) \rceil = O(\ell \log n)$ bits.

All parameters depend only on n and ℓ , so this is one generator for the whole class. It only enumerates seeds and indices; it does not compute the program-dependent repair or the auxiliary program. $\square$

9 Restrictions to $\mathcal{K}_{3,\ell}$

Overview of the section. The goal of this section is Theorem 2.4. For every $n \geq 4$ and integer $1 \leq \ell \leq n$, it gives an explicit family $\mathcal{R}_{n,\ell}$ of restrictions, generated from

$$O\left(\left(\log n + \frac{(\log n)^2}{\ell}\right) \log \log n\right)$$

seed bits, such that every length- n program P with three live states and absorbing rejection of density $\mu(P) \geq 1/8$ has a restriction $\rho \in \mathcal{R}_{n,\ell}$ with the following two properties.

1. $\mu(P|_\rho) > 1/16$.
2. $P|_\rho \in \mathcal{K}_{3,\ell}$, after the final fixed merge is appended.

This final transition sends every live state to one accepting state, preserves rejection, and reads no input. It does not change the density. The family does not depend on P ; the successful restriction may.

The starting point is an imported estimate, Theorem 3.6. Let $1 \leq k \leq n$ and $0 < \delta, \gamma < 1$. Let Z be δ -biased, let Y be γ -almost k -wise independent, and let U be uniform, all independent. Then

$$\left| \mathbb{E}[P(Z \oplus (Y \wedge U))] - \mu(P) \right| \leq C_1 n \left(\sqrt{\delta} (C_1 \log_2 n)^{4k} + 2^{-k/2} + \sqrt{\gamma} \right).$$

Conditioned on Z, Y , the string $Z \oplus (Y \wedge U)$ is a uniform completion of the restriction $\rho(Z, Y)$, which leaves coordinate i free when $Y_i = 1$ and fixes it to Z_i otherwise. The left side is therefore $|\mathbb{E}_\rho[\mu(P|_\rho)] - \mu(P)|$,

so the estimate says that a restriction built from pseudorandom strings changes the expected density only slightly. The inequality comes from Forbes and Kelley, with the Fourier-growth bound of Chattopadhyay, Hatami, Reingold and Tal giving the term $(C_1 \log_2 n)^{4k}$, after summing the levels up to k and enlarging the constant. Both estimates are imported; the restriction guarantees below are proved in the paper.

One round of the source, Lemma 3.7, takes both Z and Y to be δ -biased and outputs $\rho = \rho(Z, Y)$. For a suitable δ it has two guarantees.

1. **Density preservation:** $|\mathbb{E}_\rho[\mu(P|_\rho)] - \mu(P)| \leq \eta$.
2. **Local uniformity:** $d_{\text{TV}}(\rho_J, \mathcal{I}^J) \leq \zeta$ for every $J \subseteq [n]$ with $|J| \leq k_0$.

Here $\mathcal{I}^J$ is the ideal restriction of J , which independently leaves each coordinate free with probability $1/2$ and otherwise assigns it a uniform bit. The first guarantee is Theorem 3.6 with a choice of k and δ . The second holds because a small-bias string is close to uniform on few coordinates. We need it because our sufficient condition for membership in $\mathcal{K}_{3,\ell}$ will be expressed through local events, each depending on $O(\log n)$ coordinates, and the density estimate gives no information about such events.

We iterate h independent rounds, each acting on the coordinates that are still free, and write ρ for the resulting restriction. The ideal counterpart $\mathcal{I}_h^J$ independently leaves each coordinate free with probability 2^{-h} and otherwise assigns it a uniform bit. Lemma 9.1 shows that both errors add up over the rounds.

1. $\mathbb{E}_\rho[\mu(P|_\rho)] \geq \mu(P) - h\eta$.
2. $d_{\text{TV}}(\rho_J, \mathcal{I}_h^J) \leq h\zeta$ for every $J \subseteq [n]$ with $|J| \leq k_0$.

To see why $P|_\rho$ lands in the checkpoint class, let $c_1 < \dots < c_M$ be the colliding positions of P . Let $F(\rho)$ be the set of indices k for which c_k stays free, and $X(\rho)$ the set of indices for which c_k is fixed to a colliding map. A fixed colliding map leaves at most two reachable live states, so we place a checkpoint after every c_k with $k \in X(\rho)$. Under the ideal restriction the coordinates are independent, with

$$\Pr[k \in F(\rho)] = 2^{-h}$$

and

$$\Pr[k \in X(\rho)] \geq \frac{1 - 2^{-h}}{2}.$$

Thus free collisions are rare and checkpoints are frequent. We test this on windows of m_0 consecutive entries of the list. If the list has fewer than m_0 entries, it forms the only window. A window is good for ρ if it contains at most $s - 1$ indices of $F(\rho)$ and, when it has m_0 entries, at least one index of $X(\rho)$. Lemma 9.2 is deterministic: if every window is good for ρ , then $P|_\rho$, after the final fixed merge, belongs to $\mathcal{K}_{3,s}$. Indeed, each run of collision-list entries with no fixed colliding position has fewer than m_0 entries and lies in one window. An interval then has at most $s - 1$ free collisions and the one fixed collision that closes it.

Let G be the event that every window is good for ρ . There are at most $n + 1$ windows, and each depends on at most m_0 coordinates. With $k_0 = \min\{n, m_0\}$ and $h \geq 2$,

$$\Pr_\rho[G^c] \leq (n + 1) \left((5/8)^{m_0} + 2^{m_0 - hs} + h\zeta \right).$$

The first term bounds the ideal probability that a full window contains no fixed colliding position. The second bounds the ideal probability that some s positions of a window all stay free. The third is the cost of replacing the ideal restriction by ρ on the coordinates of one window.

The parameters are chosen as follows.

1. $m_0 = \lceil \log_2(256(n + 1)) / \log_2(8/5) \rceil$.
2. $s = \lceil 2m_0/h \rceil$.
3. $\eta = 1/(64h)$.
4. $\zeta = 1/(128(n + 1)h)$.

Then $h\eta = 1/64$ and $\Pr_\rho[\mathbf{G}^c] \leq 1/64$. Since densities lie in $[0, 1]$,

$$\mathbb{E}_\rho[\mu(P|_\rho) \mathbf{1}[\mathbf{G}]] \geq \mu(P) - h\eta - \Pr_\rho[\mathbf{G}^c] \geq \frac{1}{8} - \frac{1}{32} > \frac{1}{16}.$$

Hence some ρ in the support satisfies $\mathbf{G}$ and has $\mu(P|_\rho) > 1/16$. Taking $h = 2 + \lceil 2m_0/\ell \rceil$ gives $s \leq \ell$ and $h = O(1 + \log n/\ell)$. Each round costs $O((\log n + \log h) \log \log n)$ seed bits by Lemma 3.7, and multiplying by h gives the seed length of Theorem 2.4.

9.1 The restriction source and checkpoints

Checkpoints from fixed collisions. The residual keeps the original transition positions: assigned coordinates become fixed transitions. Suppose T_t^b sends two distinct live states to the same live state. Its image contains at most two live states, since there are three live states before the transition. If $\rho_t = b$, then

$$R_t(P|_\rho) \subseteq T_t^b(L_{t-1}^{\text{live}}) \setminus \{\perp\}.$$

Thus $|R_t(P|_\rho)| \leq 2$, and the layer after this fixed colliding transition can serve as a checkpoint. The start and the layer after the final fixed merge are also checkpoints.

One restriction round. Lemma 3.7 supplies a restriction source with density error η , local error ζ , and coordinate bound k_0 . We use its two guarantees:

- (a) For every current residual Q , the next restriction τ satisfies

$$|\mathbb{E}_\tau[\mu(Q|_\tau)] - \mu(Q)| \leq \eta.$$

- (b) On any set of at most k_0 coordinates, the restriction is within total variation distance ζ of independent choices that leave each coordinate free with probability $1/2$, and otherwise assign a uniform bit.

The same parameters work after projection to the free coordinates of any residual.

Iterating the source. Let $h \geq 1$ be an integer, and sample independent restrictions $\rho^{(1)}, \dots, \rho^{(h)}$ from the source of Lemma 3.7 on the original n coordinates, with parameters η, ζ, k_0 . For a tuple of restrictions on the same coordinate set, let Φ be the coordinatewise operation that uses the first assigned bit at each coordinate and leaves that coordinate free if every restriction leaves it free. Write $\rho^{(\leq 0)} = *^n$, and for $1 \leq a \leq h$ set

$$\rho^{(\leq a)} = \Phi(\rho^{(1)}, \dots, \rho^{(a)}).$$

The final restriction is $\rho = \rho^{(\leq h)}$. Each round acts only on the coordinates left free by the preceding rounds, so

$$P|_{\rho^{(\leq a)}} = (P|_{\rho^{(\leq a-1)}})|_{\rho^{(a)}} \quad (1 \leq a \leq h). \quad (9.1)$$

Here the restriction on the right is applied only to the free coordinates of the current residual.

Let $\mathcal{I}_h$ be the distribution on $\{0, 1, *\}$ that assigns probability 2^{-h} to $*$ and probability $(1 - 2^{-h})/2$ to each of 0 and 1. For $J \subseteq [n]$, let $\mathcal{I}_h^J$ be its product distribution on $\{0, 1, *\}^J$.

Lemma 9.1 (iterated restriction source). *The composed restriction ρ has the following properties.*

1. **Density preservation.** *Every program P with three live states and absorbing rejection satisfies*

$$\mathbb{E}_\rho[\mu(P|_\rho)] \geq \mu(P) - h\eta.$$

2. **Local uniformity.** *Every $J \subseteq [n]$ with $|J| \leq k_0$ satisfies*

$$d_{\text{TV}}(\rho_J, \mathcal{I}_h^J) \leq h\zeta.$$

Proof. For the first property, write

$$D_a = \mathbb{E}[\mu(P|_{\rho^{(\leq a)}})]$$

for the expected density after a rounds, where $0 \leq a \leq h$. We show that each round changes this quantity by at most η . Fix $1 \leq a \leq h$ and condition on $\rho^{(1)}, \dots, \rho^{(a-1)}$. The program $Q = P|_{\rho^{(\leq a-1)}}$ is then a fixed program left by earlier restrictions, and $\rho^{(a)}$ is independent of it. By (9.1) and Property 1 of Lemma 3.7,

$$\left| \mathbb{E}_{\rho^{(a)}}[\mu(P|_{\rho^{(\leq a)}})] - \mu(P|_{\rho^{(\leq a-1)}}) \right| \leq \eta.$$

Taking the expectation over the first $a-1$ rounds gives $|D_a - D_{a-1}| \leq \eta$.

Since $\rho^{(\leq 0)} = *^n$ leaves every coordinate free, $P|_{\rho^{(\leq 0)}} = P$ and therefore $D_0 = \mu(P)$. Since $\rho^{(\leq h)} = \rho$, the last term is $D_h = \mathbb{E}_\rho[\mu(P|_\rho)]$. The total change in expected density is the sum of the changes in the individual rounds:

$$\mathbb{E}_\rho[\mu(P|_\rho)] - \mu(P) = D_h - D_0 = \sum_{a=1}^h (D_a - D_{a-1}) \geq -h\eta.$$

The inequality holds because each of the h summands is at least $-\eta$. This proves the first property.

For the second property, fix J with $|J| \leq k_0$. The definition of ρ is coordinatewise, so ρ_J is a function of the tuple of projections:

$$\rho_J = \Phi(\rho_J^{(1)}, \dots, \rho_J^{(h)}).$$

The h strings in this tuple are independent, and each is within total variation distance ζ of $\mathcal{I}^J$ by Property 2 of Lemma 3.7. The tuple is therefore within distance $h\zeta$ of a tuple $(\sigma^{(1)}, \dots, \sigma^{(h)})$ of independent samples from $\mathcal{I}^J$. Applying Φ does not increase total variation distance, so

$$d_{\text{TV}}(\rho_J, \Phi(\sigma^{(1)}, \dots, \sigma^{(h)})) \leq h\zeta.$$

It remains to show that $\Phi(\sigma^{(1)}, \dots, \sigma^{(h)})$ is distributed as $\mathcal{I}_h^J$. All entries $\sigma_i^{(a)}$ are independent, and the output at coordinate i depends only on $\sigma_i^{(1)}, \dots, \sigma_i^{(h)}$, so the output coordinates are independent. Coordinate i is $*$ exactly when all h rounds leave it free, which has probability 2^{-h} . Otherwise it equals the first assigned bit, and this bit is uniform because $\mathcal{I}$ assigns 0 and 1 the same probability. $\square$

9.2 A window condition for bounded collisions

Let $c_1 < c_2 < \dots < c_M$ be the positions of the colliding transitions of the original program P . We call this the collision list. For a position j and a bit b , write T_j^b for the map that transition j applies when its input bit is b . A position c_k is on the list exactly when at least one of $T_{c_k}^0$ and $T_{c_k}^1$ is colliding.

A restriction ρ splits the list into three parts, of which we name two:

1. $F(\rho) = \{k \in [M] : \rho_{c_k} = *\}$, the free colliding positions.
2. $X(\rho) = \{k \in [M] : \rho_{c_k} \in \{0, 1\} \text{ and } T_{c_k}^{\rho_{c_k}} \text{ is colliding}\}$, the fixed colliding positions.

An index in $F(\rho)$ contributes a free colliding transition to the residual program $P|_\rho$. An index in $X(\rho)$ contributes a fixed colliding transition, after which we place a checkpoint. The remaining indices are fixed to noncolliding maps and contribute no colliding transition.

Under the ideal restriction $\rho \sim \mathcal{I}_h^{[n]}$, the coordinates of ρ are independent, and every $k \in [M]$ satisfies the following.

1. $\Pr[k \in F(\rho)] = 2^{-h}$.
2. $\Pr[k \in X(\rho)] \geq (1 - 2^{-h})/2$, because ρ_{c_k} is a uniform bit when it is not $*$, and at least one of the two maps is colliding.

Let m_0 and s be positive integers. If $M \geq m_0$, a window is a set of m_0 consecutive indices,

$$W_k = \{k, k+1, \dots, k+m_0-1\} \quad \text{for } 1 \leq k \leq M-m_0+1.$$

If $M < m_0$, the only window is $[M]$, which may be empty. A window W is good for ρ if it satisfies both conditions:

1. $|W \cap F(\rho)| \leq s-1$.
2. If $|W| = m_0$, then $W \cap X(\rho) \neq \emptyset$.

The first condition bounds the free collisions in the window, and the second guarantees a checkpoint in every full window.

Lemma 9.2 (good windows give checkpoints). *Suppose every window is good for ρ . Place checkpoints at the start, after every transition c_k with $k \in X(\rho)$, and after the final fixed merge. Then $P|_\rho$, with that merge appended, belongs to $\mathcal{K}_{3,s}$.*

Proof. Write $X(\rho) = \{a_1 < \dots < a_q\}$, where $q = 0$ is allowed, and set $a_0 = 0$ and $a_{q+1} = M+1$. The indices strictly between two consecutive fixed colliding positions form a run,

$$R_i = \{a_i + 1, \dots, a_{i+1} - 1\} \quad \text{for } 0 \leq i \leq q.$$

We first bound the free collisions in a run by showing that every run lies in a window. Every run has $|R_i| < m_0$: otherwise its first m_0 indices form a window disjoint from $X(\rho)$, which contradicts the second condition. If $M \geq m_0$, a set of fewer than m_0 consecutive indices is contained in some window W_k , and if $M < m_0$, it is contained in the only window $[M]$. The first condition for that window gives

$$|R_i \cap F(\rho)| \leq s-1.$$

Now consider the interval between two consecutive checkpoints. It contains the transitions indexed by R_i for some i , followed by the transition that closes the interval. The closing transition is $c_{a_{i+1}}$ when $i < q$, and the final merge when $i = q$. Among the indices in R_i , only those in $F(\rho)$ are colliding in $P|_\rho$, and the closing transition adds one fixed colliding transition. Original transitions outside the collision list are never colliding. The interval therefore contains at most

$$(s-1) + 1 = s$$

colliding transitions.

Finally, every checkpoint other than the start follows a fixed colliding map, so at most two live states are reachable there by the fixed-map argument in Section 9.1. At the start only the start state is reachable. Hence $P|_\rho$ with the merge appended belongs to $\mathcal{K}_{3,s}$. $\square$

9.3 A dense residual with checkpoints

Let $\mathbf{G}$ be the event that every window is good. By Lemma 9.2, it gives the required checkpoint structure. To obtain sufficient density on some restriction in $\mathbf{G}$, it is enough to arrange

$$h\eta + \Pr_\rho[\mathbf{G}^c] < \frac{1}{16}.$$

Indeed, if every restriction in $\mathbf{G}$ had residual density at most $1/16$, then, since all residual densities lie in $[0, 1]$,

$$\mathbb{E}_\rho[\mu(P|_\rho)] \leq \frac{1}{16} \Pr_\rho[\mathbf{G}] + \Pr_\rho[\mathbf{G}^c] \leq \frac{1}{16} + \Pr_\rho[\mathbf{G}^c].$$

This contradicts the lower bound $\mathbb{E}_\rho[\mu(P|_\rho)] \geq 1/8 - h\eta$ from Lemma 9.1. We choose the parameters to meet this error budget.

Proposition 9.3 (dense checkpoint residual). *Set*

$$m_0 = \left\lceil \frac{\log_2(256(n+1))}{\log_2(8/5)} \right\rceil.$$

For every integer $h \geq 2$, there is an explicit restriction source with seed length

$$O(h(N + \log_2 h)g) \tag{9.2}$$

such that, for every program P with three live states, absorbing rejection, and density at least $1/8$, some supported restriction ρ has the following properties:

(a) $\mu(P|_\rho) > 1/16$.

(b) After the final fixed merge, the residual belongs to $\mathcal{K}_{3,\ell}$, where

$$\ell = \left\lceil \frac{2m_0}{h} \right\rceil. \tag{9.3}$$

The source depends only on n and h . In particular, $\ell \leq C(1 + N/h)$ for an absolute constant C .

Proof. Write $s = \lceil 2m_0/h \rceil$, so $s = \ell$ in the statement. Use h independent rounds of Lemma 3.7 with the following parameters:

(a) $k_0 = \min\{n, m_0\}$.

(b) $\eta = 1/(64h)$.

(c) $\zeta = 1/(128(n+1)h)$.

These satisfy the hypotheses of the lemma.

Density. Property 1 of Lemma 9.1 gives

$$\mathbb{E}_\rho[\mu(P|_\rho)] \geq \mu(P) - h\eta \geq \frac{1}{8} - \frac{1}{64} = \frac{7}{64}. \tag{9.4}$$

Structure. We first bound the two ways a window can fail under the ideal restriction. A full window has no fixed colliding position with probability at most

$$\Pr_{\text{ideal}}[\text{no fixed colliding position in the window}] \leq \left(\frac{1 + 2^{-h}}{2} \right)^{m_0} \leq (5/8)^{m_0},$$

since $h \geq 2$. For a window w with $|w| \leq m_0$, a union bound over its s -subsets gives, when $s \leq |w|$,

$$\Pr_{\text{ideal}}[\text{at least } s \text{ free colliding positions in } w] \leq \binom{|w|}{s} 2^{-hs} \leq 2^{m_0 - hs}.$$

When $s > |w|$, this event is empty and the same upper bound holds.

The event that a window is bad is the union of these two events, with the first included only for a full window. It depends on at most k_0 original coordinates. Applying Property 2 of Lemma 9.1 to this event adds at most $h\zeta$. There are at most $n+1$ windows, so a union bound gives

$$\Pr_\rho[\mathbf{G}^c] \leq (n+1) \left((5/8)^{m_0} + 2^{m_0 - hs} + h\zeta \right). \tag{9.5}$$

The choice of m_0 makes the first term at most $1/(256(n+1))$. Also $hs \geq 2m_0$ and $m_0 \geq \log_2(256(n+1))$, so

$$2^{m_0-hs} \leq 2^{-m_0} \leq \frac{1}{256(n+1)}.$$

Substituting these estimates and the value of ζ into (9.5) gives

$$\Pr_\rho[G^c] \leq \frac{1}{256} + \frac{1}{256} + \frac{1}{128} = \frac{1}{64}.$$

Combining the guarantees. The required error budget holds:

$$h\eta + \Pr_\rho[G^c] \leq \frac{1}{64} + \frac{1}{64} = \frac{1}{32} < \frac{1}{16}. \quad (9.6)$$

The argument preceding the proposition therefore gives a supported restriction in $\mathbf{G}$ with residual density greater than $1/16$. Lemma 9.2 places its residual, after the final merge, in $\mathcal{K}_{3,s}$.

Seed length. We have $k_0 = O(N)$, and the reciprocals of η and ζ have logarithms $O(N + \log h)$. Since $g \geq 1$ for $n \geq 4$, the seed bound in Lemma 3.7 is $O((N + \log h)g)$ per round. Multiplying by h proves (9.2). All source parameters depend only on n and h . $\square$

Proof of Theorem 2.4. Use m_0 from Proposition 9.3 and take

$$h = 2 + \left\lceil \frac{2m_0}{\ell} \right\rceil.$$

Then $\lceil 2m_0/h \rceil \leq \ell$. Proposition 9.3 supplies a restriction family depending only on n and ℓ , containing for each target P a restriction whose residual has density greater than $1/16$ and belongs to $\mathcal{K}_{3,\ell}$ after the final fixed merge. Since $m_0 = O(\log n)$, we have $h = O(1 + \log n/\ell)$. The seed bound (9.2) is therefore

$$O\left(\left(\log n + \frac{(\log n)^2}{\ell}\right) \log \log n\right). \quad \square$$

10 Assembly and seed length

We first assemble a generator at density threshold $1/4$, then apply a separate HSG error-reduction lemma.

10.1 Completing the generator and balancing the seed costs

For each restriction and each generated center, enumerate all changes to at most $O(\ell)$ free coordinates. Specifying their positions costs $O(\ell \log n)$ seed bits. For the restriction guaranteed by Proposition 9.3, Section 8 supplies a center for which one of these choices is accepted by the residual. Inserting the fixed coordinates produces an input accepted by the original program. This constructs the generator $G_\wedge$ required by Corollary 4.5. That corollary combines it with a width-three HSG and two layer indices to obtain the width-four generator G_0 at density threshold $1/4$.

The auxiliary program and threshold choices are used only to prove that successful choices exist. Computing the generator requires only the source seeds and the positions to change; it does not require the target program or its checkpoints.

Increasing the number h of restriction steps decreases the collision bound ℓ . For the choices of h we use, each step costs $O(\log n \log \log n)$ seed bits. The centers and edits together cost $O(\ell \log n)$. Substituting $\ell \leq C(1 + \log n/h)$ gives a total of

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

Taking h of order $\sqrt{\log n / \log \log n}$ balances the first and last terms. The resulting seed length is $O((\log n)^{3/2} \sqrt{\log \log n})$.

Finally, Doron–Hoza’s HSG error reduction [DH25, Lemma 4.7] extends this constant-density generator to density threshold ε . The new generator concatenates segments of $t = \lceil 2 \ln(2/\varepsilon) \rceil$ independently generated outputs, preserving their coordinate positions and enumerating the layer indices at which the segments are joined. If r_0 is the seed length of G_0 , the new seed length is

$$tr_0 + (t-1)\lceil \log_2(n+1) \rceil = O\left((\log n)^{3/2} \sqrt{\log \log n} (1 + \log(1/\varepsilon))\right).$$

Section 10.3 states this import precisely and proves the full bound (1.1). In particular, the restriction and repair constructions themselves need only the constant density thresholds used above.

10.2 Constant-density assembly

Proposition 10.1 (constant-density generator). *Width-four ROBPs of length at most n , at density threshold $1/4$, have an explicit HSG with seed length*

$$r_0(n) = O\left((\log n)^{3/2} \sqrt{\log \log n}\right). \quad (10.1)$$

Proof. We first construct a generator $G_\wedge$ for programs with three live states and absorbing rejection at density threshold $1/8$, and then pass to width four by Corollary 4.5. Fix an integer $h \geq 2$, to be chosen at the end, and let ℓ be the collision parameter that Proposition 9.3 gives for h rounds. It satisfies $\ell \leq C(1 + N/h)$ for an absolute constant C .

The seed of $G_\wedge$ has three parts.

1. The seeds of the h restriction rounds of Proposition 9.3. They determine a restriction $\rho \in \{0, 1, *\}^n$.
2. A seed σ of the small-bias source used in Theorem 8.8, with bias $\delta = n^{-C_{\text{bias}}\ell}$. It determines a center $c(\sigma) \in \{0, 1\}^n$.
3. Edit indices $i_1, \dots, i_{r'} \in \{0, 1, \dots, n\}$, where $r' = \lceil C_{\text{edit}}\ell \rceil$ is the edit budget from the proof of Theorem 8.8. Each index uses $\lceil \log_2(n+1) \rceil$ bits; invalid encodings are decoded as 0.

List the free coordinates $M(\rho)$ in their original order, and let $e_i \in \{0, 1\}^{M(\rho)}$ be the word with a single 1 at the i -th coordinate of this list. If $i = 0$ or $i > |M(\rho)|$, let e_i be the zero word. This reindexes the free-coordinate edits of Theorem 8.8 and enumerates the same Hamming ball. The generator projects the center to the free coordinates and applies the edits there:

$$w = c(\sigma)_{M(\rho)} \oplus e_{i_1} \oplus \dots \oplus e_{i_{r'}} \in \{0, 1\}^{M(\rho)}.$$

Its output is the word $x \in \{0, 1\}^n$ that keeps the assigned bits of ρ and fills the free coordinates with w :

$$x_i = \begin{cases} \rho_i, & \text{if } i \notin M(\rho), \\ w_i, & \text{if } i \in M(\rho). \end{cases}$$

The projected center $c(\sigma)_{M(\rho)}$ is a valid center for Theorem 8.8 on the coordinates $M(\rho)$, because a projection of a small-bias string has the same bias bound.

We now show that $G_\wedge$ hits every program P with three live states and absorbing rejection of density $\mu(P) \geq 1/8$. By Proposition 9.3, some restriction ρ in the support of the first part of the seed satisfies $\mu(P|_\rho) > 1/16$, and $P|_\rho$ belongs to $\mathcal{K}_{3,\ell}$ after the final fixed merge is appended. The merge reads no bit and preserves acceptance, so the program with the merge accepts the same words as $P|_\rho$. If $M(\rho) \neq \emptyset$, Theorem 8.8 therefore gives a seed σ and edit indices $i_1, \dots, i_{r'}$ for which the word w is accepted by $P|_\rho$. If $M(\rho) = \emptyset$, the residual program is constant with positive density, and it accepts the empty word. In both cases the output x agrees with ρ on the fixed coordinates and with w on the free ones, so

$$P(x) = P|_\rho(w) = 1.$$

That is, filling the accepted residual word into the restriction gives an input accepted by the original program.

Corollary 4.5 turns $G_\wedge$ into the generator G_0 for width-four ROBPs at density threshold $1/4$. The restriction source, the small-bias source and the edit budget r' depend only on n and h . The target program appears only in the analysis, so no step of the generator needs it.

It remains to bound the seed length of G_0 . It is the sum of three contributions.

1. The h restriction rounds cost $O(h(N + \log_2 h)g)$ seed bits, by Proposition 9.3.
2. The center and the edit indices cost $O(\ell N)$ seed bits, by Theorem 8.8. Since $\ell \leq C(1 + N/h)$, this is $O(N + N^2/h)$.
3. Corollary 4.5 costs $O(N)$ additional seed bits.

The total seed length is therefore

$$O\left(h(N + \log_2 h)g + N + \frac{N^2}{h}\right). \quad (10.2)$$

The first term grows with h and the last term decreases with h . We balance them by choosing

$$h = \left\lceil 2 + \sqrt{N/g} \right\rceil. \quad (10.3)$$

Since $n \geq 4$, we have $1 \leq g \leq N$. Hence $\sqrt{N/g} \geq 1$, which gives $h \leq 4\sqrt{N/g}$ and $\log_2 h = O(N)$. Each term of (10.2) is then $O(N^{3/2}\sqrt{g})$:

1. $h(N + \log_2 h)g = O(\sqrt{N/g} \cdot N \cdot g) = O(N^{3/2}\sqrt{g})$.
2. $N^2/h \leq N^2/\sqrt{N/g} = N^{3/2}\sqrt{g}$.
3. $N \leq N^{3/2}\sqrt{g}$.

Since $N = \log_2 n$ and $g = \log_2 N$, this proves (10.1) for programs of length exactly n .

Finally, a program of length $n' < n$ is a program of length n that ignores its last $n - n'$ coordinates. Padding does not change its density, and the first n' bits of an accepted output of G_0 are accepted by the original program. $\square$

10.3 Error reduction

For $H \subseteq \{0, 1\}^n$, its *position-aligned t -fold splice* is

$$H^{\otimes t} = \left\{ x_{b_0+1:b_1}^{(1)} \circ \cdots \circ x_{b_{t-1}+1:b_t}^{(t)} : 0 = b_0 \leq b_1 \leq \cdots \leq b_t = n, x^{(j)} \in H \right\}.$$

Coordinates keep their original positions, and empty blocks are allowed.

Lemma 10.2 (HSG error reduction [DH25, Lemma 4.7]). *Suppose H hits every length- n , width-four ROBP of density greater than $1/4$. For every $0 < \varepsilon \leq 1$, the set $H^{\otimes t}$, with*

$$t = \lceil 2 \ln(2/\varepsilon) \rceil, \quad (10.4)$$

hits every such program of density at least ε .

This is the stated DH lemma with $\gamma = 1/4$ and their strict density parameter set to $\varepsilon/2$. It requires only a hitting set at constant density, not a pseudorandom generator.

Proof of Theorem 1.5. Let H be the output set of Proposition 10.1. Generate t outputs of G_0 and $t - 1$ layer indices in $\{0, \dots, n\}$, and output their position-aligned splice. Sorting these indices gives a nondecreasing sequence and represents every valid sequence; invalid individual encodings may be decoded as zero. Lemma 10.2 proves the hitting property, including at $\varepsilon = 1$. The seed length is at most

$$t r_0(n) + (t - 1) \lceil \log_2(n + 1) \rceil = O\left((\log n)^{3/2} \sqrt{\log \log n} (1 + \log(1/\varepsilon))\right).$$

Each base output and each splice can be computed in time polynomial in n and $1/\varepsilon$. This proves (1.1). $\square$

References

- [BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. [Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs](#). *SIAM Journal on Computing*, 49(5), 2020.
- [BDVY13] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. [Pseudorandomness for width-2 branching programs](#). *Theory of Computing*, 9(7):283–293, 2013.
- [BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. [Pseudorandom generators for regular branching programs](#). *SIAM Journal on Computing*, 43(3):973–986, 2014.
- [CDG26] Gil Cohen, Dean Doron, and Noam Goldgraber. [A forward-backward weight analysis of INW for permutation branching programs](#). *arXiv:2607.18168*, 2026.
- [CH22] Kuan Cheng and William M. Hoza. [Hitting sets give two-sided derandomization of small space](#). *Theory of Computing*, 18(21):1–32, 2022.
- [CHRT18] Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. [Improved pseudorandomness for unordered branching programs through local monotonicity](#). *50th ACM Symposium on Theory of Computing (STOC)*, 2018. [Full version: ECCC TR17-171](#).
- [De11] Anindya De. [Pseudorandomness for permutation and regular branching programs](#). *26th IEEE Conference on Computational Complexity (CCC)*, pages 221–231, 2011.
- [DH25] Dean Doron and William M. Hoza. [Implications of better PRGs for permutation branching programs](#). *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, LIPIcs 353, 28:1–28:20, 2025. [Full version: ECCC TR24-137](#).
- [FK18] Michael A. Forbes and Zander Kelley. [Pseudorandom generators for read-once branching programs, in any order](#). *59th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2018. [Full version: arXiv:1808.06265](#).
- [GMR⁺12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. [Better pseudorandom generators from milder pseudorandom restrictions](#). *53rd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012. [Full version: arXiv:1210.0049](#).
- [HZ20] William M. Hoza and David Zuckerman. [Simple optimal hitting sets for small-success RL](#). *SIAM Journal on Computing*, 49(4):811–820, 2020.
- [INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. [Pseudorandomness for network algorithms](#). *26th ACM Symposium on Theory of Computing (STOC)*, pages 356–364, 1994.
- [KNP11] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. [Pseudorandom generators for group products](#). *43rd ACM Symposium on Theory of Computing (STOC)*, pages 263–272, 2011.
- [Kra26] Gonen Krak. [Optimal hitting set generators via a potential-descent framework](#). *Electronic Colloquium on Computational Complexity*, Report TR26-178, 2026.
- [MRT19] Raghu Meka, Omer Reingold, and Avishay Tal. [Pseudorandom generators for width-3 branching programs](#). *51st ACM Symposium on Theory of Computing (STOC)*, 2019. [Full version: arXiv:1806.04256](#).
- [Nis92] Noam Nisan. [Pseudorandom generators for space-bounded computation](#). *Combinatorica*, 12(4):449–461, 1992.
- [NN93] Joseph Naor and Moni Naor. [Small-bias probability spaces: Efficient constructions and applications](#). *SIAM Journal on Computing*, 22(4):838–856, 1993.
- [Ste12] Thomas Steinke. [Pseudorandomness for permutation branching programs without the group theory](#). *Electronic Colloquium on Computational Complexity*, Report TR12-083, 2012.
- [SZ11] Jiří Šíma and Stanislav Žák. [Almost \$k\$ -wise independent sets establish hitting sets for width-3 1-branching programs](#). *Computer Science – Theory and Applications (CSR)*, LNCS 6651, pages 120–133, 2011.
- [SZ21] Jiří Šíma and Stanislav Žák. [A polynomial-time construction of a hitting set for read-once branching programs of width 3](#). *Fundamenta Informaticae*, 184(4):307–354, 2021. [Full version: arXiv:2101.01151](#).
- [SZ95] Michael Saks and David Zuckerman. Unpublished, 1995.