Breaking the 4^k Barrier for the k-Distinct Language

Ran Ben Basat[a]

[a]*Department of Computer Science, University College London, London, United Kingdom*

Abstract

For integers $k \leq n$, let $L_{k,n}$ be the set of words over $[n]$ of length at most k in which no symbol is repeated. We present a nondeterministic finite automaton (NFA) of size $3.918^k n^{O(1)}$, improving on the $4^{k+o(k)}n^{O(1)}$ construction of Ben-Basat, Gabizon, and Zehavi.

Our proof organizes several classical ingredients—product automata, hashing, and coefficient estimates—into a *gadget-amplification framework*: We take the product of many copies of a small local NFA gadget, whose language is a subset of $L_{r,c}$, and hash the k input symbols to copies and local colors. The hash family guarantees that, for every repetition-free input, some hash sends at most r symbols to each copy such that the resulting projection in every copy is accepted by the local gadget. Taking the nondeterministic union of the corresponding product NFAs yields a global NFA. Amplifying a 200-state gadget for $L_{6,11}$ obtained from the small Witt design $S(4, 5, 11)$, this framework gives a $3.967^k n^{O(1)}$-size NFA.

We then introduce the *compose-and-compress* technique, which deletes the expensive middle layers of these products and replaces paths across the deleted bands with sound one-symbol shortcut transitions. We apply it twice, once for enhancing the amplification framework and again for the local gadget, obtaining the stated result.

Keywords: parameterized automata, nondeterministic finite automata, k-distinct language, derandomization, separating families, Witt design

Email address: r.benbasat@cs.ucl.ac.uk (Ran Ben Basat)

1. Introduction

For a finite alphabet C and an integer $r \geq 0$, let

$$L_{r,C} \triangleq \{x_1 \cdots x_j : 0 \leq j \leq r,\ x_i \in C, \text{ and } x_i \neq x_{i'} \text{ whenever } i \neq i'\} .$$

The empty word is included. For $C = [n]$ we write $L_{r,n}$. Ben-Basat, Gabizon, and Zehavi use the name k-distinct language for the sublanguage of $L_{k,n}$ consisting of words of length exactly k [1]. This convention does not affect the size bounds in this paper. Our NFA constructions have one layer of states for each input length. If all states are accepting, they recognize $L_{k,n}$; if only the states in layer k are accepting, they recognize the traditional exact-length language. We use the prefix-closed convention because it simplifies the presentation of the techniques below.

Ben-Basat, Gabizon, and Zehavi proved a lower bound with exponential base two and an upper bound of

$$4^k k^{O(\log^2 k)} n^{O(1)} = 4^{k+o(k)} n^{O(1)} .$$

We improve the upper-bound base in the same NFA model. Every transition in this paper reads exactly one input symbol, and the size of an automaton is $|Q| + |\Delta|$.

1.1. Related work

The NFA-size question for the k-distinct language was posed publicly on Theoretical Computer Science Stack Exchange in 2014 [2]. Ben-Basat, Gabizon, and Zehavi subsequently introduced the parameterized-automata formulation of the k-distinct language [1]. Besides the ordinary NFA upper and lower bounds recalled above, they constructed a bounded-ambiguity NFA for approximate counting and a nondeterministic XOR automaton, and showed how such automata combine with problem-specific automata for k-Path, r-Dimensional k-Matching, and Module Motif. Molina Lovett and Shallit studied the special case $k = n$, in which the exact-length language consists of all permutations of the alphabet [3]. They proved asymptotically tight bounds $4^n n^{-(\lg n)/4+\Theta(1)}$ for the minimum size of a regular expression describing this language. Their work concerns regular-expression size rather than ordinary NFA size and does not provide a sub-4^k NFA construction. To our knowledge, the $O^*(4^{k+o(k)})$ NFA bound of [1] had not been improved before the present work.[1]

[1] $O^*(f(k))$ is shorthand for $f(k) \cdot n^{O(1)}$.

The algorithmic lineage is broader. Color-coding isolates a sought solution by a random coloring and derandomizes the choice with perfect hash families [4]. Divide-and-color recursively separates a solution into randomly colored parts [5], whereas Koutis introduced the algebraic multilinear-monomial method and obtained an $O^*(2^{3k/2})$ algorithm for general k-Path [6], which Williams sharpened to a randomized $O^*(2^k)$ algorithm [7]; narrow sieves extend this viewpoint to paths and packings [8]. These are the three techniques from which the automata constructions of Ben-Basat, Gabizon, and Zehavi are distilled. Representative families and their associated separating collections provide another deterministic way to retain only sets that can still be extended disjointly [9, Section 4.2]; Zehavi gives a systematic account of mixtures of representative sets, narrow sieves, and divide-and-color [10].

The ordinary $O^*(4^{k+o(k)})$ NFA relevant here arises from divide-and-color. Our construction follows a different route which we call the *gadget-amplification framework*. Starting from a local NFA gadget of capacity r over c colors, we take the product of many copies and hash every global input symbol to one copy and one local color. For every repetition-free input of at most k symbols, a covering hash family supplies a hash that assigns at most r symbols to each copy and makes each local projection one accepted by that copy. The nondeterministic union of the corresponding product NFAs is the resulting global NFA.

We note that the improved NFA construction of this paper does not translate into a new state-of-the-art algorithm for problems like k-Path. Nederlof recently gave a deterministic $2^{k+O(\sqrt{k}\log^2 k)}(n+m)\log n$-time algorithm for weighted directed k-Path in the word-RAM model, assuming that every edge weight fits in one word [11].

1.2. Overview of our results

We now state our main result.

Theorem 1 (Main theorem)**.** *For every $1 \le k \le n$, an acyclic NFA recognizing $L_{k,n}$ can be constructed deterministically such that*

$$|Q| + |\Delta| \le 2^{1.96992k} n^{O(1)} < 3.918^k n^{O(1)} \ .$$

The construction time obeys the same bound.

As mentioned above, the same transition graph, with precisely the states in layer k accepting, gives the identical bound for the exact-length-k convention of [1].

To prove the theorem, we develop the gadget-amplification framework in three stages. We first construct a constant-size NFA gadget from a (c, a, b)-separating family $\mathcal{F}$. The gadget recognizes $L_{a+b,c}$ using

$$\sum_{j=0}^{a-1} \binom{c}{j} + |\mathcal{F}| + \sum_{j=0}^{b-1} \binom{c}{j}$$

states. Although asymptotically efficient separating families are well studied [9], here we need a small construction for fixed c, a, b. We identify the small Witt design $S(4, 5, 11)$ as an $(11, 3, 3)$-separating family of size 66, which yields a compact 200-state construction for $L_{6,11}$.

The framework takes the product of many copies of a local gadget and uses a hash to assign every global input symbol to one copy and one local color. A hash succeeds on a repetition-free input when every copy receives at most the gadget capacity and the resulting local projection is accepted by that copy. A covering hash family supplies a successful hash for every repetition-free input of at most k symbols, and we take the nondeterministic union of the corresponding product NFAs. For the complete Witt gadget, optimizing a fixed histogram of local loads without compressing the product state space already yields an NFA of size $O^*(3.967^k)$ for $L_{k,n}$.

We then refine the same framework using compose-and-compress (CaC), which combines several partial gadgets into a larger one. A partial gadget is a layered NFA that accepts only repetition-free words and comes with a downward-closed family of *certified* color sets, each of size at most r. A set is certified when the gadget accepts every ordering of its elements. Every certified set S also has a certified superset of each cardinality from $|S|$ through r. CaC forms the product of several partial gadgets and deletes its large middle layers while preserving soundness.

We use CaC at two scales within the gadget-amplification framework. At the growing scale, we compose many copies of the automaton and delete their central product layers; this yields the intermediate-block NFA and an $O^*(3.925^k)$ bound. At the finite scale, we compose eleven Witt gadgets and delete the middle layers of their product to obtain a better starting partial gadget. Applying the growing-scale construction to that gadget gives the final $O^*(3.918^k)$ bound. Table 1 summarizes the progression.

The main proof analyzes random hash functions because this gives the cleanest exponent calculation. Appendix B applies known explicit splitter and restriction-family constructions to replace both random families by explicit

Stage	State accounting	Resulting size
Full-product gadget amplification with a fixed finite load histogram (§4)	All states of the raw product	$O^*(3.967^k)$
Growing compose-and-compress amplification with the Witt automaton (§6)	Witt automaton composition while compressing the product's middle layers	$O^*(3.925^k)$
Compose-and-compress of eleven Witt gadgets, followed by growing amplification (§7)	An improved finite partial gadget and the same two-tail analysis	$O^*(3.9175^k)$

Table 1: The three proof stages. The last gives the main theorem.

splitter and restriction families. The resulting deterministic procedure lists all states and ordinary transitions in $2^{1.96992k}n^{O(1)} = O^*(3.918^k)$ time.

2. Distinct-word gadgets and raw products

An NFA over an alphabet Σ is a tuple (Q, Δ, q_0, F) with $\Delta \subseteq Q \times \Sigma \times Q$. It is *layered* if Q is partitioned into $Q_0, \ldots, Q_r$, the initial state lies in Q_0, and every transition goes from Q_i to Q_{i+1}. A state in Q_i has rank i. Every layered NFA is acyclic.

Definition 2 (Complete gadget)**.** Let C be a finite color set and let $0 \le r \le |C|$. A *complete gadget of capacity r over C* is a layered NFA with layers $Q_0, \ldots, Q_r$, a unique initial state in Q_0, all states accepting, and language exactly $L_{r,C}$.

That is, a complete gadget accepts precisely the repetition-free words of length at most r. For a layered NFA H with layers $Q_0, \ldots, Q_r$, we define its *state polynomial* by

$$A_H(z) \triangleq \sum_{j=0}^{r} |Q_j| z^j \ .$$

In particular, the number of states, denoted $s(H)$, satisfies

$$s(H) = A_H(1) \ .$$

We call H *symmetric* if $|Q_j| = |Q_{r-j}|$ for every j.

For a *complete* gadget G of capacity r over a c-element color set, we also define its *set polynomial* by

$$B_G(z) \triangleq \sum_{j=0}^{r} \binom{c}{j} z^j \ .$$

The coefficient of z^j is the number of j-element color sets, and every ordering of each such set is accepted by G. Thus, A_G counts states by layer, whereas B_G counts repetition-free color sets by cardinality.

For $1 \le i \le m$, let G_i be a complete gadget over C_i with capacity r_i. Assume that the palettes C_i are disjoint. Their *raw product* has a state set $Q(G_1) \times \cdots \times Q(G_m)$, all accepting, with the initial state being the tuple of initial states of the automata. Reading a color from C_i moves only the ith coordinate. A set $S \subseteq (\bigcup_{i=1}^{m} C_i)$ is *product-feasible* if

$$|S \cap C_i| \le r_i \qquad (1 \le i \le m) \ .$$

Lemma 3 (Raw product)**.** *The raw product accepts every ordering of every product-feasible set and accepts no word with a repeated color. Its state polynomial is*

$$\prod_{i=1}^{m} A_{G_i}(z) \ ,$$

and the polynomial counting product-feasible sets by cardinality is

$$\prod_{i=1}^{m} B_{G_i}(z) \ .$$

Proof. A product state has rank equal to the sum of its local ranks, which gives the state polynomial. A product-feasible set is the disjoint union of one color set of size at most r_i from each palette, which gives the second polynomial. Any prescribed ordering can be read by interleaving the corresponding local paths. If a color repeats, it repeats inside its unique local palette, and the corresponding complete gadget has no path with that repeated label. □

As a concrete example, let G be the complete gadget of capacity two over $\{0, 1\}$ shown in Figure 1. It has four states, all accepting, and state polynomial

$$A_G(z) = 1 + 2z + z^2 \ .$$

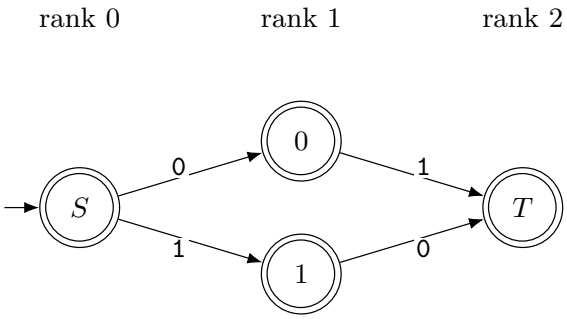


Figure 1: A complete two-color gadget of capacity two. Every arrow reads one color.

Take two copies on the disjoint palettes $\{00, 01\}$ and $\{10, 11\}$. A product state is an ordered pair, and each input symbol moves exactly one coordinate. Their raw product, which is complete over the four-color union, appears in Figure 2.

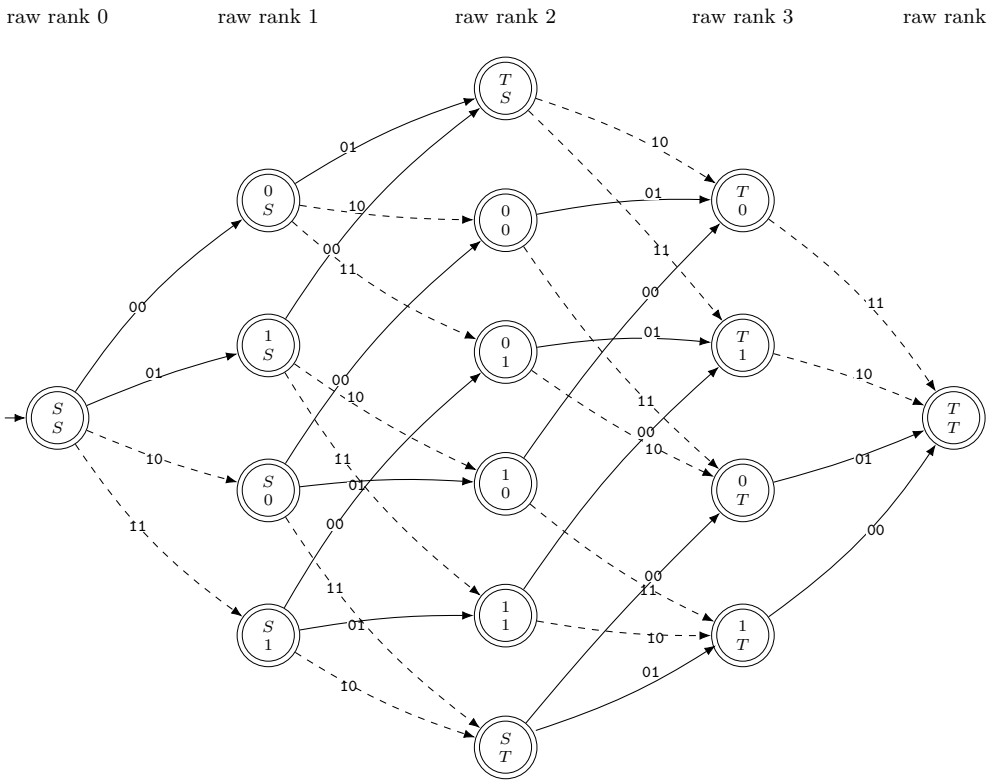


Figure 2: The raw product of two complete two-color gadgets. Solid edges move the first copy and dashed edges move the second. The graph has 16 states and 32 ordinary labeled transitions.

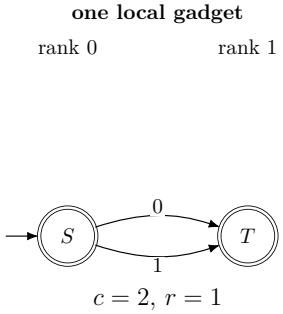
The product polynomial is

$$A_G(z)^2 = (1 + 2z + z^2)^2 = 1 + 4z + 6z^2 + 4z^3 + z^4 \ .$$

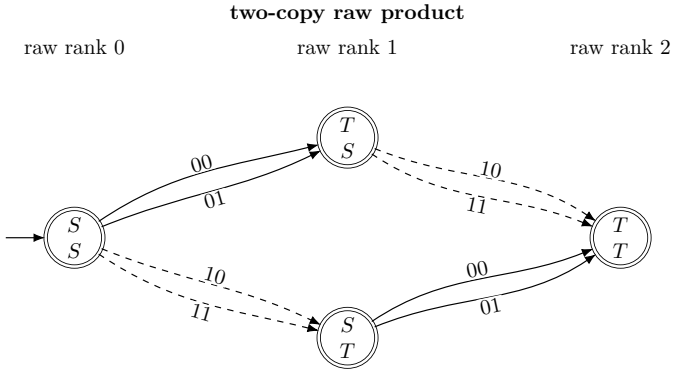
Thus its coefficients $1, 4, 6, 4, 1$ are exactly the numbers of product states in ranks $0, 1, 2, 3, 4$, respectively, as visible in Figure 2.

The raw product need not itself be a complete gadget over the union of the palettes: a repetition-free word can overload one palette. The smallest symmetric example is shown in Figure 3b. Each local gadget has two colors and capacity one. The product accepts at most one color from each palette, so it accepts 00 10 but rejects the repetition-free word 00 01. This distinction is

the reason that, when we later introduce the compose-and-compress technique, we record which color sets are guaranteed to be readable.



(a) One complete gadget with $c = 2$ and $r = 1$.



(b) The two-copy raw product, viewed as a partial gadget. Solid edges move the first copy; dashed edges move the second.

Figure 3: A complete gadget and its two-copy raw product. The product has four states with rank table $(1, 2, 1)$. It accepts 00 10, but rejects 00 01: both colors of the latter word belong to the first palette and therefore overload its capacity-one gadget.

3. Separating families and the Witt gadget

3.1. A general separator construction

Definition 4 ((c, a, b)-separating family)**.** Let C be a color set of size c and let $a, b \geq 1$. A family $\mathcal{F} \subseteq 2^C$ is a (c, a, b)-separating family if, for every two disjoint sets $P, S \subseteq C$ with $|P| = a$ and $|S| = b$, some $W \in \mathcal{F}$ satisfies

$$W \cap P = \varnothing, \qquad S \subseteq W \ .$$

Thus W avoids the colors in P and contains the colors in S.

Proposition 5 (Complete gadget from a separator)**.** *Suppose $c \geq a + b$ and $\mathcal{F}$ is a (c, a, b)-separating family. Then there is a layered NFA G over C whose layer sizes are*

$$\binom{c}{0}, \binom{c}{1}, \ldots, \binom{c}{a-1}, |\mathcal{F}|, \binom{c}{b-1}, \ldots, \binom{c}{1}, \binom{c}{0} \ . \tag{1}$$

The automaton accepts exactly $L_{a+b,C}$, so it is a complete gadget of capacity $a+b$. Its set polynomial is

$$B_G(z) \triangleq \sum_{j=0}^{a+b} \binom{c}{j} z^j \ .$$

Proof. During the first $a-1$ positions, the state stores the set of colors already read. When the ath color is read, the NFA nondeterministically guesses a member $W \in \mathcal{F}$ that avoids all a prefix colors. From a middle-layer state W, upon reading $x \in W$, the NFA may move to any $(b-1)$-set $R \subseteq W \setminus \{x\}$. Thereafter, from a state R, reading $x \in R$ moves to $R \setminus \{x\}$. This gives the layer sizes in (1).

Every path is repetition-free: the prefix is stored as a set, the guessed member of $\mathcal{F}$ avoids the whole prefix, and the suffix is consumed from a set contained in that member. Conversely, take a repetition-free word of length $a+b$. Apply the separating property with its prefix as P and its suffix as S. The resulting W gives a path for the word. A shorter repetition-free word can be extended to length $a+b$, so it appears as a prefix of such a path. Since every state is accepting, the automaton recognizes exactly $L_{a+b,C}$. □

Proposition 5 gives a general gadget construction from a small (c, a, b)-separating family. Prior work provides asymptotically efficient families when the parameters grow [9]. Here we keep c, a, b fixed and test several ways to reduce the family size, including SAT solving, puncturing existing families to reduce a and b, and lifting them to increase a and b.

Among the separating families we tried, the one in the following subsection gives the smallest NFA size bound when used in our global construction. We note that it is possible that by designing a new small (c, a, b)-separating family for suitable c, a, b values one can produce a smaller NFA for $L_{k,n}$ using our techniques.

We release the other separating families found in our search, together with a standalone exact verifier, as an open-source artifact [12], as these may be of independent interest for other applications. Here and in the artifact, *separating family* refers to the set-system notion used in Proposition 5; it is unrelated to the hash-function splitters used in Appendix B. The proofs in this paper do not require running the verifier.

3.2. The small Witt design

We use the classical small Witt design $S(4,5,11)$, a collection of five-element blocks on eleven points in which every four points lie in exactly one block; see, e.g., [13, Chapter IV]. For completeness, we give the explicit cyclic realization used in this paper. Let $X \triangleq \mathbb{Z}_{11} = \{0,1,\dots,10\}$, and consider the six base blocks

$$B_1 \triangleq \{0,1,2,3,5\},\quad B_2 \triangleq \{0,1,2,6,9\},\quad B_3 \triangleq \{0,1,2,7,8\}\ ,$$
$$B_4 \triangleq \{0,1,3,4,7\},\quad B_5 \triangleq \{0,1,3,6,8\},\quad B_6 \triangleq \{0,1,5,7,9\}\ .$$

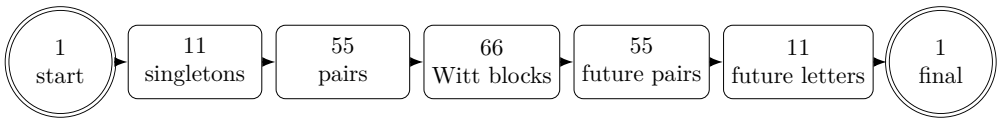
For every $i \in \{1,\dots,6\}$ and $a \in \mathbb{Z}_{11}$, define $B_i + a \triangleq \{x + a \pmod{11} : x \in B_i\}$, and let $\mathcal{B} \triangleq \{B_i + a : 1 \le i \le 6,\ a \in \mathbb{Z}_{11}\}$. The cited construction has the property that every four-subset of X lies in exactly one block of $\mathcal{B}$. It has $|\mathcal{B}| = \frac{\binom{11}{4}}{\binom{5}{4}} = 66$ blocks.

Lemma 6 (Three colors against three)**.** *For every pair of disjoint triples $P, S \subseteq X$, some block $W \in \mathcal{B}$ contains S and avoids P.*

Proof. For each $x \in X \setminus S$, the four-set $S \cup \{x\}$ lies in a unique block. Each block containing S contains exactly two points of $X \setminus S$, so exactly $8/2 = 4$ blocks contain S. Each $p \in P$ belongs to only one of these four blocks; hence the three points of P spoil at most three of them. □

Thus the 66 blocks form an $(11,3,3)$ separating family. The resulting automaton has the seven layers

layer	0	1	2	3	4	5	6
count	1	11	55	66	55	11	1 .



A Witt block avoids the first three colors
and contains the last three colors.

Figure 4: The seven layers of the Witt gadget. A middle block avoids the first three colors and contains the final three colors.

Applying Proposition 5 with $a = b = 3$ and using Lemma 6 yields the following.

Proposition 7 (Witt gadget)**.** *The automaton obtained from the Witt design is a complete gadget of capacity six over X. Its polynomials are*

$$A_W(z) = 1 + 11z + 55z^2 + 66z^3 + 55z^4 + 11z^5 + z^6 \ ,$$

$$B_W(z) = \sum_{j=0}^{6} \binom{11}{j} z^j = 1 + 11z + 55z^2 + 165z^3 + 330z^4 + 462z^5 + 462z^6 \ .$$

In particular, the gadget has $A_W(1) = 200$ states.

4. The gadget-amplification framework

This section formally develops the gadget-amplification framework and proves the first bound below 4^k. Starting from a complete local gadget of capacity r, we take the product of many copies and use a family of hashes to assign every input symbol to one copy and one local color. A hash succeeds on an input when each copy receives at most r symbols and its resulting local projection is accepted by the gadget. In this section, we use full-state accounting: for each hash function, we retain the first k layers of the raw product.

The argument has six steps.

1. A hash sends each input symbol to one of N copies of the gadget and to one local color.

2. A successful hash gives each copy a repetition-free local color set whose size does not exceed the gadget capacity.

3. We prescribe how many copies have each possible load. This prescribed histogram describes one subset of all successful hashes.

4. We count that subset using a multinomial coefficient and local-set counts.

5. We use enough hashes to cover every set of at most k input symbols.

6. We multiply the number of hashes by the size of one product automaton.

4.1. One product automaton for one hash function

Fix a complete gadget G of capacity r over a color set C of size c. Let $s(G)$ be its number of states. Fix a positive integer N and a function

$$h : [n] \longrightarrow [N] \times C \ .$$

Write

$$h(x) = (b_h(x), \gamma_h(x)) \ .$$

The value $b_h(x)$ chooses one of the N copies of G, and $\gamma_h(x)$ is the color read by that copy. When the product automaton reads x, only copy $b_h(x)$ moves. Every other copy remains in its current state.

Lemma 8 (One fixed map)**.** *Let d_G be the maximum, over a local state and a local color, of the number of possible successor states. For every $h : [n] \to [N] \times C$ and every cutoff k, there is a layered NFA A_h over $[n]$ with at most $s(G)^N$ states and at most*

$$d_G n s(G)^N$$

ordinary transitions. For every word of length at most k, A_h accepts exactly when the symbol sequence received by every local copy corresponds to a path in that copy of G.

Proof. A product state is an N-tuple of local states. Its rank is the sum of the local ranks, which tracks the number of symbols processed globally. We retain only tuples of rank at most k, and all states are accepting.

Suppose the current tuple is $(q_1, \ldots, q_N)$ and the next input symbol is x. Let $h(x) = (i, a)$. The product may replace q_i by any local successor of q_i under color a; all other coordinates are unchanged. This is exactly the raw product described above.

There are at most $s(G)^N$ tuples. For a fixed tuple and a fixed input symbol x, only one coordinate moves, and it has at most d_G possible successors. Thus there are at most $d_G n s(G)^N$ transitions. Every transition increases the total rank by one, so the automaton is layered. The acceptance statement is immediate from the coordinatewise construction. □

Two simple facts will be used repeatedly. First, if an original symbol x occurs twice, both occurrences are sent to the same pair $(b_h(x), \gamma_h(x))$. The same local copy therefore sees the same color twice. Since every path in a

complete gadget is repetition-free, the product rejects. Second, for a set S of distinct original symbols, we say that h *succeeds on* S if every copy receives a repetition-free local color set of size at most r; in that case the product accepts every ordering of S.

Write

$$B(z) \triangleq B_G(z) = \sum_{j=0}^{r} B_j z^j \ ,$$

so $B_j \triangleq [z^j]B(z)$ is the coefficient of z^j, namely the number of local color sets of size j. Fix a k-element set $S \subseteq [n]$ and choose h uniformly from all functions $[n] \to [N] \times C$.

Lemma 9 (Exact success probability)**.** *The probability that h succeeds on S is*

$$p_{k,N} = \frac{k!}{(cN)^k}[z^k]B(z)^N \ . \tag{2}$$

Proof. We count successful restrictions of h to the labeled elements of S.

For each copy, choose one local color set. If copy i receives a set of size j, that choice contributes $B_j z^j$. Therefore $[z^k]B(z)^N$ counts the choices of one set in each copy whose total size is exactly k.

Such a choice specifies exactly k distinct copy-color cells. The k labeled elements of S can be assigned bijectively to those cells in $k!$ ways. Hence the number of successful hashes is $k![z^k]B(z)^N$. The total number of hashes restricted to S is $(cN)^k$, which gives (2). □

Next, we analyze the required number of hashes.

Lemma 10 (A family covering all sets up to size k)**.** *For $p_{k,N} > 0$, there is a family $\mathcal{H}$ of*

$$O\Big(p_{k,N}^{-1} k \log(2n)\Big)$$

functions from $[n]$ to $[N] \times C$ such that every subset of $[n]$ of size at most k succeeds for at least one function in $\mathcal{H}$.

Proof. It suffices to cover every k-set: every smaller set can be extended to a k-set, and success is preserved when elements are deleted. Choose T independent random functions. A fixed k-set is missed by all of them with probability at most

$$(1 - p_{k,N})^T \leq \exp(-p_{k,N}T) \ .$$

There are $\binom{n}{k} \le n^k$ sets of size k. Therefore the expected number of uncovered k-sets is at most

$$n^k \exp(-p_{k,N} T) \ .$$

This quantity is smaller than one once

$$T > p_{k,N}^{-1}(k \log n + 1) \ .$$

Thus a family covering every k-set exists. □

Taking the nondeterministic union of the automata A_h, one for each $h \in \mathcal{H}$, recognizes $L_{k,n}$. Concretely, let q_h be the initial state of A_h. Take disjoint copies of the A_h, delete each q_h, and introduce a new initial state q_0. For every transition $q_h \xrightarrow{a} v$ in A_h, add $q_0 \xrightarrow{a} v$ with the same label a. Retain every transition whose endpoints were not deleted. All states are accepting, including q_0. Thus every transition still reads exactly one input symbol, and no ε-transitions are introduced. It remains to lower-bound the success probability using a chosen load histogram and combine the resulting hash-family bound with Lemma 8.

4.2. Load histograms and finite templates

A *load histogram* records how many copies receive each possible number of colors. We use K for the auxiliary length used in the analysis; later we choose $K = k + O(1)$. If N copies jointly receive K colors, let η_j denote the number of copies that receive exactly j colors. Necessarily

$$\sum_{j=0}^{r} \eta_j = N, \qquad \sum_{j=0}^{r} j\eta_j = K \ . \tag{3}$$

Every nonnegative integer vector satisfying (3) is a possible load histogram.

Expanding $B(z)^N$ by histograms gives the exact identity

$$[z^K]B(z)^N = \sum_{\substack{\eta_0,\dots,\eta_r \ge 0 \\ \sum_j \eta_j = N \\ \sum_j j\eta_j = K}} \binom{N}{\eta_0, \dots, \eta_r} \prod_{j=0}^{r} B_j^{\eta_j} \ . \tag{4}$$

The multinomial coefficient chooses which copies have each load. After those loads are fixed, the factor $B_j^{\eta_j}$ chooses one j-element local color set for every copy of load j.

Every summand in (4) is nonnegative. We may therefore obtain a lower bound by choosing any one feasible histogram and discarding all other successful assignments.

To specify such a histogram as the input length grows, we choose nonnegative constant integers

$$m_0, m_1, \ldots, m_r \ ,$$

with $m_j > 0$ only when $B_j > 0$.

Think of this vector as one finite template. Define

$$D \triangleq \sum_{j=0}^{r} m_j, \qquad M \triangleq \sum_{j=0}^{r} j m_j > 0 \ .$$

Thus one copy of the template uses D gadget copies and carries M input symbols. For a positive integer t, repeat the template t times. Then

$$N = tD, \qquad K = tM, \qquad \eta_j = t m_j \ . \tag{5}$$

This is why we distinguish K from k: repeating the template yields the coefficient lower bound only for lengths K divisible by M. For an arbitrary target length k, in Section 4.4 we let K be the least multiple of M with $K \geq k$, temporarily add $K - k$ fresh symbols, construct the automaton for length K, then delete every transition labeled by a fresh symbol, and retain only layers $0, \ldots, k$. Since M is fixed, $0 \leq K - k < M$, and hence $K = k + O(1)$.

It is useful to normalize the template by setting

$$p_j \triangleq \frac{m_j}{D}, \qquad \lambda \triangleq \frac{D}{M} \ .$$

Write $\mathbf{p} \triangleq (p_0, \ldots, p_r)$.

Here p_j is the fraction of copies with load j, while λ is the number of gadget copies per input symbol. The relations in (5) imply

$$\sum_j p_j = 1, \qquad \sum_j j p_j = \frac{M}{D} = \frac{1}{\lambda}, \qquad tD = \lambda K \ . \tag{6}$$

The last identity converts quantities measured per gadget copy into quantities measured per input symbol.

symbol	meaning
m_j	copies of load j in one finite template
$D = \sum_j m_j$	total gadget copies in one template
$M = \sum_j j m_j$	total input symbols carried by one template
t	number of repetitions of the template
$N = tD$	total number of gadget copies in the product
$K = tM$	input length analyzed by the repeated template
$p_j = m_j/D$	fraction of copies whose load is j
$\lambda = D/M = N/K$	number of gadget copies per input symbol

4.3. Counting the chosen histogram, one factor at a time

Substitute the histogram $\eta_j = tm_j$ into one summand of (4). We obtain

$$[z^K]B(z)^N \geq \binom{tD}{tm_0, \ldots, tm_r} \prod_{j=0}^{r} B_j^{tm_j} \ . \tag{7}$$

The multinomial coefficient assigns the loads: for every j, it chooses which tm_j copies have load j. Once the loads are fixed, each of those copies has B_j choices for its local color set, giving the second factor $\prod_{j=0}^{r} B_j^{tm_j}$. We now lower-bound the two factors on the right-hand side of (7) separately.

The multinomial factor. Let $J \triangleq \{j : m_j > 0\}$. For a positive integer u, Stirling's formula in base-two logarithmic form is

$$\begin{aligned} \log_2 u! &\geq u \log_2 u - (\log_2 \mathrm{e})u \ . \\ \log_2 u! &\leq u \log_2 u - (\log_2 \mathrm{e})u + O(\log u) \ . \end{aligned}$$

Apply the lower bound to $(tD)!$ and the upper bound to every $(tm_j)!$ with $j \in J$. Then

$$\begin{aligned} \log_2 \binom{tD}{tm_0, \ldots, tm_r} &= \log_2(tD)! - \sum_{j \in J} \log_2(tm_j)! \\ &\geq tD \log_2(tD) - (\log_2 \mathrm{e})tD \\ &\quad - \sum_{j \in J} (tm_j \log_2(tm_j) - (\log_2 \mathrm{e})tm_j + O(\log(tm_j))) \\ &\geq tD \log_2(tD) - \sum_{j \in J} tm_j \log_2(tm_j) - O(\log t) \ . \end{aligned} \tag{8}$$

The last inequality uses $\sum_{j\in J} m_j = D$ to cancel the linear terms. Since J and all positive m_j are fixed, the sum of the denominator error terms is $O(\log t)$. Now use $m_j = Dp_j$:

$$\begin{aligned}\sum_{j\in J} tm_j \log_2(tm_j) &= tD \sum_{j\in J} p_j \log_2(tDp_j) \\ &= tD \log_2(tD) \sum_{j\in J} p_j + tD \sum_{j\in J} p_j \log_2 p_j \\ &= tD \log_2(tD) + tD \sum_{j\in J} p_j \log_2 p_j \ .\end{aligned}$$

Substituting the preceding identity into (8) gives

$$\log_2 \binom{tD}{tm_0, \ldots, tm_r} \geq tD \left(-\sum_{j\in J} p_j \log_2 p_j \right) - O(\log t) \ . \tag{9}$$

The local-set factor. Taking logarithms of the second factor in (7),

$$\begin{aligned}\log_2 \left(\prod_{j=0}^{r} B_j^{tm_j} \right) &= \sum_{j\in J} tm_j \log_2 B_j \\ &= tD \sum_{j\in J} p_j \log_2 B_j \ .\end{aligned} \tag{10}$$

Combining the two factors. Define

$$\Psi_B(\mathbf{p}) \triangleq -\sum_{j:p_j>0} p_j \log_2 p_j + \sum_{j:p_j>0} p_j \log_2 B_j \ . \tag{11}$$

The first sum is the per-copy logarithmic contribution from placing the loads among the copies. The second is the per-copy logarithmic contribution from choosing the local color sets after the loads are fixed.

Add (9) and (10). By the definition of Ψ_B,

$$\log_2 [z^K] B(z)^N \geq tD\Psi_B(\mathbf{p}) - O(\log t) \ . \tag{12}$$

Since M is fixed and $K = tM$, we have $t = \Theta(K)$, and hence $O(\log t) = o(K)$. Also, (6) gives $tD = \lambda K$. Therefore

$$\log_2 [z^K] B(z)^N \geq \lambda K \Psi_B(\mathbf{p}) - o(K) \ . \tag{13}$$

4.4. From the coefficient to the final NFA size

We now translate the coefficient lower bound into the number of hashes and then combine it with the product-state cost. Recall that

$$B(z) = \sum_{j=0}^{r} B_j z^j \qquad , \qquad B_j = [z^j]B(z) \ .$$

Thus B is the generating polynomial for local color sets, while B_j is the number of sets of the particular size j.

Theorem 11 (Amplification from a fixed load histogram)**.** *For the choices above, there is an acyclic NFA recognizing $L_{k,n}$ of size*

$$2^{(E_{\text{fix}}+o(1))k} n^{O(1)} \ ,$$

where

$$E_{\text{fix}} \triangleq \lambda \log_2 s(G) + \max\left\{0, \log_2(c\lambda \mathrm{e}) - \lambda \Psi_B(\mathbf{p})\right\} \ . \tag{14}$$

Proof. We first prove the claim for lengths $K = tM$, using $N = tD = \lambda K$ copies.

Step 1: the cost of one product automaton. By Lemma 8, one hash uses at most $s(G)^N$ states, and its transition count is at most $d_G n$ times larger. Therefore the total size of one product automaton, counting both states and transitions, is at most

$$(1 + d_G n)s(G)^N = (1 + d_G n)2^{\lambda K \log_2 s(G)} \le 2^{\lambda K \log_2 s(G)} n^{O(1)} \ . \tag{15}$$

Step 2: the number of hashes. By Lemma 9,

$$p_{K,N}^{-1} = \frac{(cN)^K}{K!\,[z^K]B(z)^N} \ . \tag{16}$$

We first estimate the factorial ratio $(cN)^K/K!$. Since $N = \lambda K$, Stirling's formula gives

$$\begin{aligned} \log_2 \frac{(cN)^K}{K!} &= K \log_2(c\lambda K) - \log_2 K! \\ &= K \log_2(c\lambda K) - K \log_2 K + (\log_2 \mathrm{e})K + O(\log K) \\ &= K \log_2(c\lambda \mathrm{e}) + O(\log K) \ . \end{aligned} \tag{17}$$

Subtracting the coefficient lower bound (13) from (17) yields

$$\log_2 p_{K,N}^{-1} \le K\left(\log_2(c\lambda \mathrm{e}) - \lambda \Psi_B(\mathbf{p})\right) + o(K) \ .$$

By Lemma 10 and the preceding inequality, there is a covering family $\mathcal{H}$ with

$$\begin{aligned} |\mathcal{H}| &= O\Big(p_{K,N}^{-1} K \log(2n)\Big) \\ &\le 2^{(\max\{0,\log_2(c\lambda \mathrm{e}) - \lambda\Psi_B(\mathbf{p})\} + o(1))K} n^{O(1)} \ . \end{aligned} \tag{18}$$

The multiplicative factor $K \log(2n)$ is folded into the $n^{O(1)}$ term.

Step 3: combine the branches. As explained, the final NFA is the nondeterministic union of one product automaton per hash. Its size is therefore bounded by

$$\text{(number of hashes)} \times \text{(size of one product automaton)} \ ,$$

up to polynomial factors. Multiplication adds the two exponents in (15) and (18). Their sum is exactly E_{fix} in (14). This proves the theorem for $K = tM$.

Step 4: remove the divisibility assumption. For an arbitrary target length k, let K be the least multiple of M with $K \ge k$. Add $K - k$ fresh symbols temporarily and write

$$n_{\text{pad}} \triangleq n + K - k$$

for the padded alphabet size. Construct the automaton for $L_{K,n_{\text{pad}}}$. Then delete every transition labeled by a fresh symbol and retain only layers $0, \dots, k$. The resulting automaton recognizes $L_{k,n}$. Since M is fixed, $K = k + O(1)$, and since $n \ge k$, we also have $n_{\text{pad}} = O(n)$. The asymptotic bound is unchanged. □

4.5. The Witt gadget under the fixed-histogram analysis

To obtain a concrete NFA size bound, we apply Theorem 11 to the Witt gadget using the load template in Table 2. Appendix A explains how this template was selected; that optimization is not needed to verify the bound.

Table 2: The repeated load template for the Witt gadget. One template uses $D = 1000$ copies. Repeating it t times gives tm_j copies of each displayed load.

load j	0	1	2	3	4	5	6
local sets $B_j = \binom{11}{j}$	1	11	55	165	330	462	462
copies in one template m_j	0	0	5	28	107	292	568

The proof above uses the single-histogram bound from Section 4.3. Because the Witt gadget has only seven possible loads, one can also evaluate the full sum in (4); this gives the slightly better bound $O^*(3.9661^k)$. We omit the calculation because the later sections give stronger bounds.

Corollary 12 (Fixed-histogram Witt bound)**.** *There is an acyclic NFA recognizing $L_{k,n}$ of size*

$$3.967^k n^{O(1)} .$$

Proof. The total number of copies in one template is

$$D = 5 + 28 + 107 + 292 + 568 = 1000 .$$

The number of input symbols carried by one template is

$$\begin{aligned} M &= 2 \cdot 5 + 3 \cdot 28 + 4 \cdot 107 + 5 \cdot 292 + 6 \cdot 568 \\ &= 5390 . \end{aligned}$$

Therefore

$$\lambda = \frac{D}{M} = \frac{100}{539} .$$

Starting from (11) and substituting $p_j = m_j/D$, we obtain

$$\begin{aligned} \Psi_{B_W}(\mathbf{p}) &= -\sum_{j:m_j>0} \frac{m_j}{D} \log_2 \frac{m_j}{D} + \sum_{j:m_j>0} \frac{m_j}{D} \log_2 B_j \\ &= \frac{1}{D} \sum_{j:m_j>0} m_j \left(\log_2 D + \log_2 \frac{B_j}{m_j} \right) \\ &= \log_2 D + \frac{1}{D} \sum_{j:m_j>0} m_j \log_2 \frac{B_j}{m_j} , \end{aligned}$$

where the last equality uses $\sum_j m_j = D$. Substituting the five columns of Table 2 gives

$$\Psi_{B_W}(\mathbf{p}) > 10.25261 .$$

Define the exponent contributed by one product automaton by

$$E_{\text{branch}} \triangleq \lambda \log_2 200 .$$

By (15), one product automaton has size at most $2^{E_{\text{branch}} K} n^{O(1)}$. Using $\lambda = 100/539$ gives

$$E_{\text{branch}} < 1.41816 .$$

Define the exponent contributed by the covering family by

$$E_{\text{hash}} \triangleq \max\left\{0, \log_2(11\lambda \mathrm{e}) - \lambda \Psi_{B_W}(\mathbf{p})\right\} \ .$$

By (18), the number of hash branches is at most $2^{(E_{\text{hash}}+o(1))K} n^{O(1)}$. The numerical bounds

$$\log_2(11\lambda \mathrm{e}) < 2.47185, \qquad \lambda \Psi_{B_W}(\mathbf{p}) > 1.90215$$

therefore imply

$$E_{\text{hash}} < \max\{0, 2.47185 - 1.90215\} = 0.56970 \ .$$

Finally, (14) states that $E_{\text{fix}} = E_{\text{branch}} + E_{\text{hash}}$. Hence

$$E_{\text{fix}} < 1.41816 + 0.56970 = 1.98786, \qquad 2^{E_{\text{fix}}} < 3.967 \ . \tag{19}$$

□

This completes the fixed-histogram analysis. It gives an $O^*(3.967^k)$-size NFA, strictly improving on the previous $O^*(4^{k+o(k)})$ bound. Its weakness is equally clear: (15) pays for all 200^N product states, including the very large middle layers. The next section introduces compose-and-compress, whose purpose is to remove those layers while preserving correctness.

5. Compose-and-compress

The fixed-histogram proof counts every state of a raw product. Its middle layers are the expensive ones. Compose-and-compress (CaC) removes a band of product layers centered at the middle and replaces each path across that band by one ordinary one-symbol transition. The colors read on the omitted part of the path are hidden witnesses: they justify the shortcut but are not read by the compressed automaton.

We use the operation twice. First, the number of gadget copies grows with an intermediate block length; this creates the intermediate-block NFA used in the global construction. Second, exactly eleven Witt complete gadgets are compressed to make a better finite partial gadget. The growing construction is then applied again.

5.1. The complete two-color example

Before formalizing the procedure, let us work with an illustrative example. The complete gadget in Figure 1 is the $(2, 1, 1)$ separator gadget. We now apply compose-and-compress to its two-copy raw product from Figure 2.

Delete raw rank two and shift raw ranks three and four down by one. The six middle states disappear, so the compressed state polynomial is

$$1 + 4z + 4z^2 + z^3 \ .$$

A new transition crosses the gap whenever the raw product has a two-step path across the deleted rank. Either raw label may remain visible; the other is the hidden witness. The complete ten-state compressed NFA appears in Figure 5. A braced label denotes two ordinary transitions, one for each displayed symbol.

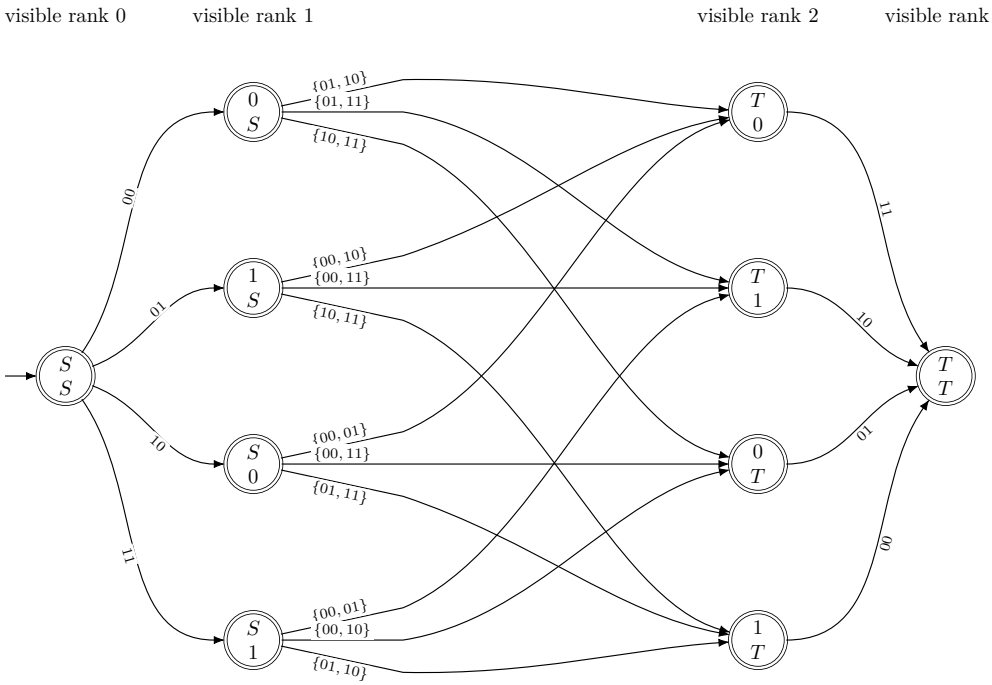


Figure 5: The complete compressed NFA after composing two $(2, 1, 1)$ gadgets. Every state appears once and every transition label is printed on its arrow.

For example, the two transitions from $(0, S)$ to $(T, 0)$ are justified by the raw paths

$$(0, S) \xrightarrow{01} (T, S) \xrightarrow{10} (T, 0) \qquad \text{and} \qquad (0, S) \xrightarrow{10} (0, 0) \xrightarrow{01} (T, 0) \ .$$

Thus one compressed transition is labeled 01 and the other is labeled 10. There are twelve braced central arcs, hence 24 central transitions, represented by twelve braced arcs, and eight inherited (raw) transitions.

The compressed NFA recognizes $L_{3,\{00,01,10,11\}}$. Soundness follows by expanding every shortcut to a raw path. A repetition-free word of length two or three can be extended by one unused color; place that color in the hidden position of a raw path and compress the resulting two-step segment. Words of length zero or one stay in the retained lower ranks.

5.2. Partial gadgets that can be composed again

As Figure 3b illustrates, a raw product of complete gadgets need not be complete over the union of their palettes. The preceding example remains a complete gadget because each two-color factor has capacity two: its capacity equals the size of its palette, so no distinct set can overload a factor. If a factor has capacity $r < |C|$ and the composed gadget has target capacity $q > r$, then a set of $r+1$ distinct colors may lie entirely in one factor's palette, and that factor cannot read it.

To allow the operation to be iterated, we therefore record only the sets that are guaranteed to work.

Definition 13 (Partial gadget)**.** A *partial gadget of capacity r over C* is a layered NFA H with layers $Q_0, \ldots, Q_r$, a unique initial state, all states accepting, and a specified family $\mathcal{U}_H \subseteq 2^C$ of *certified* sets. The NFA H and the family $\mathcal{U}_H$ must satisfy:

(P1) $\mathcal{U}_H$ is nonempty and downward closed, and every member has size at most r;

(P2) every word accepted by H is repetition-free;

(P3) for every $S \in \mathcal{U}_H$, every ordering of S is accepted by H;

(P4) if $S \in \mathcal{U}_H$ and $0 \le h \le r - |S|$, some $T \in \mathcal{U}_H$ satisfies $S \subseteq T$ and $|T| = |S| + h$.

Its certified-set polynomial is

$$B_H(z) \triangleq \sum_{j=0}^{r} |\{S \in \mathcal{U}_H : |S| = j\}| z^j \ .$$

The state polynomial and symmetry are exactly as for complete gadgets.

Every complete gadget is a partial gadget: take all subsets of C of size at most r as the certified sets. The extra family is needed only after composition; it records what the next use of the construction may safely count. For example, in the partial gadget of Figure 3b, let $C_1 = \{00, 01\}$ and $C_2 = \{10, 11\}$ be the two palettes. Its family of certified sets is $\mathcal{U}_H \triangleq \{S \subseteq C_1 \cup C_2 : |S \cap C_1| \leq 1 \text{ and } |S \cap C_2| \leq 1\}$. Thus $\{00, 10\}$ is certified, whereas $\{00, 01\}$ is not: both colors in the latter set would have to be read by the capacity-one first factor.

Lemma 14 (Product of partial gadgets)**.** *Take m copies of a partial gadget H on pairwise disjoint color sets. Their raw product is a partial gadget of capacity $R = mr$. Writing H_i for the ith copy, its certified-set family is $\mathcal{U}_{H^{\times m}} \triangleq \{\bigcup_{i=1}^{m} S_i : S_i \in \mathcal{U}_{H_i} \text{ for every } i \in [m]\}$, and the two polynomials are*

$$A_H(z)^m \qquad \text{and} \qquad B_H(z)^m \ .$$

Proof. Let C_i be the palette of H_i. Because the palettes are disjoint, every $S \subseteq \bigcup_i C_i$ decomposes uniquely as $S = \bigcup_i S_i$, where $S_i \triangleq S \cap C_i$. Because each $\mathcal{U}_{H_i}$ is nonempty, choosing one $S_i \in \mathcal{U}_{H_i}$ for every i shows that $\mathcal{U}_{H^{\times m}}$ is nonempty. If S is certified and $T \subseteq S$, then $T \cap C_i \in \mathcal{U}_{H_i}$ by downward closure, so T is certified; also $|S| = \sum_i |S_i| \leq mr$. Thus (P1) holds. Any path in the product projects to a path in each H_i, so a repeated color would contradict (P2) in the corresponding copy. Conversely, for a certified S and any ordering w of S, each projection $w|_{C_i}$ labels a path in H_i by (P3); interleaving these paths according to w gives a product path labeled w. For (P4), given $0 \leq h \leq mr - |S|$, choose integers $0 \leq h_i \leq r - |S_i|$ with $\sum_i h_i = h$. Extend each S_i by h_i colors using (P4) in H_i; the union is a certified extension of S of size $|S| + h$.

A product state is an m-tuple of local states. If their ranks are $j_1, \ldots, j_m$, then the product state has rank $j_1 + \cdots + j_m$ and contributes $z^{j_1 + \cdots + j_m}$ to the state polynomial. Multiplying the m copies of $A_H(z)$ counts exactly these tuples, so the state polynomial is $A_H(z)^m$. Likewise, multiplying $B_H(z)$ chooses one certified set S_i in each copy and records their total size in the exponent. Because the palettes are disjoint, $(S_1, \ldots, S_m) \mapsto \bigcup_i S_i$ is a bijection, so the certified-set polynomial is $B_H(z)^m$. □

5.3. The compression operation

Let G be a symmetric partial gadget of capacity r with polynomials $A(z), B(z)$. Take m copies on disjoint color sets and write

$$A(z)^m = \sum_{j=0}^{R} a_j z^j, \qquad B(z)^m = \sum_{j=0}^{R} b_j z^j, \qquad R \triangleq mr \ .$$

The raw product has capacity R. We construct from it a gadget of a smaller target capacity q, where $1 \le q < R$. To reduce the capacity by $R - q$, we delete that many consecutive raw layers and replace every path crossing the resulting gap by a one-symbol shortcut. Define

$$s \triangleq R - q, \qquad \ell \triangleq \left\lfloor \frac{q-1}{2} \right\rfloor \ .$$

Here s is the number of raw layers removed, and ℓ is the index of the last retained layer below the gap.

Specifically, we delete the states of raw layers

$$\ell + 1, \ell + 2, \ldots, \ell + s \ .$$

Raw layers $0, \ldots, \ell$ keep their ranks. Each raw layer $j \in \{\ell + s + 1, \ldots, R\}$ is retained at compressed rank $j - s$. The resulting ranks are therefore $0, \ldots, q$.

In the example of Figure 5, the starting gadget has $c = 2$ colors and capacity $r = 2$, and we take $m = 2$ copies. Hence the raw product has capacity $R = mr = 4$. We choose target capacity $q = 3$, so $s = R - q = 1$ and $\ell = \lfloor (q-1)/2 \rfloor = 1$. Thus raw layer 2 is deleted: raw layers $0, 1$ remain at compressed ranks $0, 1$, while raw layers $3, 4$ shift down to compressed ranks $2, 3$. Each new shortcut goes from raw layer 1 to raw layer 3 and replaces a two-step raw path labeled $h_1 a$; the color h_1 is hidden and a labels the shortcut. For example, the path $(0, S) \xrightarrow{01} (T, S) \xrightarrow{10} (T, 0)$ has $h_1 = 01$ and $a = 10$, and therefore yields the compressed transition $(0, S) \xrightarrow{10} (T, 0)$. The other raw path, $(0, S) \xrightarrow{10} (0, 0) \xrightarrow{01} (T, 0)$, has $h_1 = 10$ and $a = 01$, and therefore yields the parallel compressed transition labeled 01.

Generally, we keep every raw transition whose endpoints both lie in the retained lower layers or both lie in the retained upper layers, with the same label; only the layer numbers of upper states change under the shift by s. These retained raw transitions do not cross the deleted band of layers. To

replace paths that do cross it, add a transition $u \xrightarrow{a} v$ from raw layer ℓ to raw layer $\ell + s + 1$ whenever the raw product has a path from u to v labeled

$$h_1 h_2 \cdots h_s a$$

for distinct colors $h_1, \ldots, h_s, a$. Only a is read by the compressed automaton.

The proof of the following is given in Appendix E.

Lemma 15 (Compose-and-compress)**.** *The compressed automaton is a symmetric partial gadget of capacity q. Its state polynomial is*

$$A_{\mathrm{comp}}(z) = \sum_{j=0}^{\ell} a_j z^j + \sum_{j=\ell+s+1}^{R} a_j z^{j-s} \ , \tag{20}$$

and its certified-set polynomial is

$$B_{\mathrm{comp}}(z) = \sum_{j=0}^{q} b_j z^j \ . \tag{21}$$

The next section applies this lemma with a growing number of Witt gadgets. The section after that applies it once to eleven Witt complete gadgets and then feeds the resulting partial gadget back into the growing construction.

6. First use: a growing compressed product

To obtain an improved NFA construction from CaC, we first build an automaton for an intermediate length q. The reason is the transition count: applying CaC directly at length k gives the desired exponential constant for the number of states, but leaves a transition bound with a factor $2^{O(k)}$ whose hidden constant is too large.

Instead, we choose $q \to \infty$ with $q = o(k)$, construct an inner NFA for length q, and combine about k/q inner copies in an outer product. Each outer transition updates only one inner copy, so the local factor $2^{O(q)}$ occurs only once and is $2^{o(k)}$. The overhead from the outer hash family is also subexponential in k.

The construction therefore has two scales. At the inner scale, we take a growing number of copies of one fixed partial gadget and apply compose-and-compress to their raw product, obtaining an NFA for repetition-free words of

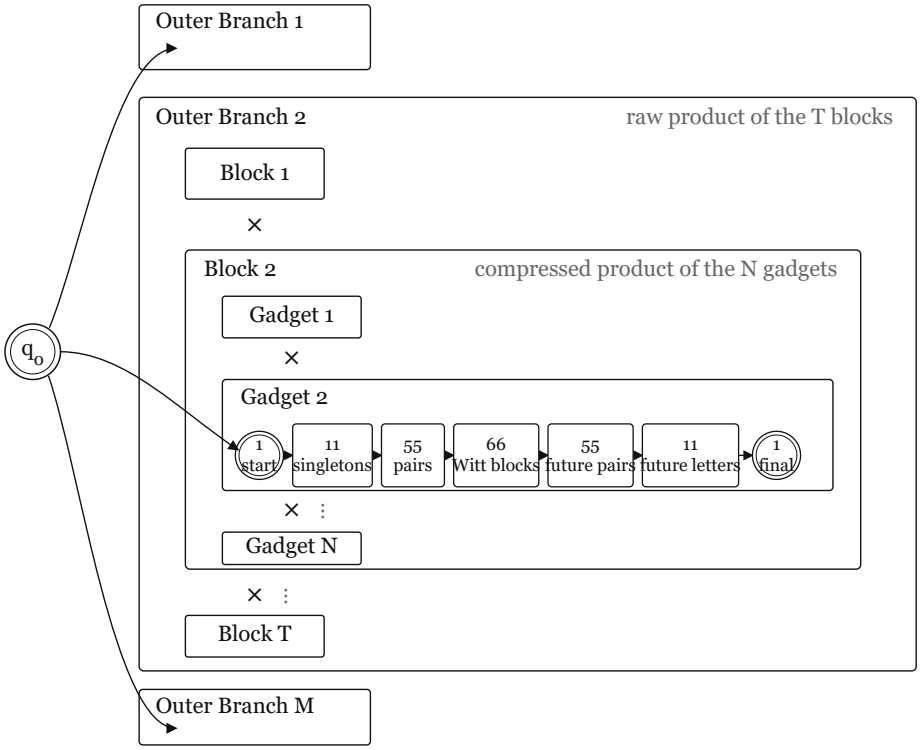


Figure 6: The hierarchy of the growing construction. The common accepting initial state has branches corresponding to the outer hash functions. Each outer branch contains T intermediate blocks, and each intermediate block contains N copies of the fixed finite gadget.

length at most q. At the outer scale, the original k letters are divided among blocks of size q, and the inner NFAs are combined by an outer hashing step to recognize $L_{k,n}$. Figure 6 illustrates these two scales: each outer branch contains T intermediate blocks, and each intermediate block is built from N copies of the fixed finite gadget.

Throughout this section, let G be a symmetric partial gadget of capacity $r \geq 1$ over a c-element color set, and write

$$A(z) = A_G(z), \qquad B(z) = B_G(z) = \sum_{j=0}^{r} B_j z^j \ .$$

Here is the optimization problem before we introduce its parameters. A branch formed from N copies has one state for each retained product state,

so the two retained tails of $A(z)^N$ determine its cost. For a fixed repetition-free q-element input set, the probability that the branch certifies the set is proportional to $[z^q]B(z)^N$, the number of certified product sets of size q. A larger coefficient reduces the number of required hash branches. Thus we must balance the states per branch against the number of branches. We use the auxiliary variables x and y to control these two costs.

First, we choose a rational number $x \in (0,1)$, and put $a_j \triangleq [z^j]A(z)^N$ and $Y \triangleq \lfloor q/2 \rfloor$. The raw product has capacity Nr. Central compression to capacity q retains the raw ranks $0, \dots, \lceil q/2 \rceil - 1$ and $Nr - q + \lceil q/2 \rceil, \dots, Nr$. Because G is symmetric, $A(z)^N$ is symmetric, and therefore

$$\sum_{j=Nr-q+\lceil q/2 \rceil}^{Nr} a_j = \sum_{j=0}^{q-\lceil q/2 \rceil} a_j = \sum_{j=0}^{Y} a_j \ ,$$

while

$$\sum_{j=0}^{\lceil q/2 \rceil - 1} a_j \le \sum_{j=0}^{Y} a_j \ .$$

Thus each retained tail contains at most $\sum_{j=0}^{Y} a_j$ states. When q is odd the two tails have the same number of layers; when q is even the upper tail has one more layer and contains the central layer.

The coefficients a_j are nonnegative, and $x^j \ge x^Y$ for every $0 \le j \le Y$. Hence

$$x^Y \sum_{j=0}^{Y} a_j \le \sum_{j=0}^{Y} a_j x^j \le A(x)^N \ ,$$

and therefore

$$\sum_{j=0}^{Y} [z^j]A(z)^N \le x^{-Y} A(x)^N \ .$$

Second, a positive rational number y selects the load histogram used to lower-bound $[z^q]B(z)^N$. The next subsection defines that histogram and proves the bound.

The final construction multiplies the number of states per branch by the number of branches, so their base-two logarithms add. We later choose x and y jointly to minimize that sum.

6.1. A load distribution chosen by one parameter

The fixed-histogram analysis chose an integral histogram by hand. We now need the same counting idea for a number of copies that grows with

an intermediate length q. It is convenient to describe the histogram by a probability vector rather than by a fixed denominator.

Choose a positive rational number y. Define

$$p_j \triangleq \frac{B_j y^j}{B(y)}, \qquad \mu \triangleq \sum_{j=0}^{r} j p_j = \frac{y B'(y)}{B(y)}, \qquad \lambda \triangleq \frac{1}{\mu} \ . \tag{22}$$

For load j, the factor B_j counts the certified local sets of size j, while y^j controls how strongly that load is favored. Dividing the weights $B_j y^j$ by their sum $B(y)$ produces the probability vector $(p_0, \ldots, p_r)$. These definitions have the following concrete meanings.

- Since $B(y) = \sum_j B_j y^j$, the values p_j are nonnegative and sum to one.
- We intend a fraction p_j of the partial-gadget copies to have load j.
- The average number of colors per copy is μ.
- The reciprocal $\lambda = 1/\mu$ is the intended number of copies per visible input color.

Suppose we take N copies. We want exactly Np_j copies to have load j. The total number of visible colors is then

$$\sum_j j N p_j = N\mu \ .$$

We therefore write

$$q = N\mu, \qquad N = \lambda q \ .$$

All numbers in (22) are rational. Hence there is a fixed positive integer M such that, whenever N is a multiple of M, every Np_j and $q = N\mu$ is an integer. Writing $N = tM$ gives $q = t(M\mu)$, and $M\mu$ is a positive integer by the choice of M. We call these values of q *allowed*. This is only a divisibility condition. Because the allowed values form an arithmetic progression with fixed step $M\mu$, the least allowed $q \geq \sqrt{k}$ satisfies $q = \sqrt{k} + O(1)$; in particular, $q = o(k)$, which is the property used in the final bounds.

The choice $p_j \propto B_j y^j$ is useful because it makes the entropy calculation collapse to a two-term expression in the following lemma.

Lemma 16 (One explicit coefficient type)**.** *For allowed values of N and $q = N\mu$,*

$$[z^q]B(z)^N \geq 2^{N(\log_2 B(y) - \mu \log_2 y) - O(\log N)} .$$

The constant in the error term depends only on the fixed polynomial B.

Proof. We repeat the fixed-histogram calculation from Section 4, now with Np_j copies of load j.

Step 1: keep one histogram. Write $B(z)^N = \prod_{i=1}^{N} B(z)$, with the ith occurrence of $B(z)$ representing gadget copy i. Choosing $B_j z^j$ from that occurrence means that copy i has load j. We retain the terms in which exactly Np_j copies have load j, for every j. Since

$$\sum_j Np_j = N, \qquad \sum_j jNp_j = N\mu = q ,$$

this is a feasible histogram for the coefficient of z^q. It contributes

$$[z^q]B(z)^N \geq \binom{N}{Np_0, Np_1, \ldots, Np_r} \prod_{j=0}^{r} B_j^{Np_j} . \tag{23}$$

Step 2: count where the loads occur. The multinomial coefficient chooses which copies have load 0, which have load 1, and so on. Applying Stirling's formula to its constantly many factorials gives

$$\log_2 \binom{N}{Np_0, \ldots, Np_r} = N \left(-\sum_{j=0}^{r} p_j \log_2 p_j \right) + O(\log N) .$$

The derivation is identical to (9), with tD replaced by N.

Step 3: count the local certified sets. Each of the Np_j copies of load j has B_j choices for its certified local set. Therefore

$$\log_2 \left(\prod_{j=0}^{r} B_j^{Np_j} \right) = N \sum_{j=0}^{r} p_j \log_2 B_j .$$

Adding Steps 2 and 3, the logarithm of the right-hand side of (23) is

$$N \left(-\sum_j p_j \log_2 p_j + \sum_j p_j \log_2 B_j \right) + O(\log N) .$$

Step 4: use the special form of p_j. By (22),

$$\log_2 p_j = \log_2 B_j + j \log_2 y - \log_2 B(y) \ .$$

Substitute this identity into the preceding entropy expression:

$$\begin{aligned} &- \sum_j p_j \log_2 p_j + \sum_j p_j \log_2 B_j \\ &\quad = - \sum_j p_j \Big(\log_2 B_j + j \log_2 y - \log_2 B(y) \Big) + \sum_j p_j \log_2 B_j \\ &\quad = \log_2 B(y) \sum_j p_j - \log_2 y \sum_j j p_j \\ &\quad = \log_2 B(y) - \mu \log_2 y \ . \end{aligned}$$

Here we used $\sum_j p_j = 1$ and $\sum_j j p_j = \mu$. Substituting this value into (23) proves the lemma. □

6.2. Building an NFA for one intermediate block

We next use $N = \lambda q$ copies of the partial gadget to recognize all repetition-free words of length at most q. Two costs must be controlled:

(a) the number of states retained after compose-and-compress; and

(b) the number of hash functions needed to cover all q-element input sets.

The rational parameter $x \in (0, 1)$ controls the first cost, while y controls the load distribution and hence the second cost. Define

$$R(x, y) \triangleq \lambda \log_2 A(x) - \frac{1}{2} \log_2 x = \frac{B(y)}{y B'(y)} \log_2 A(x) - \frac{1}{2} \log_2 x \ , \tag{24}$$

$$H(y) \triangleq \max \left\{ 0, \log_2(c \lambda \mathrm{e}) + \log_2 y - \lambda \log_2 B(y) \right\} \ , \tag{25}$$

and

$$E(x, y) \triangleq R(x, y) + H(y) \ . \tag{26}$$

The meaning of these quantities will emerge from the proof: one compressed branch has at most $2^{(R(x,y)+o(1))q}$ states, and at most $2^{(H(y)+o(1))q}$ hash branches are needed. Equivalently, $R(x, y)$ and $H(y)$ are the two base-two exponents per visible input color.

Proposition 17 (Inner NFA)**.** *For every sufficiently large allowed value $q = N\mu$ and every alphabet size $L \geq q$, there is an acyclic NFA $C_{q,L}$ recognizing $L_{q,L}$ with*

$$|Q(C_{q,L})| \leq 2^{(E(x,y)+o(1))q} L^{O(1)} \tag{27}$$

and

$$|\Delta(C_{q,L})| \leq 2^{O(q)} L^{O(1)} \ . \tag{28}$$

Proof. The construction has five steps.

Step 1: compose N partial-gadget copies and remove the middle. Take $N = \lambda q$ copies of G on disjoint color sets. Their raw capacity is

$$R_{\mathrm{raw}} \triangleq rN \ .$$

By Lemma 14, the product certified-set polynomial is $B(z)^N$. Properties (P1) and (P4) imply $B_0, B_r > 0$, so the tilted distribution in (22) has $0 < \mu < r$. Hence $q = N\mu < Nr = R_{\mathrm{raw}}$, and Lemma 15 applies with target capacity q. Thus the deleted width and the last retained lower rank are

$$s \triangleq R_{\mathrm{raw}} - q, \qquad \ell \triangleq \left\lfloor \frac{q-1}{2} \right\rfloor \ .$$

By Lemma 15, the compressed product is a partial gadget of capacity q. Its certified sets of size q are exactly those counted by $[z^q]B(z)^N$.

Step 2: count the retained states. Write

$$A(z)^N = \sum_{j=0}^{R_{\mathrm{raw}}} a_j z^j \ .$$

The coefficient a_j is the number of raw product states at rank j. Compose-and-compress retains the lower ranks $0, \ldots, \ell$ and the upper ranks $\ell+s+1, \ldots, R_{\mathrm{raw}}$. Hence the number of retained states is

$$\sum_{j=0}^{\ell} a_j + \sum_{j=\ell+s+1}^{R_{\mathrm{raw}}} a_j \ .$$

Because the partial gadget G is symmetric, $A(z)^N$ is symmetric, so $a_j = a_{R_{\mathrm{raw}}-j}$. Reflecting the upper tail around rank $R_{\mathrm{raw}}/2$ changes its index range to $0, \ldots, q-\ell-1$. Therefore

$$\sum_{j=0}^{\ell} a_j + \sum_{j=\ell+s+1}^{R_{\mathrm{raw}}} a_j = \sum_{j=0}^{\ell} a_j + \sum_{j=0}^{q-\ell-1} a_j \leq 2 \sum_{j=0}^{\lfloor q/2 \rfloor} a_j \ .$$

We now bound this lower tail by evaluating the polynomial at $x < 1$. For every $j \le \lfloor q/2 \rfloor$,

$$1 \le x^{j-\lfloor q/2 \rfloor} ,$$

so

$$\begin{aligned}\sum_{j=0}^{\lfloor q/2 \rfloor} a_j &\le x^{-\lfloor q/2 \rfloor} \sum_{j=0}^{\lfloor q/2 \rfloor} a_j x^j \\ &\le x^{-\lfloor q/2 \rfloor} A(x)^N .\end{aligned}$$

Let Q_{comp} denote the state set of the compressed product. Its number of states is at most

$$|Q_{\text{comp}}| \le 2x^{-\lfloor q/2 \rfloor} A(x)^N .$$

Taking base-two logarithms and using $N = \lambda q$ gives

$$\log_2 |Q_{\text{comp}}| \le q \left(\lambda \log_2 A(x) - \frac{1}{2} \log_2 x \right) + o(q) = (R(x,y) + o(1))q .$$

Thus, the number of states in one compressed product branch is at most

$$2^{(R(x,y)+o(1))q} .$$

Step 3: use one hash's success probability to bound the number of inner hash branches. Hash the input alphabet $[L]$ into the cN product colors. Fix a q-element input set $S \subseteq [L]$ and choose

$$h : [L] \longrightarrow [N] \times C$$

uniformly at random. The hash succeeds when its image is one of the certified q-element product sets. Let ρ denote the probability, over the random choice of h, that h succeeds on S. This probability determines how many hash branches the final nondeterministic union needs. The covering argument of Lemma 10 uses

$$O\Big(\rho^{-1} q \log(2L)\Big)$$

hashes to cover every q-element subset of $[L]$. Because $q \le L$, the factor $q \log(2L)$ is absorbed into $L^{O(1)}$. After separating this allowed polynomial dependence on L, it remains to bound ρ^{-1} as a function of q. Our goal in this step is therefore to prove

$$\rho^{-1} \le 2^{(H(y)+o(1))q} ,$$

or equivalently,

$$\frac{1}{q}\log_2 \rho^{-1} \le H(y) + o(1) \ .$$

There are $[z^q]B(z)^N$ choices for such a certified set. Once it is chosen, the q labeled elements of S may be bijected to its cells in $q!$ ways. There are $(cN)^q$ possible restrictions of h to S. Therefore

$$\rho = \frac{q!}{(cN)^q}[z^q]B(z)^N \ . \tag{29}$$

To prove this target bound, we estimate ρ^{-1}. Lemma 16 gives

$$\log_2[z^q]B(z)^N \ge N\Big(\log_2 B(y) - \mu \log_2 y\Big) - O(\log N) \ .$$

Since $N = \lambda q$ and $N\mu = q$, this becomes

$$\log_2[z^q]B(z)^N \ge q\Big(\lambda \log_2 B(y) - \log_2 y\Big) - o(q) \ . \tag{30}$$

Also, Stirling's formula and $cN = c\lambda q$ give

$$\log_2 \frac{(cN)^q}{q!} = q \log_2(c\lambda \mathrm{e}) + o(q) \ .$$

By (29),

$$\log_2 \rho^{-1} = \log_2 \frac{(cN)^q}{q!} - \log_2[z^q]B(z)^N \ .$$

Applying (30) gives

$$\begin{aligned}\log_2 \rho^{-1} &\le q \log_2(c\lambda \mathrm{e}) + o(q) \\ &\quad - \Big[q\Big(\lambda \log_2 B(y) - \log_2 y\Big) - o(q)\Big] \\ &= q\left(\log_2(c\lambda \mathrm{e}) + \log_2 y - \lambda \log_2 B(y)\right) + o(q) \ .\end{aligned}$$

Dividing by q and using $o(q)/q = o(1)$ gives

$$\frac{1}{q}\log_2 \rho^{-1} \le \log_2(c\lambda \mathrm{e}) + \log_2 y - \lambda \log_2 B(y) + o(1) \ .$$

Finally, (25) defines $H(y)$ as the maximum of zero and the three-term expression on the right. That expression is therefore at most $H(y)$. Hence

$$\frac{1}{q}\log_2 \rho^{-1} \le H(y) + o(1) \ .$$

Equivalently, $\rho^{-1} \le 2^{(H(y)+o(1))q}$. By Lemma 10, the number of required hash branches is therefore at most $2^{(H(y)+o(1))q}L^{O(1)}$, where the factor $q\log(2L)$ is absorbed into $L^{O(1)}$.

Step 4: cover every input set and add the exponents. Choose

$$O\Big(\rho^{-1} q \log(2L)\Big)$$

independent hashes. The same union-bound argument as in Lemma 10 shows that some such family covers every q-set. Since the certified family is downward closed, the same family covers every smaller set: extend it to a q-set, use a successful hash on the extension, and then restrict.

For each chosen hash $h : [L] \to [N] \times C$, take a separate copy of the compressed product. If the compressed product contains a transition $u \xrightarrow{(i,c)} v$, then in the copy associated with h add the transition $u \xrightarrow{a} v$ for every input symbol $a \in [L]$ satisfying $h(a) = (i, c)$. Thus reading a in this copy has exactly the same effect as reading its product color $h(a)$ in the compressed product. The nondeterministic union of these copies, one for each chosen hash, recognizes $L_{q,L}$. One branch has state exponent $R(x, y)$ by Step 2, and the number of branches has exponent $H(y)$ by Step 3. Therefore the union has exponent

$$R(x, y) + H(y) = E(x, y) \ ,$$

which proves (27).

Correctness is immediate. If an input symbol repeats, both occurrences receive the same product color; because every compressed path is repetition-free, every branch rejects. If the input word is repetition-free, choose a hash that certifies its underlying set. Property (P3) ensures that the corresponding branch accepts every ordering of that certified set, including the input word.

Step 5: ordinary transitions. For the final theorem we also need a bound on the number of transitions. One compressed product has no more than the $A(1)^N = 2^{O(q)}$ states of its raw product. Its color alphabet has $cN = O(q)$ elements. Even the crude bound that allows every state-color pair to connect to every state gives only $2^{O(q)}$ transitions. For one fixed hash, the construction in Step 4 replaces each compressed-product transition with at most L transitions over $[L]$, so it increases the transition count by at most a factor L. The number of hashes is $2^{O(q)} L^{O(1)}$. Hence

$$|\Delta(C_{q,L})| \le 2^{O(q)} L^{O(1)} \ ,$$

which is (28). □

6.3. From intermediate blocks to the full input

The inner NFA recognizes $L_{q,L}$. Its state bound has the desired constant $E(x, y)$, but its transition bound is only $2^{O(q)}L^{O(1)}$, with an unspecified constant in the exponent. We prevent that local transition constant from being multiplied by k/q by using an outer product in which one transition updates only one component.

Choose an allowed value $q = q(k)$ such that

$$q \longrightarrow \infty, \qquad q = o(k) \ .$$

Because the allowed values of q described after (22) are multiples of the fixed integer $M\mu$, we may take the least allowed value at least $\sqrt{k}$. Let K be the least multiple of q with $K \geq k$, and define

$$T \triangleq K/q, \qquad L \triangleq q^3 \ .$$

These parameters have simple roles:

- K is a padded target length divisible by q;
- T is the number of outer blocks;
- every outer block will contain exactly q symbols; and
- $L = q^3$ is large enough to give distinct local names to the symbols inside every block at subexponential cost.

Since $0 \leq K - k < q = o(k)$, we have $K = k + o(k)$.

The first-coordinate requirement below is exactly a splitter, while the second-coordinate requirement assigns distinct local names to the elements mapped to each first-coordinate value. The precise deterministic composition used here is proved in Appendix B from the splitter and perfect-hash constructions of Naor, Schulman, and Srinivasan [14, Sections 2.2 and 4.4, pp. 183, 186–187].

Lemma 18 (Balanced outer hashes)**.** *For every domain size $u \geq K$, there is a family of $2^{o(K)}u^{O(1)}$ functions*

$$g : [u] \longrightarrow [T] \times [L]$$

such that, for every K-element subset of $[u]$, some g sends exactly q elements to each outer block and is injective on the second coordinates inside each block.

Proof. Appendix B.4 gives the probability calculation and then the stronger globally explicit construction used by the deterministic algorithm. □

Theorem 19 (Amplification theorem)**.** *Let G be a fixed symmetric partial gadget of capacity $r \geq 1$ over a c-element color set, with polynomials A and B. Then, for every rational $x \in (0,1)$, positive rational y, and $1 \leq k \leq n$, there exists an acyclic NFA recognizing $L_{k,n}$ with*

$$|Q| + |\Delta| \leq 2^{(E(x,y)+o(1))k} n^{O(1)} .$$

Proof. The proof has five steps.

Step 1: pad the target length. Temporarily add $K - k$ fresh symbols to $[n]$. The padded alphabet has size

$$n_{\text{pad}} \triangleq n + K - k .$$

We construct an automaton for $L_{K,n_{\text{pad}}}$ and later remove the fresh symbols.

Step 2: build one branch for one outer hash. Construct the inner NFA $C_{q,L}$ from Proposition 17. Fix one function

$$g : [n_{\text{pad}}] \longrightarrow [T] \times [L]$$

from Lemma 18. Take the raw product of T copies of $C_{q,L}$. Write $g(a) = (g_1(a), g_2(a))$. When the product automata reads a symbol $a \in [n_{\text{pad}}]$, the value $g_1(a) \in [T]$ selects one inner copy, and $g_2(a) \in [L]$ is the local symbol read by that copy.

Step 3: proving the correctness of the union of branches. Suppose an input symbol a occurs twice. Both occurrences have the same value $g(a)$. The same inner copy therefore sees the same local symbol twice, and that copy rejects. Thus every branch rejects every word with a repeated symbol.

Now suppose the input word is repetition-free and has length at most K. Let S be its set of symbols. Because $n_{\text{pad}} = n + K - k \geq K$, extend S to a K-element set $S' \subseteq [n_{\text{pad}}]$, and choose an outer hash g that is successful on S'. By the definition of success in Lemma 18, for every $i \in [T]$ there are exactly q elements $a \in S'$ with $g_1(a) = i$, and their values $g_2(a)$ are pairwise distinct. Passing from S' to S only deletes elements. Hence each inner copy receives at most q symbols, and their local names remain pairwise distinct. Each copy of $C_{q,L}$ therefore accepts its local word, so the branch for g accepts the original word. Taking the nondeterministic union over the outer hashes recognizes $L_{K,n_{\text{pad}}}$.

Step 4: count states. Let

$$S_q = |Q(C_{q,L})| \ .$$

One outer product branch has at most S_q^T states. By (27),

$$S_q \le 2^{(E(x,y)+o(1))q} L^{O(1)} \ .$$

Raise this bound to the power $T = K/q$:

$$S_q^T \le 2^{(E(x,y)+o(1))K} L^{O(K/q)} \ .$$

Since $L = q^3$,

$$L^{O(K/q)} = q^{O(K/q)} = 2^{O((K/q)\log q)} = 2^{o(K)} \ .$$

Thus, the number of states in one branch is

$$2^{(E(x,y)+o(1))K} \ .$$

By Lemma 18, the outer family has at most $2^{o(K)} n_{\mathrm{pad}}^{O(1)}$ functions. Multiplying this by the number of states in one branch gives the explicit bound

$$|Q| \le 2^{(E(x,y)+o(1))K} n_{\mathrm{pad}}^{O(1)} \ .$$

Step 5: count ordinary transitions and remove the padding. Let

$$D_q = |\Delta(C_{q,L})| \ .$$

A transition of the outer product updates one of the T components. Choose the other $T-1$ local states, choose one local transition in the updated component, and choose the original input symbol that triggers it. This gives the bound

$$n_{\mathrm{pad}} D_q S_q^{T-1}$$

for one branch. By (28), $D_q = 2^{O(q)} L^{O(1)}$. Since $S_q \ge 1$, the number of transitions in one branch is therefore at most

$$\begin{aligned} n_{\mathrm{pad}} D_q S_q^{T-1} &\le n_{\mathrm{pad}} D_q S_q^T \\ &\le 2^{(E(x,y)+o(1))K} n_{\mathrm{pad}}^{O(1)} \ , \end{aligned}$$

where we used $q = o(K)$ and $L = q^3$ to absorb D_q into the $o(K)$ and polynomial terms. Multiplying by the at most $2^{o(K)} n_{\mathrm{pad}}^{O(1)}$ outer hashes from Lemma 18 gives

$$|\Delta| \le 2^{(E(x,y)+o(1))K} n_{\mathrm{pad}}^{O(1)} .$$

Finally, delete every transition labeled by one of the $K - k$ fresh symbols and retain only layers $0, \dots, k$. Every component automaton is layered, and every ordinary transition advances by one layer. Raw products, nondeterministic unions formed by merging initial states, and the deletion of transitions or terminal layers therefore preserve acyclicity. The resulting automaton is acyclic and recognizes $L_{k,n}$. We have $K = k + o(k)$ and $n_{\mathrm{pad}} = O(n)$, so the claimed bound follows. □

6.4. The sharper Witt bound

Regard the Witt complete gadget as a partial gadget by certifying every color set of size at most six. Choose

$$x_0 = \frac{173}{250} = 0.692, \qquad y_0 = \frac{1547}{1000} = 1.547 .$$

Substitution into (22) and (24)–(26) gives

quantity	value
$\mu(y_0)$	$5.2105167\ldots$
λ	$0.1919195\ldots$
$H(y_0)$	$0.5257292\ldots$
$R(x_0, y_0)$	$1.4469540\ldots$
$E(x_0, y_0)$	$1.9726832\ldots$

and therefore

$$2^{E(x_0,y_0)} = 3.9249744\ldots < 3.925 . \tag{31}$$

By folding the $o(1)$ factor into the numerical slack of Equation (31), we conclude the following.

Corollary 20 (Sharper Witt bound)**.** *There is an acyclic NFA recognizing $L_{k,n}$ of size*

$$3.925^k n^{O(1)} .$$

The finite automaton in the fixed-histogram and sharper Witt bounds is identical. The improvement from (19) to (31) is the first use of compose-and-compress: the number of Witt-gadget copies grows with q, the middle band is removed, and only the two tails are counted. The next section uses the same operation at finite scale to change the partial gadget itself.

7. Second use: an eleven-Witt partial gadget

The first use compressed a growing product while leaving the finite Witt complete gadget unchanged. We now use the same operation once at finite scale to construct a better partial gadget, and then feed that partial gadget back into the growing construction of Section 6. The eleven-copy construction below is the finite composition used in the main theorem.

Take eleven Witt complete gadgets on disjoint eleven-color sets. The raw product has 121 colors, capacity $R = 66$, and state polynomial $A_W(z)^{11}$. Set the target capacity to

$$q_{11} = 61 \ .$$

Then $s = R - q_{11} = 5$ and

$$\ell = \left\lfloor \frac{q_{11} - 1}{2} \right\rfloor = 30 \ ,$$

so CaC deletes raw ranks $31, \ldots, 35$.

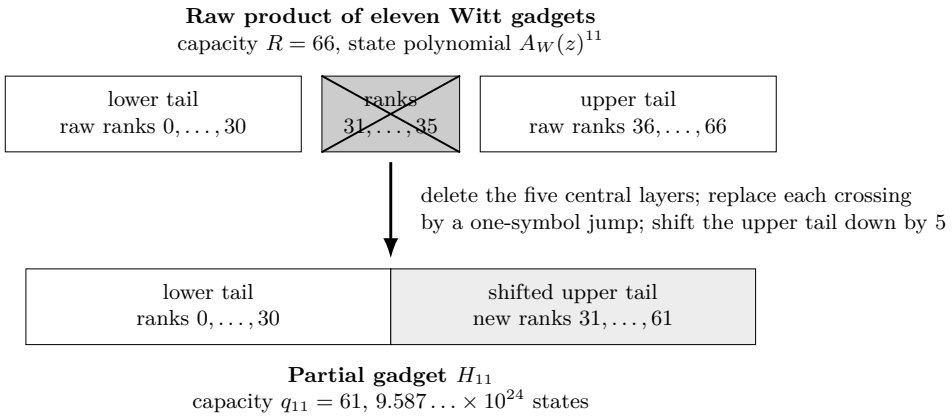


Figure 7: The one finite composition used in the main theorem. Eleven Witt complete gadgets are composed, five central layers of their raw product are deleted, and the result is a capacity-61 partial gadget.

Call the resulting partial gadget H_{11}. If

$$A_W(z)^{11} = \sum_{j=0}^{66} a_j z^j, \qquad B_W(z)^{11} = \sum_{j=0}^{66} b_j z^j \ ,$$

then Lemma 15 gives

$$A_{11}(z) = \sum_{j=0}^{30} a_j z^j + \sum_{j=36}^{66} a_j z^{j-5} \ , \tag{32}$$

$$B_{11}(z) = \sum_{j=0}^{61} b_j z^j \ . \tag{33}$$

The polynomial A_{11} is symmetric. The raw product has

$$200^{11} = 20{,}480{,}000{,}000{,}000{,}000{,}000{,}000{,}000$$

states. The five deleted layers contain

$$\sum_{j=31}^{35} [z^j] A_W(z)^{11} = 10{,}892{,}645{,}599{,}457{,}631{,}372{,}405{,}834$$

states, so

$$A_{11}(1) = 9{,}587{,}354{,}400{,}542{,}368{,}627{,}594{,}166 \ .$$

The exact coefficients are listed in Appendix C.

7.1. The final numerical bound

Choose the rational values

$$x = \frac{153}{200} = 0.765, \qquad y = \frac{81}{50} = 1.62 \ .$$

Direct evaluation gives (see Appendix D for verification)

$$A_{11}(x) = 3.481206546463651806859\ldots \times 10^{21} \ ,$$

$$B_{11}(y) = 2.439760214853457547554\ldots \times 10^{46} \ ,$$

and

$$\mu(y) = \frac{y B'_{11}(y)}{B_{11}(y)} = 57.2020375407\ldots, \qquad \lambda = \frac{1}{\mu} = 0.01748189475\ldots \ .$$

Here $c = 121$, because H_{11} uses eleven disjoint eleven-color palettes. Substituting the displayed values into the two terms of (24)–(25) gives

$$H(y) = \log_2(121\lambda \mathrm{e}) + \log_2 y - \lambda \log_2 B_{11}(y) < 0.525678 \ ,$$

$$R(x,y) = \lambda \log_2 A_{11}(x) - \frac{1}{2}\log_2 x < 1.444240 \ .$$

Consequently,

$$E(x, y) < 1.969918$$

and

$$2^{E(x,y)} < 3.917459 < 3.918 \ . \tag{34}$$

Finally, the slack in (34) absorbs the $o(1)$ term. Applying Theorem 19, together with the deterministic construction in Appendix B, proves Theorem 1.

8. Discussion

The paper develops a gadget-amplification framework and improves it at three successive levels. Full-product amplification with the Witt gadget gives an $O^*(3.967^k)$-size NFA. Applying compose-and-compress at the growing scale improves the bound to $O^*(3.925^k)$. Applying finite compression first to obtain H_{11} and then using growing amplification gives the proved $O^*(3.918^k)$ bound.

The final proved constant is not intrinsic to the amplification argument. One route to a better bound is a better starting complete gadget, obtained for example from a smaller balanced separating family or from a different finite design. A broader possibility is to start directly from a partial gadget whose certified sets have been chosen with later composition in mind, rather than obtaining partiality only by compressing a complete gadget. Such a gadget may certify fewer local color sets while having a state profile small enough to improve the combined exponent.

Stopping after one finite composition is a choice made for verifiability, not a limitation of the compose-and-compress method. In exploratory computations, starting from the Witt gadget and applying the sequence

$$\langle 5, 1\rangle, \quad \langle 3, 4\rangle, \quad \langle 3, 8\rangle, \quad \langle 2, 5\rangle, \quad \langle 2, 7\rangle, \quad \langle 2, 9\rangle \ ,$$

where $\langle m, s\rangle$ denotes composing m copies of the current gadget and compressing a central band of width s, gave a size of $O^*(3.91321^k)$. The exact state and certified-set polynomials grow rapidly along this sequence, requiring a computer-assisted computation and verification. We have preferred the slightly weaker bound from one finite composition because every finite input to that calculation can be inspected directly in the manuscript.

The most basic open problem is to close the gap between the known 2^k lower bound and the 3.918^k upper bound. A separate question is whether our construction can be leveraged to break the 4^k barrier for deterministic approximate counting of k-paths. For every fixed $\varepsilon > 0$, Lokshtanov,

Björklund, Saurabh, and Zehavi give a deterministic $(1 \pm \varepsilon)$-approximation algorithm running in $O^*(4^{k+o(k)})$ time [15]. A direct application of our NFA does not give such an algorithm, because different repetition-free words may have different numbers of accepting runs. It would therefore be interesting to determine whether the construction can be made approximately balanced, so that every repetition-free word has approximately the same known number of accepting runs, while retaining fewer than 4^k states.

Appendix A. How the Witt histogram was selected

The theorem is valid for every feasible integral template. We now explain the particular numbers used for the Witt gadget.

Before doing any calculus, let us state exactly what is being optimized. For fixed K and N, (2) reads

$$p_{K,N} = \frac{K!}{(cN)^K}[z^K]B(z)^N \ .$$

The factor $K!/(cN)^K$ is then fixed. Thus a larger lower bound on $[z^K]B(z)^N$ gives a larger success probability $p_{K,N}$, and Lemma 10 then needs fewer hash functions. This is why the coefficient is the quantity we want to make large.

A load distribution $\mathbf{p}$ used by N copies must have mean

$$\sum_j jp_j = \frac{K}{N} =: \mu \ .$$

For a rational distribution that is repeated as one histogram, (12) gives the lower bound

$$[z^K]B(z)^N \geq 2^{N\Psi_B(\mathbf{p})-o(N)} \ .$$

Consequently, when $K/N = \mu$ is fixed, the correct inner optimization problem is to maximize $\Psi_B(\mathbf{p})$ over distributions of mean μ.

There is also an outer tradeoff. The mean μ is not fixed by the problem: changing it changes

$$\lambda = \frac{N}{K} = \frac{1}{\mu} \ .$$

This changes both the hash exponent and the product-state cost $s(G)^N = s(G)^{\lambda K}$ in (14). We therefore make the choice in two stages:

(i) for each fixed mean μ, find the distribution that gives the largest coefficient contribution; and

(ii) vary μ, or equivalently λ, to minimize the total NFA exponent.

We now solve the fixed-mean optimization problem. Let $J = \{j : B_j > 0\}$, and consider real distributions $\mathbf{p} = (p_j)_{j\in J}$ satisfying

$$\sum_{j\in J} p_j = 1, \qquad \sum_{j\in J} jp_j = \mu \ .$$

Multiplying the objective by $\ln 2$ does not change its maximizer, so it is equivalent to maximizing

$$\widehat{\Psi}_B(\mathbf{p}) \triangleq (\ln 2)\Psi_B(\mathbf{p}) = \sum_{j\in J} p_j \ln \frac{B_j}{p_j} \ .$$

Write $j_{\min} = \min J$ and $j_{\max} = \max J$. The possible mean values are exactly $j_{\min} \le \mu \le j_{\max}$. Fix a target average load with $j_{\min} < \mu < j_{\max}$. We first solve the stationarity equations subject to the two displayed constraints and the explicit conditions $p_j > 0$ for every $j \in J$. The relative-entropy argument below will show that the resulting positive distribution is the unique global maximizer. Add multipliers α and θ for the two displayed constraints. The Lagrangian is

$$\mathcal{L}(\mathbf{p}, \alpha, \theta) = \sum_{j\in J} p_j \ln \frac{B_j}{p_j} + \alpha \left(\sum_{j\in J} p_j - 1 \right) + \theta \left(\sum_{j\in J} j p_j - \mu \right) \ .$$

Differentiating with respect to p_j gives one equation for each load j:

$$0 = \frac{\partial \mathcal{L}}{\partial p_j} = \ln B_j - \ln p_j - 1 + \alpha + \theta j \ .$$

Solving this equation for p_j yields

$$p_j = B_j \mathrm{e}^{\alpha - 1} \mathrm{e}^{\theta j} \ .$$

The multiplier θ appears only through the powers $\mathrm{e}^{\theta j}$. Define

$$y \triangleq \mathrm{e}^{\theta} > 0 \ .$$

This single positive parameter indexes the stationary distributions; it is not an additional load, probability, or constraint. The normalization constraint now determines the remaining constant, because

$$1 = \sum_{j\in J} p_j = \mathrm{e}^{\alpha - 1} \sum_{j\in J} B_j y^j = \mathrm{e}^{\alpha - 1} B(y) \ .$$

Consequently every stationary distribution has the tilted form

$$p_j(y) = \frac{B_j y^j}{B(y)} \qquad (y > 0) \ . \tag{A.1}$$

The mean constraint selects the value of y. Direct substitution also gives both identities used below:

$$\begin{aligned}
\mu(y) &= \sum_{j\in J} j\, p_j(y) \\
&= \frac{1}{B(y)} \sum_{j\in J} j B_j y^j = \frac{yB'(y)}{B(y)} \\
\Psi_B(\mathbf{p}(y)) &= \sum_{j\in J} p_j(y)\Big(\log_2 B_j - \log_2 p_j(y)\Big) \\
&= \sum_{j\in J} p_j(y)\Big(\log_2 B(y) - j \log_2 y\Big) \\
&= \log_2 B(y) - \mu(y) \log_2 y \ .
\end{aligned} \tag{A.2}$$

To see how the mean changes with y, fix a load j and differentiate $p_j(y) = B_j y^j / B(y)$. Here j and B_j are constants; only y^j and $B(y)$ depend on y. The quotient rule gives

$$\begin{aligned}
\frac{dp_j(y)}{dy} &= B_j \frac{j y^{j-1} B(y) - y^j B'(y)}{B(y)^2} \\
&= \frac{B_j y^j}{B(y)} \left(\frac{j}{y} - \frac{B'(y)}{B(y)} \right) \\
&= p_j(y) \left(\frac{j}{y} - \frac{B'(y)}{B(y)} \right) \ .
\end{aligned}$$

The identity $\mu(y) = yB'(y)/B(y)$ says that $B'(y)/B(y) = \mu(y)/y$. Substituting it into the last line yields

$$\frac{dp_j(y)}{dy} = \frac{p_j(y)}{y}\Big(j - \mu(y)\Big) \ . \tag{A.3}$$

Thus increasing y increases the probability of every load above the current mean and decreases the probability of every load below it.

Since $\mu(y) = \sum_{j\in J} j\, p_j(y)$, differentiating term by term and using (A.3)

gives

$$\begin{aligned}
\mu'(y) &= \sum_{j\in J} j\,\frac{dp_j(y)}{dy} \\
&= \frac{1}{y}\sum_{j\in J} jp_j(y)\Big(j-\mu(y)\Big) \\
&= \frac{1}{y}\left(\sum_{j\in J} j^2 p_j(y) - \mu(y)\sum_{j\in J} jp_j(y)\right) \\
&= \frac{1}{y}\left(\sum_{j\in J} j^2 p_j(y) - \mu(y)^2\right) \\
&= \frac{\mathrm{Var}_{\mathbf{p}(y)}(j)}{y}\ .
\end{aligned}$$

Because $y > 0$, this derivative is positive unless only one load is available. Thus in the Witt case, where all loads $0,\ldots,6$ are available, $\mu(y)$ increases continuously from 0 to 6. Therefore, given any target average load μ with $0 < \mu < 6$, exactly one $y > 0$ solves $\mu(y) = \mu$. The cases $\mu = 0$ and $\mu = 6$ arise as the limits $y \to 0$ and $y \to \infty$.

The Lagrange equations identify a stationary candidate. To prove that this candidate is the global maximizer, we compare its objective value with that of every other feasible distribution. Fix $y > 0$, write $\mathbf{q} = \mathbf{p}(y)$, and let $\mathbf{p}$ be any other distribution with the same mean $\mu(y)$. From (A.1),

$$\log_2 q_j = \log_2 B_j + j\log_2 y - \log_2 B(y)\ ,$$

so the base-two relative entropy can be expanded one term at a time:

$$\begin{aligned}
D_2(\mathbf{p}\|\mathbf{q}) &\triangleq \sum_{j\in J} p_j \log_2 \frac{p_j}{q_j} \\
&= \sum_{j\in J} p_j \log_2 p_j - \sum_{j\in J} p_j \log_2 B_j \\
&\quad - \left(\sum_{j\in J} jp_j\right)\log_2 y + \left(\sum_{j\in J} p_j\right)\log_2 B(y) \\
&= -\Psi_B(\mathbf{p}) - \mu(y)\log_2 y + \log_2 B(y)\ .
\end{aligned}$$

The last equality uses both feasibility constraints: $\sum_j p_j = 1$ and $\sum_j jp_j = \mu(y)$. Applying the second identity in (A.2) to $\mathbf{q}$ gives

$$\Psi_B(\mathbf{q}) = \log_2 B(y) - \mu(y)\log_2 y\ .$$

Comparing the last two displays yields the exact gap identity

$$D_2(\mathbf{p}\|\mathbf{q}) = \Psi_B(\mathbf{q}) - \Psi_B(\mathbf{p}) \ .$$

It remains only to note why the left-hand side is nonnegative. For each j with $p_j > 0$, apply $-\ln x \geq 1 - x$ to $x = q_j/p_j$. Then

$$(\ln 2) D_2(\mathbf{p}\|\mathbf{q}) = -\sum_{j:p_j>0} p_j \ln \frac{q_j}{p_j} \geq \sum_{j:p_j>0} (p_j - q_j) = 1 - \sum_{j:p_j>0} q_j \geq 0 \ .$$

The final inequality holds because $\mathbf{q}$ is a probability distribution, so the sum of its entries over any subset of J is at most one. Equality in all the preceding inequalities is possible only when $\mathbf{p} = \mathbf{q}$. Therefore

$$\Psi_B(\mathbf{p}) \leq \Psi_B(\mathbf{q}) \ ,$$

with equality only for $\mathbf{p} = \mathbf{q}$. This proves that the tilted distribution is the unique global maximizer for its mean.

We can now see exactly what varying y does. Define

$$\lambda(y) \triangleq \frac{1}{\mu(y)} \ .$$

For this value of λ, the tilt $\mathbf{p}(y)$ is the best continuous histogram with mean $1/\lambda(y)$. Substituting (A.2) into (14) turns the continuous search into the one-variable minimization of

$$\begin{aligned} H_{\mathrm{raw}}(y) &= \log_2(c\lambda(y)\mathrm{e}) \\ &\quad - \lambda(y)\Big(\log_2 B(y) - \mu(y)\log_2 y\Big) \ . \end{aligned}$$

Here $H_{\mathrm{raw}}(y)$ is the hash rate before truncating it at zero. The total exponent is therefore

$$E_{\mathrm{tilt}}(y) = \lambda(y)\log_2 s(G) + \max\{0, H_{\mathrm{raw}}(y)\} \ .$$

Increasing y gives the larger loads more weight through the factor y^j. This raises the mean $\mu(y)$ and lowers the number $\lambda(y) = N/K$ of gadget copies per input symbol, but it also changes the coefficient and hash contributions. The displayed function records this tradeoff completely.

Thus varying y allows us to optimize the continuous histogram for each possible mean in terms of the final NFA exponent. At the Witt minimum,

$H_{\mathrm{raw}}(y) = 0.5691\ldots > 0$, so the maximum selects $H_{\mathrm{raw}}(y)$. Substituting $\lambda(y) = 1/\mu(y)$ therefore gives

$$E_{\mathrm{tilt}}(y) = \log_2 y + \log_2 \frac{ce}{\mu(y)} + \frac{\log_2 s(G) - \log_2 B(y)}{\mu(y)} \ .$$

This is an ordinary one-variable function. A numerical minimization for the Witt values $c = 11$, $s(G) = 200$, and $B = B_W$ gives

$$y = 1.942858778\ldots \ .$$

Equivalently, differentiating the last display with respect to $t = \ln y$ shows that an interior stationary point satisfies

$$\frac{dE_{\mathrm{tilt}}}{dt} = -\frac{\mathrm{Var}_{\mathbf{p}(y)}(j)}{\mu(y)^2 \ln 2} \left(\mu(y) + \ln \frac{s(G)}{B(y)} \right) = 0 \ ,$$

or

$$B(y) = s(G) \mathrm{e}^{\mu(y)} \ .$$

Solving this equation gives the same numerical value. We round it to $y \approx 1.94286$, for which

j	0	1	2	3	4	5	6
$1000 p_j(y)$	0.023	0.488	4.742	27.640	107.402	292.133	567.572 .

Rounding these seven values gives the simple integral template in Table 2. The first two entries round to zero, so only five loads need to be displayed. The proof in the main text does not assume that this rounded template is optimal; it recomputes the final exponent directly from the five displayed integers.

There is also no asymptotic reason to sum every histogram in (4). For fixed capacity r, the number of feasible histograms is at most $(N+1)^{r+1}$, which is polynomial in N. Hence the largest single summand and the full coefficient differ by at most a polynomial factor. The denominator 1000 is used only to make the chosen approximation short and transparent.

Appendix B. Derandomization and uniform construction

The main proof uses random hash functions only to estimate how many hashes are needed. This appendix replaces those hashes by deterministic families and shows that the complete NFA can be listed within the asymptotic bound of Theorem 1. There are two families to construct:

(i) the *inner hashes* in Proposition 17, which map the alphabet $[L]$ into the colors of a product partial gadget; and

(ii) the *outer hashes* in Lemma 18, which divide the input symbols evenly among T outer blocks and are injective inside each block.

We use the globally explicit splitter and k-restriction constructions of Naor, Schulman, and Srinivasan [14, Sections 3 and 4.4, pp. 184–187].

For reference, the parameters have the same meanings as in the main proof:

k	requested maximum word length,
q	intermediate block capacity, chosen as $\Theta(\sqrt{k})$,
K	the least multiple of q with $K \geq k$,
$T = K/q$	number of outer blocks,
$L = q^3$	alphabet size of one inner NFA.

The divisibility adjustment adds $K - k$ fresh symbols to the input alphabet. We call the resulting alphabet the *padded alphabet* and denote its size by

$$n_{\mathrm{pad}} \triangleq n + K - k \ . \tag{B.1}$$

The symbol n_{pad} is therefore notation, not a second input parameter. At the end we delete all transitions labeled by the fresh symbols and keep only ranks $0, \ldots, k$. Because $K - k < q = O(\sqrt{k})$ and $n \geq k$, we have $n_{\mathrm{pad}} = O(n)$.

The two replacements used below are worth keeping separate. The inner hash family is obtained by deterministic search on the small domain $[q^3]$ and costs $2^{O(q \log q)} = 2^{o(k)}$ time. The outer hash family is assembled from three explicit splitter families and has $2^{o(K)} \log(2n_{\mathrm{pad}})$ members.

Appendix B.1. One deterministic hash-family lemma

We first state the precise part of the k-restriction framework that we use. In both applications below, Γ is the set of assignments of hash values to one fixed t-element input set that make a product-automaton branch succeed. The lemma turns the fraction of assignments that belong to Γ into a deterministic family of hash functions such that every t-element input set receives an assignment in Γ.

Lemma 21 (Deterministic realization of allowed hash patterns)**.** *Let u, t, b be positive integers with $1 \leq t \leq u$ and $b \leq u$. Let $\varnothing \neq \Gamma \subseteq [b]^t$ be closed under*

permutations of its coordinates: if $(x_1, \ldots, x_t) \in \Gamma$*, then* $(x_{\pi(1)}, \ldots, x_{\pi(t)}) \in \Gamma$ *for every permutation* π *of* $[t]$*. Put*

$$\delta = \frac{|\Gamma|}{b^t} \ .$$

Assume that membership in Γ *can be tested in time* τ*. Then one can deterministically construct a family* $\mathcal{R}$ *of functions* $h : [u] \to [b]$ *such that, for every* $S = \{s_1 < \cdots < s_t\} \subseteq [u]$*, some* $h \in \mathcal{R}$ *satisfies*

$$(h(s_1), \ldots, h(s_t)) \in \Gamma \ .$$

If $\delta = 1$*, one function suffices. If* $0 < \delta < 1$*, the family has size at most*

$$\left\lceil \frac{t \ln u + 1}{-\ln(1-\delta)} \right\rceil = O\Big(\delta^{-1} t \log(2u)\Big) \ . \tag{B.2}$$

In either case it can be constructed in time

$$O\left(\delta^{-1} \left(\frac{\mathrm{e}u^2}{t}\right)^t \tau\right) \le \delta^{-1} u^{2t+O(1)} \tau \ . \tag{B.3}$$

Proof. Apply the one-constraint case of the k-restriction theorem of Naor, Schulman, and Srinivasan [14, Theorem 1, p. 185] with domain size u, restriction size t, alphabet size b, one constraint, and allowed set Γ. Exactly $|\Gamma|$ of the b^t possible assignments belong to Γ, so the fraction of allowed assignments is $\delta = |\Gamma|/b^t$. The theorem therefore gives a family of size at most the bound in (B.2). Its construction time is

$$O\left(\delta^{-1} \binom{u}{t} \tau \, |\mathcal{H}_{u,t,b}|\right) \ ,$$

where the explicit t-wise independent space used in that theorem satisfies $|\mathcal{H}_{u,t,b}| \le u^t$ because $b \le u$. Using $\binom{u}{t} \le (\mathrm{e}u/t)^t$ gives (B.3). □

The four quantities in this lemma have direct meanings in our applications: u is the domain of a hash, t is the size of the input set that must be covered, b is the number of hash values, and δ is exactly the success probability of one uniformly random hash on a fixed t-set.

Appendix B.2. The hash family in Lemma 10

We first derandomize the family used in the fixed-histogram analysis. Fix the gadget G, the number N of copies, and the target length k as in Section 4. Each hash maps an input symbol to a pair $(i, a) \in [N] \times C$, where i selects one of the N gadget copies and a is a color of that copy. Thus the hash has cN possible values, so the parameter b in Lemma 21 is $b = cN$. Let $\Gamma_{k,N} \subseteq ([N] \times C)^k$ contain the ordered assignments with the following two properties:

(a) no copy-color cell occurs twice; and

(b) each copy receives at most r cells, where r is the capacity of G.

The condition is unchanged when the k coordinates are permuted. Moreover,

$$|\Gamma_{k,N}| = k![z^k]B_G(z)^N \ ,$$

so its density is the success probability in (2):

$$\frac{|\Gamma_{k,N}|}{(cN)^k} = p_{k,N} \ .$$

Membership is tested in $k^{O(1)}$ time by recording the used cells and the load of every copy.

Applying Lemma 21 directly with domain $[n]$ would cost $n^{O(k)}$ time. We therefore perform the standard size reduction first. An (u, t, ℓ)-splitter is a family of maps from $[u]$ to $[\ell]$ such that, for every t-element set $S \subseteq [u]$, some map assigns either $\lfloor t/\ell \rfloor$ or $\lceil t/\ell \rceil$ elements of S to each value in $[\ell]$. In particular, an (u, t, t^2)-splitter contains a map that is injective on every prescribed t-set. Lemma 2 of Naor, Schulman, and Srinivasan [14, Lemma 2, p. 186] gives a globally explicit (n, k, k^2)-splitter $\mathcal{A}$ of size

$$|\mathcal{A}| = O(k^6 \log k \log(2n)) \ .$$

Now apply Lemma 21 on the reduced domain $[k^2]$ with allowed patterns $\Gamma_{k,N}$. In the fixed-histogram application $N = \Theta(k)$ and c is fixed, so $cN \le k^2$ for all sufficiently large k; the finitely many remaining values can be handled directly. This gives maps $r : [k^2] \to [N] \times C$ forming a family $\mathcal{R}$ of size

$$|\mathcal{R}| = O\Big(p_{k,N}^{-1} k \log(2k)\Big) \ .$$

Define

$$\mathcal{H}_{\text{fix}} \triangleq \{r \circ a : a \in \mathcal{A},\ r \in \mathcal{R}\} \ .$$

For a k-set $S \subseteq [n]$, choose $a \in \mathcal{A}$ injective on S and then choose $r \in \mathcal{R}$ that realizes an allowed pattern on the set $a(S)$. Thus some member of $\mathcal{H}_{\text{fix}}$ succeeds on S. A smaller set is covered by extending it to a k-set, exactly as in Lemma 10. We have

$$|\mathcal{H}_{\text{fix}}| \le p_{k,N}^{-1} k^{O(1)} \log(2n) \ ,$$

and the family can be listed in time complexity of

$$p_{k,N}^{-1} k^{O(k)} n^{O(1)} \ .$$

This already makes the fixed-histogram NFA a uniform fixed-parameter construction. The factor $k^{O(k)}$ does not preserve the base displayed in Theorem 11; preserving the final base requires the two-scale construction below.

Appendix B.3. Explicit inner hashes

We now derandomize the hashes in Proposition 17. Using the notation of that proposition: G is the fixed partial gadget, q is an allowed intermediate length, $N = \lambda q$ copies are composed, and the input alphabet for the resulting inner NFA is $[L]$, where later $L = q^3$.

Let $D = [N] \times C$ be the color set of the raw product. Let $\Gamma_q \subseteq D^q$ contain the ordered q-tuples for which

(a) all q copy-color cells are distinct; and

(b) in each copy, the set of local colors is certified by G.

This is precisely the event counted in (29). Therefore

$$|\Gamma_q| = q![z^q]B_G(z)^N \qquad \text{and} \qquad \frac{|\Gamma_q|}{(cN)^q} = \rho \ .$$

The partial gadget is fixed, so its certified family is part of its constant-size description. Membership in Γ_q is consequently testable in $q^{O(1)}$ time. For the Witt complete gadget the local test is simply $|S| \le 6$. For the eleven-Witt partial gadget H_{11}, a set is certified exactly when it uses at most six colors from each Witt palette and at most 61 colors in total.

Apply Lemma 21 with

$$u = L = q^3, \qquad t = q, \qquad b = cN, \qquad \delta = \rho \ .$$

For large q, $b = O(q) \le L$. We obtain a deterministic family of

$$O\Big(\rho^{-1} q \log(2L)\Big) \tag{B.4}$$

inner hashes $h : [L] \to [N] \times C$. Its construction time is

$$\rho^{-1} L^{2q+O(1)} = 2^{O(q \log q)} \ .$$

To justify the last equality, property (P4) gives at least one certified product set of size q, hence $[z^q] B_G(z)^N \ge 1$. Since $q! \ge (q/\mathrm{e})^q$ and $cN = c\lambda q$, equation (29) yields

$$\rho \ge (c\lambda \mathrm{e})^{-q} \ .$$

Thus $\rho^{-1} = 2^{O(q)}$, while $L^{2q+O(1)} = 2^{O(q \log q)}$.

We must also construct the compressed product, rather than merely count its states. The raw product of $N = \Theta(q)$ copies of a fixed partial gadget has $2^{O(q)}$ states and ordinary transitions, so it can be generated and its unreachable states removed in $2^{O(q)}$ time. Let the deleted band have width s, as in Lemma 15. For every reachable state u at the lower boundary, compute all states w reachable from u after exactly s raw transitions. For every raw transition $w \xrightarrow{a} v$ into the retained upper boundary, add the compressed transition $u \xrightarrow{a} v$.

No search over hidden color sets is required. Choose any path from the initial state to u. If a continuation from u repeated a color, either inside the continuation or from the chosen prefix, their concatenation would be a path of the raw product labeled by a repeated color. This contradicts partial-gadget soundness, because every raw state is accepting. Hence every path found by the layered reachability computation is a valid witness for the shortcut. Even a cubic-time reachability computation in the raw product takes $2^{O(q)}$ time.

Finally, relabel the compressed product by every hash in (B.4) and take their nondeterministic union. This deterministically lists the inner NFA $C_{q,L}$ from Proposition 17. Its preprocessing cost is $2^{O(q \log q)}$, in addition to time polynomial in the number of states and transitions that are output.

Appendix B.4. Explicit outer hashes

We first give the short probability calculation behind Lemma 18 and then replace it by the standard explicit splitter construction.

Nonuniform existence proof. Choose one function $g : [u] \to [T] \times [L]$ uniformly at random. Its two coordinates play different roles. The first coordinate maps the $K = qT$ labeled elements of a fixed K-set to T blocks. The fraction of assignments placing exactly q elements in every block is

$$\rho_{\mathrm{bal}} \triangleq \frac{K!}{(q!)^T T^K} \ .$$

Stirling's formula gives

$$-\log_2 \rho_{\mathrm{bal}} = O(T \log q) = o(K) \ ,$$

because $T = K/q$ and $q \to \infty$.

Condition on balanced first coordinates. Inside one block, the probability that the q second coordinates are distinct is

$$\frac{(L)_q}{L^q} = \prod_{i=0}^{q-1} \left(1 - \frac{i}{L}\right) \ .$$

The T blocks are independent. For $L = q^3$ and sufficiently large q, $i/L \le 1/2$; hence $\log(1 - z) \ge -2z$ gives

$$-\log \rho_{\mathrm{inj}} \le \frac{2T}{L} \sum_{i=0}^{q-1} i = O\left(\frac{K}{q^2}\right) = o(K) \ .$$

Thus one random function succeeds on the fixed set with probability $\rho_{\mathrm{bal}}\rho_{\mathrm{inj}} = 2^{-o(K)}$. Taking $2^{o(K)} K \log(2u)$ independent functions and applying a union bound over the at most u^K possible K-sets proves the existence claim.

Globally explicit construction. We next replace the random family by published splitter and perfect-hash families. Recall that

$$K = qT, \qquad L = q^3, \qquad q = \Theta(\sqrt{K}) \ ,$$

and that the outer hashes are defined on the temporary alphabet $[n_{\mathrm{pad}}]$ from (B.1). We use three explicit splitter families from Naor, Schulman, and Srinivasan [14, Lemma 2 and Theorem 3(i),(iv), pp. 186–187]:

(i) an $(n_{\mathrm{pad}}, K, K^2)$-splitter $\mathcal{A}$, with

$$|\mathcal{A}| = K^{O(1)} \log(2n_{\mathrm{pad}}) \ , \tag{B.5}$$

that contains a map injective on every prescribed K-set;

(ii) a (K^2, K, T)-splitter $\mathcal{B}$, with

$$|\mathcal{B}| = O\left(\frac{K^{2T+O(1)} \log K}{T!}\right) = 2^{O(T \log K)} = 2^{o(K)} ,$$

that maps exactly q elements of every prescribed K-set to each value in $[T]$; and

(iii) a (K^2, q, L)-splitter $\mathcal{P}$, with

$$|\mathcal{P}| = q^{O(1)} \log K ,$$

that contains a map injective on every prescribed q-set. Here $L = q^3 \geq q^2$, so Theorem 3(iv) applies.

These are globally explicit families: each family can be listed in time polynomial in its domain size and its own output size.

For $a \in \mathcal{A}$, $b \in \mathcal{B}$, and a tuple $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_T) \in \mathcal{P}^T$, define

$$g_{a,b,\boldsymbol{\phi}}(x) \triangleq \Big(b(a(x)),\ \phi_{b(a(x))}(a(x))\Big) \in [T] \times [L] . \tag{B.6}$$

Let $\mathcal{G}$ contain all maps in (B.6).

Lemma 22 (Explicit outer family)**.** *For every K-set $S \subseteq [n_{\mathrm{pad}}]$, some $g \in \mathcal{G}$ sends exactly q elements of S to each outer bucket and is injective on the second coordinates inside every bucket. Moreover,*

$$|\mathcal{G}| \leq 2^{o(K)} \log(2n_{\mathrm{pad}}) , \tag{B.7}$$

and the complete family can be listed in $2^{o(K)} n_{\mathrm{pad}}^{O(1)}$ time.

Proof. Fix $S \subseteq [n_{\mathrm{pad}}]$ with $|S| = K$. Choose $a \in \mathcal{A}$ injective on S. Choose $b \in \mathcal{B}$ such that

$$|a(S) \cap b^{-1}(i)| = q \qquad \text{for every } i \in [T].$$

For each $i \in [T]$, choose $\phi_i \in \mathcal{P}$ injective on the q-element set $a(S) \cap b^{-1}(i)$. The resulting map $g_{a,b,\boldsymbol{\phi}}$ has the required two properties.

The number of maps is

$$|\mathcal{G}| = |\mathcal{A}|\,|\mathcal{B}|\,|\mathcal{P}|^T .$$

Since $T = K/q = \Theta(\sqrt{K})$,

$$\log_2 |\mathcal{B}| = O(T \log K) = o(K)$$

and

$$T \log_2 |\mathcal{P}| = O(T(\log q + \log\log K)) = o(K) \ .$$

Together with (B.5), this proves (B.7). Enumerating the choices and evaluating each composed map on $[n_{\mathrm{pad}}]$ adds only a polynomial factor in n_{pad}. □

Appendix B.5. Uniform form of the main construction

Theorem 23 (Uniform amplification)**.** *Under the hypotheses of Theorem 19, the NFA in that theorem can be constructed deterministically in*

$$2^{(E(x,y)+o(1))k} n^{O(1)} \tag{B.8}$$

time. The output consists of the complete state list and the complete list of ordinary one-symbol transitions.

Proof. For bounded k the claim can be satisfied by a direct finite construction. For larger k, choose the allowed value q, the padded value K, and $T = K/q$ exactly as in Section 6. Thus $q = \Theta(\sqrt{k})$ and $K = k + o(k)$.

First construct the inner NFA $C_{q,L}$, with $L = q^3$, by the deterministic procedure above. Its extra preprocessing time is $2^{O(q \log q)} = 2^{o(k)}$. Next enlarge the alphabet temporarily to $[n_{\mathrm{pad}}]$, construct the family $\mathcal{G}$ from Lemma 22, and for every $g \in \mathcal{G}$ list the raw product of T copies of $C_{q,L}$.

Let $S_q = |Q(C_{q,L})|$ and $D_q = |\Delta(C_{q,L})|$. A branch for one outer hash has at most S_q^T states. Store the outgoing local transitions of $C_{q,L}$ by local input symbol, and precompute $g(x)$ for every $x \in [n_{\mathrm{pad}}]$. Enumerating the product states takes $O(S_q^T)$ time. For each transition, choose the updated component, one of its local transitions, the states of the other $T - 1$ components, and an original input symbol that induces the local transition. Hence all states and transitions of one branch can be listed in

$$O\Big(S_q^T + n_{\mathrm{pad}} D_q S_q^{T-1}\Big)$$

time. The second term is also the transition bound used in the proof of Theorem 19. The nondeterministic union is formed without epsilon transitions by merging the initial states and taking the union of their outgoing transitions.

The state and transition estimates from Theorem 19 therefore remain unchanged. By Lemma 22, the number of outer branches is only $2^{o(K)} n_{\mathrm{pad}}^{O(1)}$. The total listing time is consequently

$$2^{(E(x,y)+o(1))K} n_{\mathrm{pad}}^{O(1)} .$$

The construction of the inner family and all three outer splitter families is subexponential in K and is absorbed by this bound. Finally delete every transition labeled by one of the $K - k$ fresh symbols and retain only ranks $0, \ldots, k$. Since $K = k + o(k)$ and $n_{\mathrm{pad}} = O(n)$, the result is (B.8). □

Corollary 24 (Uniform form of the main theorem)**.** *The NFA in Theorem 1 can be constructed deterministically in*

$$3.918^k n^{O(1)}$$

time.

Proof. The eleven-Witt partial gadget H_{11} is fixed. Its finite compose-and-compress step is constructed once by the layered-reachability procedure used for the inner NFA. Apply Theorem 23 with the rational values x and y from Section 7, and then use (34). □

If only rank k is accepting, the same output graph recognizes the traditional exact-length language. Ben-Basat, Gabizon, and Zehavi show that an explicit acyclic NFA of size s for that language yields a deterministic minimum-weight simple k-path algorithm in time $O(sn^2 \log W)$ for integer weights of magnitude at most W [1, Section 3]. Thus the uniform construction above also gives such an algorithm with running time $3.918^k n^{O(1)} \log W$. This observation records the algorithmic consequence of uniformity; it is not used elsewhere in the paper.

Appendix C. Coefficients of the eleven-Witt partial gadget

The next two tables expand (32) and (33). They contain all exact integers used in the final substitution; no numerical search or external certificate is needed to verify the displayed bound.

j	$[z^j]A_{11}$	$[z^j]B_{11}$
0	1	1
1	121	121
2	7260	7260
3	286891	287980
4	8372595	8495410
5	191926328	198792594
6	3590148903	3843323484
7	56226518975	63140310750
8	750820249872	899749078800
9	8664220979865	11296832619050
10	87283865030813	126523976934740
11	773660153828188	1276728613011390
12	6070667004845190	11703086455223240
13	42373390765351866	98121692243791370
14	264118233584540500	756881012184576840
15	1474717582817269098	5398390512819437562
16	7395064892682784108	35756962984086163842
17	3337662195639920709 0	220781732808203697900
18	13584323676756848908 2	127502595460637078700
19	49944330218306720280	690727917346907362615 0
20	166155668174645604679 4	351934410708150537359 40
21	501004687780722295427 4	169041462406342601641390
22	137149840615937725851 00	767012962690500474869800
23	341441193763237280185 22	329378464163820342811815 0
24	774370258160501338256 25	134088091220178101972658 00
25	160261587109715509889 99	518239187718615018087880 14
26	303156976457717708620 074	190409252871662685087548 244
27	524962205541521065357 435	665843978673145925551573 290
28	833326882669836119261 367	221836745548167675741674 8920
29	121411624159985027534 9740	704800480663713660006345 9390
30	162520772667489411523 2127	213704778434206147697849 82696

Table C.1: Exact coefficients of H_{11}, degrees 0 through 30.

j	$[z^j]A_{11}$	$[z^j]B_{11}$
31	1625207726674894115232127	6188316098869881004415111 9406
32	1214116241599850275349740	1712336990590342762542725 77125
33	8333268826693836119261367	4529682909307691167688089 26075
34	5249622055415210653574 35	1145955372380013863124868 534200
35	3031569764577177086200 74	2773354543347515465375527 514310
36	1602615871097155099889 99	6421766099824627001832732 034410
37	774370258160501338256 25	1422805629944058381349982 7476760
38	341441193763237280185 22	3016237854958009465542829 8010380
39	137149840615937725851 00	6117255088273234004495609 2020660
40	501004687780722295427 4	1186626689047838447453269 62163008
41	166155668174645604679 4	2200821435709043517037897 8914669128
42	499444330218306720280	3900887830915798488465237 11227800
43	135843236767568489082	6603728077666569712351488 43838400
44	33376621956399207090	1066941562469213756447700 967936800
45	7395064892682784108	1643728762638690913078626 400224480
46	1474717582817269098	2412106515472011756720476 125466880
47	264118233584540500	3367408233588917777131894 279473600
48	42373390765351866	4465709629768930125562108 491700800
49	6070667004845190	5616109549756970662661370 246585600
50	773660153828188	6684380843158579956024573 450993024
51	87283865030813	7511917616038063290570634 987749504
52	8664220979865	7949016579455335405379951 674583040
53	750820249872	7894911698912352592545592 448689920
54	56226518975	7331500055535166639173837 399632640
55	3590148903	6336791071517629004643473 671190016
56	191926328	5069778473880165126986746 880645376
57	8372595	3729376854292201572403735 168608000
58	286891	2501473076022710659977161 145907200
59	7260	1513894347870970784258386 016755200
60	121	8154717020592006191706457 12949760
61	1	3838882243507117248139172 87823360

Table C.2: Exact coefficients of H_{11}, degrees 31 through 61.

Appendix D. Interval verification of the final numerical bound

This exact-arithmetic Python 3 script verifies the inequalities in Section 7.

```python
from fractions import Fraction as Q
from math import comb
from mpmath import iv

iv.dps = 60

def mul(p, q):
    r = [0] * (len(p) + len(q) - 1)
    for i, a in enumerate(p):
        for j, b in enumerate(q):
            r[i + j] += a * b
    return r

def power(p, n):
    r = [1]
    for _ in range(n):
        r = mul(r, p)
    return r

def evaluate(p, x):
    r = Q(0)
    for a in reversed(p):
        r = r * x + a
    return r

def I(x):
    return iv.mpf(x.numerator) / x.denominator

A_raw = power([1, 11, 55, 66, 55, 11, 1], 11)
B_raw = power([comb(11, j) for j in range(7)], 11)

# Raw ranks 31,...,35 are deleted; raw j >= 36 shifts to j-5.
A_11 = A_raw[:31] + A_raw[36:]
B_11 = B_raw[:62]

assert sum(A_raw[31:36]) == 10892645599457631372405834
assert sum(A_11) == 95873544005423686627594166

x, y = Q(153, 200), Q(81, 50)
Ax, By = evaluate(A_11, x), evaluate(B_11, y)
Bprime = [(j + 1) * B_11[j + 1] for j in range(61)]
mu = y * evaluate(Bprime, y) / By
lambda_ = 1 / mu

log2 = lambda z: iv.ln(z) / iv.ln(2)
H = log2(121 * I(lambda_) * iv.e) + log2(I(y)) - I(lambda_) * log2(I(By))
R = I(lambda_) * log2(I(Ax)) - iv.mpf("0.5") * log2(I(x))
E = H + R
base = iv.power(2, E)

assert H.b < iv.mpf("0.525678").a
assert R.b < iv.mpf("1.444240").a
assert E.b < iv.mpf("1.969918").a
assert base.b < iv.mpf("3.917459").a < iv.mpf("3.918").a

print("E =", E)
print("2^E =", base)
print("verified: 2^E < 3.917459 < 3.918")
```

Appendix E. Proof of the compose-and-compress lemma

Proof of Lemma 15. We verify the partial-gadget properties one at a time. The raw product is a partial gadget by Lemma 14; compose-and-compress changes only its state layers and the way paths cross the deleted band.

State polynomial. The retained lower layers are the raw layers $0, \dots, \ell$, so they contribute $\sum_{j=0}^{\ell} a_j z^j$. A retained raw layer $j \geq \ell + s + 1$ is shifted down by s, so it contributes $a_j z^{j-s}$. Adding the two parts gives (20).

Soundness. Take any path in the compressed automaton. Every retained raw transition is also a transition of the raw product. For each compressed jump, choose one raw path that witnesses its existence and replace the jump by that path. The endpoints match by construction, so after all replacements we obtain one continuous path in the raw product. If a visible color appeared twice on the compressed path, it would still appear twice on the expanded raw path. This is impossible because the raw-product partial gadget accepts no word with a repeated color. Hence every compressed path is repetition-free.

Completeness below the cut. Let S be a certified product set of size $j \leq q$, and fix an arbitrary ordering $a_1 a_2 \cdots a_j$. When $j \leq \ell$, the raw accepting path has length at most ℓ. It stays entirely in the retained lower tail, so the same path exists in the compressed automaton.

Completeness across the cut. Now suppose $j > \ell$. The compressed path must cross from the lower tail to the upper tail after reading $a_1, \dots, a_\ell$. The deleted band has width s, so we need s additional colors to witness that crossing. By (P4), there is a certified set $T \supseteq S$ with $|T| = |S| + s$. Write $T \setminus S = \{h_1, \dots, h_s\}$; these colors are distinct and do not belong to S. By (P3), the raw product accepts every ordering of T. In particular, it accepts $a_1 \cdots a_\ell h_1 \cdots h_s a_{\ell+1} \cdots a_j$. After the prefix $a_1 \cdots a_\ell$, the next $s + 1$ raw transitions read $h_1 \cdots h_s a_{\ell+1}$. The definition of compose-and-compress therefore adds a jump with the same endpoints and visible label $a_{\ell+1}$. Replace those $s + 1$ raw transitions by that jump. The remaining suffix transitions are retained raw transitions. The resulting compressed path reads exactly $a_1 a_2 \cdots a_j$, because the colors $h_1, \dots, h_s$ are hidden rather than read.

Certified sets. The raw product has b_j certified sets of size j. The argument above shows that every such set remains readable when $j \leq q$. We discard the larger certified sets because the new capacity is q. This gives the truncation

in (21). Downward closure is preserved. Property (P4) is also preserved: if a retained certified set has size j and we request $h \le q - j$ additional colors, the product version of (P4) extends it to size $j + h \le q$, which is still retained.

Symmetry. For $0 \le j \le \ell$, the compressed layer of rank j has size a_j. Its mirror rank is $q - j$. That rank comes from raw rank $q - j + s = R - j$ and therefore has size $a_{R-j} = a_j$, using the symmetry of the raw product. Thus the compressed state profile is symmetric. □

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During preparation of this work, we used OpenAI's ChatGPT and Codex (models 5.6, 5.5, and 5.4 Pro) and Google's Gemini (model 3.1 Pro) to assist with proof verification, proof simplification, parameter optimization, literature survey, figure drawing, language editing, and organization. We independently reviewed and verified all mathematical arguments, citations, figures, code, and numerical calculations, edited the resulting material as needed, and takes full responsibility.

References

[1] R. Ben-Basat, A. Gabizon, M. Zehavi, The k-distinct language: Parameterized automata constructions, Theoretical Computer Science 622 (2016) 1–15. doi:10.1016/j.tcs.2016.01.029.

[2] R. Ben-Basat, Bounds on the size of the smallest NFA for $L_{k\text{-distinct}}$, Theoretical Computer Science Stack Exchange, question 20468; posted 6 January 2014; accessed 27 July 2026 (Jan. 2014). https://cstheory.stackexchange.com/questions/20468/bounds-on-the-size-of-the-smallest-nfa-for-l-k-distinct

[3] A. Molina Lovett, J. Shallit, Optimal regular expressions for permutations, in: 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), Vol. 132 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, pp. 121:1–121:12. doi:10.4230/LIPIcs.ICALP.2019.121.

[4] N. Alon, R. Yuster, U. Zwick, Color-coding, Journal of the ACM 42 (4) (1995) 844–856. doi:10.1145/210332.210337.

[5] J. Kneis, D. Mölle, S. Richter, P. Rossmanith, Divide-and-color, in: Graph-Theoretic Concepts in Computer Science, Vol. 4271 of Lecture Notes in Computer Science, Springer, 2006, pp. 58–67. doi:10.1007/11917496_6.

[6] I. Koutis, Faster algebraic algorithms for path and packing problems, in: Automata, Languages and Programming, Vol. 5125 of Lecture Notes in Computer Science, Springer, 2008, pp. 575–586. doi:10.1007/978-3-540-70575-8_47.

[7] R. Williams, Finding paths of length k in $O^*(2^k)$ time, Information Processing Letters 109 (6) (2009) 315–318. doi:10.1016/j.ipl.2008.11.004.

[8] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Narrow sieves for parameterized paths and packings, Journal of Computer and System Sciences 87 (2017) 119–139. doi:10.1016/j.jcss.2017.03.003.

[9] F. V. Fomin, D. Lokshtanov, F. Panolan, S. Saurabh, Efficient computation of representative families with applications in parameterized and exact algorithms, Journal of the ACM 63 (4) (2016) 29:1–29:60. doi:10.1145/2886094.

[10] M. Zehavi, Mixing color coding-related techniques, in: Algorithms – ESA 2015, Vol. 9294 of Lecture Notes in Computer Science, Springer, 2015, pp. 1037–1049. doi:10.1007/978-3-662-48350-3_86.

[11] J. Nederlof, Weighted k-path and other problems in almost $O^*(2^k)$ deterministic time via dynamic representative sets, in: Proceedings of the 2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS 2025), IEEE Computer Society, 2026, pp. 2813–2824. doi:10.1109/FOCS63196.2025.00143.

[12] R. Ben-Basat, Verified (c, k, k)-separating-family rows, GitHub repository, commit a7b41b15d4cb83226aa42a905f5f63364a8f5917; accessed 27 July 2026 (2026). https://github.com/ranbenbasat/verified-splitters-artifact/tree/a7b41b15d4cb83226aa42a905f5f63364a8f5917

[13] T. Beth, D. Jungnickel, H. Lenz, Design Theory, Volume I, 2nd Edition, no. 69 in Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1999. doi:10.1017/CBO9780511549533.

[14] M. Naor, L. J. Schulman, A. Srinivasan, Splitters and near-optimal derandomization, in: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1995, pp. 182–191. doi:10.1109/SFCS.1995.492475.

[15] D. Lokshtanov, A. Björklund, S. Saurabh, M. Zehavi, Approximate counting of k-paths: Simpler, deterministic, and in polynomial space, ACM Transactions on Algorithms 17 (3) (2021) 26:1–26:44. doi:10.1145/3461477.