Powers of Three and Structural Transport in the Collatz Map: Exact Window Distributions and Boundary Estimates

Y.Ueoka
Independent Researcher, Corresponding Author

Tsumugi (ChatGPT)
AI Co-researcher

29 November 2026 — Preprint, version 1.0

Abstract

We organize earlier ABN/CPE results on the formation and transport of finite structural patterns. For pure multiplication by powers of three, a fixed low-bit window has an exact periodic distribution, including geometric run lengths and explicit two-time correlations. For the accelerated Collatz map, we give a sharp information-transport law, a uniform one-step bound for joint run histograms, and an affine, restartable comparison with a moving window of the pure-power sequence. A sufficient condition for exact agreement above the two lowest mask cells is also recorded. Finite-state joint counts retain input-output dependence, and completed-interval identities express the actual structural balance in terms of pure-power profiles and exact boundary corrections. Full-period sampling, spatial statistics, and fixed-orbit time averages are distinct; no universal long-time transfer or convergence is asserted. We also present exact spatial CKC balance laws, finite-word transport estimates, and a deterministic calendar for a protected upper K-run. The results are finite identities, estimates and counting theorems, not a proof of convergence.

1 Notation and scope

For positive odd n let $F(n) = (3n+1)/2^{s(n)}$, $s(n) = \nu_2(3n+1)$. For every nonnegative integer n, the occupied-cell mask of canonical ABN is $E(n) = n \operatorname{XOR} 2n$, with bits $e_i = b_i \operatorname{XOR} b_{i-1}$ in low-to-high order, $b_{-1} = 0$. This mask, not the ordinary binary word of n, defines CPE: maximal occupied runs $C(r)$ and intervening zero runs $K(d)$. For every positive integer n (odd or even), write $B(n) = \text{bitlength}(n) + 1$, $H_C(n) = \text{wt}_2(E(n))$, $Z(n) = B(n) - H_C(n)$ and $\mu(n)$ for the number of C-runs. The full CPE word has two occupied endpoints only for odd n; the accelerated map F is used only on positive odd integers. Set $\text{bitlength}(0) = 0$, and leave $B(0)$ undefined. The empty mask has zero weight, zero runs and zero cells. For a mask word U, $z(U)$ counts zero cells directly in that word, without applying E again; it is distinct from $Z(U)$. For an arbitrary finite window, zeros outside the window are not counted as internal K-runs. For a nonnegative integer n, let $C_W(n)$ count starts of occupied runs within the lowest W bits of $E(n)$, resetting the bit before the window to zero. It need not equal $\mu(n)$.

The original formation-and-transport programme is due to Y.Ueoka. This paper complements Papers A–C [1, 2, 3] by collecting selected Phase 4 results A104–A106 and A109–A112 and Phase 5 M07/M09 identities. The original notes distinguish proofs, finite tests, and open transfer questions. We keep that distinction. In particular, the standard order of 3 modulo powers of 2 is used as an elementary input, not a novelty claim.

2 Pure powers: exact finite-window distributions

Theorem 2.1 (A110, Lemma 1 and Theorem 2)**.** *For $W \geq 3$, let $P_W = 2^{W-2}$. For any fixed positive odd n, the residues of $3^t n$ for $0 \leq t < P_W$ are exactly the residues congruent to n or $3n$*

modulo 8, each once. Conditional on either low-three-bit residue, the mask bits $e_3, \dots, e_{W-1}$ are independent fair bits under uniform sampling over the full period.

Proof. For $k \geq 1$, $\nu_2(3^{2^k} - 1) = k + 2$: the first case is $3^2 - 1 = 8$, and squaring increases the valuation by one since $3^{2^k} \equiv 1 \pmod 8$. The order divides the order of the odd residue group, a power of two, and is therefore exactly 2^{W-2}. Powers of 3 lie in the two classes 1 and 3 modulo 8, whose total size equals the order. Multiplication by n permutes the odd residues. Once b_2 is fixed, the upper bits are transformed bijectively by $e_i = b_i \text{ XOR } b_{i-1}$, with inverse $b_i = b_2 \text{ XOR } e_3 \text{ XOR} \cdots \text{XOR } e_i$. Thus every upper mask word occurs once in each class. □

Corollary 2.2 (A110, Corollary 4)**.** *In the fair part of the window, a completely bounded C/K word $0\,1^{r_1}0^{d_1}\cdots 0^{d_{k-1}}1^{r_k}0$ of total length $L + 2$ has frequency $2^{-(L+2)}$ at any fixed admissible position. Conditional on a C-run starting, the frequency of a run of length r is 2^{-r} whenever the terminating zero lies within the fair part. The analogous internal K-run formula follows by exchanging zero and one.*

Proof. A specified word of length $L + 2$ has the asserted probability by the preceding theorem. For a run of length r, divide $2^{-(r+2)}$ by the start probability $1/4$. This is a formula for complete runs, not a normalized distribution of truncated boundary runs. □

Theorem 2.3 (A110, Theorem 3)**.** *Let $C_W(t) = C_W(3^t n)$, with uniform full-period averaging denoted by $\mathbb{E}$. For $W \geq 3$,*

$$\mathbb{E}C_W = \begin{cases} (W+3)/4, & n \equiv 1, 3 \pmod 8, \\ (W+1)/4, & n \equiv 5, 7 \pmod 8. \end{cases}$$

For $W \geq 4$, $\mathrm{Var}(C_W) = (W-1)/16$. In particular $C_W/W \to 1/4$ in mean square under these period measures.

Proof. For low residues $1, 3, 5, 7$ the pair (C_3, e_2) equals $(1,0), (2,1), (1,1), (1,0)$ respectively. The two residues in each orbit class occur equally. For $i \geq 3$ let $X_i = e_i(1 - e_{i-1})$. Conditional on $e_2 = a$, $\mathbb{E}X_3 = (1-a)/2$, and $\mathbb{E}X_i = 1/4$ for $i \geq 4$. This gives the means. For $h = W - 3 \geq 2$, the conditional variances are $(1-a)/4$ for X_3 and $3/16$ for later terms. Adjacent covariances are $-(1-a)/8$ for (X_3, X_4) and $-1/16$ thereafter; nonadjacent terms are independent. The conditional variance sums to $(h+1)/16$. Conditional means in either class differ by $1/2$, so the total variance is $(h+2)/16$. For $W = 4$ directly the four equiprobable cases have variance $3/16$. The mean-square conclusion follows from the moments. □

The full count polynomial is also explicit. Put $T(z) = \left(\begin{smallmatrix} 1 & z \\ 1 & 1 \end{smallmatrix}\right)$ and $V_a^{(h)} = (T(z)^h(1,1)^\top)_a$. For the two residue classes it is respectively $zV_0^{(W-3)} + z^2V_1^{(W-3)}$ and $z(V_0^{(W-3)} + V_1^{(W-3)})$. Each new bit contributes z precisely when a run starts. The sum of coefficients is P_W.

3 Two-time structure of the pure-power orbit

Theorem 3.1 (A111, Theorems 1–2)**.** *The triangular mask map E extends to a bijective isometry of $\mathbb{Z}_2$ with $\nu_2(E(x) - E(y)) = \nu_2(x - y)$ for $x \neq y$. For $h \geq 1$ the first differing bit of $E(3^{t+h}n)$ and $E(3^t n)$ is at position*

$$\nu_2(3^h - 1) = \begin{cases} 1, & h \text{ odd}, \\ 2 + \nu_2(h), & h \text{ even}. \end{cases}$$

Proof. The first differing bit is unchanged by triangular XOR, whose inverse is cumulative XOR from the low end. Multiplication by an odd number preserves the valuation. For odd h, reduce modulo 4; for even $h = 2^k u$ with u odd, the geometric sum $(3^h - 1)/(3^{2^k} - 1)$ is odd, and the squaring calculation above applies. □

Theorem 3.2 (A111, Theorem 3)**.** *For $W \geq 4$ and $J = P_W/2$,*

$$\mathbb{E}[(C_W(t+J) - C_W(t))^2] = 1/2, \quad \mathrm{Cov}(C_W(t), C_W(t+J)) = (W-5)/16.$$

Proof. Only the highest bit in the width-W window changes under the half-period shift. Writing that bit as a and the preceding one as z, the difference is $(1-2a)(1-z)$, whose square is $1-z$. The preceding bit is fair, including $W = 4$ by the low-residue table. The two times have identical marginal distributions; hence the covariance is the variance minus half the mean squared difference. □

The correlation coefficient is $(W-5)/(W-1)$, tending to one. Full-period marginal uniformity does not imply independence between times. More generally, for $W \geq 4$ and $h \geq 1$, with $K = \min(W, \nu_2(3^h - 1))$, the shared low window contributes equally, and the remaining $W - K$ bits contain at most $\lceil (W-K)/2 \rceil$ starts. The same variance identity bounds the covariance below by $(W-1)/16 - \frac{1}{2}\lceil (W-K)/2 \rceil^2$ (A111, Corollary 4).

4 Transport and one-step structural comparison

Theorem 4.1 (A104, Theorem 1.1)**.** *Let $n \equiv m \pmod{2^K}$ be positive odd integers and $S_t = \sum_{i<t} s(F^i(n)) < K$. The first t division exponents coincide, and $F^t(n) \equiv F^t(m) \pmod{2^{K-S_t}}$. Their masks agree on these lowest $K - S_t$ bits. For $m = n + 2^K$ the next mask bit differs.*

Proof. At step j, induction gives agreement modulo 2^{K-S_j}; the next valuation is strictly smaller than that precision, so it is shared. Dividing gives precision $K - S_{j+1}$. For the specified pair the difference at step t is $3^t 2^{K-S_t}$, and the triangular XOR preserves the first differing bit. □

Theorem 4.2 (A109, Theorem 2)**.** *Let $H_k(x)$ be the histogram over ordered types of k consecutive maximal C-runs and the intervening K-runs. For any $k \geq 1$ and positive odd n,*

$$\|H_k(F(n)) - H_k(3n)\|_1 \leq \begin{cases} 3, & s(n) = 1, \\ 4, & s(n) \geq 2. \end{cases}$$

Proof. Write $v = E(3n) \gg s$, where $\gg$ denotes a right shift. The exact boundary identities of Paper A give $E(3n) = 1 + 2^s v$, $v_0 = 1$, and $E(F(n)) = v$ XOR 2. In low-to-high order the old word begins with $1\,0^{s-1}v$; the new word is v with bit 1 flipped. Beyond the first run affected by that flip, the run sequences agree. More explicitly, when $v_2 = 0$ take the common suffix at the first start at position at least 3 in v; when $v_2 = 1$ take the next start after the run containing position 2. If absent the suffix is empty. The numbers of C-runs preceding the common suffix are as follows (bits beyond support are zero).

(v_1, v_2)	old $s = 1$	new	old $s \geq 2$	new
$(0,0)$	1	1	2	1
$(0,1)$	2	1	3	1
$(1,0)$	1	1	2	1
$(1,1)$	1	2	2	2

A histogram entry starting in the common suffix has a matching entry of the same type. Each earlier start contributes at most one unmatched entry, irrespective of k. The indicated sums bound the norm. The bound 4 is attained for $k = 1, n = 13$ and $k = 2, n = 17$ in the original note. □

This is a comparison of one accelerated step with one pure multiplication, not a bound of $4t$ for iterates. Such an iteration would need a separate control on amplification by multiplication.

5 Restartable comparison with a moving window

Theorem 5.1 (A112, Theorem 1)**.** *At any starting odd $x = F^j(n)$, write $x_i = F^i(x)$ and $V_h = \sum_{i<h} s(x_i)$. For $h \geq 1$,*

$$2^{V_h} F^h(x) = 3^h x + A_h, \qquad A_h = \sum_{i<h} 3^{h-1-i} 2^{V_i}.$$

Here A_h is positive odd and $E(F^h(x)) = E(3^h x + A_h) \gg V_h$.

Proof. Iteration gives $A_{h+1} = 3A_h + 2^{V_h}$ with $A_0 = 0$. The summand with $i = 0$ is odd, and all others even. Left shifts commute with the mask operation: $E(2^v x) = 2^v E(x)$ for $v \geq 0$. General right shifts do not commute with E. □

Lemma 5.2 (A112, Lemma 2)**.** *For $z, a \geq 0$, let $d_H(u, v) = \mathrm{wt}_2(u \text{ XOR } v)$. Then $d_H(E(z + a), E(z)) \leq 2\,\mathrm{wt}_2(a)$. More precisely for $a = 2^j$ and $r = \nu_2(\lfloor z/2^j \rfloor + 1)$ the XOR difference is $2^j + 2^{j+r+1}$.*

Proof. Adding 2^j flips the consecutive binary interval $j, \ldots, j + r$, leaving its two boundaries as the only mask changes. Add each occupied binary digit of a successively and apply the triangle inequality. □

Theorem 5.3 (A112, Theorem 3 and Corollary 4)**.** *The full-mask distance between $E(2^{V_h} F^h(x))$ and $E(3^h x)$ is bounded by $2\,\mathrm{wt}_2(A_h) \leq 2\sum_{i<h} \mathrm{wt}_2(3^i)$. Let L be the width-W low window of $E(F^h(x))$, and P the width-W low window of $E(3^h x) \gg V_h$. Put $d = d_H(L, P)$. Then*

$$|C_W(L) - C_W(P)| \leq d, \quad \|G_\ell(L) - G_\ell(P)\|_1 \leq 2\ell d,$$

where in this formula C_W is applied directly to mask words (without applying E again) and G_ℓ counts all length-ℓ binary subwords, $1 \leq \ell \leq W$. Also $d \leq 2\,\mathrm{wt}_2(A_h)$.

Proof. Apply the lemma to the affine identity and use subadditivity of binary digit sum and invariance of Hamming distance under a common left shift. A common right shift and restriction to a finite window cannot increase Hamming distance. One flipped bit changes the run count by at most one and affects at most ℓ subword positions, each changing the histogram by at most two in norm. Sum over the d flips. □

The normalized histogram error is at most $4\ell\,\mathrm{wt}_2(A_h)/(W - \ell + 1)$. The universal budget $2\sum_{i<h} \mathrm{wt}_2(3^i)$ is independent of starting time and initial value and is $O(h^2)$, since $\mathrm{wt}_2(3^i) \leq 1 + i\log_2 3$. The sharper budget $2\,\mathrm{wt}_2(A_h)$ depends on the actual trajectory. Padding with many zeros is not evidence of meaningful structural agreement.

Theorem 5.4 (M09, safe-window identity (S1))**.** *If $R_h = A_h/2^{V_h} < 1$, set $U = E(3^h x) \gg (V_h + 2)$ and $y = F^h(x)$. Then*

$$U = E(y) \gg 2, \qquad E(y) = 4U + 3 - 2(\mathrm{wt}_2(U) \bmod 2).$$

Proof. The affine identity and $0 < A_h < 2^{V_h}$ give $\lfloor 3^h x/2^{V_h} \rfloor = y - 1$. For a shifted mask only its new bottom bit depends on a discarded binary bit, and so it agrees with the mask of the shifted integer above that bit. Since y is odd, $y - 1$ differs only at bit zero, hence their masks agree from bit two up. Finally $E(y)$ has lowest bit one and even total weight, fixing its second bit as $1 - (\mathrm{wt}_2(U) \bmod 2)$. □

The preceding proof is a fresh reconstruction of an existing identity, not a new priority claim. The condition is sufficient, not asserted necessary. Endpoint agreement is not a statement of individual run identity at every intermediate time.

Theorem 5.5 (M09, S2–S3: boundary-safe structural counts)**.** *Under the preceding safe-window hypothesis, write* $p = \mathrm{wt}_2(U) \bmod 2$*,* $a = U \bmod 2$*,* $c(U)$ *for the number of maximal occupied runs in* U *and* $z(U) = \mathrm{bitlength}(U) - \mathrm{wt}_2(U)$*, with all counts zero for* $U = 0$*. Then*

$$B(y) = \mathrm{bitlength}(U) + 2, \quad \mu(y) = c(U) + 1 - (1-p)a, \quad Z(y) = z(U) + p.$$

The number of boundary-independent complete C-runs, defined here as those starting strictly above bit 2 of $E(y)$*, is*

$$c_{\mathrm{full}} = c(U) - a = \mu(y) - 1 - \mathbf{1}_{\{y \equiv 3 \pmod 8\}}.$$

Each of these runs has identical endpoints in the aligned pure-power mask (shifted by V_h*). The number of consecutive CKC words formed by these runs is* $\max(c_{\mathrm{full}} - 1, 0)$*.*

Proof. The two lowest bits of $E(y)$ are $1, 1-p$ in low-to-high order, followed by U. The length, zero count and run count follow by concatenation, including the possible fusion at the cut. For a run starting at position strictly larger than 2, its preceding zero, occupied cells and terminating zero all belong to the common part of the two masks; its maximal endpoints therefore agree. These runs constitute a contiguous suffix of the ordered run list. If $a = 1$, the first run of U touches the cut and is excluded. From the first three ordinary binary bits of odd y, $pa = 1$ is equivalent to $y \equiv 3 \pmod 8$: indeed $p = 1$ means $e_1 = 0$, and $a = 1$ means $e_2 = 1$. Substitution gives the second count identity. Counting neighboring pairs in the suffix proves the CKC formula. □

A run starting at the cut might happen to agree on both sides; the count deliberately excludes it because agreement of the preceding bit is not guaranteed by the safe-window identity. These are endpoint matches, not a claim that tagged chains survive every intermediate operation.

6 Exact joint counts by finite-state transport

The complete-period measures in Section 2 are not the only useful distributions. A105–A106 count finite classes of initial words jointly with their actual accelerated outputs. The joint distribution, unlike separate input and output bounds, retains the dependence needed for a descent test.

Theorem 6.1 (A105, Lemma 2.1 and Theorem 2.2)**.** *Fix* $t \geq 1$*,* $L \geq 2$*, a set of specified mask positions* $I \subseteq \{0, \ldots, L-1\}$ *and prescribed bits* ϵ_i*. Let*

$$\mathcal{C} = \{n > 0 : \ n \text{ odd}, \ B(n) = L, \ e_i(n) = \epsilon_i \ (i \in I)\}.$$

Then $P^{(t)}_{L,I,\epsilon}(u, v) = \sum_{n \in \mathcal{C}} u^{\mu(n)} v^{\mu(F^t(n))}$ *is computable by a product of polynomial-weighted finite-state matrices with at most* $16 \cdot 8^t$ *states, independently of* L*.*

Proof. Read ordinary input bits b from low to high. One accelerated step keeps a state $(p, c, h) \in \{0,1\}^3$, initially $(0, 1, 0)$. Here p is the previous input bit, c the carry, and h indicates whether a nonzero output bit has appeared. Set

$$q = b + p + c, \qquad z = q \bmod 2, \qquad (p', c', h') = (b, \lfloor q/2 \rfloor, h \vee z).$$

Emit z if $h' = 1$, and otherwise emit nothing. Ordinary binary addition proves that the emitted word is the low-to-high binary expansion of $(3x+1)/2^{\nu_2(3x+1)}$, with zero padding. Cascade t such devices; a device receiving no bit does not update. There are at most 8^t cascade states. Store also the previous binary and previous mask bits at each of the input and final output, all initially zero, giving the four additional bits. Multiply a transition by u or v at each respective mask start $0 \to 1$ (at the output only when a bit is emitted).

For $L \geq 3$ the input consists of a 1, $L-3$ arbitrary bits, and a 1. For $L = 2$ it is the single bit 1. Append $2t + 5$ zeros. Two trailing zero inputs suffice to clear one device's previous bit and carry; because its input integer is positive, it has started emitting by that time, and thereafter

transmits every zero. Induction along the cascade clears all stages after at most $2t$ trailing input zeros; the remaining zeros count the final mask boundary. Reject transitions violating prescribed input mask bits, including the top mask bit processed during padding. Each allowed word thus traces exactly one path, contributing exactly its prescribed monomial. Addition of path weights is multiplication of the corresponding transfer matrices, with the initialized state and summation over terminal states. The exponents belong to the polynomial weights, not to the finite-state memory. □

Corollary 6.2 (A105, Theorem 3.1)**.** *For nonempty $\mathcal{C}$, every member has $\mu(F^t(n)) < \mu(n)$ if and only if every nonzero coefficient $[u^a v^b]P^{(t)}_{L,I,\epsilon}$ satisfies $b < a$. This can hold even when $\max b = \min a$ on the support.*

Proof. The support is exactly the set of realizable ordered pairs. For strictness of the comparison, take $L = 12$, $I = \{0, \ldots, 7\}$, $E(n) \equiv 69 \pmod{256}$ and $t = 1$. The class consists of $1219, 1475, 1731, 1987$ and the polynomial is $2u^4v^3 + u^5v^3 + u^5v^4$, as verified by direct integer evaluation. All pairs descend, yet the separated extremes are both 4. The original A105 also gives the same polynomial for $L = 14$, $E(n) \equiv 37 \pmod{1024}$ and $t = 2$. □

Theorem 6.3 (A106, Theorem 1 and Proposition 1)**.** *Fix a finite collection of C/K types $h_j(n)$, each counting a specified complete local run pattern, and $t \geq 1$. For $L \geq 3$,*

$$Q_{L,t}(u, v, \mathbf{w}) = \sum_{n>0 \text{ odd},\ B(n)=L} u^{\mu(n)} v^{\mu(F^t(n))} \prod_j w_j^{h_j(n)}$$

is exactly computable by a finite-state weighted matrix product whose state set is independent of L. In a nonempty class specified by $(\mu, \mathbf{h}) = (m, \mathbf{k})$ the largest b with $[u^m v^b \mathbf{w}^{\mathbf{k}}]Q_{L,t} > 0$ is the sharp finite-class upper bound on $\mu(F^t(n))$.

Proof. In the mask word padded by one zero on each side, $C(a)K(d)C(b)$ is counted by occurrences of $0\,1^a0^d1^b0$, including overlaps; a single complete $C(r)$ is counted by $0\,1^r0$. For any fixed complete alternating run pattern, concatenate its run bits and add on each side one bit opposite to the corresponding end bit. For example, $K(d)C(r)K(e)$ is counted by $1\,0^d1^r0^e1$, and $C(r)K(d)$ by $0\,1^r0^d1$. A single complete K-run is counted by $1\,0^d1$. Virtual end zeros delimit C-runs only, and never count as external K-runs. Store a suffix of length at most one less than the longest selected word, initialized by the virtual low zero, and multiply by w_j each time the corresponding pattern ends. Take the product of this finite memory with the preceding cascade. Padding zeros count the virtual upper zero; extra zeros cannot create spurious occurrences because each pattern contains an occupied cell and ends in at most one zero. If T_0, T_1 are the weighted transitions for each ordinary bit and α, β the initial and terminal vectors, the expression is

$$Q_{L,t} = \alpha T_1 (T_0 + T_1)^{L-3} T_1 T_0^{2t+5} \beta.$$

The state count depends on t and selected pattern lengths, not L. The coefficient records every input exactly once, so its maximum in the indicated nonempty fiber is attained and is the sharp upper bound. □

For clarity, the following finite enumeration from A106 uses $t = 2$. The five selected types are $\#C(1)$, $H_2(1,2,1)$, $H_2(1,1,1)$, $H_2(2,1,1)$, $H_2(1,1,2)$, where $H_2(a,d,b)$ counts $C(a)K(d)C(b)$. The last three columns count *integers* in classes certified to descend uniformly at time 2, not the number of classes.

L	Population	Actual descent	μ only	μ + 2 types	μ + 5 types
12	512	196	5	16	57
14	2048	809	7	29	139
16	8192	3339	7	183	478
18	32768	13516	9	544	1385
20	131072	54649	9	1779	4171

For $L = 20$, 650 of the 670 classes defined by the five types and μ still have more than one possible output count. This is not a defect in the bound: a coarse variable need not determine a unique output to certify a uniform inequality. Neither the fixed-t finite-state construction nor this finite population establishes a time-uniform closure or a fixed-orbit law. A new independent implementation in the research archive checks the transfer construction against direct enumeration and reproduces the A105/A106 tables.

7 Exact pure-power profiles over completed intervals

A complementary approach in M07 and M09 removes the accelerated map from the right-hand side of a finite structural balance, without a safety assumption. The data at the beginning of each interval still depend on the actual trajectory. For positive odd a set $\epsilon(a) = 1$ if $a \equiv 1 \pmod 4$ and -1 otherwise.

Lemma 7.1 (M07, L1 and endpoint identity)**.** *For $D \geq 1$ and positive odd a,*

$$E(2^D a - 1) = 1 + 2^D + 2^{D+1} E((a-1)/2),$$
$$Z(2^D a - 1) = D - 1 + Z(a) + \epsilon(a) - \mathbf{1}_{\{a=1\}}.$$

If $q > 1$ is odd, $v = \nu_2(q-1)$ and $y = (q-1)/2^v$, then $Z(y) = Z(q) - v + 1 + \epsilon(q)$.

Proof. Put $u = (a-1)/2$. The ordinary binary word of $2^D a - 1$ has D low ones followed by a zero and the bits of u. Taking its adjacent differences gives the first identity with disjoint supports. For $u > 0$ its mask has width $B(u) + D + 1$ and weight $H_C(u) + 2$; hence its zero count is $Z(u) + D - 1$. Also $E(2u+1) = (2E(u)) \,\mathrm{XOR}\, 3$, giving $Z(a) = Z(u) - 1 + 2(u \bmod 2) = Z(u) - \epsilon(a)$. For $a = 1$, the first formula is $1 + 2^D$, giving $Z = D - 1$ as required. For the endpoint, $q - 1 = 2^v y$; adding one flips ordinary bit zero alone and hence flips precisely mask bits 0 and 1. The weight changes by zero when $v = 1$ and by two otherwise, while the width of $q - 1$ is unchanged. Since $Z(q-1) = Z(y) + v$, the endpoint formula follows. □

Theorem 7.2 (M07, T1–T3: completed-interval balance)**.** *Let $D \geq 1$, a positive odd, $n = 2^D a - 1$, $q = 3^D a$, $v = \nu_2(q-1)$ and $y = (q-1)/2^v$. For $0 \leq k < D$,*

$$F^k(n) = 2^{D-k} 3^k a - 1, \qquad F^D(n) = y,$$

and the cumulative division exponent is $D + v$. The exact balance is

$$Z(y) - Z(n) = Z(3^D a) - Z(a) - (D-1) - |v-2| - \epsilon(a) + \mathbf{1}_{\{a=1\}}.$$

Proof. For $r \geq 2$ and odd c, $3(2^r c - 1) + 1 = 2(2^{r-1} 3c - 1)$ with odd parenthesis; induction gives the first $D - 1$ steps. The last step starts at $2 \cdot 3^{D-1} a - 1$ and has exponent $1 + v$. Apply the preceding lemma to the initial and final points. Since $\epsilon(q) = -1$ for $v = 1$ and $+1$ for $v \geq 2$, $1 - v + \epsilon(q) = -|v-2|$, giving the balance. □

Corollary 7.3 (M07.7b and M09c, T4–T6a)**.** *Starting at any fixed positive odd n_0, recursively set $D_j = \nu_2(n_j + 1)$, $a_j = (n_j + 1)/2^{D_j}$, $v_j = \nu_2(3^{D_j} a_j - 1)$ and $n_{j+1} = (3^{D_j} a_j - 1)/2^{v_j}$. For every $J \geq 1$, $n_J = F^{\sum_{j<J} D_j}(n_0)$, and*

$$Z(n_J) - Z(n_0) = \sum_{j<J} \big[Z(3^{D_j} a_j) - Z(a_j) - (D_j - 1) - |v_j - 2| - \epsilon(a_j) + \mathbf{1}_{\{a_j=1\}} \big].$$

Moreover $3 \mid a_{j+1}$ if and only if v_j is even.

Proof. Apply the interval theorem at each successive starting point and telescope. The recurrence gives $2^{v_j} n_{j+1} \equiv -1 \pmod 3$, so $n_{j+1} \equiv (-1)^{v_j+1} \pmod 3$. Since dividing $n_{j+1}+1$ by a power of two does not affect divisibility by three, the final assertion follows. □

Theorem 7.4 (M09R1c, U1–U6: synchronized profile)**.** *For positive odd b and integers $1 \le k < D$, the difference from the initial zero count of $n = 2^D b - 1$ is*

$$Z(F^k(n)) - Z(n) = Z(3^k b) - Z(b) - k + ((-1)^k - 1)\epsilon(b) + \mathbf{1}_{\{b=1\}}.$$

In particular the difference is independent of D in this range. If k is even and $b > 1$, strict Z-descent at k is equivalent to $Z(3^k b) - Z(b) < k$.

Proof. Apply the initial-point formula of the lemma to $2^{D-k}3^k b - 1$ and $2^D b - 1$, using $\epsilon(3^k b) = (-1)^k \epsilon(b)$ and $3^k b > 1$. The terms $D-1$ cancel. For even k and $b > 1$ the remaining correction vanishes. □

Corollary 7.5 (M09R1c, T6b–T6d: factor shift)**.** *Write $a = 3^f b$ with $3 \nmid b$. For $n = 2^D a - 1$ and $\tilde{n} = 2^{D+f} b - 1$ one has $F^f(\tilde{n}) = n$ and $F^{D+f}(\tilde{n}) = F^D(n)$. If $\Delta(D, a) = Z(F^D(n)) - Z(n)$, then*

$$\begin{aligned}\Delta(D, a) - \Delta(D+f, b) = f + Z(b) - Z(a) \\ + \epsilon(b) - \epsilon(a) + \mathbf{1}_{\{a=1\}} - \mathbf{1}_{\{b=1\}}.\end{aligned}$$

Proof. Use the interval trajectory formula on $\tilde{n}$ until time $f < D+f$. The endpoints are therefore equal. The difference of balances equals $Z(\tilde{n}) - Z(n)$, evaluated by the lemma. □

The profile and factor shift originate in Y.Ueoka's synchronized-parameter and factor-moving proposals; the M09 notes credit the research-thread ChatGPT for the derivations and checks. The interval length $D = \nu_2(n+1)$ is fixed by a given initial integer. Increasing D at fixed b changes the initial integer; it does not extend a fixed initial interval. In the telescoping sum, the successive a_j are not independent samples of the pure-power measure, and may be divisible by three. Factor shifts change the starting point and the balance by the displayed correction; independent replacements of starting points do not preserve the fixed-orbit telescoping sum without correction. These exact identities therefore do not by themselves imply eventual descent.

8 Spatial distributions and local interactions

The earlier R174/R178/R180/R181 studies distinguish spatial counting on a single mask from the complete-period measure of Section 2. They also identify the structural information lost when only marginal histograms are retained.

Theorem 8.1 (R180.1–2: pure multiplication)**.** *Extend the input mask e by zeros on both sides. Set $d_{-1} = d_0 = 0$. Then the mask e' of $3n$ satisfies, for $i \ge 0$,*

$$d_{i+1} = e_i(1 - d_i), \qquad e'_i = e_i \,\mathrm{XOR}\, d_{i-1} \,\mathrm{XOR}\, d_i.$$

Equivalently $e'_i = 1 - e_i$ if $e_{i-1} = 1$, and $e'_i = e_i$ if $e_{i-1} = e_{i-2} = 0$. In the remaining case $e'_i = e_i \,\mathrm{XOR}\, (r \bmod 2)$, where r is the length of the C-run ending at $i-2$.

Proof. For the ordinary addition $n + 2n$ let c_i be the incoming carry and y_i the output bit. Set $d_i = c_i \mathrm{XOR} b_{i-1}$. If $b_i = b_{i-1}$ then $c_{i+1} = b_i$, hence $d_{i+1} = 0$; otherwise $c_{i+1} = c_i$ and $d_{i+1} = 1 - d_i$. Since $y_i = b_i \,\mathrm{XOR}\, d_i$, the adjacent difference is the displayed formula. If the previous mask bit is 1, the two d bits differ; if the previous two are zero both are zero. Otherwise $d_i = 0$ and d_{i-1} is the parity accumulated along the immediately preceding C-run. These cases include the zero-padded boundaries. □

Theorem 8.2 (R174.3/R178.1: joint-distribution balance)**.** *Write the mask low to high as* $C(r_1)K(g_1)\cdots K(g_{\mu-1})C(r_\mu)$*, and suppose* $\mu \geq 2$*. Let* $M = \mu - 1$ *and* $\Pi(a,g,b)$ *be the empirical distribution of the ordered triples* (r_{i+1}, g_i, r_i)*. Put* $\pi(a,g,b) = 2^{-(a+g+b)}$*,* $V = \frac{1}{2}\|\Pi - \pi\|_1$*, and* $I = \mathbf{1}_{\{r_\mu=1\}}$*. Define*

$$f(a,g,b) = \mathbf{1}_{b=1} + \mathbf{1}_{(g=1,\ b\ even)\,or\,(g=2,\ b\ odd)} + \mathbf{1}_{g=1,\ b\ odd,\ a\geq 2}.$$

Then

$$\mu(3n) - \mu(n) = 1 - I + M \sum_{a,g,b\geq 1} f(a,g,b)(\pi - \Pi)(a,g,b), \qquad |\mu(3n) - \mu(n) - (1-I)| \leq 2MV.$$

Proof. Use the preceding local rule to count output starts. A C-run starting at s with length r has zero output on $s+1,\ldots,s+r-1$ and an occupied output at $s+r$. The output start at s is present for the first run, and for a later run precisely when the preceding gap has length at least 3, or length 2 after an even C-run. For gap 1 after an odd C-run, the output at s is zero; when $r = 1$, the start lost at s is recovered at $s+1$. The cell $s+r$ is a start for every $r \geq 2$. The second cell of a gap of length at least 2 can be occupied but is adjacent to $s+r$, and no later gap cell starts a run. The same count applies to the upper zero padding. Thus the run count is $2\mu - z - J - D$, where z counts length-one C-runs, J counts the middle event in f, and D counts its last event. In particular $z + J + D = I + M\sum f\Pi$. Under π, the three event weights are respectively $1/2, 1/3, 1/6$, by summing geometric series over the specified parities and lengths. Their sum is one. Substitution proves the identity. The middle and last events of f are disjoint, so $0 \leq f \leq 2$. The signed difference $\pi - \Pi$ has zero total mass and its positive and negative parts each have mass V. Hence $|\sum f(\pi - \Pi)| \leq 2V$, proving the stated bound. □

The reference π is a deterministic comparison distribution. No independent sampling of triples at a single time is assumed. If $V(3^t n) \to 0$, then $\mu(3^t n) \to \infty$ (a distribution supported on boundedly many points cannot approach the fixed infinite-support π in total variation), and the relative change of μ tends to zero. The premise is not proved here.

Theorem 8.3 (R178.3: C-to-K transport)**.** *Let* $L_C(n)$ *and* $L_K(n)$ *denote the largest C- and K-run lengths, with* $L_K = 0$ *if there is no K-run. Then*

$$Z(3n) \geq \sum_i (r_i - 1) + \sum_{i<\mu} (g_i - 1 - \mathbf{1}_{r_i\ odd})_+,$$

$$L_K(3n) \geq \max\{0, L_C(n) - 1, L_K(n) - 2\}.$$

Consequently for $j \geq 1$*,* $L_K(3^j n) \geq \max\{0, L_C(n) - (2j-1), L_K(n) - 2j\}$.

Proof. The local rule forces all but the first cell of each input C-run to zero. In each internal gap all but the first cell and, if the preceding run is odd, the second cell are guaranteed zero. These intervals are disjoint and internal to the output support: the low bit is one and the width increases under multiplication by three. Summing gives the first bound, and taking the largest guaranteed interval gives the second. Iterating the latter bound proves the final claim. □

Theorem 8.4 (R180.3 and R181 correction: finite-window tail error)**.** *Fix* $L \geq 1$ *and in the local rule for* e' *replace the parity of a preceding run of length* $r > L$ *by zero. Call the resulting finite bit word* $\tilde{e}$*. It is determined at each position by the* $L+3$ *input bits ending there. Put* $h_L = \#\{i : r_i > L\}$*. Then*

$$d_H(e', \tilde{e}) \leq h_L, \quad \|N_C(e') - N_C(\tilde{e})\|_1 \leq 3h_L, \quad |\mu(3n) - \mu(\tilde{e})| \leq h_L.$$

The internal K-run histogram has the same $3h_L$ *bound; each individual C-run count differs by at most* $2h_L$*. When both C-run counts are positive the normalized C-run distributions differ in total variation by at most* $\min(1, 2h_L/\mu(3n))$.

Proof. Only the cell two positions above the end of an input C-run can be wrong, and only if its length exceeds L. The $L+3$-bit window detects whether the preceding run exceeds L and determines its length otherwise. Each bit flip splits, joins, changes or deletes at most two old runs and creates at most two new runs, with at most three total removals and insertions; the number of runs changes by at most one. This applies also to internal zero runs, with the exterior excluded. Sum these bounds over the differing positions. For normalized C histograms, bring both distributions to denominator $\mu(3n)$; the numerator of the ℓ^1 error is at most $3h_L + h_L$. No integer interpretation of the approximate word is needed. □

The correction in R181 is essential: for $n = 11, L = 1$, the exact word of 33 is 1100011 (low to high), but the approximation is 110001 after removing high padding (the R181 note printed 100011; its odd-weight conclusion is unchanged). The latter has three occupied cells and is not a canonical positive-integer ABN mask. It cannot be iterated as if it were such a mask.

Theorem 8.5 (R178.2/R180.4: marginal and triple-count nonclosure)**.** *The separate C- and K-run histograms do not determine even $\mu(3n)$. The complete CKC triple-count histogram together with the two marginal histograms does not determine the output C-run length histogram.*

Proof. For $11, 13$ the input C-length multiset is $\{1, 3\}$ and the K-length multiset is $\{1\}$; direct masks give $\mu(33) = 2$ and $\mu(39) = 3$. For $246727, 246663$, both inputs have six C(1) runs, with gaps $(2, 2, 3, 3, 3)$ and $(2, 3, 2, 3, 3)$ respectively. Thus all triple counts agree. Computing $E(740181)$ and $E(739989)$ gives respectively C-length multisets $\{9, 3, 3, 3\}$ and $\{6, 6, 3, 3\}$. The difference is the arrangement of consecutive C(1)K(2) patterns, not a discrepancy in the count of isolated triples. □

Theorem 8.6 (R181.1–2: spatial word counts and run distributions)**.** *For $k \geq 1$ let $P_{n,k}$ be the distribution of the $B(n)$ length-k words ending at positions $0, \ldots, B(n) - 1$, with low zero padding. Write $\Delta_k(n) = \frac{1}{2}\|P_{n,k} - 2^{-k}\|_1$, where 2^{-k} is the uniform word distribution. Fix $R \geq 1$. Let $p_C^{(R)}, p_K^{(R)}$ be the normalized run distributions in bins $1, \ldots, R, > R$, and $\pi^{(R)} = (2^{-1}, \ldots, 2^{-R}, 2^{-R})$. If $\Delta_{R+2} \leq \epsilon < 1/4$, then*

$$\mathrm{TV}(p_C^{(R)}, \pi^{(R)}) \leq \frac{3\epsilon + B^{-1}}{2(1/4 - \epsilon)}, \quad \mathrm{TV}(p_K^{(R)}, \pi^{(R)}) \leq \frac{3\epsilon}{2(1/4 - \epsilon)}.$$

Proof. The start rates are $P_{n,2}(01) = \mu/B$ and $P_{n,2}(10) = (\mu - 1)/B$, each within ϵ of $1/4$ by marginalization. The counts divided by B for C-length $r \leq R$ are $P_{n,r+2}(01^r0)$ plus the top-run correction $B^{-1}\mathbf{1}_{r_\mu = r}$; for K-length r they are $P_{n,r+2}(10^r1)$. The tail bins equal $P_{n,R+2}(01^{R+1})$ and $P_{n,R+2}(10^{R+1})$. The corresponding suffix events are disjoint within each family and under uniform input have total weight $1/4$ with individual weights $2^{-(r+2)}$ and $2^{-(R+2)}$. Coarse graining has ℓ^1 error at most 2ϵ. Add at most $1/B$ for the C top correction, then at most ϵ to normalize from mass $1/4$ to the observed start rate. Divide by twice that rate to obtain the bounds. Both denominators are positive under the hypothesis. □

Theorem 8.7 (R181.3–4: deterministic transport of spatial words)**.** *For $k, L \geq 1$ and $\tau = B(3n) - B(n) \in \{1, 2\}$,*

$$\Delta_k(3n) \leq \Delta_{k+L+2}(n) + \frac{kh_L}{B(n)} + \frac{\tau}{B(3n)} \leq (k+1)\Delta_{k+L+2}(n) + k2^{-(L+2)} + \frac{2}{B(n)+1}.$$

For every k, $\Delta_k(n) \geq \max(0, 1 - B(n)/2^k)$.

Proof. The approximate rule is the current bit XOR a function of the preceding $L+2$ bits. For a length-$k+L+2$ input word, fixing its first $L+2$ bits makes the remaining k bits map bijectively to the approximate output k-word. Thus uniform words map to uniform words; total variation cannot increase under this map. Compare the approximate output on the original $B(n)$ ending

positions with the exact output: h_L possible changes affect at most kh_L words. Including the τ new positions changes the empirical distribution by at most $\tau/B(3n)$. Finally the tail count identity above gives $h_L/B = P_{n,L+2}(01^{L+1}) \le 2^{-(L+2)} + \Delta_{k+L+2}(n)$. The support-size bound follows since at most $B(n)$ distinct words are observed. The transport bound requires a longer input window and is not a contraction on a fixed set of word statistics. □

Theorem 8.8 (ABT branch P42-B: fixed-width ensemble)**.** *For $B = b+1$ (thus b is the ordinary binary width), $b \ge 2$, and $Z = k \ge 1$, the number of positive odd integers with exactly $j \ge 1$ K-runs is*

$$\binom{k-1}{j-1}\binom{b-k}{j} \quad \text{if } k \equiv b-1 \pmod 2,$$

and zero otherwise, with out-of-range binomial coefficients zero. For any nonempty such (b,k) class the mean K-run count under uniform sampling of its integers is $k(b-k)/(b-1)$. For $k=0$ there is one integer if b is odd and none otherwise.

Proof. The mask has width $b+1$, two occupied endpoints, and an even occupied-cell count. Each such word determines a unique odd integer by inverse adjacent XOR. Composing the k zeros into j positive gaps and the $b+1-k$ ones into $j+1$ positive runs gives the product. The parity condition follows from the even occupied count. For the mean, choose k positions uniformly among the $b-1$ interior cells. A gap starts at the first interior position if it is zero, or at a later position if it is zero and the preceding position is one. Summing the probabilities gives $k/(b-1)+k(b-1-k)/(b-1)$ (for $b=2$ only the first term is present). The all-occupied word yields the last statement. This is an ensemble of initial integers, not a fixed-orbit time measure. □

Theorem 8.9 (Phase 4 R85, R112 correction; Phase 5 M08)**.** *Fix $T \ge 1$, set $h = \lceil \log_2(3^T) \rceil$, $s = \lceil \log_4(3^T) \rceil$, $L_C = 2s+2$, $L_K = h+1$. Replace each C-run length $r \ge L_C$ by $L_C + ((r-L_C) \bmod 2)$ and each K-run length g by $\min(g, L_K)$, obtaining the canonical mask of $\bar{n}$. For $1 \le t \le T$,*

$$H_C(3^t n) = H_C(3^t \bar{n}), \quad B(3^t n) = B(3^t \bar{n}) + B(n) - B(\bar{n}),$$

$$Z(3^t n) - Z(n) = Z(3^t \bar{n}) - Z(\bar{n}) + H_C(n) - H_C(\bar{n}).$$

For fixed μ, T the number of representative masks is at most $(L_C+1)^\mu L_K^{\mu-1}$.

Proof. Parity-preserving shortening gives an even occupied count and preserves the occupied endpoints, so the representative is canonical. For multiplication by $m = 3^t$ the binary input b_i and carry $c_i \in [0, m-1]$ give $y_i = (mb_i + c_i) \bmod 2$ and $c_{i+1} = \lfloor (mb_i + c_i)/2 \rfloor$. On a constant input of zeros or ones the carry after j cells is respectively $\lfloor c/2^j \rfloor$ or $m - 1 - \lfloor (m-1-c)/2^j \rfloor$. Thus after h cells the carry is fixed and after $h+1$ the last output bit is also fixed; the latter extra cell is the R112 clarification. On an alternating pair of input bits, let $u = b_i + 2b_{i+1} \in \{1,2\}$ and $q = m/3$. After j pairs the carry is $uq + \lfloor (c - uq)/4^j \rfloor$. After s pairs this is either $uq - 1$ or uq, both fixed by the pair update, emitting respectively 11 or 00 on each subsequent pair. The extra pair makes the preceding output bit agree as well.

A K-run corresponds to constant ordinary input bits and a C-run to alternating bits (including the closing transition to upper zero padding). For a saturated representative K-run, insert copies of the constant input immediately after its first h cells, before its next existing cell. The carry is fixed, and each inserted output equals the next existing output bit. For a saturated representative C-run, fix the phase of the first s input pairs, and insert copies of that same pair immediately after these $2s$ cells, before the next existing pair. Such a pair exists because the representative length is at least $2s+2$. At that point the carry is a fixed point of $c \mapsto \lfloor (mu + c)/4 \rfloor$; every inserted output pair equals the next existing constant pair. The output transitions, including those on either side of the insertion, are therefore unchanged. The later input phase is unchanged, even if the C-run has an unpaired final cell. This insertion also works in a top C-run: it precedes the last input transition, so all subsequent input and output cells are translated by the inserted length.

In either type of run the highest occupied ordinary output bit is above the highest occupied input bit because $m \geq 3$. It lies in the translated suffix, not the inserted block, and the output width consequently increases by exactly the inserted length. Apply these insertions to the representative to recover n. The two identities follow since H_C counts output transitions including the closing one, and subtraction using $Z = B - H_C$ proves the third. Counting allowed positive lengths gives the representative bound. □

The M08 correction $H_C(n) - H_C(\bar{n})$ must not be dropped when C-runs are shortened. In the completed-interval balance of Section 7, take $T = D$ and apply this identity to a; the valuation v and sign $\epsilon(a)$ still refer to the original a, not the representative. The compression concerns pure multiplication on a fixed finite horizon, not equality of accelerated future integers or a single finite representative system for all times.

9 Finite protection of an upper pure-power structure

The following Phase 6 results concern a particular upper block separated by a long K-run. They complement the lower-window transport and will also be relevant to coupled dynamics.

Theorem 9.1 (P6-386–388: protected upper mask)**.** *Let a, b, T be integers with $T \geq 0$, and $n_0 = 2^a - \delta_0$, where δ_0 is odd, $0 < \delta_0 \leq 2^b$, $a - b = d + 1$, $b \geq 0$, $d \geq 1$. Suppose $3^T 2^b < 2^a$. For $0 \leq t \leq T$, put $n_t = F^t(n_0)$, $S_t = \sum_{j<t} s(n_j)$ and $k_t = a - S_t$. Then $n_t = 3^t 2^{k_t} - \delta_t$, $0 < \delta_t < 2^{k_t}$, $k_t \geq 1$, and for $t < T$,*

$$s(n_t) = \nu_2(3\delta_t - 1), \quad \delta_{t+1} = (3\delta_t - 1)/2^{s(n_t)}, \quad 2^{S_t}\delta_t \leq 3^t \delta_0.$$

For $w \geq 1$ such that $3^t 2^b \leq 2^{a-w}$, the w ordinary bits directly below k_t are all one and

$$E(n_t) \gg k_t = U_t := E(3^t - 1) + 1.$$

In particular this last equality holds up to $t = \lfloor d/\log_2 3 \rfloor$ independently of the lower structure, after alignment by k_t.

Proof. Induct on t: the lifted deficit satisfies $2^{S_{t+1}}\delta_{t+1} = 3 \cdot 2^{S_t}\delta_t - 2^{S_t}$, so the upper bound holds. Indeed $0 < (3\delta_t - 1)2^{S_t} < 3^{t+1}\delta_0 < 2^a$ for $t < T$, proving that the valuation of $3\delta_t - 1$ is strictly below k_t and is the valuation of $3n_t + 1$. Division gives the displayed recurrence and positive odd deficit. At time t, write $n_t = (3^t - 1)2^{k_t} + (2^{k_t} - \delta_t)$. The inequality on w implies $\delta_t \leq 2^{k_t - w}$ (and $w \leq k_t$ since the deficit is positive), hence the top w bits of the lower term are one. The coefficient $3^t - 1$ is even; adjacent XOR therefore gives bit one at k_t, and above it exactly the shifted mask of that coefficient. Take $w = 1$ and $3^t \leq 2^d$ for the final assertion. The lower trajectory and absolute positions still depend on δ_0. □

Theorem 9.2 (P6-409–410: calendar of a specified K-run)**.** *For $t \geq 1$, let $\ell(t)$ be the number of zeros immediately above bit zero of U_t before the next occupied bit. Then*

$$\ell(t) = \begin{cases} 0 & t \text{ odd}, \\ 1 + \nu_2(t) & t \text{ even}. \end{cases}$$

For $D \geq 2$, $T \geq 1$, set $M = \lfloor T/2^{D-1} \rfloor$. Then

$$\#\{1 \leq t \leq T : \ell(t) \geq D\} = M,$$

$$\sum_{\substack{t \leq T \\ \ell(t) \geq D}} \ell(t) = (D+1)M - \mathrm{wt}_2(M),$$

$$\sum_{t=1}^{T} \ell(t) = T - \mathrm{wt}_2(T) + \lfloor T/2 \rfloor.$$

Proof. The first occupied bit of $E(3^t-1)$ is at $\nu_2(3^t-1)$, since shifting does not change the lowest occupied bit under XOR. The valuation formula of Section 3 gives 1 for odd t and $2+\nu_2(t)$ for even t. Adding 1 creates the new low endpoint, so the intervening length is as stated. The threshold times are $2^{D-1}u$ for $1 \le u \le M$ and their lengths are $D+\nu_2(u)$. The identity $\sum_{u=1}^{M}\nu_2(u) = \sum_{j\ge1}\lfloor M/2^j\rfloor = M - \mathrm{wt}_2(M)$ follows by summing the binary expansion. It yields both sums (the empty sum is zero). On the protected part of an actual orbit the upper-mask theorem transfers this specific run, not all births or subsequent lifetimes of K-runs. □

10 Limits and source provenance

For $x=1$, $F^t(1)=1$, $V_t=2t$, while $3^t < 2^{2t-1}$ for $t \ge 3$. Thus $E(3^t) \gg V_t = 0$. Moreover $\mu(3^t)$ is unbounded: in an arbitrarily wide fixed window, the pure-power period theorem realizes an arbitrarily large number of separated starts, recurring at multiples of that period. These are the A112 obstruction statements. They do not contradict exact fixed-window period distributions.

The existing research does not infer full-width spatial statistics at one large time from complete-period finite-window sampling, nor long-time statistics of the accelerated orbit from pure multiplication. The original goal is to understand structural formation and transport; this paper preserves established finite identities and bounds without asserting convergence. More general coupled-block results remain in the source archive. Historical priority is not claimed.

Research provenance and contributions. Y.Ueoka originated the ABN/CPE programme and the pure-power formation and transport approach, including the selection of structural observables, the simultaneous input–output viewpoint in A111, the synchronized parameters and factor shifts in M09, and the protected upper structure and subsequent isolation proposals in R53/R59. The original derivations and checks belong to the respective historical ChatGPT research sessions, as recorded in their notes; the Gray correspondence was independently identified in a separate ChatGPT conversation. Tsumugi (ChatGPT) is the AI co-researcher responsible for the present proof reconstructions, compilation, corrections, verification and exposition, not a retrospective name for all those sessions. The original M07/M09 derivations retain their recorded attribution wherever a particular contributor is not identified in the surviving consolidation. The independent ChatGPT referee of R34 proposed the phase-explicit insertion argument in Theorem 8.9 and the general pattern guards in Theorem 6.3, and suggested domain and estimate clarifications. The authors acknowledge these contributions. Y.Ueoka is the human corresponding author responsible for submission and correspondence.

Source concordance. The result labels in each theorem identify the original propositions. The following table supplies their archival locations and the manuscript result numbers; physical PDF page numbers are one-based. These are preserved project research records, not independently verified public editions. The standalone A112 archive is dated 12 November 2026, the M09 R2bj handoff 19 November 2026. The ABT P47 and Phase 6 R270 archives are identified by their recorded revision labels. No unverified Zenodo version is assigned to these results.

Results	Original labels	Archive location
2.1–2.3	A110	A112 cumulative PDF, 879–881
3.1–3.2	A111	same, 882–884
4.1	A104	same, 859–861
4.2	A109	same, 876–878
5.1–5.3	A112	same, 885–887; standalone A112 TeX
5.4–5.5	M09 S1–S3	M09 R2bj MAIN NOTE TeX, sec. 4
6.1–6.2	A105	A112 cumulative PDF, 863–864
6.3	A106	same, 865–867
7.1–7.5	M07/M09	M09 R2bj MAIN NOTE TeX, secs. 2–3b
8.1	R180.1–2	A112 cumulative PDF, 469–470
8.2	R174.3/R178.1	same, 454–455, 465–466
8.3	R178.3	same, 466
8.4	R180.3/R181	same, 469–471
8.5	R178.2/R180.4	same, 466, 470
8.6–8.7	R181.1–4	same, 471–472
8.8	ABT P42-B	ABT P47, P42 appendix, sec. 440
8.9	R85/R112/M08	A112 PDF, 230–231, 274; M09 R2bj TeX
9.1	P6-386–388	Phase 6 R270 main TeX, R53
9.2	P6-409–410	Phase 6 R270 main TeX, R59

The complete script listings in Appendix A preserve the executable checks of the selected results. An independent ChatGPT review of the R34 manuscript found no major mathematical defect and recommended five clarifications; these were implemented in R35. The final edition preserves the numbered statements and proofs of R35. The review is not formal verification or human peer review.

11 Related work and scope of the comparisons

The run rewriting approach of Sinyor [4], building on the earlier binary-run work of Błażewicz and Pettorossi, and the binary algorithm of Kaufman [5] are relevant antecedents to the local updates. Paper A explains the relation between ordinary binary runs and the CPE coordinates. The present paper uses these coordinates to expose run distributions, input–output correlations, boundary corrections and finite-horizon structural balances; it does not claim that the underlying Collatz map or the use of a binary scan is new.

Finite-state descriptions also precede this work. Caucal and Rispal [6] describe transducers for the nonaccelerated Collatz function and explicit composition constructions using suitable bases. Section 6 instead fixes the number of odd-to-odd operations, removes low output zeros as part of each stage, and uses polynomial weights to retain joint CPE observables. The cited constructions and the weighted count here have different outputs and counting objectives; a full equivalence or minimal-state comparison is not asserted.

The 2-adic conjugacy of the one-division Collatz map to the binary shift [7] is background for congruence and finite-information transport. The mask E used here is a triangular adjacent-bit transform, not that conjugacy. Dupuy and Weirich [8] study average digit equidistribution for ordinary base expansions of powers. The exact complete-period measure of Section 2 is a fixed low-window statement for the ABN support mask, and is distinct from spatial digit statistics of an entire large integer. These references motivate comparison, not a claim that the finite-window results establish the corresponding long-time or global spatial laws. Historical priority for individual bounds remains open to a more comprehensive search.

References

[1] Y. Ueoka and Tsumugi (ChatGPT), Alternating Binary Notation and Collatz Phase Expression: Canonical Representations and Exact Structural Dynamics, https://aixiv.science/abs/aixiv.261128.000003.

[2] Y. Ueoka and Tsumugi (ChatGPT), Finite-Time Structural Descent in the Accelerated Collatz Map, https://aixiv.science/abs/aixiv.261128.000004.

[3] Y. Ueoka and Tsumugi (ChatGPT), Witness-Preserving Gluing and Variational Induction in the Collatz Problem, https://aixiv.science/abs/aixiv.261128.000005.

[4] J. Sinyor, The $3x + 1$ Problem as a String Rewriting System (2010), https://doi.org/10.1155/2010/458563.

[5] R. Kaufman, A Reduced Forward Collatz Algorithm: How Binary Strings Change Their Length Under $3x + 1$ (2023), https://arxiv.org/abs/2301.07466.

[6] D. Caucal and C. Rispal, On the Powers of the Collatz Function, MCU 2024, LNCS 15270 (2025), 115–129; author manuscript, https://www-igm.univ-mlv.fr/~caucal/Accepte.pdf.

[7] D. J. Bernstein and J. C. Lagarias, The $3x + 1$ Conjugacy Map, Canadian Journal of Mathematics 48 (1996), 1154–1169, https://doi.org/10.4153/CJM-1996-060-x.

[8] T. Dupuy and D. E. Weirich, Bits of 3^n in binary, Wieferich primes and a conjecture of Erdős, Journal of Number Theory 158 (2016), 268–280, https://doi.org/10.1016/j.jnt.2015.05.022.

A Reproducible finite checks

The four scripts below use Python 3.10 or later and the standard library only. Run each separately with assertions enabled (without the `-O` option), for example `python verify_r31.py`. The original A112 script also writes `A112_RESULTS.json` in its own directory. The scripts are printed for independent reconstruction; the PDF also embeds the original plain-text files. Wrapped listing lines are continuations of the same numbered source line. Finite checks supplement the analytic proofs, not replace them. Counts can overlap and should not be added as distinct integers.

Script	Replay highlights
A112	6,750 additions; 23,400 segments; 538,200 local-pattern checks.
R31	80 full periods; 81,237 histogram comparisons; 15,262 safe windows.
R32	120 joint-DP comparisons; all eight A105 rows and five A106 populations; 8,192 completed intervals and 126,976 interior times.
R33	8,692 input masks; 34,768 tail checks; 23,903 compression times; 94,095 protected upper times; calendar and ensemble checks.

`CHECK_A112.py`

```
#!/usr/bin/env python3
"""Independent integer audits for A112. Pure integer operations; no sympy."""
from __future__ import annotations
from collections import Counter
import json, random
from pathlib import Path

ROOT=Path(__file__).resolve().parent

def E(n):
    assert n >= 0
    return n^(n<<1)

def nu2(n):
    assert n>0
    return (n&-n).bit_length()-1

def orbit_segment(n,h):
    assert n>0 and n%2
    x=n; S=0; A=0
    for _ in range(h):
        A=3*A+(1<<S)
        s=nu2(3*x+1)
        x=(3*x+1)>>s
        S+=s
    assert (x<<S)==3**h*n+A
    return x,S,A

def C(z,W):
    z &= (1<<W)-1
    return (z & ~(z<<1)).bit_count()

def histogram(word,W,ell):
    return Counter((word>>i)&((1<<ell)-1) for i in range(W-ell+1))

def l1_hist(a,b):
    return sum(abs(a[k]-b[k]) for k in a.keys()|b.keys())

def check_power_add():
    count=0
    for x in range(450):
```

```
        for j in range(15):
            z=(x>>j)+1
            r=nu2(z)
            diff=E(x+(1<<j))^E(x)
            assert diff==(1<<j)|(1<<(j+r+1)),(x,j,hex(diff))
            count+=1
    return count

def check_segments():
    rng=random.Random(112112)
    cases=[i for i in range(1,800,2)]+[rng.getrandbits(128)|1 for _ in range(120)]
    seg_count=pattern_count=0
    maxima={}
    for n in cases:
        current=n
        for start in range(5):
            if start:
                current,_,_=orbit_segment(current,1)
            for h in range(1,10):
                y,S,A=orbit_segment(current,h)
                U=3**h*current
                b=E(U+A)
                p=E(U)
                assert b==E(y)<<S
                defects=(b^p).bit_count()
                budget=2*A.bit_count()
                coarse=2*sum((3**r).bit_count() for r in range(h))
                assert defects<=budget<=coarse,(n,start,h,defects,budget,coarse)
                assert abs(C(b,b.bit_length()+1)-C(p,p.bit_length()+1))<=defects
                maxima[h]=max(maxima.get(h,0),defects)
                for W in (3,5,9,21,49):
                    actual=E(y)&((1<<W)-1)
                    pure=(E(U)>>S)&((1<<W)-1)
                    d=(actual^pure).bit_count()
                    assert d<=defects
                    assert abs(C(actual,W)-C(pure,W))<=d
                    for ell in (1,2,3,4,5):
                        if ell>W:continue
                        error=l1_hist(histogram(actual,W,ell),histogram(pure,W,ell))
                        assert error<=2*ell*d,(n,start,h,W,ell,error,d)
                        pattern_count+=1
                seg_count+=1
    return seg_count,pattern_count,maxima

def check_obstruction():
    rows=[]
    for t in range(1,73):
        x,S,A=orbit_segment(1,t)
        assert x==1 and S==2*t
        pure=E(3**t)>>S
        if t>=3: assert pure==0 and E(x)==3
        if t in (1,2,3,4,8,16,32,64,72):
            rows.append({'t':t,'S_t':S,'B_3t':(3**t).bit_length()+1,'shifted_pure':pure,'
    true_ABT':E(x),'global_hamming':(E(3**t+A)^E(3**t)).bit_count()})
    return rows

def check_no_uniform_pure_mu():
    # For any K (tested here K<=8) search one residue-period timestamp with
    # >= K ABT one-runs in E(3**t) mod 2^W, then periodic recurrence.
    rows=[]
    for K in range(2,9):
        W=4*K+3
        # constructive CRT: find eligible low-bit word on reachable orbit under powers of 3.
        # Use t through period for moderate W; for K>=5, avoid huge enumeration by
```

```
        # synthesize target 3^t as any residue 1 or3 mod8 and note subgroup theorem.
        if K<=4:
            P=1<<(W-2)
            found=None
            x=1
            mask=(1<<W)-1
            for t in range(P):
                if C(E(x),W)>=K:
                    found=t;break
                x=(x*3)&mask
            assert found is not None
            rows.append({'K':K,'W':W,'one_witness_t':found,'period':P})
        else:
            # E(3^t) base has e0=1,e1=1,e2=0 at even t; free bits e3..e_(W-1)
            # exist in subgroup orbit and can realize at least K starts.
            rows.append({'K':K,'W':W,'one_witness_t':'exists by subgroup transitivity','period':1<<(W-2)})
    return rows

if __name__=='__main__':
    result={'plus_power_checks':check_power_add()}
    seg, pats, maxima=check_segments()
    result.update(segment_checks=seg,pattern_checks=pats,largest_observed_ABT_Hamming_by_h=maxima,obstruction_n1=check_obstruction(),fixed_K_periodic_existences=check_no_uniform_pure_mu())
    (ROOT/'A112_RESULTS.json').write_text(json.dumps(result,ensure_ascii=False,indent=2)+'\n',encoding='utf8')
    print(json.dumps({k:v for k,v in result.items() if k not in ('obstruction_n1','fixed_K_periodic_existences')},ensure_ascii=False,indent=2))
```

verify_r31.py

```
from collections import Counter
from itertools import groupby
from random import Random
from fractions import Fraction

def E(n): return n^(n<<1)
def v2(n): return (n&-n).bit_length()-1
def F(n): return (3*n+1)>>v2(3*n+1)
def c(w): return (w & ~(w<<1)).bit_count()
def runs(n): return tuple(len(list(g)) for _,g in groupby(bin(E(n))[2:][::-1]))
def hist(a,k): return Counter(a[i:i+2*k-1] for i in range(0,len(a)-2*k+2,2))
periods=0; samples=0
for W in range(3,13):
    P=1<<(W-2); M=(1<<W)-1
    for n in range(1,16,2):
        x=n&M; seq=[]; words=[]
        for _ in range(P):
            words.append(x); seq.append(c(E(x)&M)); x=3*x&M
        assert len(set(words))==P and all(x%8 in (n%8,3*n%8) for x in words)
        mean=Fraction(sum(seq),P); assert mean==Fraction(W+(3 if n%8 in (1,3) else 1),4)
        if W>=4:
            var=sum((x-mean)**2 for x in seq)/P
            assert var==Fraction(W-1,16)
            assert Fraction(sum((x-seq[(i+P//2)%P])**2 for i,x in enumerate(seq)),P)==Fraction(1,2)
        periods+=1; samples+=P
print('periods',periods,'timestamps',samples,'PASS')
checks=0
for n in range(1,1<<15,2):
    a,b=runs(3*n),runs(F(n)); cap=3 if v2(3*n+1)==1 else 4
    for k in range(1,max(len(a),len(b))//2+2):
        x,y=hist(a,k),hist(b,k); assert sum((x-y).values())+sum((y-x).values())<=cap
```

```
        checks+=1
print('A109 histogram checks',checks,'PASS')
rng=Random(31); safe=0; total=0
for n in [*range(1,2048,2),*(rng.getrandbits(128)|1 for _ in range(512))]:
    y=n; V=A=0
    for h in range(1,21):
        A=3*A+(1<<V); V+=v2(3*y+1); y=F(y); total+=1
        assert (y<<V)==3**h*n+A
        if A<1<<V:
            U=E(3**h*n)>>(V+2)
            assert U==E(y)>>2 and E(y)==4*U+3-2*(U.bit_count()%2)
            safe+=1
print('M09 safe windows',safe,'of',total,'PASS')
```

verify_r32.py

```
from collections import Counter, defaultdict
from random import Random
from itertools import groupby

def E(n): return n^(n<<1)
def mu(n):
    w=E(n); return (w & ~(w<<1)).bit_count()
def Z(n): return E(n).bit_length()-E(n).bit_count()
def v2(n): return (n&-n).bit_length()-1
def F(n): return (3*n+1)>>v2(3*n+1)
def eps(n): return 1 if n%4==1 else -1
def iterate(n,t):
    for _ in range(t): n=F(n)
    return n

def dp(L,t,fixed=None,patterns=()):
    fixed=fixed or {}; k=max((len(p)-1 for p in patterns),default=0)
    initial=(((0,1,0),)*t,0,0,0,0,'0' if k else '')
    counts={(initial,(0,)*(2+len(patterns))):1}
    choices=[(1,)] if L==2 else [(1,)]+[(0,1)]*(L-3)+[(1,)]
    choices += [(0,)]*(2*t+5)
    for i,opts in enumerate(choices):
        nxt=defaultdict(int)
        for (state,exponents),num in counts.items():
            stages,bi,ei,bo,eo,suffix=state
            for b in opts:
                e=b^bi
                if i in fixed and e!=fixed[i]: continue
                x=b; new=list(stages)
                for j,(p,c,h) in enumerate(stages):
                    if x is None: break
                    z=x+p+c; bit=z%2
                    new[j]=(x,z//2,int(h or bit))
                    x=bit if h or bit else None
                de=int(e==1 and ei==0); df=0; bnext=bo; enext=eo
                if x is not None:
                    bnext=x; enext=x^bo; df=int(enext==1 and eo==0)
                word=suffix+str(e)
                inc=(de,df)+tuple(int(word.endswith(p)) for p in patterns)
                nxt[(tuple(new),b,e,bnext,enext,word[-k:] if k else ''),tuple(a+d for a,d in
    zip(exponents,inc))]+=num
        counts=nxt
    out=Counter()
    for (state,power),num in counts.items():
        assert all(p==c==0 and h==1 for p,c,h in state[0])
        out[power]+=num
    return out

```

```
def direct(L,t,fixed=None,patterns=()):
    fixed=fixed or {}; out=Counter()
    for n in range((1<<(L-2))|1,1<<(L-1),2):
        w=E(n)
        if any((w>>i)&1!=e for i,e in fixed.items()): continue
        word='0'+bin(w)[2:][::-1]+'0'
        out[(mu(n),mu(iterate(n,t)))+tuple(sum(word.startswith(p,i) for i in range(len(word)))
     for p in patterns)]+=1
    return out

rng=Random(32)
patterns=('010','010010','01010','011010','010110')
checks=0
for L in range(2,12):
    for t in (1,2,3):
        for fixed in ({}, {i:rng.randrange(2) for i in rng.sample(range(L), min(3,L))}):
            for pats in ((),patterns):
                assert dp(L,t,fixed,pats)==direct(L,t,fixed,pats)
                checks+=1
print('DP vs integer enumeration',checks,'PASS',flush=True)
for L,K,t,w in ((12,8,1,69),(14,10,2,37)):
    fixed={i:(w>>i)&1 for i in range(K)}
    assert dp(L,t,fixed)==Counter({(4,3):2,(5,3):1,(5,4):1})
print('A105 two strict joint-count examples PASS',flush=True)
for L,K,t,expected in ((12,8,1,(128,16,7)),(12,8,2,(128,10,8)),(14,10,1,(512,70,34))
    ,(14,10,2,(512,57,47)),(14,10,3,(512,66,58)),(16,10,1,(512,17,2)),(16,10,2,(512,25,4))
    ,(16,10,3,(512,22,8))):
    g=defaultdict(list)
    for n in range((1<<(L-2))|1,1<<(L-1),2): g[E(n)% (1<<K)].append((mu(n),mu(iterate(n,t))))
    actual=(len(g),sum(all(b<a for a,b in x) for x in g.values()),sum(max(b for a,b in x)<min(
    a for a,b in x) for x in g.values()))
    assert actual==expected,(L,K,t,actual,expected)
print('A105 all 8 original cohort rows PASS',flush=True)
for L,expected in ((12,(512,196,5,16,57)),(14,(2048,809,7,29,139)),(16,(8192,3339,7,183,478))
    ,(18,(32768,13516,9,544,1385)),(20,(131072,54649,9,1779,4171))):
    q=direct(L,2,patterns=patterns)
    verified=[]
    for p in (0,2,5):
        groups=defaultdict(lambda:[0,True,set()])
        for (a,b,*freq),num in q.items():
            g=groups[(a,*freq[:p])]; g[0]+=num; g[1]&=b<a; g[2].add(b)
        verified.append(sum(g[0] for g in groups.values() if g[1]))
    actual=(sum(q.values()),sum(num for (a,b,*_),num in q.items() if b<a),*verified)
    assert actual==expected,(L,actual,expected)
    if L==20: assert (len(groups),sum(len(g[2])>1 for g in groups.values()))==(670,650)
    print('A106',L,actual,'PASS',flush=True)

def starts(w):
    bits=bin(w)[2:][::-1] if w else ''
    pos=0; out=[]
    for b,g in groupby(bits):
        r=len(list(g))
        if b=='1': out.append((pos,pos+r-1))
        pos+=r
    return out
safe=0
for n in [*range(1,2048,2),*(rng.getrandbits(128)|1 for _ in range(512))]:
    y=n; A=V=0
    for h in range(1,21):
        A=3*A+(1<<V); V+=v2(3*y+1); y=F(y)
        if A>=1<<V: continue
        P=E(3**h*n); U=P>>(V+2); w=E(y); p=U.bit_count()%2; a=U&1
        c=len(starts(U)); z=U.bit_length()-U.bit_count()
        assert (w.bit_length(),mu(y),Z(y))==(U.bit_length()+2,c+1-(1-p)*a,z+p)
```

```
        interior=[(l-2,r-2) for l,r in starts(w) if l>2]
        other=[(l-V-2,r-V-2) for l,r in starts(P) if l>V+2]
        assert interior==other and len(interior)==c-a==mu(y)-1-int(y%8==3)
        safe+=1
print('S2/S3 safe windows',safe,'PASS',flush=True)
blocks=profiles=shifts=0
for a in list(range(1,256,2))+[(rng.getrandbits(128)|1) for _ in range(128)]:
    for D in range(1,33):
        n=(a<<D)-1; q=3**D*a; v=v2(q-1); y=(q-1)>>v
        assert E(n)==1+(1<<D)+(E((a-1)//2)<<(D+1))
        assert Z(n)==D-1+Z(a)+eps(a)-int(a==1)
        x=n
        for k in range(1,D):
            x=F(x); assert x==(3**k*a<<(D-k))-1
            assert Z(x)-Z(n)==Z(3**k*a)-Z(a)-k+((-1)**k-1)*eps(a)+int(a==1)
            profiles+=1
        assert F(x)==y and Z(y)-Z(n)==Z(q)-Z(a)-(D-1)-abs(v-2)-eps(a)+int(a==1)
        assert ((y+1)%3==0)==(v%2==0)
        b=a; f=0
        while b%3==0: b//=3; f+=1
        nn=(b<<(D+f))-1
        assert iterate(nn,f)==n and iterate(nn,D+f)==y
        assert Z(nn)-Z(n)==f+Z(b)-Z(a)+eps(b)-eps(a)+int(a==1)-int(b==1)
        shifts+=1; blocks+=1
print('M07/M09 completed blocks',blocks,'interior times',profiles,'factor shifts',shifts,'PASS
    ',flush=True)
```

verify_r33.py

```
from collections import Counter, defaultdict
from itertools import groupby
from math import comb
from fractions import Fraction as Q
from random import Random

def E(n): return n^(n<<1)
def runs(w): return [len(list(g)) for _,g in groupby(bin(w)[2:][::-1])] if w else []
def Z(w): return w.bit_length()-w.bit_count()
def v2(n): return (n&-n).bit_length()-1
def mu(w): return len(runs(w)[::2])
def diff(a,b): return sum(abs(a[k]-b[k]) for k in a.keys()|b.keys())
def local(w,L=None):
    e=[(w>>i)&1 for i in range(w.bit_length()+2)]; o=0; r=0; prev=0
    for i,x in enumerate(e):
        p=e[i-1] if i else 0
        pp=e[i-2] if i>1 else 0
        if p: y=1-x
        elif pp: y=x^(prev%2 if L is None or prev<=L else 0)
        else: y=x
        o|=y<<i
        if x: r+=1
        elif p: prev=r; r=0
    return o

def spatial(w,k,B=None):
    B=w.bit_length() if B is None else B
    mask=(1<<k)-1
    # encode low-to-high words in ordinary binary integer orientation (bijection)
    return Counter((w>>(i-k+1) if i>=k-1 else w<<(k-1-i))&mask for i in range(B))
def delta(w,k):
    B=w.bit_length(); c=spatial(w,k); N=1<<k
    return Q(sum(abs(x*N-B) for x in c.values())+(N-len(c))*B,2*B*N)
def hist(w):
    a=runs(w); return Counter(a[::2]),Counter(a[1::2])
```

```
def compress(w,T):
    h=(3**T).bit_length(); s=(h+1)//2; lc=2*s+2; lk=h+1
    a=runs(w); a=[min(r,lc+((r-lc)%2)) if i%2==0 else min(r,lk) for i,r in enumerate(a)]
    return a

def from_runs(a):
    b=0; n=0; j=0
    for i,r in enumerate(a):
        for _ in range(r):
            b^=int(i%2==0); n|=b<<j; j+=1
    assert b==0
    return n
rng=Random(33); ns=list(range(1,1<<14,2))+[(rng.getrandbits(rng.randrange(16,257))|1) for _ in range(500)]
for n in ns:
    w,y=E(n),E(3*n); a=runs(w); r,g=a[::2],a[1::2]
    assert local(w)==y
    d=sum(x==1 for x in r)+sum((gap==1 and b%2==0) or (gap==2 and b%2==1) for b,gap in zip(r,g))+sum(gap==1 and b%2==1 and c>=2 for b,gap,c in zip(r,g,r[1:]))
    assert mu(y)==2*len(r)-d
    assert Z(y)>=sum(x-1 for x in r)+sum(max(0,gap-1-b%2) for b,gap in zip(r,g))
    assert max(runs(y)[1::2],default=0)>=max(max(r)-1,max(g,default=0)-2,0)
    for L in (1,2,4,8):
        z=local(w,L); H=sum(x>L for x in r); C,K=hist(y); c,k=hist(z)
        assert (y^z).bit_count()<=H and diff(C,c)<=3*H and diff(K,k)<=3*H
        assert abs(mu(y)-mu(z))<=H and all(abs(C[j]-c[j])<=2*H for j in C.keys()|c.keys())
print('pure multiplication, joint balance, transport, tail approximation:',len(ns),'inputs;',4*len(ns),'tail checks PASS',flush=True)
assert sorted(runs(E(11))[::2])==sorted(runs(E(13))[::2]) and mu(E(33))==2 and mu(E(39))==3
assert sorted(runs(E(740181))[::2])==[3,3,3,9] and sorted(runs(E(739989))[::2])==[3,3,6,6]
assert bin(local(E(11),1))[2:][::-1]=="110001"
assert local(E(11),1).bit_count()==3
print('nonclosure examples and noncanonical approximation PASS',flush=True)
sp=0; rb=0
for n in ns[::8]:
    w,y=E(n),E(3*n); B=w.bit_length(); r,g=runs(w)[::2],runs(w)[1::2]
    for k,L in ((1,1),(2,2),(3,4),(4,6)):
        H=sum(x>L for x in r); D=delta(w,k+L+2)
        assert delta(y,k)<=D+Q(k*H,B)+Q(y.bit_length()-B,y.bit_length())
        assert Q(H,B)<=Q(1,2**(L+2))+D
        assert delta(w,k)>=max(0,1-Q(B,2**k)); sp+=1
    for R in (1,2,3):
        eps=delta(w,R+2)
        if eps>=Q(1,4): continue
        ref=[Q(1,2**i) for i in range(1,R+1)]+[Q(1,2**R)]
        for rr,cor in ((r,Q(1,B)),(g,0)):
            assert rr
            p=[Q(rr.count(i),len(rr)) for i in range(1,R+1)]+[Q(sum(x>R for x in rr),len(rr))]
            assert sum(abs(a-b) for a,b in zip(p,ref))/2 <= (3*eps+cor)/(2*(Q(1,4)-eps)); rb+=1
print('spatial transport',sp,'run-distribution bounds',rb,'PASS',flush=True)
count=0
for b in range(2,15):
    c=Counter((Z(E(n)),mu(E(n))-1) for n in range((1<<(b-1))|1,1<<b,2))
    for k in range(b):
        for j in range(b):
            expect= int(b%2==1 and j==0) if k==0 else (comb(k-1,j-1)*comb(b-k,j) if 1<=j<=min(k,b-k) and k%2==(b-1)%2 else 0)
            assert c[k,j]==expect
        num=sum(c[k,j] for j in range(b))
        if num: assert Q(sum(j*c[k,j] for j in range(b)),num)==Q(k*(b-k),b-1)
    count+=sum(c.values())
print('P42 fixed-width ensemble',count,'integers PASS',flush=True)
cc=0
```

```
for n in ns[::4]:
    for T in (1,2,3,5):
        nbar=from_runs(compress(E(n),T)); D=E(n).bit_length()-E(nbar).bit_length()
        for t in range(1,T+1):
            a,b=E(3**t*n),E(3**t*nbar)
            assert a.bit_count()==b.bit_count() and a.bit_length()-b.bit_length()==D
            cc+=1
print('R85/M08 compression',cc,'time comparisons PASS',flush=True)
ct=0
for _ in range(3000):
    b=rng.randrange(0,100); d=rng.randrange(1,100); a=b+d+1; delta0=rng.randrange(1,1<<b,2) if
     b else 1
    n=(1<<a)-delta0; y=n; S=0; deficit=delta0; p=1; t=0
    while p<=1<<d:
        k=a-S; assert y==p*(1<<k)-deficit and 0<deficit<1<<k
        assert E(y)>>k==E(p-1)+1
        ct+=1
        if 3*p>1<<d: break
        ss=v2(3*y+1); assert ss==v2(3*deficit-1)
        y=(3*y+1)>>ss; deficit=(3*deficit-1)>>ss; S+=ss; p*=3; t+=1
print('protected upper masks',ct,'times PASS',flush=True)
ell=[]
for t in range(1,2049):
    u=E(3**t-1)+1; l=v2(u-1)-1; ell.append(l); assert l==(1+v2(t) if t%2==0 else 0)
for T in (1,2,5,31,100,511,1024,2048):
    assert sum(ell[:T])==T-T.bit_count()+T//2
    for D in range(2,14):
        M=T//(1<<(D-1)); a=[l for l in ell[:T] if l>=D]
        assert len(a)==M and sum(a)==(D+1)*M-M.bit_count()
print('upper K-run calendar 2048 times, 96 threshold tests PASS',flush=True)
```