Coupled Collatz Dynamics in Alternating Binary Notation: Lifted Trajectories and Exact Branching Laws

Y.Ueoka
Independent Researcher, Corresponding Author

Tsumugi (ChatGPT)
AI Co-researcher

29 November 2026 — Preprint, version 1.0

Abstract

We preserve selected earlier results on lifted accelerated Collatz trajectories and their decomposition into signed powers of two. Prescribed division histories yield an affine evolution, whereas actual trajectories agree only within an exactly characterized common prefix. We give the first-branch direction and the first nonzero correction to two-trajectory comparison. Two separated families also give exact finite-time representations of powers of two and three. Negative-cell deletion is governed by an exact matching law, and canonical pair extraction by explicit branching and reunion criteria. Mirrored upper blocks, two contact conventions, signed separation and restart balances are also included. Prescribed contact heights, exact time-injection cancellation and extraction information are included. These results concern finite-time arithmetic and do not establish the Collatz conjecture. This is Paper E of the ABN/CPE research series.

1 Scope and notation

For positive odd n, let $F(n) = (3n+1)/2^{s(n)}$, $s(n) = \nu_2(3n+1)$, $s_j(n) = s(F^j(n))$ and $S_T(n) = \sum_{j<T} s_j(n)$. Here ν_2 denotes the exponent of 2 in a nonzero integer. For any nonzero integer x define $H(x) = 3x + 2^{\nu_2(x)}$. The image is nonzero because the odd part of x cannot be $-1/3$. For a positive integer exponent sequence $s = (s_0, s_1, \ldots)$, put $S_j(s) = \sum_{i<j} s_i$ and define $G(x, 0, s) = x$, $G(x, j+1, s) = 3G(x, j, s) + 2^{S_j(s)}$ for every integer x. Empty sums vanish. In G the exponent sequence is a prescribed input, not an assertion about the trajectory of x.

For positive odd n, canonical ABN consists of alternating signed powers of two, with highest sign positive and lowest sign negative: $n = \sum_{i=1}^{2m} (-1)^{i-1} 2^{k_i}$, $k_1 > \cdots > k_{2m} = 0$. For every nonnegative integer n, define the occupied-cell mask $E(n) = n \text{ XOR } 2n$, with $E(0) = 0$ and all bits above the ordinary width equal to zero; its occupied and intervening zero runs define CPE, not the ordinary binary run decomposition. We use the canonical representation from Paper A. A signed decomposition need not be canonical. The present paper preserves the Phase 5 and Phase 6 coupled-trajectory programme rather than restarting proof exploration. Here and throughout, iteration counts and bit positions are integers; F^0, H^0 are identity maps on their stated domains, and $S_0 = 0$. We use $\text{bit}_i(x) = \lfloor x/2^i \rfloor \bmod 2$ for nonnegative x and $i \geq 0$, with the additional convention $\text{bit}_{-1}(x) = 0$. In all count formulas $N(x) = \text{popcount}(E(x))$ is the number of occupied ABN cells, also denoted $H_C(x)$ in Papers A–D; it is not the lifted map H.

Notation guide. The same symbol m may be a local integer index or count, as specified in each section. For nonnegative x the main width and count conventions are:

Symbol	Meaning
$E(x)$	Occupied-cell mask x XOR $2x$; not the ordinary binary word.
$N(x) = H_C(x)$	Number of occupied ABN cells; $N(0) = 0$.
$\beta(x)$	Ordinary binary width; $\beta(0) = 0$.
$B(x)$	Mask width $\beta(x) + 1$ for $x > 0$; $B(0) = 0$.
$Z(x), m(x)$	Number of zero cells and internal zero-runs, respectively, of the mask of a positive odd x.
Γ	Band adjacency graph in Section 5; distinct from the mask E.
g_t, d_t	Ordinary-block gap (Section 9) and signed mask gap (Section 10).
J_t, δ_t	Contact clock and its increment (Section 11).

2 Lifted and prescribed dynamics

Theorem 2.1 (P6-1–4)**.** *For positive odd n and $T \geq 0$,*

$$H^T(n) = 2^{S_T(n)} F^T(n), \qquad \nu_2(H^T(n)) = S_T(n).$$

For nonzero integer x and $k \geq 0$, $H^T(2^k x) = 2^k H^T(x)$. For any prescribed sequence,

$$G(x, T, s) = 3^T x + \sum_{j<T} 3^{T-1-j} 2^{S_j(s)}.$$

In particular $G(n, T, s(n)) = H^T(n)$.

Proof. Induct using $H(2^{S_T} F^T(n)) = 2^{S_T}(3F^T(n)+1)$. The scaling identity follows from $\nu_2(2^k x) = k+\nu_2(x)$, first for one step and then by iteration. The explicit expression for G follows by induction from its recurrence. □

Proposition 2.2 (P6-5–9, P6-31–33)**.** *For any integer decomposition $n = b + A$ of a positive odd n,*

$$H^T(n) = 3^T A + G(b, T, s(n)).$$

In particular A may be a sum of signed two-cell terms $\pm 2^u \pm 2^v$, $u > v \geq 0$. If distinct occupied positions of the canonical ABN are extracted, with their original signs, then $\nu_2(A)$ equals the least extracted position, for a nonempty extraction. If b is positive odd and $A \neq 0$, the common-prefix criterion in Theorem 3.1 applies with $v = \nu_2(A)$.

Proof. For fixed s, subtract the two explicit expressions for G at $b + A$ and b. The sum of the extracted cells divided by its lowest power of two is odd, since only the lowest term contributes an odd summand. The final assertion is Theorem 3.1; in particular, it concerns a finite actual prefix, not the prescribed history of an arbitrary signed remainder. □

For example, $103 = (2^7 - 2^5) + (2^3 - 1)$. Inserting a zero pair changes the displayed decomposition but does not automatically give a canonical extraction. In particular a smaller positive remainder need not share the full exponent history of the original integer.

Proposition 2.3 (P6-9a–d)**.** *Let $n = a2^K + b$ with $a, K \geq 1$ integers and b positive odd. For all $T \geq 0$,*

$$G(b, T, s(n)) = 2^{S_T(n)} F^T(n) - 3^T a 2^K.$$

Thus $2^K \mid G(b, T, s(n))$ if and only if $S_T(n) \geq K$. When $S_T(n) \geq K$, put $Q_T = G(b, T, s(n))/2^K$; then

$$Q_T \equiv -3^T a \pmod{2^{S_T(n)-K}}.$$

If $F^J(n) = 1$ and $S_J(n) \geq K$, then for $t \geq 0$,

$$Q_{J+t} = 2^{S_J(n)+2t-K} - 3^{J+t} a.$$

Proof. The first identity is the preceding proposition. If $S_T < K$, the first term has valuation S_T while the second is divisible by 2^K; their difference cannot be divisible by 2^K. If $S_T \geq K$, division yields the congruence. Under the final hypothesis the subsequent odd states equal 1 and each exponent equals 2, yielding the last formula. The final formula is conditional on reaching 1. □

3 Exact agreement and first branching

Theorem 3.1 (P6-10)**.** *For distinct positive odd x, y let $v = \nu_2(x - y) \geq 1$. Then*

$$(s_j(x) = s_j(y) \text{ for all } 0 \leq j < T) \quad \Longleftrightarrow \quad S_T(y) < v.$$

Equivalently the length-T exponent word of y is shared exactly by the positive odd integers in the residue class

$$x \equiv y \pmod{2^{S_T(y)+1}}.$$

This residue formulation also includes $x = y$ (without evaluating $\nu_2(0)$). During agreement, $2^{S_j(y)}(F^j(x) - F^j(y)) = 3^j(x - y)$ for $j \leq T$.

Proof. Subtract the two recurrences while their exponents agree. If $S_T(y) < v$, then at step $j < T$ the difference of the two numerators has valuation $v - S_j(y) > s_j(y)$, so the numerator valuations coincide. This proves agreement inductively. Conversely, agreement through T steps gives the displayed difference identity at $j = T$; both terminal states are odd, so $S_T(y) + 1 \leq v$. For distinct inputs this is equivalent to the displayed congruence; equality of inputs gives both agreement and congruence directly. □

Theorem 3.2 (P6-11, P6-21–25)**.** *With the preceding notation let $J = \min\{T \geq 1 : S_T(y) \geq v\}$, $\delta = v - S_{J-1}(y)$, and $r = s_{J-1}(y)$. Then $1 \leq J \leq v$, and the first differing index is $J - 1$, where*

$$s_{J-1}(x) = \begin{cases} \delta, & r > \delta, \\ q > \delta, & r = \delta. \end{cases}$$

Define $\Delta_T = H^T(x) - H^T(y) - 3^T(x - y)$. Then

$$\Delta_T = \sum_{j<T} 3^{T-1-j}(2^{S_j(x)} - 2^{S_j(y)}), \quad \Delta_{T+1} = 3\Delta_T + 2^{S_T(x)} - 2^{S_T(y)}.$$

In particular $\Delta_T = 0$ for $T \leq J$, while

$$\Delta_{J+1} = 2^{S_J(x)} - 2^{S_J(y)} \neq 0, \quad \nu_2(\Delta_{J+1}) = \min(S_J(x), S_J(y)).$$

Proof. The cumulative exponents increase by at least one. The preceding theorem determines the first differing index. At that step the numerator difference has valuation δ, while the numerator for y has valuation $r \geq \delta$. Unequal valuations give the smaller one; equal valuations give a strictly larger valuation upon addition. Subtract the two formulas for G using the two actual histories to obtain the difference sum and recurrence. The terms through $j = J - 1$ vanish, and the next is a nonzero difference of powers of two, with valuation equal to the smaller exponent. □

For $x = 103, y = 7$, $v = 5$ and the first four exponents are $(1, 1, 2, 1)$ and $(1, 1, 2, 3)$; $J = 4$. For $x = 17, y = 1$, $J = 2$ and $\Delta_3 = 16$. The identity $H^T(n) = 3^T(n - b) + H^T(b)$ is not true in general: for $n = 3, b = 1, T = 2$ the two sides are 32 and 34. This is a limitation of replacing a prescribed history by the remainder's actual history, not a failure of affine evolution with a fixed history.

4 Separated finite-time families

Theorem 4.1 (P6-26–27)**.** *For $T \geq 0$ and integer $M \geq \max(1, 2T)$ let $U_T = H^T(2^M + 1)$, $V_T = H^T(2^M - 1)$, $W_T = H^T(1)$. Then*

$$U_T = 3^T 2^M + 4^T, \quad V_T = 3^T 2^M - 2^T, \quad W_T = 4^T,$$

so $3^T = (U_T - W_T)/2^M$ and $2^T = U_T - V_T - W_T$.

Proof. $H^t(1) = 4^t$ and $H^t(-1) = -2^t$. For $0 \leq t < T$ we have $2t < M$ and $t < M$. Thus the valuations of $3^t 2^M + 4^t$ and $3^t 2^M - 2^t$ are respectively $2t$ and t. Applying H yields the asserted expressions at the next time, including the possible final boundary equality $2T = M$. Induct from $t = 0$ and subtract the resulting identities. □

Proposition 4.2 (P6-28–30)**.** *For an integer $n = \sum_{j=0}^{d} c_j 2^j$ with integer coefficients and $M \geq \max(1, 2d)$,*

$$n = \sum_{j=0}^{d} c_j (U_j - V_j - W_j).$$

For $n = \sum_{j=0}^{e} d_j 3^j$ and $M \geq \max(1, 2e)$, $n = 2^{-M} \sum_{j=0}^{e} d_j (U_j - W_j)$. Also for $M \geq \max(1, 2T)$ and positive odd x, y,

$$H^T(x) - H^T(y) = 2^{-M}(x - y)(U_T - W_T) + \Delta_T.$$

Proof. Substitute the two power identities into the expansions and the defining equation of Δ_T. □

These are finite-time representations. The separation parameter depends on the time horizon; no infinite-time separation at fixed M follows. The identities do not provide a new convergence statement by themselves.

5 Canonical negative-cell removal

For any positive integer z, write the ordinary binary one-runs as $[m_i, p_i - 1]$, indexed upwards, $1 \leq i \leq r$. Their endpoint pairs give the canonical ABN

$$z = \sum_{i=1}^{r} (2^{p_i} - 2^{m_i}), \qquad 0 \leq m_1 < p_1 < m_2 < \cdots < m_r < p_r.$$

Write $N(z) = \text{popcount}(E(z)) = 2r$. Here r counts ordinary binary one-runs, not CPE occupied runs. The endpoint representation is unique by uniqueness of binary expansion. Define $D_{m_i}(z) = z + 2^{m_i}$, i.e. remove a specified negative cell of the original canonical word and then canonicalize.

Lemma 5.1 (P6-34–39, P6-50–51)**.** *For a positive integer x let $k = \nu_2(x)$ and let ℓ be the second-lowest occupied position in the canonical ABN of $3x$. Then $H(x) = D_k(3x)$ and $\nu_2(H(x)) = \ell > k$. For any higher negative position $m > k$ of $3x$, $\nu_2(D_m(3x)) = k$. In particular, for positive odd x, $s(x) = \ell$. If n, b are positive odd and $A = n - b$, then $H(n) = 3A + H(b)$, whereas the identity with actual $H^T(b)$ in place of prescribed $G(b, T, s(n))$ need not hold for $T > 1$.*

Proof. The lowest occupied position of $3x$ is its negative endpoint at k. The next is the positive endpoint at ℓ; the remaining terms are multiples of $2^{\ell+1}$. Removal leaves valuation exactly ℓ. Adding a higher power leaves the valuation k unchanged. The last assertion follows by subtracting $3n + 1$ and $3b + 1$; the counterexample in Section 3 proves the limitation. □

For example, $3 \cdot 13 = 2^6 - 2^5 + 2^3 - 1$; removing -1 gives 40, whose canonical ABN is $2^6 - 2^5 + 2^4 - 2^3$, not the uncanonical three-cell expression. Thus deletion is an integer operation, followed by canonicalization.

Theorem 5.2 (P6-42–48, P6-50)**.** *For a subset $I \subseteq \{1, \dots, r\}$ define*

$$z_I = z + \sum_{i \in I} 2^{m_i}, \qquad M(I) = \#\{i < r : i \in I,\ i+1 \notin I,\ m_{i+1} = p_i + 1\}.$$

Then $N(z_I) = N(z) - 2M(I)$. For a single removal at i, the band is replaced by $2^{p_i} = 2^{p_i+1} - 2^{p_i}$ unless $i < r$ and $m_{i+1} = p_i + 1$, in which case the two bands merge into $2^{p_{i+1}} - 2^{p_i}$. In particular, with the band indices of $z = 3x$,

$$N(H(x)) = N(3x) - 2\mathbf{1}_{\{r \geq 2,\ m_2 = p_1 + 1\}}.$$

Proof. As an integer, $z_I = \sum_{i \notin I}(2^{p_i} - 2^{m_i}) + \sum_{i \in I} 2^{p_i}$. The unchanged intervals and the replacement single bits $\{p_i\}$ are disjoint and ordered. Consecutive intervals touch exactly when the lower band is selected, the upper is not, and the original gap consists of one zero bit. Each touch decreases the number of one-runs by one, with no other changes; every resulting run has two canonical endpoints. The single-band formulas follow by addition; applying the first lemma to $3x$ proves the H identity. □

The last identity only measures the correction after tripling, not the change from x to $H(x)$: $N(7) = 2$, $N(21) = 6$, and $N(22) = 4$.

Theorem 5.3 (P6-49, P6-52–53)**.** *Let Γ be the path subgraph on the r bands with edge $(i, i+1)$ exactly when $m_{i+1} = p_i + 1$, and let $\nu(\Gamma)$ be its maximum matching size. Then*

$$\min_I N(z_I) = N(z) - 2\nu(\Gamma).$$

If z is odd, and only positive odd z_I with the same first T actual exponents as z are admitted, the minimum is $N(z) - 2\nu(\Gamma_T)$, where Γ_T is the subgraph induced by $m_i > S_T(z)$.

Proof. The edges counted by $M(I)$ are a matching: shared endpoints would have to be both selected and unselected. Conversely, select only the lower endpoints of a maximum matching. Every matching edge is then realized, so the maximum equals the matching size. For the constrained problem, oddness excludes $m_1 = 0$; for nonempty I the difference $z_I - z$ has valuation $\min_{i \in I} m_i$. Theorem 3.1 gives exactly $m_i > S_T(z)$ for all selected indices. These vertices form an upper terminal segment, so every edge leaving a selected lower endpoint stays within the induced graph. The same matching argument applies there; the empty choice is always allowed. □

For $85 = (1010101)_2$ selecting the first and third negative endpoints gives $102 = (1100110)_2$ and decreases N from eight to four. This optimization changes the integer; it is not a freedom to choose the actual Collatz update of z.

6 Prescribed upper blocks and width resonance

Theorem 6.1 (P6-173–181, P6-185–186)**.** *Let b, U be positive odd, $L \geq 1$, $n = b + 2^L U$. Define*

$$j = \min\{i \geq 0 : S_{i+1}(b) \geq L\}, \quad h = L - S_j(b), \quad k = s_j(b), \quad q = F^{j+1}(b), \quad g = 3^{j+1}.$$

The first differing exponent is at index j, $1 \leq h \leq k$, and

$$\begin{aligned} k > h: &\quad s_j(n) = h, \quad F^{j+1}(n) = 2^{k-h} q + gU, \\ k = h: &\quad s_j(n) = k + e_U, \quad F^{j+1}(n) = (q + gU)/2^{e_U}, \quad e_U = \nu_2(q + gU) \geq 1. \end{aligned}$$

Proof. Theorem 3.1 fixes the first differing index, since $\nu_2(n - b) = L$. Before it, the normalized difference is $3^j 2^{L - S_j(b)} U$. Therefore the numerator at index j for n equals $2^k q + 2^h gU$. Comparing valuations proves both formulas. □

Proposition 6.2 (P6-178–179, P6-183). *For fixed b, L in Theorem 6.1, take $U = 2^d - 1$, $d \geq 1$, and write $n_d = b + 2^L(2^d - 1)$. In the case $k = h$, $e = \nu_2(q - g)$ is finite and positive, and*

$$e_U = \begin{cases} d & d < e, \\ e + \nu_2((q-g)/2^e + g) & d = e, \\ e & d > e. \end{cases}$$

Consequently the limit of $F^{j+1}(n_d)/n_d$ as $d \to \infty$ equals $g/2^L$ if $k > h$, and $g/2^{L+e}$ otherwise. Neither limit equals 1; a limit below (above) 1 implies strict numerical descent (increase) at this time for all sufficiently large d.

Proof. The actual odd output q is not divisible by 3, whereas $3 \mid g$, so $q \neq g$. Both are odd. Compare valuations in $q + gU = q - g + g2^d$; at $d = e$ the sum is positive and its residual factor even. For $d > e$ the output is $g2^{d-e} + (q-g)/2^e$ in the equality case. Compare leading coefficients of 2^d with $n_d = 2^{L+d} + b - 2^L$. The denominators are positive even integers, while g is odd. □

For $b = 1, L = 4$ the first differing index is 1 for every d; $d = 1, 2, 3, 4$ give first two exponents $(2, 3), (2, 4), (2, 8), (2, 5)$, respectively. This width resonance concerns a family of different initial values, not unbounded time on a fixed orbit.

Theorem 6.3 (P6-187–192). *Fix a positive odd b, $m \geq 1$, $L = S_m(b)$, $q = F^m(b)$ and $g = 3^m$. For every positive odd target w with $3 \nmid w$, there are infinitely many $e \geq 1$ with $2^e w \equiv q \pmod{3^m}$ and $2^e w > q$. For each such e, the unique positive odd U giving $n = b + 2^L U$ with the first $m - 1$ exponents of b, last exponent $s_{m-1}(b) + e$, and $F^m(n) = w$ is*

$$U = (2^e w - q)/3^m.$$

In particular $w = q$ is obtained precisely for e a positive multiple of $2 \cdot 3^{m-1}$, with $U = q(2^e - 1)/3^m$. In that case for every $t \geq 0$,

$$F^{m+t}(n) = F^{m+t}(b), \quad S_{m+t}(n) = S_{m+t}(b) + e, \quad H^{m+t}(n) = 2^e H^{m+t}(b).$$

Proof. Here $j = m - 1$ and $h = k$ in Theorem 6.1. Thus the output condition is exactly $q + gU = 2^e w$. For $a \geq 1$, $\nu_3(4^a - 1) = 1 + \nu_3(a)$: write $a = 3^h u$ with $3 \nmid u$, use $4^u \equiv 1 + 3u \pmod 9$, then factor $X^3 - 1 = (X - 1)(X^2 + X + 1)$, the second factor having valuation 1 when $X \equiv 1 \pmod 3$. Hence 2 has order $2 \cdot 3^{m-1}$ modulo 3^m, the number of units. Both q, w are units, so there is one exponent class modulo that order and arbitrarily large positive representatives. The quotient U is positive odd. For $w = q$ the congruence is exactly the order condition. Equal odd states have identical subsequent trajectories, so the cumulative exponent difference stays e and the lifted identity follows. □

Proposition 6.4 (P6-193–195). *In Theorem 6.3, if $b < 2^{L-1}$, the upper part $2^L U$ consists of actual canonical endpoint pairs of n, with the remainder b obtained by removing them. For fixed b, m, the additional exponent $e_U = \nu_2(q + gU)$ is bounded over $U = 2^d - 1$, but unbounded over positive odd U. Its relative natural density on odd U satisfies*

$$\mathrm{dens}_{\mathrm{odd}}(e_U \geq E) = 2^{-(E-1)}, \qquad \mathrm{dens}_{\mathrm{odd}}(e_U = E) = 2^{-E} \quad (E \geq 1).$$

Proof. The binary bit at $L - 1$ separates b from the shifted one-runs of U, so their endpoint pairs concatenate without a merge. The single-pair bound follows from Proposition 6.2: apart from one width, the exponents are at most e. For arbitrary odd U, the inequality $e_U \geq E$ specifies one odd residue $U \equiv -qg^{-1} \pmod{2^E}$. It contains infinitely many positive integers and has relative density $2^{-(E-1)}$; taking differences gives the mass. In particular arbitrarily large exponents are possible, necessarily requiring multiple upper pairs once the single-pair bound is exceeded. □

These are existence results for designed initial integers. The densities do not assert time frequencies along a fixed orbit. An unshifted Mersenne term $2^K - 1$ cannot be removed leaving an odd remainder of an odd input; such an extraction is not an instance of the two-positive-odd comparison.

7 Extraction from a fixed canonical word

Theorem 7.1 (P6-196–202). *Fix positive odd n with its canonical endpoint pairs indexed as above, $m_1 = 0$. Take nonempty $I \subseteq \{2, \dots, r\}$, $A_I = \sum_{i \in I}(2^{p_i} - 2^{m_i})$, $b_I = n - A_I$, $L = \min_{i \in I} m_i$, $U = A_I/2^L$. The remaining pairs are canonical, b_I is positive odd, and U odd. Set $j = \min\{i \geq 0 : S_{i+1}(n) \geq L\}$, $h = L - S_j(n)$, $k = s_j(n)$, $y = F^{j+1}(n)$, $g = 3^{j+1}$. The first differing index of n, b_I is j. Moreover,*

$$\begin{aligned} k > h: &\quad s_j(b_I) = h, \quad F^{j+1}(b_I) = 2^{k-h}y - gU, \\ k = h: &\quad s_j(b_I) = k + e_I, \quad F^{j+1}(b_I) = (y - gU)/2^{e_I}, \quad e_I = \nu_2(y - gU) \geq 1. \end{aligned}$$

In the second case $0 < F^{j+1}(b_I) < y$. In the first case, enlarging I strictly while preserving its lowest index strictly decreases the output of b_I at $j + 1$.

Proof. Deleting whole disjoint binary one-runs cannot merge the remaining ones; the bottom run remains. The lowest bit of the deleted sum fixes its valuation L. Apply Theorem 3.1 with reference n and subtract the normalized common-history states. The numerator for b_I at j is $2^k y - 2^h gU$ and is positive. Comparing valuations yields the formulas, including positivity of $y - gU$ in the equality case. Strictly adding positive upper pairs increases U, giving the asserted decrease in the first case. □

There are $2^{r-1} - 1$ eligible nonempty subsets, and 2^{r-i} with lowest selected index i ($2 \leq i \leq r$). The latter all share the same first differing index, but need not share the exponent at that index.

Proposition 7.2 (P6-203–204, P6-207). *Fix the lowest selected index $i \geq 2$ in Theorem 7.1 and take the case $k = h$. Let $U_0 = 2^{p_i - m_i} - 1$, and for $a > i$ let $V_a = 2^{m_a - m_i}(2^{p_a - m_a} - 1)$. For any additional subset $J \subseteq \{i + 1, \dots, r\}$ and $E \geq 1$,*

$$e_{\{i\} \cup J} \geq E \quad \Longleftrightarrow \quad \sum_{a \in J} V_a \equiv g^{-1}y - U_0 \pmod{2^E}.$$

If $i < r$, let $D = m_{i+1} - m_i$ and $e_0 = \nu_2(y - gU_0)$. All subsets in this layer have $e_I = e_0$ when $e_0 < D$ and $e_I \geq D$ when $e_0 \geq D$.

Proof. Write $U = U_0 + \sum_{a \in J} V_a$ and use the equality case of Theorem 7.1. Since g is odd, it has an inverse modulo 2^E. Each V_a is a multiple of 2^D when $i < r$, so the two positive numerators agree modulo 2^D, proving the valuation assertions. Terms with $m_a - m_i \geq E$ do not affect the displayed congruence. □

Theorem 7.3 (P6-205–206). *For the fixed word in Theorem 7.1, immediate reunion $F^{j+1}(b_I) = y$ occurs if and only if $k > h$ and*

$$gU = (2^{k-h} - 1)y.$$

Necessarily $2 \cdot 3^j$ divides $k - h$. There is at most one such subset in each lowest-selected-index layer.

Proof. The equality case of Theorem 7.1 has output strictly below y; the other case gives the equation by subtraction. Since $3 \nmid y$, it implies $3^{j+1} \mid 2^{k-h} - 1$; the order computation of Theorem 6.3 applies. For a fixed layer the equation fixes U, and disjoint binary runs make the map from subsets to their sums injective. □

For instance, $21 = 1 + 4 + 16$ has I consisting of the two upper runs, $b_I = 1$, $L = h = 2$, $j = 0$, $k = 6$, $y = 1$, $U = 5$. The criterion gives $3 \cdot 5 = 2^4 - 1$, so both reach 1 in one step. The source R17 printed $3 + 4 + 16$ in this example; the corrected lower run is 1. This criterion is not an assertion that a suitable subset exists for every n.

8 Mirrored blocks and interface arithmetic

In Sections 8–12 use $\beta(x) = \text{bitlength}(x)$, with $\beta(0) = 0$, and $N(x) = \text{popcount}(E(x))$ for nonnegative x, with $E(0) = 0$. For positive x the mask width is $B(x) = \beta(x) + 1$, not $\beta(x)$. The distinction matters in comparing the contact conventions of R31 and R132.

Theorem 8.1 (P6-266–268)**.** *For positive odd a and $M \geq \max(1, \beta(a-1)+1)$,*

$$E(a(2^M - 1)) = (1 + 2^M)(1 + E(a-1)), \quad N(a(2^M - 1)) = 2 + 2N(a-1).$$

Fix positive odd b and $T \geq 0$, put $y_t = F^t(b)$, $S_t = S_t(b)$. Choose $M \geq \max(1, \beta(3^T - 1) + 1)$ and $L \geq \max_{t \leq T}(S_t + \beta(y_t) + 1)$. For $X = b + 2^L(2^M - 1)$, the actual exponent prefix through T agrees with that of b, and for $t \leq T$,

$$F^t(X) = y_t + 2^{L-S_t}3^t(2^M - 1), \quad E(F^t(X)) = E(y_t) + 2^{L-S_t}(1 + 2^M)(1 + E(3^t - 1)).$$

The added occupied count is $2 + 2N(3^t - 1) \equiv 2 \pmod 4$.

Proof. Put $v = a - 1$, an even integer less than 2^{M-1}. The product is $2^M v + (2^M - 1 - v)$. Its lower M ordinary bits are the complement of v; the upper bits are those of v. The adjacent-bit XOR defining E has boundary bits 1 at 0 and M, and the two disjoint shifted copies of $E(v)$ on the remaining positions. The width hypothesis ensures $E(v) < 2^M$ and evenness ensures its bit 0 vanishes. This proves the first identity and count, since $N(v)$ is even. For the trajectory, $S_T < L$ gives the common prefix by Theorem 3.1. The difference formula there gives the integer identity. The separation $L - S_t \geq \beta(y_t) + 1$ ensures the two masks have disjoint support, so the mirror formula applies at each t. This construction chooses the initial widths for a prescribed finite horizon; it does not give permanent isolation. □

Theorem 8.2 (P6-269–271)**.** *Let $y, W \geq 0$, $k \geq 1$, $q = \lfloor y/2^k \rfloor$, $r = y \bmod 2^k$, $b = \text{bit}_{k-1}(r)$ and $p(u) = u \bmod 2$. Then*

$$N(y + 2^k W) - N(y) = N(q + W) - N(q) - 2b(p(q + W) - p(q)).$$

Let D be the binary carry mask for addition of q, W: its bit i is the carry into position i (bit 0 is zero). Then

$$E(y + 2^k W) \,\text{XOR}\, E(y) \,\text{XOR}\, 2^k E(W) = 2^k E(D).$$

For $q = 1$, odd $W > 0$, $h = \nu_2(W + 1)$ and $\eta = \text{bit}_{h+1}(W)$, the right side is $2^{k+1} + 2^{k+h+1}$, and the occupied-count increment is $N(W) - 2 + 2b - 2\eta$. For positive odd y, W, if $k \geq \beta(y) + 1$ the increment is $N(W)$; if $k = \beta(y)$ it is $N(W) - 2$.

Proof. The masks of r and $2^k q$ meet only at position k, with bits $b, p(q)$. Their XOR thus has count $N(r) + N(q) - 2bp(q)$; replace q by $q + W$ and subtract. The ordinary sum satisfies $q + W = q \,\text{XOR}\, W \,\text{XOR}\, D$; E is linear over XOR, including shifts. If $q = 1$, the carry is supported exactly at $1, \ldots, h$, hence $E(D) = 2 + 2^{h+1}$. Also $E(W + 1) = E(W) \,\text{XOR}\, (1 + 2^{h+1})$, so its count is $N(W) - 2\eta$. Substitution proves the formula. In the final cases $q = 0$ and $b = 0$ or 1 respectively. Vanishing carry correction does not preclude cancellation of overlapping occupied cells. □

9 First geometric contact and restarting

Theorem 9.1 (P6-272–273,277)**.** *Let $X = b + 2^L a$ with positive odd a, b, $L \geq \beta(b) + 1$. Put $y_t = F^t(b)$, $S_t = S_t(b)$, $k_t = L - S_t$, $g_t = k_t - \beta(y_t)$, $\tau = \min\{j : S_{j+1} \geq L\}$ and $t_c = \min\{t : g_t \leq 0\}$. Then $F^t(X) = y_t + 2^{k_t}3^t a$ for $0 \leq t \leq \tau$, and the first differing exponent index is τ. The formal gap satisfies*

$$g_{t+1} - g_t = -[\beta(3y_t + 1) - \beta(y_t)] \in \{-1, -2\},$$

so $\lceil g_0/2 \rceil \le t_c \le g_0$ *and* $g_{t_c} \in \{0, -1\}$. *If* $t_c \le \tau$ *this is actual first adjacency or one-bit overlap of the ordinary binary blocks. If* $\tau < t_c$, *then* $g_\tau = 1$, $s_\tau = \beta(y_\tau) + 1$, $F(y_\tau) = 1$ *and* $y_\tau = (4^m - 1)/3$ *for some* $m \ge 1$; *also* $\tau + 1 \le g_0$.

Proof. The common-history formula and first difference follow from Theorem 3.1. Subtract $\beta(y_{t+1}) = \beta(3y_t + 1) - s_t$ from $k_{t+1} = k_t - s_t$. The bounds $2^{\beta(y_t)} < 3y_t + 1 < 2^{\beta(y_t)+2}$ yield the two increments and the hitting-time estimates. If $\tau < t_c$ then $\beta(y_\tau) + 1 \le k_\tau \le s_\tau \le \beta(y_\tau) + 1$; equality forces $3y_\tau + 1 = 2^{\beta(y_\tau)+1}$, with even exponent by reduction modulo 3. The gap at that index is 1, hence $\tau + 1 \le g_0$. □

The R30 statement about possibly skipping adjacency into deep overlap is superseded by R31: at first contact the overlap is at most one ordinary bit. This is not the mask-contact convention of Paper B (which occurs one cell earlier).

Theorem 9.2 (P6-274–275)**.** *At actual first contact* $t = t_c \le \tau$ *in Theorem 9.1, write* $W = 3^t a$ *and* $I = N(F^t(X)) - N(y_t) - N(W)$. *Then* $I = -2$ *for* $g_t = 0$ *and* $I = -2 - 2\eta$ *for* $g_t = -1$, *where* $h = \nu_2(W + 1)$, $\eta = \mathrm{bit}_{h+1}(W)$. *Before first contact the corresponding difference is zero. In the overlap case the two highest ordinary bits of* y_t *are 10. If* $a = 2^M - 1$ *and* $M \ge \nu_2(3^t - 1) + 2$, *then* $\eta = 1$ *for odd* t *or* $t = 2^j u$ *with* $j \ge 1$, *odd* $u \equiv 1 \pmod 4$; *it is zero for* $u \equiv 3 \pmod 4$.

Proof. Separation and adjacency use Theorem 8.2. For overlap, $t \ge 1$, $g_{t-1} = 1$ and the preceding lifted width increases by 2. Writing $B_0 = \beta(y_{t-1})$ and $s = s_{t-1}$ gives $\beta(y_t) = B_0 + 2 - s$ and $y_t < 3 \cdot 2^{B_0 - s} = 3 \cdot 2^{\beta(y_t)-2}$. Since $k_t \ge 1$, the leading bits are 10. The interface formula therefore has $q = 1, b = 0$. For the mirror case set $v = \nu_2(3^t - 1)$ and $w = (3^t - 1)/2^v$. Modulo 2^{v+2} we have $W \equiv -1 - 2^v w$, hence $h = v$ and $\eta = 1$ precisely when $w \equiv 1 \pmod 4$. For odd t, $3^t \equiv 3 \pmod 8$ gives this residue. For $t = 2^j u$, repeated squaring of $9 = 1 + 8$ gives $3^{2^j} = 1 + 2^{j+2} w_j$ with $w_j \equiv 1 \pmod 4$; indeed $w_{j+1} = w_j + 2^{j+1} w_j^2$. Binomial expansion gives $w \equiv u \pmod 4$. □

Proposition 9.3 (P6-276)**.** *Every positive odd* $n > 1$ *has a separated highest canonical pair at either* n *or* $F(n)$: *the selected state has the form* $b + 2^L(2^M - 1)$ *with positive odd* b, $M \ge 1$, $L \ge \beta(b) + 1$.

Proof. If n has at least two ordinary one-runs, remove its top run. Otherwise $n = 2^m - 1$ for $m \ge 2$, and $F(n) = 2^m + (2^{m-1} - 1)$ has two separated runs. □

Theorem 9.4 (P6-278–280)**.** *At actual first contact in Theorem 9.1 write* $z = y_t$, $k = k_t$, $W = 3^t a$. *For overlap* $k = \beta(z) - 1$, *write* $z = 2^k + r$, $h = \nu_2(W + 1)$, $V = (W + 1)/2^h$, $J = k + h$. *Then*

$$F^t(X) = r + 2^J V, \qquad J - \beta(r) \ge h + 1, \qquad 3r < z,$$

with r, V *positive odd. For adjacency* $k = \beta(z)$, *write the top ordinary one-run of* z *as* $2^k - 2^j$, *with remainder* r *and* $m = k - j \ge 1$. *Then* $F^t(X) = r + 2^j A$, $A = 2^m(W + 1) - 1$. *If* $r > 0$, r *is odd,* $j - \beta(r) \ge 1$, *and* $3r < z$. *In each positive-remainder case the two masks are disjoint and their sum is the mask of the new state, so Theorem 9.1 can be applied afresh. If the adjacent remainder is zero, this particular restart ends, not necessarily the actual orbit. For an initial Mersenne block* $a = 2^M - 1$ *at overlap, writing* $v = \nu_2(3^t - 1)$ *gives* $h = \min(M, v)$ *if* $M \ne v$, *and* $h \ge M + 1$ *otherwise.*

Proof. In overlap the leading bits 10 imply $0 < r < 2^{k-1}$, with odd r. Substitute $z = 2^k + r$ and factor $W + 1$; the gap and $3r < z$ follow. In adjacency the top one-run has a zero immediately below it if $j \ge 1$, so $r < 2^{j-1}$ there; if $j = 0$, $r = 0$. Substituting gives the stated integer identity. For positive r, $2r < 2^j \le 2^k - 2^j$ proves the comparison. In both cases the separated ordinary blocks imply separated masks. For resonance write $W + 1 = 2^M 3^t - 2^v w$ for odd $w = (3^t - 1)/2^v$ and compare valuations. All these are new decompositions of the actual state, not changes to the actual trajectory. □

Theorem 9.5 (P6-281–282). *For positive odd W, $m \geq 1$, $h = \nu_2(W+1)$, $V = (W+1)/2^h$ and $\eta = \mathrm{bit}_{h+1}(W)$,*

$$E(2^m(W+1)-1) = 1 + 2^m(E(W)-1), \quad E(W) = 2^h E(V) \,\mathrm{XOR}\, (1+2^{h+1}).$$

Thus the upper coefficient's occupied count is preserved at adjacency restart and decreases by 2η at overlap restart. At actual first contact the net count change is

$$N(F^t(X)) - N(X) = [N(y_t) - N(b)] + [N(3^t a) - N(a)] - \ell,$$

where $\ell = 2$ at adjacency and $\ell = 2 + 2\eta$ at overlap. For any finite chain of valid restarts the corresponding changes sum to the total net change.

Proof. The first mask formula extends the bottom ordinary one-run by m bits. The second is the carry identity for adding 1 to W, using $W+1 = 2^h V$; XOR flips precisely the bits at 0 and $h+1$, originally 1 and η. The count claims follow. Initially the masks are disjoint; at contact apply Theorem 9.2 and subtract the initial count. Sum the identities for a finite chain of actual states by telescoping. The growth of both coefficients must be included. □

Proposition 9.6 (P6-283). *For a positive-remainder restart, $r < b$ if $t \leq 2$. This need not hold at $t = 3$: $2159 = 111 + 2^{11}$ has first contact at 3, with $F^3(2159) = 7289 = 377 + 2^8 \cdot 27 = 121 + 2^{10} \cdot 7$, and $121 > 111$.*

Proof. The bound $y_{i+1} + 1 \leq (3/2)(y_i + 1)$ gives $r < y_t/3 \leq 3b/4 + 5/12 < b$ for $b \geq 3, t \leq 2$. When $b = 1$, the base remains 1 and there is no positive-remainder restart. For the example the base goes $111, 167, 251, 377$, the common exponents are all 1 and the gaps are $4, 2, 1, -1$. The overlap decomposition in Theorem 9.4 gives the stated result. □

For instance, $47 = 15 + 2^5$ restarts at $71 = 7 + 2^6$, then at $161 = 1 + 2^5 \cdot 5$. Reaching comparison base 1 does not mean the actual state is 1. The count budget can have either sign; no monotonicity or bound on the number of restarts is asserted.

10 Signed isolation and mask contact

We now use the mask-contact convention of R131–132, with positive odd q, b, $\sigma \in \{1, -1\}$ and $n = 2^k q + \sigma b$. Set $L_+(x) = E(x)$, $L_-(x) = E(x-1) + 1$ on positive odd x, $\lambda_\sigma(x) = \beta(L_\sigma(x))$, $h_\sigma(x) = \mathrm{popcount}\, L_\sigma(x)$, $z_\sigma(x) = \lambda_\sigma(x) - h_\sigma(x)$; $m_\sigma(x)$ counts internal zero-runs of this mask. All these masks have leading and trailing 1; the minus mask is an auxiliary nonnegative word, not the canonical ABN of a negative number. Define $Z(x) = B(x) - N(x)$ and $m(x)$ to be the number of internal zero-runs of $E(x)$ for positive odd x. For the same positive-side comparison state $y_t = b_t$ and shift $K = L - S_t = k - S_t$, the ordinary gap is $g_t = K - \beta(y_t)$, whereas the mask gap is $d_t = K - \beta(y_t) - 1 = g_t - 1$. The diagrams below show the two endpoint conventions, with positions increasing to the right and with $w = \beta(y_t)$.

Ordinary support intervals			Mask support intervals		
$\underbrace{[0,\ldots,w-1]}_{\text{lower}}$	$\underbrace{[w,\ldots,K-1]}_{g_t}$	$\underbrace{[K,\ldots]}_{\text{upper}}$	$\underbrace{[0,\ldots,w]}_{\text{lower}}$	$\underbrace{[w+1,\ldots,K-1]}_{d_t}$	$\underbrace{[K,\ldots]}_{\text{upper}}$

The interval diagram describes the separated case $K > w+1$; intervals denote potential support, not necessarily occupied cells. At $K = w+1$, masks are adjacent while ordinary blocks have one zero between them. At $K = w$, ordinary blocks are adjacent while the masks overlap in one occupied endpoint. On the negative side $\lambda_-(x) = \beta(x) + 1 - \mathbf{1}_{x=1}$, so the positive-side shift relation is not used without this correction.

Theorem 10.1 (P6-655,656; signed separation)**.** *Define* $b_0 = b$, $s_i = \nu_2(3b_i + \sigma)$, $b_{i+1} = (3b_i + \sigma)/2^{s_i}$, $S_t = \sum_{i<t} s_i$, $d_t = k - S_t - \lambda_\sigma(b_t)$. *If* $d_0 \geq 1$ *then* n *is positive odd, and* $1 \leq d_j - d_{j+1} \leq 2$. *If* $d_t \geq 1$, *for all* $j \leq t$,

$$F^j(n) = 2^{k-S_j} 3^j q + \sigma b_j, \quad E(F^j(n)) = L_\sigma(b_j) + 2^{k-S_j} L_\sigma(3^j q).$$

In particular $d_0 \geq 2T + 1$ *guarantees separation through* T. *At a separated time* t,

$$Z(F^t(n)) - Z(n) = z_\sigma(3^t q) - z_\sigma(q) - S_t - h_\sigma(b_t) + h_\sigma(b),$$
$$m(F^t(n)) = m_\sigma(b_t) + m_\sigma(3^t q) + 1.$$

Proof. Put $X_j = 2^{S_j} b_j$. Its recurrence is $X_{j+1} = 3X_j + \sigma 2^{S_j}$, so $2X_j \leq X_{j+1} \leq 4X_j$ and the ordinary width increases by one or two. Since $\lambda_+(x) = \beta(x) + 1$ and $\lambda_-(x) = \beta(x) + 1 - \mathbf{1}_{x=1}$, the only extra correction occurs on first entry into the minus fixed point 1. There $3b_j - 1 = 2^{s_j}$ with $s_j \geq 3$, $\beta(b_j) = s_j - 1$, so the lifted width increases by 2 and the gap decreases by 1. This proves the gap bound. With $d_t \geq 1$ all earlier $S_j < k$, and comparing the two terms in $3F^j(n) + 1 = 2^{k-S_j} 3^{j+1} q + \sigma(3b_j + \sigma)$ yields exact division by 2^{s_j} until t. For $K = k - S_j$, positive-side blocks have disjoint binary supports and separated masks. On the minus side write $2^K(u-1) + (2^K - b_j)$ with $u = 3^j q$; the lower K bits are the complement of $b_j - 1$, so their interior adjacent differences are unchanged, the bottom bit is 1 and the boundary bit at K is 1 (the upper part starts with bit 0). This gives precisely $L_-(b_j) + 2^K L_-(u)$ under separation. The initial positivity follows from $b < 2^k$. In either case the zero count is $z_\sigma(u) + k - S_j - h_\sigma(b_j)$, and one internal zero-run separates the two masks. Subtract at time zero. □

Theorem 10.2 (P6-658; signed first mask contact)**.** *With Theorem 10.1's notation and* $d_0 \geq 1$, *let* $t = \min\{j \geq 1 : d_j \leq 0\}$, $K = k - S_t$, $u = 3^t q$, $l = L_\sigma(b_t)$, $h = L_\sigma(u)$, $a = \mathrm{bit}_{K-1}(l)$, $c = \mathrm{bit}_1(h)$ *and* $I = \mathbf{1}_{d_t=-1}$. *Then* $\lceil d_0/2 \rceil \leq t \leq d_0$, $d_t \in \{0, -1\}$, $K \geq 1$, *and*

$$F^t(n) = 2^K u + \sigma b_t, \qquad E(F^t(n)) = l \,\mathrm{XOR}\, (2^K h).$$

The balances at that instant are

$$Z(F^t(n)) - Z(n) = z_\sigma(u) - z_\sigma(q) - S_t - h_\sigma(b_t) + h_\sigma(b) + 2I,$$
$$m(F^t(n)) = m_\sigma(b_t) + m_\sigma(u) + I(a + c - 1).$$

Proof. The gap bound gives the time and overshoot. Just before contact, $k - S_{t-1} \geq \lambda_\sigma(b_{t-1}) + 1 > s_{t-1}$: for $b_{t-1} > 1$ use $3b_{t-1} + \sigma < 2^{\beta(b_{t-1})+2}$, and for 1 use $s = 2$ or 1. Thus exact division still applies. The final gap gives $K = \lambda_\sigma(b_t)$ or $\lambda_\sigma(b_t) - 1$. For $b_t > 1$ this is at least $\beta(b_t)$; for $b_t = 1$ the previous strict inequality $k - S_{t-1} > s_{t-1}$ gives $K \geq 1$. Thus $b_t < 2^K$ in all cases. Positive-side disjoint ordinary bits give the XOR formula. On the minus side the lower complement satisfies $E(2^K - b_t) = 2^K \,\mathrm{XOR}\, L_-(b_t)$, and the upper contribution $2^K E(u-1)$ combines with 2^K to give $2^K L_-(u)$. When $d_t = 0$ the two masks are adjacent; when $d_t = -1$ they share the occupied bit at K, which cancels. The top bit survives since $u \geq 3$ gives upper mask width at least two. Thus the count loses $2I$ relative to the separate sum and the width is unchanged. At adjacency the two end occupied runs join; at overlap the end runs disappear exactly on the sides with $a = 0$ and $c = 0$, respectively. Both sides have remaining occupied bits since $K \geq 1$, the low mask has bit 0 occupied, and $\beta(h) \geq 2$. At overlap, deleting the last low bit removes one occupied run if and only if $a = 0$, and deleting the first high bit removes one occupied run if and only if $c = 0$. The surviving sides cannot join across the now-zero position K. Thus the total occupied-run count is reduced by $(1 - a) + (1 - c)$ from the sum of the two separate counts. Subtracting one gives the claimed internal zero-run count. At adjacency the corresponding reduction is exactly one. The first follows by the width and occupied-count calculation as in Theorem 10.1. □

The positive-side separation and Z balance were already used in Paper B (P6-655,658,662); they are included here only as part of the signed framework, not as additional independent descent results. The ordinary-block contact of Section 9 and the mask contact here differ by one in the positive-side gap. The R47 one-step merger classification (P6-351–353) was likewise already published in Paper B. The bottom-run extensions $R_j(n) = 4^j n + (4^j - 1)/3$ satisfy $F(R_j(n)) = F(n)$ and change the mask by $E(R_j(n)) = (4^j - 1) + 4^j E(n)$; hence along this family $N/B \to 1$, $m/B \to 0$, $Z/B \to 0$ (P6-354). These are limits along designed initial values, not time frequencies; the limits alone do not prove nonuniqueness from exact density. Distinct inputs with a shared actual exponent prefix cannot merge during that prefix by Theorem 3.1 (P6-355).

11 Prescribed contact times and sampling over initial heights

Theorem 11.1 (P6-308–310)**.** *Fix positive odd a, b and put $y_t = F^t(b)$, $S_t = S_t(b)$, $J_t = S_t + \beta(y_t)$, and $\delta_t = J_{t+1} - J_t$. Then $\delta_t \in \{1, 2\}$. Every $L > \beta(b)$ belongs to a unique interval $J_{t-1} < L \le J_t$, $t \ge 1$. For $X_L = b + 2^L a$:*

1. *At $L = J_t$ the first ordinary-block contact is actual adjacency at t. With $k = \beta(y_t)$ and $W = 3^t a$,*

$$F^t(X_L) = y_t + 2^k W, \quad E(F^t(X_L)) = (E(y_t) - 2^k) + 2^k(E(W) - 1).$$

   In particular the upper mask bits of relative indices $i \ge 1$ are preserved at positions $k + i$, and the occupied count is $N(y_t) + N(W) - 2$.

2. *The only other possible height is $L = J_t - 1$ when $\delta_{t-1} = 2$. If $y_t > 1$, this gives actual first one-bit overlap at t, with $F^t(X_L) = y_t + 2^{\beta(y_t)-1} W$. If $y_t = 1$, the first exponent difference occurs at $t - 1$, before formal contact.*

Proof. Theorem 9.1 gives the increments and formal contact classification. At $L = J_t$, $S_t < L$, so the exponent prefix of length t agrees by Theorem 3.1; before t the gap is positive and at t it is zero. The ordinary blocks are disjoint and XOR-linearity of E removes precisely their coinciding endpoint bits at k. For $L = J_t - 1$, the common prefix criterion is $S_t < L$, equivalent to $\beta(y_t) > 1$. Otherwise $S_t = L$ while $S_{t-1} < J_{t-1} < L$, giving the asserted first difference. The increments exhaust all heights. □

Proposition 11.2 (P6-311)**.** *For an integer $M \ge 1$, choose uniformly among all heights giving actual first contact at times $1, \dots, M$, with a, b fixed as above. Each time has multiplicity $c_t = 1 + \mathbf{1}_{\{\delta_{t-1}=2,\ y_t>1\}}$. For any event indicator $A_t \in \{0, 1\}$, set $p = M^{-1} \sum_{t=1}^{M} A_t$, $q = \sum c_t A_t / \sum c_t$. Then*

$$\frac{p}{2 - p} \le q \le \frac{2p}{1 + p}.$$

Restricting to adjacency heights $L = J_t$ gives $q = p$. In particular pure-$3^t a$ window frequencies (excluding the cancelled relative bit 0) can be transferred to adjacency samples at those heights, not to a single fixed initial integer.

Proof. The multiplicities follow from Theorem 11.1. For Mp event times the total event weight is between Mp and $2Mp$, while the complementary weight is between $M(1 - p)$ and $2M(1 - p)$. Taking the extreme ratios gives the inequalities including $p = 0, 1$. At adjacency there is exactly one height per time; the upper mask transfer follows from the preceding theorem. □

12 Time-labelled additive components and exact cancellation

Theorem 12.1 (P6-320–321; lifted time ledger)**.** *Write a positive odd initial integer as the sum of its canonical pairs $P_i = 2^{\ell_i}(2^{d_i} - 1)$, corresponding to its maximal ordinary one-runs. With*

$n_t = F^t(n)$ *and* $S_t = S_t(n)$,

$$2^{S_T} n_T = \sum_i 2^{\ell_i} 3^T (2^{d_i} - 1) + \sum_{j<T} 2^{S_j} 3^{T-1-j} (2^1 - 1).$$

For a restart at time t, replace the first sum by $3^u \sum_i P_i^{(t)}$, where the pairs are those of n_t, and the second by $\sum_{j<u} 2^{S_{t+j}-S_t} 3^{u-1-j}$; the left side is then $2^{S_{t+u}-S_t} n_{t+u}$. An arbitrary exact signed additive decomposition can also replace the first pair sum, without implying canonical extraction.

Proof. Each maximal ordinary one-run contributes the difference of powers at its two endpoints; the endpoints are exactly the alternating signed canonical cells. Substitute the pair decomposition into the affine identity of Section 2, applied to n or to n_t. Each injection has value $1 = 2^1 - 1$, so it too is a two-cell term with its own insertion time. The actual history is that of the total state, not independently accelerated component histories. □

Proposition 12.2 (P6-322)**.** *For nonnegative x, y define $Q = (x + y)$ XOR x XOR y. Then*

$$E(x + y) = E(x) \text{ XOR } E(y) \text{ XOR } E(Q).$$

In general $N(x + y)$ can exceed $N(x) + N(y)$; e.g. $7 + 14 = 21$ has counts $(2, 2, 6)$. In the actual trajectory $99 = 3 + 96 \mapsto 149$ the count increases from 4 to 8, although the lifted sum is formed from two initial pairs and one injection. Thus additive components alone do not bound the number of canonical pairs by their number.

Proof. Q is the mask of input carries at all ordinary binary positions. The equality follows from XOR-linearity of E. The displayed counts follow by direct evaluation; $3 \cdot 99 + 1 = 298 = 2 \cdot 149$, and shifting by one preserves the occupied count. No claim of ordinary-additive linearity of E is made. □

General addition lemma (reviewer-suggested separation). For arbitrary positive odd A, D with $S = \nu_2(A + D) \geq 2$, the mask identity

$$(E(A) \text{ XOR } E(D)) \bmod 2^{S+1} = 2 + 2^S$$

holds. The signed coefficients $\alpha_i(x) = \text{bit}_{i-1}(x) - \text{bit}_i(x)$ have sum -2 at position 0, 1 at position 1, and zero at $2 \leq i < S$; the input carry mask satisfies $Q \bmod 2^{S+1} = 2^{S+1} - 2$. In the separate case $S = 1$, the mask XOR modulo 4 is zero.

Proof of the lemma. Write the ordinary bits as a_i, b_i. Both bottom bits are 1 and a carry enters position 1. Since the sum has valuation S, the output is zero at $1 \leq i < S$, so inductively $a_i + b_i = 1$ there and the carry persists. The output at S is 1, hence $a_S = b_S$. Adjacent XOR differences give the mask identity, and signed adjacent differences give the stated signed coefficients. The carry enters precisely positions $1, \ldots, S$ within this window. When $S = 1$, both pairs of bits at positions 0 and 1 agree between the summands, yielding the asserted XOR modulo 4. This is a general integer addition fact, not a Collatz-specific property. □

Theorem 12.3 (P6-324–325; two-boundary cancellation)**.** *For $T \geq 1$ let $A = 3^T n$, $D = \sum_{j<T} 2^{S_j} 3^{T-1-j}$, and $S = S_T \geq 2$. Both A, D are positive odd and $A + D = 2^S n_T$. Then*

$$(E(A) \text{ XOR } E(D)) \bmod 2^{S+1} = 2 + 2^S.$$

Consequently the two counts restricted to positions $0, \ldots, S$ differ by at most 2. For the signed coefficients $\alpha_i(x) = \text{bit}_{i-1}(x) - \text{bit}_i(x)$, with bit -1 defined as zero,

$$\alpha_0(A) + \alpha_0(D) = -2, \quad \alpha_1(A) + \alpha_1(D) = 1, \quad \alpha_i(A) + \alpha_i(D) = 0 \quad (2 \leq i < S).$$

The signed sum through position $S-1$ is zero, and the carry mask $Q = (A+D)\,\mathrm{XOR}\,A\,\mathrm{XOR}\,D$ satisfies $Q \bmod 2^{S+1} = 2^{S+1}-2$. Conditional on the fixed exponent word, $n \equiv 3^{-T}(2^S - D) \pmod{2^{S+1}}$. The same cancellation statements hold for the accelerated map with any fixed positive odd constant c in $3n+c$, replacing D by cD and using its actual exponent history (the congruence then uses cD).

Proof. Apply the preceding general addition lemma to the positive odd summands of the lifted identity. The mask XOR has two occupied positions in the window, so the occupied counts differ by at most two. The initial weighted terms $-2\cdot 2^0 + 1\cdot 2^1 = 0$ cancel, and all the remaining terms below S vanish; the unsigned coefficient sum is not being asserted to vanish. Reduction of the lifted identity modulo 2^{S+1} gives the congruence using the oddness of n_T. For positive odd c, the lifted affine correction is cD, also odd, so the same argument applies. □

Proposition 12.4 (P6-326; composition of time injections)**.** *For $u, v \geq 1$ and restart time $t \geq 0$ let $R_{t,u} = S_{t+u} - S_t$ and $D_{t,u} = \sum_{j<u} 2^{S_{t+j}-S_t} 3^{u-1-j}$. Then*

$$3^u n_t + D_{t,u} = 2^{R_{t,u}} n_{t+u}, \quad D_{t,u+v} = 3^v D_{t,u} + 2^{R_{t,u}} D_{t+u,v}.$$

Every window with $R_{t,u} \geq 2$ satisfies Theorem 12.3 with $A = 3^u n_t$, $D = D_{t,u}$. No independence across windows follows from this identity.

Proof. Apply the lifted identity from the state n_t. Split the injection sum for $u+v$ at u and factor the powers of 3 and 2 respectively. The cancellation argument applies to the new positive odd pair because their sum has valuation $R_{t,u}$. □

Theorem 12.5 (P6-327–328; information retained in canonical extraction)**.** *Keep the lowest canonical pair P_0 of a positive odd integer and choose any subset of the remaining pairs $P_i = 2^{\ell_i}(2^{d_i}-1)$, $1 \leq i \leq m$, in increasing order of ℓ_i, obtaining $b_\epsilon = P_0 + \sum_i \epsilon_i P_i$. For distinct selections ϵ, η and first differing index j,*

$$\nu_2(3^T(b_\epsilon - b_\eta)) = \ell_j \qquad (T \geq 0).$$

Thus under uniform selection, for each $k \geq 1$, $E(3^T b_\epsilon) \bmod 2^k$ has exactly $2^{q(k)}$ equiprobable values, where $q(k) = \#\{i \geq 1 : \ell_i < k\} \leq \lfloor (k-1)/2 \rfloor$. Its Shannon entropy in bits is $q(k)$ for all T. Moreover the class of $\nu_2(3^T b_\epsilon + 1)$ being bounded in T or unbounded can change under extraction only through the pair starting at position 2. Such a pair exists exactly when the original integer is 5 modulo 8, and its inclusion switches the two classes.

Proof. The first differing term is an odd multiple of 2^{ℓ_j}, all higher terms multiples of 2^{ℓ_j+1}. Multiplication by 3 preserves the valuation. Choices below k therefore give distinct residues, with identical multiplicities from the higher choices. The map E is invertible on a fixed low window by successive XOR, and $\ell_i \geq 2i$ by the separating zero bits. For the last assertion, powers of 3 modulo 8 alternate between 1 and 3. For $b \equiv 1, 3 \pmod 8$ the valuation of $3^T b + 1$ is at most 2, and for $b \equiv 5, 7 \pmod 8$ it is either 1 or at least 3. In the latter cases $-b^{-1}$ is 1 or 3 modulo 8 and hence in the subgroup generated by 3 modulo 2^k for every $k \geq 3$ (the order of 3 is 2^{k-2}, since $\nu_2(3^{2^j} - 1) = j + 2$ for $j \geq 1$ by repeated squaring, and the order modulo 8 is 2; the subgroup consists exactly of those two residue classes). Arbitrarily large valuations are therefore attained. Higher pairs at positions $\ell_i \geq 3$ vanish modulo 8. A pair beginning at 2 requires low ordinary bits 101, and inclusion changes the residue from 1 to 5, proving the assertion. □

For example $109 = 1 + 12 + 96$ yields comparisons $1, 13, 97, 109$, with residues $1, 5, 1, 5$ modulo 8. These comparisons are not successive states of one trajectory. The finite-prefix superposition P6-323 is already contained in Theorem 3.1 and the difference formula of Section 3. Time-labelled additive components here are exact integer identities; they do not constitute a unique canonical time-labelled rewrite system. In particular the cancellation theorem applies also to positive odd constants other than 1, so it is not a convergence proof specific to $3n+1$.

13 Provenance and remaining editorial scope

The source is the Phase 6 consolidated R270 main note, P6-1–12, P6-21–33, preserving the Phase 5 R2jr–R2ju material cited there (Sections 175–181). The source assigns the lifted formulation, prescribed-history G, two-cell extraction and rearrangement, and two-history comparison to Y.Ueoka; the associated formal proofs, first-branch criterion and finite-time calculations to the contemporary ChatGPT. The present reconstruction and verification are by Tsumugi (ChatGPT), not a retrospective name for all historical ChatGPT research sessions. Y.Ueoka is the human corresponding author responsible for submission and correspondence. The original source, including its attribution and subsequent corrections, remains in the archive. These basic identities are not claimed to be historically new. Sections 5–7 reconstruct P6-R04/R05 (P6-34–53) and R15–R17 (P6-173–207, selected results). The original R04/R05 negative-cell and pair-extraction ideas and the R15 internal/Mersenne extraction question are attributed to Y.Ueoka in the source. The original formal results and proofs are by the contemporary ChatGPT; the present selection, reorganization, correction of the R17 example and checks are by Tsumugi. Sections 8–10 reconstruct R29–R33 (P6-266–283) and R131–R132 (P6-655,656,658), distinguishing the ordinary-bit and mask contact conventions and the already published positive-side result. The restarting proposal belongs to Y.Ueoka, as do the earlier isolated-three-power formation and lower-budget integration directions; the original formal derivations and checks belong to the contemporary ChatGPT. The present reconstruction and new finite checks are by Tsumugi. R47 is cited without re-counting the classification in Paper B. The original statements and corrections are preserved. Sections 11–12 reconstruct R39 (P6-308–311) and R41–R42 (P6-320–328, with P6-323 cited rather than counted twice). The original problem settings on arbitrary pair extraction, restart and pure-three-power formation belong to Y.Ueoka; the original contact-height and cancellation derivations and checks belong to the contemporary ChatGPT. Tsumugi reconstructed the proofs and performed the present checks. The selected scope remains fixed; remaining merger-based induction and R271 belong to Paper F. Other archived time-labelled representations are not asserted to have been exhaustively audited or included. Neither the original R271 finite-rule obstruction nor the conjecture is resolved by this paper. The manuscript has undergone an internal proof review and replay of all four finite-check scripts. An independent ChatGPT reviewer recommended minor revisions; R42 addresses all five recommendations. The reviewer suggested the explicit cylinder form, separation of the general addition lemma, notation and contact clarifications, conclusion and reproducibility manifest. This is an independent reconsideration, not human peer review or a separate-model guarantee. The reviewer did not provide a target PDF hash; exact byte identity of the reviewed file is not verified. The R42 revision has not undergone an additional independent review. The literature comparison below is focused, not a comprehensive priority search.

Source concordance. All numbered results are mapped below to the preserved Phase 6 R270 TeX archive (`references/R270_flat/MAIN_NOTE.tex` in the BATON research archive). The label prefix P6- is omitted in the table. These are archival revision identifiers, not independently verified public edition identifiers. The opening sections preserve the Phase 5 material identified above. Where a result combines several source propositions, the table records their joint provenance, not a claim of a one-to-one correspondence. The source file is identified by its SHA-256 fingerprint:

`3c4a00d052cf0e54aa759b8c7d9ef4a311bcc11fc04b2858c126d18f58696ae6`

Result	Original labels (P6-)	Archive section or revision
2.1	1–4	Phase 6 opening; R03 where noted
2.2	5–9, 31–33	Phase 6 opening; R03 where noted
2.3	9a–d	Phase 6 opening; R03 where noted
3.1	10	Phase 6 opening / R03
3.2	11, 21–25	Phase 6 opening / R03
4.1	26–27	R03
4.2	28–30	R03
5.1	34–39, 50–51	R04–R05
5.2	42–48, 50	R04–R05
5.3	49, 52–53	R04–R05
6.1	173–181, 185–186	R15–R16
6.2	178–179, 183	R15–R16
6.3	187–192	R15–R16
6.4	193–195	R15–R16
7.1	196–202	R17
7.2	203–204, 207	R17
7.3	205–206	R17

Result	Original labels (P6-)	Archive section or revision
8.1	266–268	R29–R30
8.2	269–271	R29–R30
9.1	272–273,277	R31–R33
9.2	274–275	R31–R33
9.3	276	R31–R33
9.4	278–280	R31–R33
9.5	281–282	R31–R33
9.6	283	R31–R33
10.1	655,656	R131–R132
10.2	658	R131–R132
11.1	308–310	R39
11.2	311	R39
12.1	320–321	R41–R42
12.2	322	R41–R42
12.3	324–325	R41–R42
12.4	326	R41–R42
12.5	327–328	R41–R42

14 Relation to earlier work

The congruence in Theorem 3.1 describes the odd-input Syracuse cylinder of a given finite exponent word, including the final odd bit. The classical parity-vector approach encodes finite histories by congruence classes and is related to the 2-adic shift conjugacy of Bernstein and Lagarias [5]. Their map uses one division by 2 per step, whereas our map F groups a whole odd-to-odd transition. In that convention the affine formula of Section 2 is the usual finite Syracuse composition (cf. Tao, equation (1.7), [6]); neither this formula nor congruence coding is presented as a new discovery. Our explicit first-branch formulas, the separation of prescribed from actual histories, and the analysis of selected canonical subwords use these elementary identities as inputs. Inverse-tree methods, for example Applegate and Lagarias [7], likewise construct predecessors; our target and reunion formulas distinguish freely designed upper blocks from extraction within a fixed canonical word.

Sinyor's binary-run rewriting [8] is a related structural approach. The ordinary one-runs used to index our endpoint pairs are standard binary runs, while the CPE occupied and zero runs belong to the adjacent-XOR mask, not to the original binary word. Sections 5–12 focus on matching under negative-cell deletion, interface cancellation, restarting, and information retained under prescribed

extraction. Papers A–D [1, 2, 3, 4] contain the complementary representation, descent, gluing, and finite-window results. In particular the contact-height construction transfers samples between distinct initial integers rather than proving fixed-orbit statistical behavior. This comparison is not a claim of historical priority for individual structural formulas. Neither convergence nor a new unconditional density result follows from these finite-time statements.

15 Conclusion

The central structural results are the maximum-matching characterization of negative-cell deletion, exact tests for extracting and reuniting canonical pairs within a fixed word, contact and restart balances for coupled blocks, signed interface cancellation, and the information count retained by pure tripling of canonical extractions. The affine iteration and finite congruence formulas are classical ingredients, and the positive-side result already in Paper B is recalled for a unified signed treatment, not counted as a new discovery. The general addition lemma isolates the finite arithmetic behind the time-injection application. None of these statements asserts that a given initial integer necessarily admits a successful extraction or a finite sequence of restarts leading to convergence. The fixed-orbit convergence problem remains open. The source record is retained for later research, not as a claim of comprehensive historical priority.

References

[1] Y. Ueoka and Tsumugi (ChatGPT), *Alternating Binary Notation and Collatz Phase Expression: Canonical Representations and Exact Structural Dynamics*, aiXiv (2026), https://aixiv.science/abs/aixiv.261128.000003.

[2] Y. Ueoka and Tsumugi (ChatGPT), *Finite-Time Structural Descent in the Accelerated Collatz Map: Two-Step Vacancy Balances, Finite Compression, and Window Scores*, aiXiv (2026), https://aixiv.science/abs/aixiv.261128.000004.

[3] Y. Ueoka and Tsumugi (ChatGPT), *Witness-Preserving Gluing and Variational Induction in the Collatz Problem*, aiXiv (2026), https://aixiv.science/abs/aixiv.261128.000005.

[4] Y. Ueoka and Tsumugi (ChatGPT), *Powers of Three and Structural Transport in the Collatz Map: Exact Window Distributions and Boundary Estimates*, aiXiv (2026), https://aixiv.science/abs/aixiv.261128.000006.

[5] 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.

[6] T. Tao, Almost all orbits of the Collatz map attain almost bounded values, Forum of Mathematics, Pi 10 (2022), e12, https://arxiv.org/abs/1909.03562.

[7] D. Applegate and J. C. Lagarias, Lower bounds for the total stopping time of $3x+1$ iterates, Mathematics of Computation 72 (2003), 1035–1049, https://arxiv.org/abs/math/0103054.

[8] J. Sinyor, The $3x+1$ problem as a string rewriting system, Discrete Dynamics in Nature and Society (2010), Article 458563, https://doi.org/10.1155/2010/458563.

A Reproducible finite checks

The four scripts use Python 3.10 or later and its standard library only. Execute each separately with assertions enabled (without `-O`); for example, `python verify_r37.py`. All four were replayed for R41 and their stdout matched the saved R37–R40 logs byte for byte. The independent reviewer reported the same replay result; the supplied replay logs match all four stored logs. The additional reviewer script was replayed for R42 under Python 3.13.5, matching its supplied log exactly. The PDF includes the complete plain-text code and embeds the scripts and corresponding logs. Wrapped listing lines are continuations of the same source line. Finite checks are not replacements for the analytic proofs; counts overlap.

Script	Selected replay counts
R37	211,328 prefix/correction checks; 16,896 lift/cut checks; 325 separated families.
R38	19,600 removal subsets; 51,669 prefix checks; 26,880 upper blocks; 11,812 fixed extractions.
R39	25,000 general interface checks; 15,360 ordinary-contact trials; 12,288 signed contact cases per sign.
R40	24,576 adjacency heights; 65,536 carry masks; 20,480 injection windows; 40,960 composition windows.

Separate code package. The companion archive `Paper_E_R42_reproducibility.zip` contains the four original scripts and logs, the reviewer's additional checks and log, a README and a SHA-256 manifest. It is a local companion deliverable, not yet a deposited repository. The original code and log fingerprints are:

File	SHA-256
verify_r37.log	fc2c741bac1fb58c2b8bf6a19777cfe5db796925d6e26447f1b7ab2e0c34d13b
verify_r37.py	5899ca188bf04bdccdaedd8fd969c52208f3c82f4d2b694081d533090ffdf24f
verify_r38.log	b4655668e584b1fd7fc405749422537dd12603168ed245e79daeecc1a2147164
verify_r38.py	f269eff7c00da20c723724a29d6bf0937f91620dc4c4db5e09139ee04e2992bf
verify_r39.log	bade38e13b0d6de6caddb90a707828f23b1e8a36522b1023edafe5862bde5b09
verify_r39.py	5e159bc4164357e1eda079a4e0f4775e2b399ab99de077058883648a91f62900
verify_r40.log	38c216308e2a4e3ccbc1cd5ded694f3e8ccdff3fb54f107a29f6eca2b4c26d82
verify_r40.py	5d57280f3fb9799bfa25af17c28f3452bae0191c96b5c7d51c1bb80211f1d969

`verify_r37.py`

```
from itertools import product

def v(x):
    assert x
    return (abs(x)&-abs(x)).bit_length()-1

def H(x): return 3*x+(1<<v(x))
def hist(n,T):
    x=n; S=[0]; X=[n]
    for _ in range(T):
        y=3*x+1; a=v(y); x=y>>a; S.append(S[-1]+a); X.append(x)
    return S,X

def G(b,S,T):
    for j in range(T): b=3*b+(1<<S[j])
    return b

P={n:hist(n,12) for n in range(1,256,2)}
count=0
for x,y in product(P,repeat=2):
    if x==y: continue
```

```
    S,X=P[x]; R,Y=P[y]; k=v(x-y)
    J=next(j for j in range(1,13) if R[j]>=k)
    d=k-R[J-1]; r=R[J]-R[J-1]; q=S[J]-S[J-1]
    assert (q==d if r>d else q>d)
    for t in range(13):
        assert (S[:t+1]==R[:t+1])==(R[t]<k)
        D=(X[t]<<S[t])-(Y[t]<<R[t])-3**t*(x-y)
        assert D==sum(3**(t-1-j)*((1<<S[j])-(1<<R[j])) for j in range(t))
        if t<=J: assert D==0
        if t==J+1: assert D==(1<<S[J])-(1<<R[J]) and v(D)==min(S[J],R[J])
        count+=1
print('prefix and correction:',count)
count=0
for a,K,b in product(range(1,5),range(1,13),range(1,64,2)):
    n=(a<<K)+b; S,X=hist(n,10); h=n
    for t in range(11):
        assert h==X[t]<<S[t]
        g=G(b,S,t); assert h==3**t*(a<<K)+g
        assert (g%(1<<K)==0)==(S[t]>=K)
        if S[t]>=K: assert (g//(1<<K)+3**t*a)%(1<<(S[t]-K))==0
        h=H(h); count+=1
print('lift and cut:',count)
count=0
for T in range(65):
    for M in range(max(1,2*T),max(1,2*T)+5):
        U,V,W=(1<<M)+1,(1<<M)-1,1
        for _ in range(T): U,V,W=H(U),H(V),H(W)
        assert (U,V,W)==(3**T*(1<<M)+4**T,3**T*(1<<M)-2**T,4**T)
        assert U-V-W==2**T and (U-W)//(1<<M)==3**T
        count+=1
print('separated families:',count)
```

verify_r38.py

```
from itertools import product

def v(x):
    assert x
    return (abs(x)&-abs(x)).bit_length()-1

def N(x): return (x^(2*x)).bit_count()
def hist(x,t):
    S=[0]; X=[x]
    for _ in range(t):
        a=v(3*x+1); x=(3*x+1)>>a; S.append(S[-1]+a); X.append(x)
    return S,X

def runs(x):
    a=[]; i=0
    while x:
        if not x&1: x>>=1; i+=1; continue
        m=i
        while x&1: x>>=1; i+=1
        a.append((m,i))
    return a

def subsets(r):
    for mask in range(1<<r): yield {i for i in range(r) if mask>>i&1}

def match(edges):
    # A left-to-right greedy maximum matching for a subgraph of a path.
    k=0; last=-2
    for i in edges:
        if i>last+1: k+=1; last=i
```

```
    return k

count=0; steps=0; constraints=0
for z in range(1,2048):
    a=runs(z); r=len(a); assert N(z)==2*r
    e=[i for i in range(r-1) if a[i+1][0]==a[i][1]+1]
    opt=2*r; S=hist(z,8)[0] if z&1 else []
    opts=[2*r]*len(S)
    for I in subsets(r):
        w=z+sum(1<<a[i][0] for i in I); M=sum(i in I and i+1 not in I for i in e)
        assert N(w)==2*r-2*M; opt=min(opt,N(w)); count+=1
        if z&1 and w&1:
            R=hist(w,8)[0]
            for t,s in enumerate(S):
                same=R[:t+1]==S[:t+1]
                assert same==(not I or min(a[i][0] for i in I)>s)
                if same: opts[t]=min(opts[t],N(w))
                constraints+=1
    assert opt==2*r-2*match(e)
    for t,s in enumerate(S): assert opts[t]==2*r-2*match([i for i in e if a[i][0]>s])
    b=runs(3*z); k=v(z); h=3*z+(1<<k)
    assert b[0][0]==k and v(h)==b[0][1]
    assert N(h)==N(3*z)-2*(len(b)>1 and b[1][0]==b[0][1]+1)
    for m,p in b[1:]: assert v(3*z+(1<<m))==k
    steps+=1
print('removal subsets:',count,'actual H steps:',steps,'prefix checks:',constraints)
count=0; resonance=0
for b,L in product(range(1,128,2),range(1,13)):
    S,X=hist(b,L+1); j=next(i for i in range(L) if S[i+1]>=L); h=L-S[j]; k=S[j+1]-S[j]; q=X[j
    +1]; g=3**(j+1)
    for U in set(range(1,40,2))|{(1<<d)-1 for d in range(1,21)}:
        n=b+(U<<L); R,Y=hist(n,j+1)
        assert R[:j+1]==S[:j+1] and R[j+1]!=S[j+1]
        if k>h: assert R[j+1]-R[j]==h and Y[j+1]==(q<<(k-h))+g*U
        else:
            a=v(q+g*U); assert R[j+1]-R[j]==k+a and Y[j+1]==(q+g*U)>>a
        count+=1
    if k==h:
        e=v(q-g)
        for d in range(1,25):
            actual=v(q+g*((1<<d)-1)); expected=d if d<e else e if d>e else e+v(((q-g)>>e)+g)
            assert actual==expected; resonance+=1
print('upper blocks:',count,'width resonances:',resonance)
count=0; coalescence=0; density=0
for b,m in product(range(1,64,2),range(1,5)):
    S,X=hist(b,m+6); L=S[m]; q=X[m]; g=3**m; order=2*3**(m-1)
    assert len({pow(2,e,g) for e in range(order)})==order
    for w in range(1,24,2):
        if w%3==0: continue
        e=next(e for e in range(1,order+1) if (pow(2,e,g)*w-q)%g==0)
        while (w<<e)<=q: e+=order
        for ee in [e,e+order]:
            U=((w<<ee)-q)//g; n=b+(U<<L); R,Y=hist(n,m+6)
            assert R[:m]==S[:m] and R[m]==S[m]+ee and Y[m]==w
            if b<(1<<(L-1)): assert runs(n)==runs(b)+[(l+L,u+L) for l,u in runs(U)]
            count+=1
    for e in [order,2*order]:
        U=q*((1<<e)-1)//g; R,Y=hist(b+(U<<L),m+6)
        assert Y[m:]==X[m:] and all(R[i]-S[i]==e for i in range(m,m+7)); coalescence+=1
    for E in range(1,9):
        a=[v(q+g*U) for U in range(1,1<<(E+1),2)]
        assert sum(x>=E for x in a)==2 and sum(x==E for x in a)==1; density+=1
print('designed targets:',count,'reunion families:',coalescence,'residue counts:',density)
count=0; mods=0; shield=0; coalescence=0
```

```
for n in range(1,4096,2):
    a=runs(n); r=len(a); S,X=hist(n,n.bit_length()); layer={}
    for I in subsets(r):
        if not I or 0 in I: continue
        i=min(I); L=a[i][0]; A=sum((1<<a[i0][1])-(1<<a[i0][0]) for i0 in I); U=A>>L; b=n-A
        assert runs(b)==[a[i0] for i0 in range(r) if i0 not in I]
        j=next(j for j in range(len(S)-1) if S[j+1]>=L); h=L-S[j]; k=S[j+1]-S[j]; y=X[j+1]; g
    =3**(j+1)
        R,Y=hist(b,j+1); assert R[:j+1]==S[:j+1] and R[j+1]!=S[j+1]
        same=Y[j+1]==y
        assert same==(k>h and g*U==((1<<(k-h))-1)*y)
        if same:
            assert (k-h)%(2*3**j)==0; layer[i]=layer.get(i,0)+1; coalescence+=1
        if k>h:
            assert R[j+1]-R[j]==h and Y[j+1]==(y<<(k-h))-g*U
        else:
            e=v(y-g*U); assert R[j+1]-R[j]==k+e and Y[j+1]==(y-g*U)>>e and Y[j+1]<y
            U0=(1<<(a[i][1]-L))-1
            for E in range(1,9):
                assert (e>=E)==((g*(U-U0)-(y-g*U0))%(1<<E)==0); mods+=1
            if i+1<r:
                D=a[i+1][0]-L; e0=v(y-g*U0); assert (e==e0 if e0<D else e>=D); shield+=1
        count+=1
    assert all(c==1 for c in layer.values())
print('fixed canonical extractions:',count,'modular checks:',mods,'shield checks:',shield,'
    immediate reunions:',coalescence)
```

verify_r39.py

```
"""Finite consistency checks of the R39 reconstruction, not a universal proof."""
from collections import Counter
from random import Random
R=Random(39)
counts=Counter()
def E(n): return n^(n<<1)
def N(n): return E(n).bit_count()
def v(n): return (n&-n).bit_length()-1
def bit(n,j): return (n>>j)&1
def F(n):
    a=3*n+1; return a>>v(a)
def m(mask): return (mask&~(mask<<1)).bit_count()-1
def Z(n): return E(n).bit_length()-N(n)
def carry(q,w):
    c=0; D=0; i=0
    while q or w or c:
        D|=c<<i; c=((q&1)+(w&1)+c)>=2; q>>=1;w>>=1;i+=1
    return D
for a in range(1,512,2):
    for M in range(max(1,(a-1).bit_length()+1),max(1,(a-1).bit_length()+1)+8):
        p=a*((1<<M)-1)
        assert E(p)==(1+(1<<M))*(1+E(a-1)) and N(p)==2+2*N(a-1)
        counts['mirror']+=1
for b in range(1,256,2):
    for T in range(9):
        y=b; S=0; hist=[]
        for t in range(T+1):
            hist.append((y,S)); S+=v(3*y+1); y=F(y)
        L=max(S+y.bit_length()+1 for y,S in hist); M=max(1,(3**T-1).bit_length()+1)
        x=b+(1<<L)*((1<<M)-1)
        for t,(y,S) in enumerate(hist):
            assert x==y+(1<<(L-S))*3**t*((1<<M)-1)
            assert E(x)==E(y)+(1<<(L-S))*(1+(1<<M))*(1+E(3**t-1))
            if t<T: assert v(3*x+1)==v(3*y+1)
            counts['embedded times']+=1; x=F(x)
```

```
for _ in range(25000):
    y=R.getrandbits(R.randrange(1,65)); W=R.getrandbits(R.randrange(1,65)); k=R.randrange
    (1,65)
    q,r=divmod(y,1<<k); b=bit(r,k-1); D=carry(q,W)
    assert N(y+(W<<k))-N(y)==N(q+W)-N(q)-2*b*((q+W)%2-q%2)
    assert E(y+(W<<k))^E(y)^(E(W)<<k)==E(D)<<k
    counts['general interface']+=1
for W in range(1,1024,2):
    h=v(W+1); eta=bit(W,h+1); V=(W+1)>>h
    for k in range(1,8):
        for b in (0,1):
            y=(1<<k)+(b<<(k-1))
            assert N(y+(W<<k))-N(y)==N(W)-2+2*b-2*eta
            assert E(y+(W<<k))^E(y)^(E(W)<<k)==(1<<(k+1))+(1<<(k+h+1))
            assert E((1<<k)*(W+1)-1)==1+(1<<k)*(E(W)-1)
            assert E(W)==(E(V)<<h)^ (1+(1<<(h+1)))
            assert N(V)==N(W)-2*eta
            counts['overlap arithmetic and transport']+=1
for t in range(1,1025):
    q=v(3**t-1)
    for M in (max(1,q-1),q,q+1,q+2):
        W=3**t*((1<<M)-1); h=v(W+1)
        assert h==min(M,q) if M!=q else h>=M+1
        if M>=q+2: assert bit(W,h+1)==int(t%2==1 or (t>>v(t))%4==1)
        counts['time-width resonance']+=1
for b in range(1,512,2):
    if b>1:
        X=b if N(b)>2 else F(b); assert N(X)>=4
        counts['top pair extraction']+=1
    for g0 in range(1,13):
        L=b.bit_length()+g0
        for a in (1,3,7,31,R.randrange(1,256,2)):
            x=b+(a<<L); X=x; y=b; S=0;t=0
            while True:
                k=L-S; g=k-y.bit_length(); W=3**t*a
                assert x==y+(W<<k)
                if g<=0:
                    assert (g in (0,-1)) and (g0+1)//2<=t<=g0
                    h=v(W+1); eta=bit(W,h+1); loss=2 if g==0 else 2+2*eta
                    assert N(x)-N(y)-N(W)==-loss
                    assert N(x)-N(X)==N(y)-N(b)+N(W)-N(a)-loss
                    if g==-1:
                        r=y-(1<<k); J=k+h; A=(W+1)>>h
                        assert 0<r<(1<<(k-1)) and J-r.bit_length()>=h+1 and 3*r<y
                    else:
                        j=k-1
                        while j>0 and bit(y,j-1): j-=1
                        r=y-((1<<k)-(1<<j)); J=j; A=(1<<(k-j))*(W+1)-1
                    assert x==r+(A<<J)
                    if r:
                        assert r%2 and J-r.bit_length()>=1 and 3*r<y
                        assert E(x)==E(r)+(E(A)<<J)
                        if t<=2: assert r<b
                        counts['positive restart']+=1
                    counts['actual ordinary contact']+=1;break
                assert N(x)==N(y)+N(W)
                s=v(3*y+1)
                if s>=k:
                    assert g==1 and s==y.bit_length()+1 and F(y)==1 and t+1<=g0
                    assert v(3*x+1)!=s
                    counts['early exponent branch']+=1;break
                assert v(3*x+1)==s
                newy=F(y); newg=k-s-newy.bit_length(); assert g-newg in (1,2)
                x=F(x);y=newy;S+=s;t+=1
```

```
            counts['ordinary contact trials']+=1
assert F(F(F(2159)))==7289
for sigma in (1,-1):
    def mask(x): return E(x) if sigma==1 else E(x-1)+1
    def z(x): return mask(x).bit_length()-mask(x).bit_count()
    for b in range(1,512,2):
        for d0 in range(1,13):
            for q in (1,3,7,R.randrange(1,256,2)):
                k=mask(b).bit_length()+d0; x=(q<<k)+sigma*b; n=x; S=0; t=0; y=b; oldd=d0
                while True:
                    K=k-S; u=3**t*q; l,h=mask(y),mask(u); d=K-l.bit_length()
                    assert x==(u<<K)+sigma*y
                    if t: assert oldd-d in (1,2)
                    bal=z(u)-z(q)-S-l.bit_count()+mask(b).bit_count()
                    if d<=0:
                        assert d in (0,-1) and K>=1 and (d0+1)//2<=t<=d0
                        assert E(x)==l^(h<<K)
                        I=int(d==-1)
                        assert Z(x)-Z(n)==bal+2*I
                        assert m(E(x))==m(l)+m(h)+I*(bit(l,K-1)+bit(h,1)-1)
                        counts[f'signed contact {sigma:+}']+=1;break
                    assert E(x)==l+(h<<K)
                    assert Z(x)-Z(n)==bal and m(E(x))==m(l)+m(h)+1
                    counts[f'signed separated times {sigma:+}']+=1
                    a=3*y+sigma;s=v(a);assert v(3*x+1)==s
                    oldd=d;x=F(x);y=a>>s;S+=s;t+=1
for label,n in counts.items(): print(f'{label}: {n}')
print('All checks passed.')
```

verify_r40.py

```
"""Finite reproducibility checks for Paper E, R40. No full-orbit claim."""
from collections import Counter
from fractions import Fraction
from itertools import combinations, product


def E(n):
    assert n >= 0
    return n ^ (2*n)


def nu(n):
    assert n != 0
    return (n & -n).bit_length()-1


def F(n, c=1):
    x = 3*n+c
    return x >> nu(x)


def hist(n, T, c=1):
    xs, S = [n], [0]
    for _ in range(T):
        s = nu(3*xs[-1]+c)
        xs.append((3*xs[-1]+c)>>s)
        S.append(S[-1]+s)
    return xs, S


def pairs(n):
    out = []
    i = 0
```

```
    while n >> i:
        if not ((n >> i) & 1):
            i += 1
            continue
        j = i+1
        while (n >> j) & 1:
            j += 1
        out.append((i, 2**j-2**i))
        i = j
    return out


def corr(A,D,S):
    assert nu(A+D)==S
    if S<2:
        return
    mask=(1<<(S+1))-1
    assert ((E(A)^E(D))&mask)==2+(1<<S)
    assert abs((E(A)&mask).bit_count()-(E(D)&mask).bit_count())<=2
    assert ((A+D)^A^D)&mask==mask-1
    def alpha(x,j):
        return (0 if j==0 else (x>>(j-1))&1)-((x>>j)&1)
    assert alpha(A,0)+alpha(D,0)==-2
    assert alpha(A,1)+alpha(D,1)==1
    assert all(alpha(A,j)+alpha(D,j)==0 for j in range(2,S))
    assert sum((alpha(A,j)+alpha(D,j))*2**j for j in range(S))==0


counts=Counter()
for b in range(1,512,2):
    ys,S=hist(b,12)
    J=[s+y.bit_length() for s,y in zip(S,ys)]
    for a in range(1,16,2):
        mult=[]
        for t in range(1,13):
            d=J[t]-J[t-1]
            assert d in (1,2)
            k=ys[t].bit_length(); W=3**t*a
            L=J[t]; X=b+2**L*a
            actual,R=hist(X,t)
            assert R==S[:t+1]
            assert actual[-1]==ys[t]+2**k*W
            assert E(actual[-1])==E(ys[t])-2**k+2**k*(E(W)-1)
            assert E(actual[-1]).bit_count()==E(ys[t]).bit_count()+E(W).bit_count()-2
            assert all(L>J[j] for j in range(t))
            counts['adjacency_heights']+=1
            if d==2:
                L-=1; X=b+2**L*a
                actual,R=hist(X,t)
                if ys[t]>1:
                    assert R==S[:t+1]
                    assert actual[-1]==ys[t]+2**(k-1)*W
                    counts['overlap_heights']+=1
                else:
                    assert R[:-1]==S[:t] and R[-1]!=S[t]
                    counts['early_branch_heights']+=1
            mult.append(1+int(d==2 and ys[t]>1))
        for event in ([int((3**t*a)%8==1) for t in range(1,13)], [int(t%3==0) for t in range
    (1,13)], [1]*12, [0]*12):
            p=Fraction(sum(event),12); q=Fraction(sum(c*x for c,x in zip(mult,event)),sum(mult
    ))
            assert p/(2-p)<=q<=2*p/(1+p)
            counts['selection_bias']+=1

```

```
for x in range(256):
    for y in range(256):
        q=(x+y)^x^y
        assert E(x+y)==E(x)^E(y)^E(q)
        carry=0
        for i in range(10):
            assert ((q>>i)&1)==carry
            carry=(((x>>i)&1)+((y>>i)&1)+carry)//2
        counts['carry_masks']+=1
assert [E(x).bit_count() for x in (7,14,21,99,149)]==[2,2,6,4,8]
assert F(99)==149
for n in range(1,512,2):
    for c in (1,3,5,7,9):
        xs,S=hist(n,10,c)
        for t in range(4):
            ps=pairs(xs[t]); assert sum(p for _,p in ps)==xs[t]
            def D(t,u): return sum(2**(S[t+j]-S[t])*3**(u-1-j) for j in range(u))
            for u in range(1,5):
                A=3**u*xs[t]; d=D(t,u); s=S[t+u]-S[t]
                assert A+c*d==2**s*xs[t+u]
                assert A==sum(3**u*p for _,p in ps)
                corr(A,c*d,s)
                mod=2**(s+1)
                assert xs[t]%mod==(pow(3,-u,mod)*(2**s-c*d))%mod
                counts['time_injection_windows']+=1
                for v in range(1,3):
                    assert D(t,u+v)==3**v*d+2**s*D(t+u,v)
                    counts['cocycle_windows']+=1

for n in range(1,512,2):
    ps=pairs(n); assert ps[0][0]==0
    sels=list(product((0,1),repeat=len(ps)-1))
    bs=[ps[0][1]+sum(e*p for e,(_,p) in zip(sel,ps[1:])) for sel in sels]
    assert (any(l==2 for l,_ in ps[1:]))==(n%8==5)
    for i,j in combinations(range(len(bs)),2):
        l=next(l for e,f,(l,_) in zip(sels[i],sels[j],ps[1:]) if e!=f)
        assert nu(bs[i]-bs[j])==l
        for T in (0,1,7):
            assert nu(3**T*(bs[i]-bs[j]))==l
            counts['extraction_distances']+=1
    for T in (0,1,7):
        for k in range(1,13):
            q=sum(l<k for l,_ in ps[1:]); assert q<=(k-1)//2
            freq=Counter(E(3**T*b)%(2**k) for b in bs)
            assert len(freq)==2**q and set(freq.values())=={2**(len(ps)-1-q)}
            counts['extraction_windows']+=1
    # Check the residue-class statement and the switch by extraction.
    for i,j in combinations(range(len(bs)),2):
        changed=any(e!=f and l==2 for e,f,(l,_) in zip(sels[i],sels[j],ps[1:]))
        assert ((bs[i]%8 in (5,7))!=(bs[j]%8 in (5,7)))==changed
        counts['class_switches']+=1
for b in range(1,64,2):
    for k in range(3,11):
        mod=2**k
        hits=any((pow(3,T,mod)*b+1)%mod==0 for T in range(2**(k-2)))
        assert hits==(b%8 in (5,7))
        counts['high_valuation_residue_classes']+=1
for name,count in counts.items():
    print(f'{name}: {count}')
print('all R40 finite checks passed')
```