Pair Predicate Decomposition of Collatz-type Maps (xn+1)/2^d and the Structural Closure of 3n+1

Yukihiro Honda

February 2026 (v5.1)

Abstract

We decompose the binary representation of a natural number n into 2-bit pairs and introduce a system of 16 Boolean predicates defined on each pair, with the left-bit projection m4 and right-bit projection m6 serving as basis. Within this framework, we decompose the additive structure of Collatz-type maps T(n)=(xn+1)/2^d and classify the carry behavior at each pair position as Generate/Propagate/Kill (GPK).

Our main result is the following. For x=3, the GPK of the upper stage (m4-stage) of the addition coincides, at every pair position, with the intra-pair predicates m2(AND) and m7(XOR) of the input n itself (Theorem B). That is, the carry structure of 3n+1 can be read directly from the encoding of n. We call this property **structural closure**. In contrast, we establish that this coincidence fails for all x≥5 (Theorem C). For x≥5, the GPK depends on correlations between adjacent pairs, but since all 16 predicates are per-pair independent, they cannot express inter-pair correlations.

From these results it follows that the m4-stage carry structure of the map closes within the predicate space only for x=3 (Classification Theorem). This per-pair closure of the m4-stage, combined with exhaustive finite case analysis of the scan and re-pairing transitions (Tables 5.1, 5.2), yields a deterministic finite-state system in which the complete one-step dynamics are algebraically determined. Within this system, the activation condition for carry generation (G-dominance) and the activation result (carry-induced destruction of G-blocks) are inseparably coupled within the same algebraic identity; no configuration can maintain G-dominance indefinitely, rendering divergence of 3n+1 trajectories structurally impossible (Proposition 5.4). In contrast, for x≥5, where this closure fails, non-trivial cycles are known to exist.

1. Introduction

1.1 Background

The Collatz conjecture asserts that for any positive integer n, repeated application of the map

$$n \mapsto \{n/2 \text{ if } n \text{ is even} \\ 3n+1 \text{ if } n \text{ is odd}$$

eventually reaches 1 in finitely many steps. Since its formulation in 1937, it has resisted resolution, famously prompting Erdős to remark that "mathematics is not yet ready for such problems" [1].

The focus of this paper is not the resolution of the Collatz conjecture itself, but a more fundamental question: **What is the internal structure of the map $T(n)=(xn+1)/2^d$**, and **what is special about the parameter x=3**?

The variant 5n+1 is a well-studied counterpart. Unlike 3n+1, it does not exhibit convergence to 1 and possesses multiple non-trivial cycles, such as 13→33→83→13 and 27→17→43→27 [2]. Why the dynamical behavior differs so sharply as a function of x is not well understood.

1.2 Prior Work

The analysis of Collatz-type maps has a long history. Lagarias [1] provides a comprehensive survey of the problem's breadth and difficulty. Wirsching [3] developed binary-representation-based analysis and studied the statistical properties of the Collatz map.

In practice, the odd-to-odd transition is captured by the Syracuse function

$$T(n)=(3n+1)/2^{v_2(3n+1)}$$

where $v_2(m)$ denotes the 2-adic valuation (number of trailing zeros) of m. The Syracuse function streamlines trajectory tracking by eliding even steps, but it does not penetrate the internal structure of the map. The arithmetic operation 3n+1 remains a black box, and the trailing-zero count can only be determined after computation.

Binary-based approaches include Terras's stopping time analysis [4] and Everett's binary-operation description [5], but both presuppose the arithmetic execution of 3×n.

1.3 Our Approach

This paper treats natural numbers not as arithmetic objects but as **information structures**.

Specifically, we decompose the binary representation of n into a sequence of 2-bit pairs and introduce a system of 16 Boolean predicates defined on each pair (a_i,b_i). Of these 16 predicates, m4 (the left-bit projection of each pair) and m6 (the right-bit projection) form a basis; the remaining 14 are derived from m4 and m6 by bitwise Boolean operations.

Under this basis decomposition, the operation xn+1 is described as follows. Decomposing xn+1=(x−1)n+n+1, the term (x−1)n corresponds to a left shift of the bit string of n. The shift amount $s=\log_2(x-1)$ determines which bit positions of m4/m6 are referenced. Based on this reference pattern, the addition at each pair position is decomposed into bit-level Boolean operations (AND, XOR, and carry propagation).

The core discovery is that when x=3, the shift amount is 1 bit, and the inputs to one of the two addition stages (the m4-stage) coincide with the pair (a_i,b_i) of n itself. Consequently, the positions of carry Generation and Propagation can be read directly as the intra-pair predicates $m2=AND(a_i,b_i)$ and $m7=XOR(a_i,b_i)$.

For x≥5, this degeneracy does not occur, and references to adjacent pairs become necessary. This difference arises from an essential property of the 16 predicates—per-pair independence—and is confirmed by exhaustive verification over all 784 predicate combinations.

1.4 Main Results

The main results of this paper are summarized as follows.

> **Theorem A (Unified Algorithm).** For any odd number n and parameter x of the xn+1 type, the next odd number $n'=T(n)$ is computable by an O(k) bit-scan of m4/m6, where $k=\lceil\log_2(n)\rceil/2$ is the number of pairs. The multiplication xn is decomposed into index shifts of the reference pattern and intra-pair 1-bit additions.

> **Theorem B (Structural Privilege of 3n+1).** In the m4-stage carry structure of 3n+1, the Generate positions coincide with m2(AND) and the Propagate positions coincide with m7(XOR), at every pair position.

> **Theorem C (Limitation of the 16 Predicates).** For x≥5, the overall carry structure (G,P) of xn+1 cannot be expressed as any combination of AND, OR, XOR of any two of the 16 predicates m1–m16 of n.

Combining Theorems B and C, x=3 is the **unique** parameter for which the m4-stage carry structure of the map closes within the complete space of 16 predicates. This m4-stage closure serves as the foundation: the four pair types (G, P, P, K) determined by m2/m7/m9 become the input alphabet of a deterministic finite-state transducer (DFST) that captures the complete scan dynamics (§5.7).

Proposition 5.4 (Self-annihilation of G-dominance). In the closed GPK system of 3n+1, the activation condition for carry generation (G-dominance) and the activation result (carry-induced destruction of G-blocks) are inseparably coupled within the same algebraic identity 3n+1=2n+n+1. No configuration can maintain G-dominance indefinitely. Exhaustive finite case analysis of scan transitions (Table 5.1, 16 patterns) and re-pairing transitions (Table 5.2, 16 patterns) yields Lemmas 5.1–5.8, which collectively establish that divergence of 3n+1 trajectories is structurally impossible (Remark 5.3).

1.5 Organization

Section 2 defines the pair predicate system and establishes basis properties. Section 3 derives the decomposition of xn+1 into pair-wise addition and exhibits the x-dependence of reference patterns. Section 4 formulates the carry problem and its resolution by scanning. Section 5 establishes the structural privilege of 3n+1 and the limitation theorem for the 16 predicates; §5.7 presents the complete finite case analysis (§5.7.1–5.7.6), including the DFST permutation property, the four-phase closure of the one-step map, and the structural impossibility of divergence via exhaustive scan and re-pairing transition tables. Section 6 discusses the m4↔m6 exchange principle under division by 2. Section 7 provides a methodological note. Section 8 applies the framework to cycle analysis. Section 9 presents discussion and open problems. Section 10 states the conclusion.

2. Pair Predicate System

This section rigorously defines the 2-bit pair decomposition of natural numbers and the system of 16 Boolean predicates defined thereon. We show that m4 and m6 form a basis, and confirm the completeness and per-pair independence of the system.

2.1 Two-Bit Pair Decomposition

Definition 2.1 (Even-digit padding). For a positive integer n, write its binary representation as $\beta(n)=d_{L-1}d_{L-2}\ldots d_1 d_0$, where $d_{L-1}=1$ (most significant bit) and $L=\lfloor \log_2 n\rfloor+1$. When L is odd, prepend a single zero to make the digit count even:

$$\beta(n)=\{\beta(n) \text{ if } L \text{ is even}$$
$$0\beta(n) \text{ if } L \text{ is odd}$$

Denote the padded digit count by 2k. Then $k=\lceil L/2\rceil$ is the *pair count*.

Definition 2.2 (Pair sequence). Group the padded binary representation $\beta(n)=s_1 s_2 \ldots s_{2k}$ into consecutive 2-bit pairs from the left:

$$P(n)=((a_0,b_0),\ (a_1,b_1),\ \ldots,\ (a_{k-1},b_{k-1}))$$

where $a_i=s_{2i+1}$ and $b_i=s_{2i+2}$ (1-indexed). That is, a_i is the left bit and b_i is the right bit of each pair.

Example 2.1. $n=27=11011_2$ → padding: 011011 → pairs: ((0,1),(1,0),(1,1)). Pair count k=3.

Example 2.2. $n=17=10001_2$ → padding: 010001 → pairs: ((0,1),(0,0),(0,1)). Pair count k=3.

Definition 2.3 (Zipper structure and reconstruction). To reconstruct the original binary representation from the pair sequence P(n), interleave the left and right bits:

$$\beta(n)=a_0\ b_0\ a_1\ b_1\ \ldots\ a_{k-1}\ b_{k-1}$$

We call this interleaving the *zipper structure*. Reconstruction is unique: removing the leading-zero padding recovers n uniquely.

2.2 Definition of m4 and m6

Definition 2.4 (Projections m4, m6). For the pair sequence $P(n)=((a_0,b_0),\ldots,(a_{k-1},b_{k-1}))$, define two bit strings:

$$m4(n)=(a_0,a_1,\ldots,a_{k-1})$$

$$m6(n)=(b_0,b_1,\ldots,b_{k-1})$$

m4 is the left-bit projection (LEFT) and m6 is the right-bit projection (RIGHT) of each pair.

Proposition 2.1 (Reconstruction). The number n is uniquely reconstructed from the pair (m4(n),m6(n)).

Derivation. The zipper structure recovers β(n), and removal of the leading zero uniquely determines n. □

Example 2.3. n=27: m4 =(0,1,1), m6 =(1,0,1). Zipper: 0 1 1 0 1 1=011011_2=27. √

2.3 The 16 Predicates

There are $2^{2^2}=16$ Boolean functions of two inputs, forming a complete system. For each pair (a_i,b_i), we define the 16 predicates m1–m16 as follows.

Definition 2.5 (The 16 predicates).

Pred.	Name	Definition f(a,b)	Truth values (0,0)(0,1)(1,0)(1,1)
m1	FALSE	0	0 0 0 0
m2	AND	$a\wedge b$	0 0 0 1
m3	L>R	$a\wedge\neg b$	0 0 1 0
m4	LEFT	a	0 0 1 1
m5	R>L	$\neg a\wedge b$	0 1 0 0
m6	RIGHT	b	0 1 0 1
m7	XOR	$a\oplus b$	0 1 1 0
m8	OR	$a\vee b$	0 1 1 1
m9	NOR	$\neg a\wedge\neg b$	1 0 0 0
m10	XNOR	$\neg(a\oplus b)$	1 0 0 1
m11	NOT_R	$\neg b$	1 0 1 0
m12	R→L	$a\vee\neg b$	1 0 1 1
m13	NOT_L	$\neg a$	1 1 0 0
m14	L→R	$\neg a\vee b$	1 1 0 1
m15	NAND	$\neg(a\wedge b)$	1 1 1 0
m16	TRUE	1	1 1 1 1

Each predicate is applied bitwise to the pair sequence of n:

$$m_j(n)=(f_j(a_0,b_0), f_j(a_1,b_1), \ldots, f_j(a_{k-1},b_{k-1}))$$

Remark (on the role of the complete system). In the subsequent analysis, only a handful of these 16 predicates—most notably m2 (AND), m7 (XOR), m4 (LEFT), m6 (RIGHT), and m9 (NOR)—appear explicitly in the main results. The reader may therefore wonder why all 16 are catalogued here. The reason is that the completeness of the system is indispensable for the *exhaustive verification* carried out in Section 5. To establish Theorem C, we must show that *no* pair of predicates from the full 16-predicate space can express the carry structure of xn+1 for x≥5. This impossibility claim requires testing all (16/2)×{AND,OR,XOR}=360 candidate combinations (and indeed the full 784-combination search described in §5.5), which is only meaningful when the predicate space is verifiably complete. Without defining the entire system, one could not rule out the possibility that some overlooked predicate might succeed where the obvious ones fail. In a sense, the predicates that go unused in our theorems earn their place by *failing* the exhaustive search—their collective failure is precisely what certifies that the structural closure of 3n+1 (Theorem B) is unique. We ask the reader's indulgence for this apparent redundancy; it is, in fact, the backbone of the impossibility argument.

2.4 Basis Theorem

Theorem 2.1 (m4/m6 basis). All 16 predicates m1–m16 can be expressed by bitwise Boolean operations on m4 and m6.
Derivation. We give the expression for each predicate directly:

Predicate	Expression in m4/m6
m1	0 (constant zero)
m2	m4 AND m6
m3	m4 AND (NOT m6)
m4	m4 (basis)
m5	(NOT m4) AND m6
m6	m6 (basis)
m7	m4 XOR m6
m8	m4 OR m6
m9	(NOT m4) AND (NOT m6)
m10	NOT (m4 XOR m6)
m11	NOT m6
m12	m4 OR (NOT m6)
m13	NOT m4
m14	(NOT m4) OR m6
m15	NOT (m4 AND m6)
m16	1 (constant one)

All operations are applied bitwise. □

Remark 2.1. The choice of basis is not unique. For instance, {m3($a \wedge \neg b$), m6(b)} or {m4(a), m8($a \vee b$)} also generate all 16 predicates. However, m4/m6 is the unique basis pair that directly preserves the numerical information of n, requiring only the zipper structure to reconstruct n with no additional operations (direct reconstructibility). Other basis pairs can also reconstruct n—for example, the complement pair (m13,m11)=($\neg a,\neg b$) recovers n via bitwise negation, and (m13,m6)=($\neg a$,b) via negation of the left bits alone—but all such pairs require additional logical operations.

Note that not every pair of predicates forms a basis. For example, {m2(AND), m7(XOR)} are both symmetric in (a,b), meaning f(a,b)=f(b,a). Since any Boolean composition of symmetric functions remains symmetric, this pair can only generate the 8 symmetric predicates (m1, m2, m7, m8, m9, m10, m15, m16) and cannot produce asymmetric predicates such as m4(a,b)=a.

2.5 Per-Pair Independence

Definition 2.6 (Per-pair independence). A function $g:\{0,1\}^{2k}\to\{0,1\}^k$ is *per-pair independent* if the i-th component of the output depends only on the i-th pair (a_i,b_i) of the input:

$$g(n)_i=h(a_i,b_i) \text{ for some } h:\{0,1\}^2\to\{0,1\}$$

Proposition 2.2. All 16 predicates m1–m16 are per-pair independent.

Derivation. By Definition 2.5, each $f_j(a_i,b_i)$ is a function of (a_i,b_i) alone. □

Proposition 2.3 (Closure). The bitwise AND, OR, XOR, and NOT of per-pair independent functions are again per-pair independent.

Derivation. If $g_1(n)_i=h_1(a_i,b_i)$ and $g_2(n)_i=h_2(a_i,b_i)$, then $(g_1 \text{ AND } g_2)(n)_i=h_1(a_i,b_i)\wedge h_2(a_i,b_i)$ depends only on (a_i,b_i). The other operations are analogous. □

Corollary 2.1. Any Boolean composition of m1–m16 is per-pair independent and cannot express correlations between positions i and j ($i\neq j$).

This property is used essentially in Section 5 to establish that the carry structure of 5n+1 cannot be described by the 16 predicates.

2.6 Complement Pairs

The 16 predicates possess a natural symmetry.

Definition 2.7 (Complement pair). Predicates m_j and m_{17-j} are called a complement pair. They satisfy $m_j(a,b)+m_{17-j}(a,b)=1$.

Pair	Predicate	Complement	Type
1	m1 (FALSE)	m16 (TRUE)	Constant
2	m2 (AND)	m15 (NAND)	Logical
3	m3 (L>R)	m14 (L→R)	Implication
4	m4 (LEFT)	m13 (NOT_L)	Projection
5	m5 (R>L)	m12 (R→L)	Implication
6	m6 (RIGHT)	m11 (NOT_R)	Projection
7	m7 (XOR)	m10 (XNOR)	Exclusive
8	m8 (OR)	m9 (NOR)	Logical

Proposition 2.4. $m_j(n) = \neg\, m_{17-j}(n)$ holds for all n.

Derivation. Verify $f_j(a,b)+f_{17-j}(a,b)=1$ for each row of the truth table. □

2.7 Summary of This Section

Established results:

1. A natural number n is uniquely decomposed into a 2-bit pair sequence P(n), and is completely reconstructed from m4 (left-bit string) and m6 (right-bit string).

2. The 16 two-input Boolean functions on pairs (a_i,b_i) form a complete system, all expressible as bitwise operations on m4 and m6.

3. All 16 predicates are per-pair independent, and this property is preserved under Boolean composition. Consequently, correlations between distinct pair positions cannot be described within the 16-predicate framework.

4. The 16 predicates form 8 complement pairs, with predicate numbers j and 17−j paired.

Section 3 derives how xn+1-type maps are described on this predicate system.

3. Decomposition of xn+1

This section decomposes the Collatz-type map xn+1 into pair-wise addition. We show that the shift amount $s=\log_2(x-1)$ determines the reference pattern on the pair structure, and derive the cases x=3 and x=5 in detail.

3.1 Decomposition Principle

Proposition 3.1 (Additive decomposition). For a positive odd number n and parameter x:

$$xn+1=(x-1)n+n+1$$

The term $(x-1)n$ corresponds to a left shift of the binary representation of n.

Derivation. When $x-1$ is a power of 2, $(x-1)n=2^s n$ is exactly an s-bit left shift of n. This paper treats the case $x\in\{3,5,9,17,\dots\}$ where $x-1$ is a power of 2. □

Remark 3.1. When $x-1$ is not a power of 2 (e.g., $x=7$, $x-1=6=2\cdot3$), additional additions beyond a simple shift arise. The framework is in principle extensible, but for clarity we focus on $x-1=2^s$. The primary objects of study are x=3 (s=1) and x=5 (s=2).

3.2 Shift and Pair Structure

Let $P(n)=((a_0,b_0),\dots,(a_{k-1},b_{k-1}))$ be the pair sequence of n, with corresponding binary representation $\beta(n)=a_0\ b_0\ a_1\ b_1\ \dots\ a_{k-1}\ b_{k-1}$. The binary representation of $2^s n$ appends s zeros at the end.

Case s=2 (x=5, 4n):

$$P(4n)=((a_0,b_0),\ (a_1,b_1),\ \dots,\ (a_{k-1},b_{k-1}),\ (0,0))$$

A zero pair (0,0) is appended. **Pair boundaries are preserved.**

$$m4(4n)=(a_0,a_1,\dots,a_{k-1},0)=m4(n)\|0$$

$$m6(4n)=(b_0,b_1,\dots,b_{k-1},0)=m6(n)\|0$$

where ‖ denotes bit-string concatenation.

Case s=1 (x=3, 2n):

$$\beta(2n)=a_0\ b_0\ a_1\ b_1\ \dots\ a_{k-1}\ b_{k-1}\ 0$$

The digit count becomes odd, so a leading zero is prepended:

$$\beta(2n)_{padded}=0\ a_0\ b_0\ a_1\ b_1\ \ldots\ a_{k-1}\ b_{k-1}\ 0$$

Grouping into pairs:

$$P(2n)=((0,a_0),\ (b_0,a_1),\ (b_1,a_2),\ \ldots,\ (b_{k-1},0))$$

Pair boundaries shift by 1 bit. As a result:

$$m4(2n)=(0,b_0,b_1,\ldots,b_{k-1})=0\|m6(n)$$

$$m6(2n)=(a_0,a_1,\ldots,a_{k-1},0)=m4(n)\|0$$

Proposition 3.2 (m4/m6 crossover under 2n). m4(2n) equals m6(n) with a leading zero prepended, and m6(2n) equals m4(n) with a trailing zero appended. That is, the 2n operation swaps the roles of m4 and m6.

3.3 Pair Addition Formulation

We describe $xn+1=2^s n+n+1$ as addition over pair sequences. Below we use LSB (least significant pair) ordering. Let the pair sequence of n in LSB order be $(a_0,b_0),(a_1,b_1),\ldots$

At pair position i, the contribution from n is (a_i,b_i), and the contribution from $2^s n$ is determined by the shift amount. Addition at each pair position proceeds in two stages: right bit (m6) then left bit (m4).

Definition 3.1 (Two-stage pair addition). At pair position i:
m6-stage (right bit):

$$sum_{R,i}=r_i^{(6)}+b_i+c_{in,i}$$

where $r_i^{(6)}$ is the right bit of $2^s n$ at pair position i, and $c_{in,i}$ is the input carry to pair i.

$$new\_m6_i=sum_{R,i} \bmod 2,\ c_{mid,i}=\lfloor sum_{R,i}/2\rfloor$$

m4-stage (left bit):

$$sum_{L,i}=r_i^{(4)}+a_i+c_{mid,i}$$

where $r_i^{(4)}$ is the left bit of $2^s n$ at pair position i.

$$new\_m4_i=sum_{L,i} \bmod 2,\ c_{out,i}=\lfloor sum_{L,i}/2\rfloor$$

Carry propagation: $c_{in,0}=1$ (effect of +1), $c_{in,i+1}=c_{out,i}$.

3.4 Reference Pattern for x=5

In 5n+1=4n+n+1, the term 4n is a 1-pair shift (§3.2), so at pair position i:

$$r_i^{(4)}=a_{i-1},\ r_i^{(6)}=b_{i-1}\ (a_{-1}=b_{-1}=0)$$

Proposition 3.3 (Reference pattern for 5n+1). In the pair addition for 5n+1:

$$\text{m6-stage inputs:}(b_{i-1},\ b_i) \text{ — adjacent m6 bits}$$

$$\text{m4-stage inputs:}(a_{i-1},\ a_i) \text{ — adjacent m4 bits}$$

Derivation. Substitute $r_i^{(6)}=b_{i-1}$ into the m6-stage: $sum_{R,i}=b_{i-1}+b_i+c_{in,i}$. Substitute $r_i^{(4)}=a_{i-1}$ into the m4-stage: $sum_{L,i}=a_{i-1}+a_i+c_{mid,i}$. □

Remark 3.2. For 5n+1, both the m6-stage and the m4-stage have isomorphic reference patterns (adjacent same-type reference). This is because s=2 is even (§3.6 generalizes).

3.5 Reference Pattern for x=3

In 3n+1=2n+n+1, the term 2n is a 1-bit shift. By §3.2, and tracking the pair boundary shift precisely in LSB order:

$$r_i^{(6)}=a_{i-1}\ (a_{-1}=0),\ r_i^{(4)}=b_i$$

Proposition 3.4 (Reference pattern for 3n+1). In the pair addition for 3n+1:

$$\text{m6-stage inputs:}(a_{i-1},\ b_i) \text{ — cross-reference between m4 of the previous pair and m6 of the current pair}$$

$$\text{m4-stage inputs:}(b_i,\ a_i)=(a_i,\ b_i) \text{ — within the current pair (order is immaterial by commutativity)}$$

Derivation. Substitute $r_i^{(6)}=a_{i-1}$ into the m6-stage: $sum_{R,i}=a_{i-1}+b_i+c_{in,i}$. Substitute $r_i^{(4)}=b_i$ into the m4-stage: $sum_{L,i}=b_i+a_i+c_{mid,i}=a_i+b_i+c_{mid,i}$. □

Key observation. The m4-stage inputs (a_i,b_i) are precisely pair i of n itself. This coincidence is the foundation of Theorem B in Section 5.

3.6 General Reference Patterns

The parity of the shift amount s qualitatively determines the reference pattern structure.

Proposition 3.5 (Classification of reference patterns). For $x-1=2^s$:
When s is even (s=2t): The shift corresponds to t full pairs. Pair boundaries are preserved.

$$r_i^{(4)}=a_{i-t},\ r_i^{(6)}=b_{i-t}$$

The m6-stage references m6 bits and the m4-stage references m4 bits (**isomorphic reference**).

When s is odd (s=2t+1): The shift is t pairs plus 1 bit, and pair boundaries shift by 1 bit.

$$r_i^{(4)}=b_{i-t},\ r_i^{(6)}=a_{i-t-1}$$

The m6-stage references m4-derived bits and the m4-stage references m6-derived bits (**cross-reference**).

Table 3.1: Principal reference patterns.

x	s	Pair shift	Boundary	m6-stage input	m4-stage input	Type
3	1	0+1bit	shifted	(a_{i-1},b_i)	(a_i,b_i)	cross + intra-pair
5	2	1 pair	preserved	(b_{i-1},b_i)	(a_{i-1},a_i)	isomorphic adjacent
9	3	1+1bit	shifted	(a_{i-2},b_i)	(a_i,b_{i-1})	cross + distant
17	4	2 pairs	preserved	(b_{i-2},b_i)	(a_{i-2},a_i)	isomorphic distant

3.7 Worked Example: n=27 under 5n+1

n=27: m4 =(0,1,1), m6 =(1,0,1). LSB order: a=(1,1,0), b=(1,0,1).

Pair 0 (LSB, i=0):

$$m6\text{-}stage{:}b_{-1}+b_0+c_{in}=0+1+1=2 \Rightarrow new\_m6_0=0,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_{-1}+a_0+c_{mid}=0+1+1=2 \Rightarrow new\_m4_0=0,\ c_{out}=1$$

Pair 1 (i=1):

$$m6\text{-}stage{:}b_0+b_1+1=1+0+1=2 \Rightarrow new\_m6_1=0,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_0+a_1+1=1+1+1=3 \Rightarrow new\_m4_1=1,\ c_{out}=1$$

Pair 2 (i=2):

$$m6\text{-}stage{:}b_1+b_2+1=0+1+1=2 \Rightarrow new\_m6_2=0,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_1+a_2+1=1+0+1=2 \Rightarrow new\_m4_2=0,\ c_{out}=1$$

Overflow pair (i=3):

$$m6\text{-}stage{:}b_2+0+1=1+0+1=2 \Rightarrow new\_m6_3=0,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_2+0+1=0+0+1=1 \Rightarrow new\_m4_3=1,\ c_{out}=0$$

Result (LSB order): ((0,0),(1,0),(0,0),(1,0)) → MSB order: ((1,0),(0,0),(1,0),(0,0))

$$5\times27+1=136=10001000_2\ ✓$$

3.8 Worked Example: n=27 under 3n+1

Same n=27: LSB order: a=(1,1,0), b=(1,0,1).

Pair 0 (i=0):

$$m6\text{-}stage{:}a_{-1}+b_0+c_{in}=0+1+1=2 \Rightarrow new\_m6_0=0,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_0+b_0+c_{mid}=1+1+1=3 \Rightarrow new\_m4_0=1,\ c_{out}=1$$

Pair 1 (i=1):

$$m6\text{-}stage{:}a_0+b_1+1=1+0+1=2 \Rightarrow new\_m6_1=0,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_1+b_1+1=1+0+1=2 \Rightarrow new\_m4_1=0,\ c_{out}=1$$

Pair 2 (i=2):

$$m6\text{-}stage{:}a_1+b_2+1=1+1+1=3 \Rightarrow new\_m6_2=1,\ c_{mid}=1$$

$$m4\text{-}stage{:}a_2+b_2+1=0+1+1=2 \Rightarrow new\_m4_2=0,\ c_{out}=1$$

Overflow pair (i=3):

$$m6\text{-}stage{:}a_2+0+1=0+0+1=1 \Rightarrow new\_m6_3=1,\ c_{mid}=0$$

$$m4\text{-}stage{:}0+0+0=0 \Rightarrow new\_m4_3=0,\ c_{out}=0$$

Result (LSB order): ((1,0),(0,0),(0,1),(0,1)) → MSB order: ((0,1),(0,1),(0,0),(1,0))

$$3\times27+1=82=01010010_2\ ✓$$

3.9 Summary of This Section

1. Decomposing xn+1=(x−1)n+n+1, the term (x−1)n is an $s=\log_2(x-1)$-bit left shift, and the parity of s determines whether pair boundaries are preserved or shifted.

2. Addition at pair position i has a two-stage structure (m6-stage and m4-stage), with each stage's inputs described as references to m4/m6 bits of n.

3. For x=5 (s=2, even): isomorphic adjacent reference of m6–m6 and m4–m4. For x=3 (s=1, odd): cross-reference of m4→m6 and m6→m4, with the m4-stage inputs coinciding with the current pair (a_i,b_i) itself.

4. The multiplication xn is never explicitly performed. The shift of (x−1)n is absorbed into index offsets of the reference pattern, and all operations reduce to bit references and 1-bit additions (with carry propagation).

4. The Carry Problem and Its Resolution

The previous section derived the two-stage pair addition structure. This section formulates the carry propagation structure, shows that it can be resolved by sequential scanning, and that parallelization is possible.

4.1 The Problem

The output of pair position i depends on the input carry $c_{in,i}$, which equals the output carry $c_{out,i-1}$ of pair i−1, which in turn depends on pair i−2, and so on. This chain runs from LSB (pair 0) to MSB:

$$c_{in,0}=1 \rightarrow c_{out,0}=c_{in,1} \rightarrow c_{out,1}=c_{in,2} \rightarrow \dots$$

Within each pair, two stages of carry exist:

$$c_{in,i} \rightarrow c_{mid,i} \rightarrow c_{out,i}$$

Therefore, to determine the output bits (new $m4_i$, new $m6_i$) of pair i, in principle all carries from pair 0 through pair i−1 must be settled.

Question. Is this sequential dependence essential? Can the carry at each pair be expressed in "closed form" from the m4/m6 bit strings?

4.2 Local Carry Classification (GPK)

Despite the sequential carry dependence, the *behavior of each pair with respect to carry* can be classified independently of the carry value.

Definition 4.1 (GPK classification). For two bits $p,q\in\{0,1\}$ and an input carry $c\in\{0,1\}$, the carry output of the addition p+q+c is $\lfloor(p+q+c)/2\rfloor$. We classify:

- **Generate (G):** p+q=2 (i.e., p=q=1). Carry output is 1 regardless of c.
- **Propagate (P):** p+q=1 (i.e., p≠q). Carry output equals c.
- **Kill (K):** p+q=0 (i.e., p=q=0). Carry output is 0 regardless of c.

In Boolean predicates:

$$G(p,q)=p\wedge q=AND(p,q)$$

$$P(p,q)=p\oplus q=XOR(p,q)$$

$$K(p,q)=\neg p\wedge\neg q=NOR(p,q)$$

G,P,K are mutually exclusive and exhaustive.

Remark 4.0. The GPK classification is not introduced for the first time in this paper. It is the standard technique for classifying carry behavior at each digit of carry-lookahead adders [6], with over half a century of history. Our contribution is to apply this classification to the pair addition of Collatz-type maps and to show that for 3n+1, the GPK coincides with the intra-pair predicates m2/m7/m9 of n.

4.3 GPK of Each Stage

Based on the reference patterns of Section 3, we describe the GPK of each stage.

m6-stage GPK:

$$G_{mid}[i]=AND(r_i^{(6)}, b_i),\ P_{mid}[i]=XOR(r_i^{(6)}, b_i)$$

m4-stage GPK:

$$G_{out}[i]=AND(r_i^{(4)}, a_i),\ P_{out}[i]=XOR(r_i^{(4)}, a_i)$$

For x=5:

$$G_{mid}[i]=b_{i-1}\wedge b_i,\ P_{mid}[i]=b_{i-1}\oplus b_i$$

$$G_{out}[i]=a_{i-1}\wedge a_i,\ P_{out}[i]=a_{i-1}\oplus a_i$$

For x=3:

$$G_{mid}[i]=a_{i-1}\wedge b_i,\ P_{mid}[i]=a_{i-1}\oplus b_i$$

$$G_{out}[i]=a_i\wedge b_i,\ P_{out}[i]=a_i\oplus b_i$$

4.4 Serial Composition of GPK Across Two Stages

The m6-stage and m4-stage are connected in series (the carry output of the m6-stage becomes the carry input to the m4-stage). Serial composition yields the overall GPK for pair i.

Proposition 4.1 (Serial composition). When two GPK stages (G_1,P_1) and (G_2,P_2) are connected in series (output of stage 1 feeds stage 2), the overall GPK is:

$$G_{12}=G_2\vee(P_2\wedge G_1),\ P_{12}=P_2\wedge P_1$$

Derivation. Case analysis on carry input c. If $G_2=1$: stage 2 generates, output is 1. If $G_2=0,P_2=1$: stage 2 passes through stage 1's output, which is 1 if $G_1=1$, c if $P_1=1$, or 0 if $K_1=1$. If $K_2=1$: output is 0. Overall Generate occurs when $G_2\vee(P_2\wedge G_1)$. Overall Propagate occurs when $P_2\wedge P_1$. □

Definition 4.2 (Pair GPK). The overall GPK for pair position i:

$$G_i = G_{out}[i] \lor (P_{out}[i] \land G_{mid}[i]),\ P_i = P_{out}[i] \land P_{mid}[i]$$

4.5 Carry Resolution by Scanning

Once the local GPK is computed for all pairs, carry determination reduces to a simple scan from the LSB.

Algorithm 4.1 (Carry scan).

```
Input:  Pair GPK sequence (G_0, P_0), ..., (G_{k-1}, P_{k-1})
        Initial carry c_0 = 1

c ← 1
for i = 0 to k-1:
    c_in[i] ← c
    c ← G_i OR (P_i AND c)
```

Proposition 4.2. Algorithm 4.1 determines all pair input carries in O(k) time.

Derivation. The loop runs k times; each iteration involves constant-time Boolean operations. □

Output computation after carry determination:

For each pair i, once $c_{in,i}$ is known:

$$c_{mid,i} = G_{mid}[i] \lor (P_{mid}[i] \land c_{in,i})$$

$$new\_m6_i = r_i^{(6)} \oplus b_i \oplus c_{in,i}$$

$$new\_m4_i = r_i^{(4)} \oplus a_i \oplus c_{mid,i}$$

This is also computed independently per pair in O(1).

4.6 Cost of Local GPK Computation

Key observation. Computing local GPK values $G_{mid}[i], P_{mid}[i], G_{out}[i], P_{out}[i]$ requires only **referencing m4/m6 bits and taking AND/XOR**. The reference targets differ by x (Table 3.1), but the type and count of operations are identical.

- x=3: References m4 bits at positions i and i−1, and m4/m6 bits at position i.

- x=5: References m4 bits at positions i and i−1 (m4 with m4), and m6 bits at positions i and i−1 (m6 with m6).
- In all cases, **each pair's local GPK is obtained in O(1) without additional arithmetic**.

Therefore, computing local GPK for all pairs is O(k), and the subsequent scan is also O(k), so complete carry determination is achieved in O(k) time.

4.7 Parallelization via GPK Tree

The scan is O(k) time but inherently sequential. In parallel computation environments, binary tree composition of GPK values compresses this to O(logk) depth.

Proposition 4.3 (Associativity of GPK composition). GPK composition (the rule of Proposition 4.1) is associative.

Derivation. For three stages A,B,C:

$$G_{(AB)C}=G_C\vee(P_C\wedge G_B)\vee(P_C\wedge P_B\wedge G_A)$$

$$G_{A(BC)}=(G_C\vee(P_C\wedge G_B))\vee((P_C\wedge P_B)\wedge G_A)=G_C\vee(P_C\wedge G_B)\vee(P_C\wedge P_B\wedge G_A)$$

Both coincide. For P: $P_{(AB)C}=P_C\wedge P_B\wedge P_A=P_{A(BC)}$. □

Algorithm 4.2 (GPK tree composition).

```
Input:  Pair GPK sequence (G_0, P_0), ..., (G_{k-1}, P_{k-1})
Output: Prefix GPK: (G_{0..j}, P_{0..j}) for all j

Level 0: Each pair’s (G_i, P_i)

Level 1: Compose adjacent pairs
  (G_{01}, P_{01}), (G_{23}, P_{23}), ...

Level 2: Compose further
  (G_{0123}, P_{0123}), ...

... completes in log_2(k) levels
```

Proposition 4.4. GPK tree composition completes O(k) total work in O(logk) depth. Each level uses only AND and OR operations.

Derivation. At each level l of the binary tree, $k/2^l$ compositions are performed, each involving a constant number of AND/OR operations. The total number of levels is $\lceil \log_2 k \rceil$. □

Remark 4.1. The GPK tree is an optimization for parallel computation and is not required for correctness verification. The sequential scan (Algorithm 4.1) determines all carries in O(k) time.

4.8 Worked Example: n=27, x=5 Carry Determination

We revisit the example of §3.7 from the GPK perspective.

n=27, m4 =(0,1,1), m6 =(1,0,1). LSB order: a=(1,1,0), b=(1,0,1).

Local GPK computation:

Pair i	b_{i-1}	b_i	G_{mid}	P_{mid}	a_{i-1}	a_i	G_{out}	P_{out}	G_i	P_i
0	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	1	1	1	0	1	0
2	0	1	0	1	1	0	0	1	0	1

GPK sequence (from LSB): **P, G, P**

Scan:

Step	c_{in}	GPK	c_{out}	Interpretation
i=0	1	P	1	Propagates initial carry 1
i=1	1	G	1	Self-generates (independent of c_{in})
i=2	1	P	1	Propagates carry 1

All pairs have c_{in}=1. Consistent with the step-by-step computation of §3.7.

Structural interpretation. Since pair 1 is Generate, even if pair 0 had killed the carry, carry would still reach pair 2. GPK describes the "terrain" of carry propagation, and the initial value c_0=1 flows over this terrain.

4.9 Worked Example: n=27, x=3 Carry Determination

n=27, LSB order: a=(1,1,0), b=(1,0,1).

Local GPK computation (3n+1 reference pattern):

Pair i	a_{i-1}	b_i	G_{mid}	P_{mid}	a_i	b_i	G_{out}	P_{out}	G_i	P_i
0	0	1	0	1	1	1	1	0	1	0
1	1	0	0	1	1	0	0	1	0	1
2	1	1	1	0	0	1	0	1	1	0

Note: The m4-stage G_{out},P_{out} are AND, XOR of (a_i,b_i), which is none other than m2, m7 of n.

n=27 pairs (LSB order): (1,1),(1,0),(0,1)

- m2(AND): 1,0,0 → G_{out}=1,0,0 √
- m7(XOR): 0,1,1 → P_{out}=0,1,1 √

GPK sequence (from LSB): **G, P, G**

Scan:

Step	c_{in}	GPK	c_{out}
i=0	1	G	1
i=1	1	P	1
i=2	1	G	1

All pairs have c_{in}=1. Consistent with the step-by-step computation of §3.8.

4.10 Answer to the Question of §4.1

We organize the answer to the question posed in §4.1: "Is the sequential dependence of carry essential?"

Answer 1 (Computational). The local GPK of each pair is obtained in O(1) by merely referencing m4/m6 bits. Subsequent carry determination completes in a single O(k) scan. The multiplication xn is absorbed into reference index shifts and does not appear at the scan stage.

Answer 2 (Structural). While the carry *values* are determined sequentially, each pair's *behavior* with respect to carry (whether G, P, or K) is independent of the carry value. Thus, "what the carry does" is determined locally; only "how far the carry reaches" is a global question.

Answer 3 (Parallelization). The global propagation can also be compressed to O(logk) depth via GPK tree composition. This is the same principle as carry-lookahead adders [6], and associativity (Proposition 4.3) guarantees its correctness.

4.11 Summary of This Section

1. Carry propagation reduces to GPK (Generate/Propagate/Kill) classification of each pair. GPK is computed by AND and XOR alone, reading bits according to the reference pattern (Table 3.1).

2. Once all GPK values are known, a single O(k) scan from the LSB determines all carries. The multiplication xn is absorbed at the reference pattern stage; the scan consists only of AND/OR. Overall complexity is O(k).

3. When parallelization is needed, binary tree composition of GPK achieves O(logk) depth (GPK tree).

4. The reference targets of local GPK differ between x=3 and x=5, but the **computational structure is identical**.

5. The Structural Privilege of 3n+1

This section is the core of the paper. We establish that the m4-stage carry structure of 3n+1 coincides with intra-pair predicates of n (Theorem B), and that this property is exclusive to x=3 (Theorem C), using the per-pair independence of the 16 predicates.

5.1 Theorem B: m4-Stage Coincidence

Theorem 5.1 (Theorem B: m4-stage coincidence for 3n+1). In the pair addition for 3n+1, the Generate and Propagate of the m4-stage coincide with predicates m2(AND) and m7(XOR) of n, at every pair position. That is, for any odd n and LSB-ordered pair position i ($0 \le i \le k-1$):

$$G_{out}[i]=m2(n)_i=a_i \wedge b_i$$

$$P_{out}[i]=m7(n)_i=a_i \oplus b_i$$

Derivation. By Proposition 3.4, the m4-stage inputs for 3n+1 are $(r_i^{(4)},a_i)=(b_i,a_i)$. By commutativity:

$$G_{out}[i]=AND(b_i,a_i)=a_i \wedge b_i=m2(n)_i$$

$$P_{out}[i]=XOR(b_i,a_i)=a_i \oplus b_i=m7(n)_i$$

These are exactly the definitions of m2 and m7. □

Corollary 5.1. For 3n+1, the m4-stage Kill coincides with m9(NOR):

$$K_{out}[i]=\neg a_i \wedge \neg b_i=m9(n)_i$$

Derivation. Since G,P,K are mutually exclusive and exhaustive, $K=\neg G \wedge \neg P=\neg(a_i \wedge b_i) \wedge \neg(a_i \oplus b_i)$. By truth table, this is true only when $a_i=b_i=0$, matching the definition of m9(NOR). □

Interpretation. Restating Theorem 5.1:

- Pair $(a_i,b_i)=(1,1) \rightarrow m2=1 \rightarrow$ the m4-stage **generates** carry
- Pair $(a_i,b_i) \in \{(0,1),(1,0)\} \rightarrow m7=1 \rightarrow$ the m4-stage **propagates** carry
- Pair $(a_i,b_i)=(0,0) \rightarrow m9=1 \rightarrow$ the m4-stage **kills** carry

The m4-stage carry landscape of 3n+1 is completely determined by "reading" the pair type of n. The predicates m2, m7, m9 from the 16-predicate system of Section 2 literally describe the carry dynamics of 3n+1.

5.2 The m6-Stage

In contrast to the m4-stage, the m6-stage does *not* coincide with intra-pair predicates of n.

Proposition 5.1. The m6-stage GPK of 3n+1 is:

$$G_{mid}[i]=a_{i-1}\wedge b_i,\ P_{mid}[i]=a_{i-1}\oplus b_i$$

This is a cross-reference between the m4 bit at position i−1 and the m6 bit at position i, and cannot be determined from a single pair position.

Remark 5.1. The overall pair GPK $G_i=G_{out}[i]\vee(P_{out}[i]\wedge G_{mid}[i])$ contains the m6-stage cross-reference term, so even for 3n+1, the *overall* pair GPK is not per-pair independent. However, the fact that the m4-stage is per-pair independent means that the "upper stage" of the carry structure is directly readable from the predicate table of n.

5.3 Theorem C: Limitation of the 16 Predicates

Definition 5.1. A function $f:N_{odd}\rightarrow\{0,1\}^k$ is *16-predicate representable* if it can be expressed as a finite bitwise Boolean composition of m1–m16. By Corollary 2.1, every 16-predicate representable function is per-pair independent.

Theorem 5.2 (Theorem C: Limitation of the 16 predicates). For x≥5 (x−1 a power of 2), the m4-stage GPK (G_{out},P_{out}) of xn+1 is not 16-predicate representable.

Derivation. Two steps.
Step 1: Identifying the reference structure. For x=5, the m4-stage inputs are (a_{i-1},a_i) (Proposition 3.3):

$$G_{out}[i]=a_{i-1}\wedge a_i,\ P_{out}[i]=a_{i-1}\oplus a_i$$

This depends simultaneously on positions i−1 and i.

Step 2: Contradiction with per-pair independence. By contradiction. Suppose $(G_{out}[0],G_{out}[1],\ldots)$ is 16-predicate representable. By Corollary 2.1, it must be per-pair independent: there exists $h:\{0,1\}^2\rightarrow\{0,1\}$ such that $G_{out}[i]=h(a_i,b_i)$ for all i and all n. But $G_{out}[i]=a_{i-1}\wedge a_i$ depends on a_{i-1}. Fixing $a_i=1,b_i=1$:

- $a_{i-1}=0$ gives $G_{out}[i]=0$
- $a_{i-1}=1$ gives $G_{out}[i]=1$

Despite $(a_i,b_i)=(1,1)$ being identical, $G_{out}[i]$ varies with a_{i-1}. No function h of (a_i,b_i) alone can capture this variation. Contradiction.

□

5.4 Exhaustive Verification

We supplement the derivation of Theorem 5.2 with a concrete counterexample.

Proposition 5.2 (Exhaustive non-coincidence). For n=31, the overall pair GPK (G_i) of 5n+1 does not coincide with any of the 784 combinations: 16 single predicates plus 16×16×3=768 binary combinations (AND, OR, XOR of any two predicates).

Verification. n=31=011111_2, pairs ((0,1),(1,1),(1,1)), k=3.
LSB order: a=(1,1,0), b=(1,1,1).

Overall GPK for 5n+1 (MSB order): G=(1,1,0), P=(0,0,1).

The 16-predicate values for n=31 (MSB order, duplicates removed):

$$\{(0,0,0),\ (0,1,1),\ (1,0,0),\ (1,1,1)\}$$

Since pairs 1 and 2 are both (1,1), any per-pair independent function must give the same output at both positions. Therefore, the 3-bit patterns attainable by any Boolean composition of the 16 predicates are limited to those with equal second and third components.

However, G=(1,1,0) has second component 1 and third component 0, violating this constraint.

All **784 combinations** were verified computationally; none match. □

□

Remark 5.2. The essence of Proposition 5.2 is the consequence of per-pair independence that "identical pairs can only produce identical outputs." Because GPK depends on relationships between adjacent pairs, even identical pairs can have different GPK values if their neighboring pairs differ. This is precisely the information that is in principle indescribable within the 16-predicate framework.

5.5 Classification Theorem

Theorem 5.3 (Classification Theorem). For parameters x with $x-1=2^s$, the m4-stage GPK of xn+1 is 16-predicate representable if and only if s=1 (i.e., x=3).

Derivation.
(s=1): By Theorem 5.1, G_{out}=m2, P_{out}=m7. √

(s=2, x=5): m4-stage inputs are (a_{i-1},a_i). Not representable by Theorem 5.2. ✗

(s≥3): The reference distance is $\lfloor s/2 \rfloor \geq 1$ pairs. The same argument as Theorem 5.2 (dependence on a_{i-t} with $t \geq 1$) contradicts per-pair independence. ✗

Therefore s=1 is the only representable case.

□

Corollary 5.2. x=3 is the **unique** xn+1-type parameter for which the m4-stage carry structure can be read as intra-pair predicates of n.

5.6 Structural Interpretation

Why only s=1 is special. In $xn+1=2^s n+n+1$, the term $2^s n$ is an s-bit left shift. The pair structure has width 2 bits, so:

- s=1 (less than pair width): The shift swaps m4 and m6 roles but the **reference stays within the same pair**. $r_i^{(4)}=b_i$ is the right bit of position i itself.
- s=2 (equal to pair width): $r_i^{(4)}=a_{i-1}$ references the previous pair. Inter-pair correlation is unavoidable.
- s≥3 (exceeds pair width): Even more distant pair correlations.

Descriptive power of 16 predicates vs Requirements of the map

Only when s=1 do the requirements fall within the descriptive power.

Relation to the Collatz conjecture. Regarding why the Collatz conjecture is formulated for 3n+1, the Classification Theorem (5.3) offers the following structural explanation: x=3 is the unique parameter for which part of the carry structure can be read directly as internal information of n within the predicate space. This transparency has a direct structural consequence: the m4-stage carry mechanism deterministically annihilates the conditions required for sustained trajectory growth (Proposition 5.4, Remark 5.3). The closed, finite, scale-invariant GPK system of 3n+1 cannot maintain the Generate-dominant configurations necessary for divergence.

5.7 Structural Consequence: Complete Finite Case Analysis

The structural closure established in Theorem 5.1 has a direct deterministic consequence for the trajectory behavior of 3n+1. In this section, we derive this consequence through a **complete finite case analysis** of the scan transition and re-pairing operations, using the standard GPK (Generate/Propagate/Kill) classification that has been the foundation of carry-lookahead adder design for over half a century [6, 7, 8].

5.7.1 Complete Scan Transition Table

For x=3 (s=1, odd), at pair position i in LSB order, the input state is completely described by three variables:

- **Pair type** $(a_i,b_i)\in\{(0,0),(0,1),(1,0),(1,1)\}$ — 4 values
- **Input carry** $c_{in}\in\{0,1\}$ — 2 values
- **Previous left bit** $a_{i-1}\in\{0,1\}$ — 2 values (with $a_{-1}=0$)

Total: 4×2×2=16 patterns. For each pattern, the output is uniquely determined by the reference pattern of 3n+1 (Proposition 3.4):

$$sum_R=a_{i-1}+b_i+c_{in},\ new\_m6_i=sum_R \bmod 2,\ c_{mid}=\lfloor sum_R/2\rfloor$$

$$sum_L=b_i+a_i+c_{mid},\ new\_m4_i=sum_L \bmod 2,\ c_{out}=\lfloor sum_L/2\rfloor$$

Table 5.1: Complete scan transition table for 3n+1.

#	Input (a,b)	GPK	c_{in}	a_{i-1}	sum_R	new m6	c_{mid}	sum_L	new m4	c_{out}	Output	Out GPK
1	(0,0)	K	0	0	0	0	0	0	0	0	(0,0)	K
2	(0,0)	K	0	1	1	1	0	0	0	0	(0,1)	P
3	(0,0)	K	1	0	1	1	0	0	0	0	(0,1)	P
4	(0,0)	K	1	1	2	0	1	1	1	0	(1,0)	P
5	(0,1)	P	0	0	1	1	0	1	1	0	**(1,1)**	**G**
6	(0,1)	P	0	1	2	0	1	2	0	1	(0,0)	K
7	(0,1)	P	1	0	2	0	1	2	0	1	(0,0)	K
8	(0,1)	P	1	1	3	1	1	2	0	1	(0,1)	P
9	(1,0)	P	0	0	0	0	0	1	1	0	(1,0)	P
10	(1,0)	P	0	1	1	1	0	1	1	0	**(1,1)**	**G**
11	(1,0)	P	1	0	1	1	0	1	1	0	**(1,1)**	**G**
12	(1,0)	P	1	1	2	0	1	2	0	1	(0,0)	K
13	(1,1)	G	0	0	1	1	0	2	0	1	(0,1)	P
14	(1,1)	G	0	1	2	0	1	3	1	1	(1,0)	P
15	(1,1)	G	1	0	2	0	1	3	1	1	(1,0)	P
16	(1,1)	G	1	1	3	1	1	3	1	1	**(1,1)**	**G**

All 16 entries are derived from the five-line arithmetic above, which is itself a direct consequence of the algebraic identity $3n+1=2n+n+1$.

Remark. The scan defines a *deterministic finite state transducer* (DFST) with state $(c_{in},a_{i-1})\in\{0,1\}^2$ (4 states), input alphabet $\{(0,0),(0,1),(1,0),(1,1)\}$ (4 symbols), and deterministic output. The initial state is $(c_{in},a_{i-1})=(1,0)$ (the +1 effect and boundary condition). Each output (c_{out},a_i) — where a_i is the m4 bit of the current input pair — uniquely determines the next state.

Remark (DFST permutation property). For each of the four DFST states, the map from input pair type to output pair type is a *permutation* of $\{K,P(0,1),P(1,0),G\}$:

State (c_{in},a_{i-1})	K→	P(0,1)→	P(1,0)→	G→
(0, 0)	K	G	P(1,0)	P(0,1)
(0, 1)	P(0,1)	K	G	P(1,0)
(1, 0)	P(0,1)	K	G	P(1,0)
(1, 1)	P(1,0)	P(0,1)	K	G

Each row contains exactly one occurrence of each pair type. This is not assumed but directly verified from Table 5.1. The permutation property means that the scan, viewed as a local transformation at each pair position, preserves the multiplicity of pair types: it does not structurally favor any type over others. Note that states (0,1) and (1,0) induce the same permutation.

5.7.2 Structural Properties from the Scan Table

Lemma 5.1 (G-block entry elimination).
(a) The only scan case where a G-input produces a G-output is Case 16, which requires $c_{in}=1$ and $a_{i-1}=1$.

(b) The **leftmost** G-block's leading G-position is always converted to P in the scan output.

(c) When carry from a G-block propagates rightward through a P(1,0) bridge, each P(1,0) position is converted to K(0,0) (Case 12), destroying the bridge.

Derivation. (a) Direct inspection of Table 5.1: only Case 16 maps G-input to G-output, requiring both $c_{in}=1$ and $a_{i-1}=1$.
(b) At the leading position of the leftmost G-block, the predecessor is either the boundary ($a_{-1}=0$, Case 16 fails), K ($c_{out}=0$, Case 16 fails), or P(0,1) ($a_{i-1}=0$, Case 16 fails). If the predecessor is P(1,0) with $c_{in}=1$, the carry must originate from an earlier G-block—but no earlier G-block exists. Therefore, the leftmost G-block's leading G is converted to P (Cases 13, 14, or 15).

(c) P(1,0) with $c_{in}=1$ and $a_{i-1}=1$ (from a preceding G or P(1,0)) triggers Case 12: output K(0,0), $c_{out}=1$. The carry passes through, but the bridge position is destroyed.

□

Consequence (left-to-right consumption). A non-leftmost G-block may temporarily retain its leading G if an unbroken P(1,0) bridge delivers carry from a preceding G-block (Case 16 at the entry). However, such a bridge is simultaneously converted to K(0,0) by part (c). By Lemma 5.5 (below), the K barrier persists through re-pairing. Therefore, when the leftmost G-block is fully consumed, the carry supply to the next G-block is severed, and that block becomes the new leftmost—its leading G then subject to part (b). G-blocks are consumed sequentially from left to right.

Example. For $n=1003=1111101011_2$, pairs (LSB): G, P(1,0), P(1,0), G, G. The scan converts G-block 1's leading G to P (Case 15), the P(1,0) bridge to K (Case 12×2), while G-block 2's leading G is preserved (Case 16). However, the bridge is now K(0,0), so at the next step, G-block 2's leading G is subject to part (b). Total G-count: $3 \to 2$.

> **Lemma 5.2 (Impossibility of K→G).** No Kill position in the input can produce a Generate position in the output.
>
> *Derivation.* Cases 1–4 of Table 5.1 cover all four contexts for K-input (0,0). The outputs are K(0,0), P(0,1), P(0,1), and P(1,0) respectively. None is G(1,1). □

> **Lemma 5.3 (P→G constraint).** The only cases producing G-output from P-input are Cases 5, 10, and 11. In all three cases, $c_{out}=0$.
>
> *Derivation.* Inspection of P-input rows (Cases 5–12) in Table 5.1. Cases producing G-output: Case 5 (P(0,1), $c_{in}=0,a_{i-1}=0$), Case 10 (P(1,0), $c_{in}=0,a_{i-1}=1$), Case 11 (P(1,0), $c_{in}=1,a_{i-1}=0$). All have $c_{out}=0$. □

Remark (Case 10 chaining). A consecutive run of P(1,0) inputs can trigger a chain of Case 10 applications: each produces G-output with $c_{out}=0$ and new_m4=1, providing $a_{i-1}=1$ and $c_{in}=0$ for the next position. Thus P(1,0) runs of length M can produce G-blocks of length M in the output. However: (i) this consumes M P-positions from the input, (ii) all generated G-positions have $c_{out}=0$, so the resulting G-block carries no internal carry chain, and (iii) at the next scan step, Lemma 5.1(b) applies when this new G-block is the leftmost, or the left-to-right consumption mechanism governs its lifetime.

> **Lemma 5.4 (Carry output of G-positions).** For all four G-input cases (Cases 13–16), $c_{out}=1$.
>
> *Derivation.* Direct inspection of Table 5.1, Cases 13–16. This is a computational confirmation of the Generate property: $G_{out}=a_i \wedge b_i=1 \wedge 1=1$, which produces $c_{out}=1$ unconditionally, regardless of c_{in} or a_{i-1}. □

5.7.3 Re-pairing Transition Table

After computing 3n+1, the division by 2^d removes d trailing zeros. When d is odd, the remaining bit string requires re-pairing with a 1-bit boundary shift (§6). Each new pair is formed from the left bit of one old pair and the right bit of the adjacent old pair:

$$new\_m6_i' = old\_m4_A,\ new\_m4_i' = old\_m6_B$$

where A and B are adjacent pairs in the scan output. (When d is even, pairs are preserved and no re-pairing occurs.)

Table 5.2: Re-pairing transition table (d odd).

#	Pair A	A GPK	Pair B	B GPK	new m6'	new m4'	New pair	New GPK
R1	(0,0)	K	(0,0)	K	0	0	(0,0)	K
R2	(0,0)	K	(0,1)	P	0	1	(1,0)	P
R3	(0,0)	K	(1,0)	P	0	0	(0,0)	K
R4	(0,0)	K	(1,1)	G	0	1	(1,0)	P
R5	(0,1)	P	(0,0)	K	0	0	(0,0)	K
R6	(0,1)	P	(0,1)	P	0	1	(1,0)	P
R7	(0,1)	P	(1,0)	P	0	0	(0,0)	K
R8	(0,1)	P	(1,1)	G	0	1	(1,0)	P
R9	(1,0)	P	(0,0)	K	1	0	(0,1)	P
R10	(1,0)	P	(0,1)	P	1	1	**(1,1)**	**G**
R11	(1,0)	P	(1,0)	P	1	0	(0,1)	P
R12	(1,0)	P	(1,1)	G	1	1	**(1,1)**	**G**
R13	(1,1)	G	(0,0)	K	1	0	(0,1)	P
R14	(1,1)	G	(0,1)	P	1	1	**(1,1)**	**G**
R15	(1,1)	G	(1,0)	P	1	0	(0,1)	P
R16	(1,1)	G	(1,1)	G	1	1	**(1,1)**	**G**

New G is produced if and only if $m4_A=1$ and $m6_B=1$ (Cases R10, R12, R14, R16). Note that this requires pair A to have left bit 1 (i.e., A is P(1,0) or G(1,1)) and pair B to have right bit 1 (i.e., B is P(0,1) or G(1,1)).

5.7.4 Lemmas from the Re-pairing Table

Lemma 5.5 (G-block non-mergeability). Two G-blocks separated by one or more non-G pairs in the scan output remain separated after re-pairing (d odd). The separating barrier may shift position but does not vanish.

Derivation. Consider two G-blocks separated by a single P. If P = P(0,1): the G–P boundary gives $(m6_P, m4_G)=(1,1)$=G, but the P–G boundary gives $(m6_G, m4_P)=(1,0)$=P. The barrier shifts left but persists. If P = P(1,0): the G–P boundary gives P, the P–G boundary gives G. The barrier shifts right but persists. For separations of $k \geq 2$ non-G pairs, the argument extends: at each boundary, since non-G pairs have at least one bit equal to 0 (P has $a \oplus b=1$; K has $a=b=0$), the re-pairing formula $(m4_A, m6_B)$ cannot produce (1,1) at every boundary position. At least one P barrier remains. □

> **Lemma 5.6 (G-block tail growth bound).** Under re-pairing (d odd), a G-block can extend by at most 1 position at its tail.
>
> *Derivation.* Let the last G-position of a block be at position k, with successor pair [next]. Under re-pairing, the new pair at position k is $(m6_{next}, m4_G) = (m6_{next}, 1)$. This is G iff $m6_{next} = 1$, i.e., [next] is P(0,1) or G(1,1) — a tail extension of +1. However, the new pair at position k+1 is $(m6_{next+1}, m4_{next})$. If [next] = P(0,1), then $m4_{next} = 0$, so the new pair has $m4' = 0$, which cannot be G. The tail extension is bounded at 1. The tail extension requires the specific configuration [next] = P(0,1), which is consumed in the process and unavailable at the next step. For the leftmost G-block whose leading G is always eliminated (Lemma 5.1(b)), the net change per step is at most 0 (−1 head + at most +1 tail). □

Remark (Integrated G-block lifecycle). For the **leftmost** G-block of initial length L, combining the scan transition (Lemma 5.1(b)) with the re-pairing transition (Lemma 5.6), the net G-count change in a single step depends on the parity of d and the type of the head conversion:

- **d even:** Scan converts the leading G to P (−1). No re-pairing occurs. G–G boundaries are preserved. Net change: −1.
- **d odd, head converted to P(0,1):** Scan produces P(0,1) followed by (L−1) G-positions. Re-pairing at the P(0,1)–G boundary gives Case R8: $(m4_A, m6_B) = (0,1) \to$ P, destroying one additional G-position. G–G internal boundaries give Case R16 (G preserved). Net change: −2.
- **d odd, head converted to P(1,0):** Scan produces P(1,0) followed by (L−1) G-positions. Re-pairing at the P(1,0)–G boundary gives Case R12: $(m4_A, m6_B) = (1,1) \to$ G, recovering the boundary. Net change: −1.

The most typical case—G-block preceded by K(0,0), giving $P_{head} = P(0,1)$ via Case 13—yields −2 per step. In all cases, the leftmost G-block loses at least one G per step. A leftmost G-block of initial length L is therefore fully consumed in at most L steps. (Non-leftmost G-blocks may temporarily retain their leading G via carry from the left; see Lemma 5.1(a) and the left-to-right consumption mechanism.)

> **Lemma 5.7 (Isolated G lifetime).** An isolated G-block (length 1) that is the leftmost G-block, or whose predecessor is K or P(0,1), is converted to P at the next scan step. An isolated G receiving carry via a P(1,0) bridge from a preceding G-block may survive that step, but the bridge is destroyed (Lemma 5.1(c)), and the isolated G is converted once the carry supply is severed.
>
> *Derivation.* An isolated G consists entirely of its leading position. When the conditions for Lemma 5.1(b) are met (leftmost, or no carry from the left), the leading G is converted to P. When carry arrives via P(1,0) bridge, Case 16 preserves the G, but the bridge is simultaneously converted to K(0,0) (Lemma 5.1(c)). At the subsequent step, the K barrier blocks carry, and Lemma 5.1(b) applies. □

Remark (P→G products under re-pairing). When d is odd, an isolated G (length 1) produced by a P→G transition (Lemma 5.3) undergoes re-pairing with its neighbors. Let A be the predecessor pair and C the successor pair in the scan output. The re-pairing creates two new boundary pairs:

- A–G boundary: $new\_m4' = m6_G = 1$, $new\_m6' = m4_A$. New pair is G iff $m4_A=1$ (A is P(1,0) or G).
- G–C boundary: $new\_m6' = m4_G = 1$, $new\_m4' = m6_C$. New pair is G iff $m6_C=1$ (C is P(0,1) or G).

An isolated G can thus expand to at most 2 G-positions under re-pairing, or vanish entirely, depending on the neighbor pair types. Crucially, all P→G outputs have $c_{out}=0$ (Lemma 5.3), so the successor pair is computed with $c_{in}=0$. This carry severance prevents P→G products from initiating new carry chains, structurally limiting their ability to seed large G-blocks.

5.7.5 The Division Exponent and Growth

> **Lemma 5.8 (d=1 iff pair 0 is G).** For odd n, in the scan output (LSB order), d=1 if and only if the input pair at position 0 is G(1,1).
>
> *Derivation.* For odd n, $b_0=1$ (the least significant bit is 1), so pair 0 is either G(1,1) or P(0,1). The scan initial state is $(c_{in},a_{i-1})=(1,0)$. If pair 0 = G(1,1): Case 15 applies, output = (1,0), trailing bits = …10, so d=1. If pair 0 = P(0,1): Case 7 applies, output = (0,0), trailing bits = …00, so d≥2. □

Remark (General division exponent formula). For odd n, the full division exponent d is determined by the input pair sequence as follows. Let j≥0 be the number of leading P(0,1) pairs (i.e., pairs 0 through j−1 are all P(0,1), and pair j is the first non-P(0,1) pair). Then:

- If pair j is G(1,1): d=2j+1 (odd)
- If pair j is K(0,0) or P(1,0): d=2j (even, with j≥1)

Derivation. By the DFST initial state $(c_{in},a_{i-1})=(1,0)$: input P(0,1) at pair 0 triggers Case 7, producing output K(0,0) with $(c_{out},new\_m4)=(1,0)$, maintaining the DFST in state (1,0). Each subsequent P(0,1) repeats Case 7, extending the trailing-zero output by two bits per pair. When the first non-P(0,1) pair is reached at position j in DFST state (1,0): K(0,0) triggers Case 3 (output P(0,1), first non-zero bit at position 2j, so d=2j); P(1,0) triggers Case 11 (output G(1,1), first non-zero bit at position 2j, so d=2j); G(1,1) triggers Case 15 (output P(1,0) with bits (m6,m4)=(0,1), first non-zero bit at position 2j+1, so d=2j+1). For j=0 with pair 0 = G: Case 15 gives d=1, recovering Lemma 5.8.

Remark (Expected division exponent under uniform distribution). If pair types are independently and uniformly distributed (each of {K,P(0,1),P(1,0),G} with probability 1/4, subject to the constraint that pair 0 of odd n has $b_0=1$, giving equal probability 1/2 to G and P(0,1) at pair 0), the d-formula yields E[d]=2 by geometric series evaluation. The net expected bit change per step is $\log_2 3 - E[d] \approx 1.585 - 2 = -0.415$ bits, consistent with the known contraction rate of Collatz trajectories. By the DFST permutation property, the scan does not structurally bias the output toward any particular pair type, so the uniform distribution assumption is self-consistent at the single-step level.

Remark (d=1 re-pairing lifecycle). When d=1 (pair 0 = G, Case 15 output = P(1,0)), the division removes only bit 0, and re-pairing produces new pairs: new pair i=(old_m6$_{i+1}$, old_m4$_i$). Since the scan output has m4$_0$=1 (from Case 15), new pair 0=(old_m6$_1$, 1). If the original pair 1 was also G (within the same G-block), then old_m6$_1$=1, giving new pair 0 = G(1,1). Thus d=1 can recur at the next step. However, the G-block loses at least 1 G per step (integrated lifecycle above), so d=1 persists for at most L consecutive steps, where L is the initial G-block length containing pair 0. When old_m6$_1$≠1 (pair 1 is K or P(1,0)), new pair 0 = P(0,1), and d≥2 at the next step. This is consistent with the arithmetic bound of Corollary 5.3.

Growth condition. Each step of 3n+1 adds $\log_2 3 \approx 1.585$ bits and removes d bits. Net bit growth is positive only when d=1. By Lemma 5.8, sustained growth requires pair 0 to be G(1,1) at every step. Pair 0, as the leftmost pair, is always in or to the left of the leftmost G-block. By Lemma 5.1(b), the leftmost G-block's leading G is eliminated at each scan step; by Lemma 5.6, its net length change is at most 0. The leftmost G-block containing pair 0 is therefore consumed in finite time, after which d≥2 and the trajectory contracts.

> **Corollary 5.3 (Arithmetic bound on consecutive d=1).** Let n be odd and let $n_0=n$, $n_{j+1}=(3n_j+1)/2^{d_j}$ be the odd-to-odd trajectory. If $d_j=1$ for L consecutive steps j=0,1,…,L−1, then $n \equiv 2^{L+1}-1 \pmod{2^{L+1}}$, i.e., the lowest L+1 bits of n are all 1.

Derivation. By induction on L. **Base case** (L=1): By Lemma 5.8, $d_0=1$ iff pair 0 of n is G(1,1), i.e., $(a_0,b_0)=(1,1)$. Since n is odd, $b_0=1$, and the condition is $a_0=1$, meaning the lowest two bits are 11, so $n \equiv 3 \pmod 4$.

Inductive step: Suppose L consecutive d=1 steps require $n \equiv 2^{L+1}-1 \pmod{2^{L+1}}$. For L+1 consecutive steps, we additionally need $d_1=1$ for $n_1=(3n+1)/2$. By Lemma 5.8, $n_1 \equiv 3 \pmod 4$, i.e., $(3n+1)/2 \equiv 3 \pmod 4$. This gives $3n+1 \equiv 6 \pmod 8$, hence $n \equiv 7 \pmod 8$, meaning the lowest 3 bits are 111. Continuing, L+1 consecutive d=1 requires $n \equiv 2^{L+2}-1 \pmod{2^{L+2}}$.

□

Consequence. Since n is a finite natural number with at most $\lfloor \log_2 n \rfloor+1$ bits, the maximum number of consecutive d=1 steps starting from n is at most $\lfloor \log_2 n \rfloor$. This is an *independent* arithmetic derivation of the finiteness of d=1 streaks, requiring only modular arithmetic—no GPK analysis, no transition tables, no G-block dynamics. The fact that two entirely different methods (GPK structural analysis and modular arithmetic) reach the same conclusion reinforces the robustness of the result.

5.7.6 Proposition 5.4: Complete Derivation

> **Proposition 5.4 (Self-annihilation of G-dominance).** In the GPK system of 3n+1, the **activation condition** for carry generation (G-dominance, i.e., high concentration of (1,1)-pairs) and the **activation result** (carry-induced destruction of G-blocks) are inseparably coupled within the same algebraic identity. No configuration can maintain G-dominance indefinitely.

Derivation. The result follows from Lemmas 5.1–5.8, which are derived exhaustively from Tables 5.1 and 5.2.

Step 1: Inseparable coupling. Maintaining G-dominance requires a high concentration of (1,1)-pairs. By Theorem 5.1, each (1,1)-pair is assigned Generate. By Lemma 5.4, every G-position outputs c_{out}=1 unconditionally (all four G-cases in Table 5.1 confirm this). This carry enters the output bit formula as an XOR operand, deterministically transforming the output pair types at G-block boundaries. By Lemma 5.1(b), the leftmost G-block's leading G is always converted to P. By Lemma 5.1(c), any P(1,0) bridge carrying this carry to subsequent G-blocks is simultaneously destroyed (converted to K). This coupling—(1,1)-pairs produce G, G produces carry, carry destroys G and the bridges connecting G-blocks—is a direct consequence of the identity 3n+1=2n+n+1 and cannot be circumvented within the closed system.

Step 2: Left-to-right G-block consumption. G-blocks are consumed sequentially from left to right:

- Lemma 5.1(b): The leftmost G-block's leading G is eliminated at each scan step (−1 per step)
- Lemma 5.6: Tail extension bounded at +1 per re-pairing; for the leftmost block, net change ≤0
- Lemma 5.1(c): P(1,0) bridges between G-blocks are converted to K during carry propagation
- Lemma 5.5: K barriers persist through re-pairing, so G-blocks cannot merge
- When the leftmost G-block is fully consumed, the K barrier severs carry to the next G-block, which becomes the new leftmost and begins its own consumption

The leftmost G-block of initial length L has a finite lifetime bounded by L (net change ≤0 per step). Non-leftmost blocks may be temporarily preserved by carry from the left, but the bridge destruction ensures that each block eventually becomes the leftmost. The total consumption time for all G-blocks is finite.

Carry asymmetry bridge. A natural question arises: even though each individual G-block has a finite lifetime, could P→G transitions continuously replenish the G supply, maintaining G-dominance indefinitely? The carry structure of Table 5.1 rules this out through an asymmetry between destruction and creation.

Destruction chains. When the leftmost G-block's leading G is consumed (Lemma 5.1(b)), the output carry is c_{out}=1 (Lemma 5.4: all four G-input cases produce c_{out}=1). This carry does not stop at the consumed position. It flows rightward into the next position as c_{in}=1. If that next position is a P(1,0) bridge connecting two G-blocks, the carry converts it to K(0,0) (Lemma 5.1(c), Table 5.1 Case 12), producing another c_{out}=1 that propagates further. The newly created K barrier then permanently severs the carry connection to subsequent G-blocks (Lemma 5.5). In this way, a single consumption event at the G-block head cascades rightward: it eliminates one G, destroys the bridge, and isolates the next G-block—all from one carry chain.

Creation does not chain. In contrast, when a P position is converted to G (Cases 5, 10, 11), the output carry is c_{out}=0 (Lemma 5.3). This means the newly created G produces no carry that flows to its right neighbor. The next position receives c_{in}=0 from this event and is structurally unaffected by it. The creation is a local, one-position event with no rightward propagation. Furthermore, Case 10 chaining—the only mechanism that creates a G-block of length M from M consecutive P(1,0) positions—consumes those M P-positions as fuel. This fuel is removed from the pair sequence in the same step and is not regenerated.

This asymmetry is summarized as follows: each G-destruction event produces c_{out}=1, which cascades into bridge destruction and carry severance affecting multiple downstream positions. Each G-creation event produces c_{out}=0, affecting only the single position where the creation occurs and consuming P-fuel in the process. Both facts are per-case readings from Table 5.1, not quantitative estimates. Since destruction cascades while creation does not, P→G generation cannot systematically compensate for the left-to-right consumption process.

Step 3: New G-generation is structurally bounded. Given this carry asymmetry, new G-positions arise only from P→G transitions (Lemma 5.3: Cases 5, 10, 11) and re-pairing (Table 5.2: Cases R10, R12, R14, R16). By Lemma 5.2, K→G is impossible. The P→G transitions all produce c_{out}=0 (Lemma 5.3), meaning newly generated G-blocks carry no internal carry chain. Case 10 chaining can convert a P(1,0) run of length M into a G-block of length M, but this consumes M P-positions from the input. The same Lemmas 5.1–5.7 apply to these newly created G-blocks: they have finite lifetime and cannot merge with existing blocks.

Dimensional separation. The DFST permutation property (§5.7.1) ensures that for each fixed DFST state, the map from input pair type to output pair type is a bijection on {K,P(0,1),P(1,0),G}. However, the DFST state (c_{in},a_{i-1}) varies by position, determined by the carry chain and the preceding input. Different positions therefore undergo different permutations, and global type counts are *not* preserved. For example, two consecutive G-inputs at positions 0 and 1 yield P(1,0) and G respectively (states (1,0) and (1,1)), reducing the G-count from 2 to 1. This is precisely the mechanism of Lemma 5.1(b): the leftmost G-block's leading G is systematically converted to P.

A natural question remains: even though G-blocks are consumed at the left, could P→G transitions elsewhere sustain enough G-positions to continually replenish pair 0 = G? The growth condition d=1 requires pair 0 = G(1,1) (Lemma 5.8)—a *positional* condition on a specific location, not a condition on aggregate G-count. The G-block degradation established by Lemmas 5.1–5.7 operates in this positional dimension: head elimination, bridge destruction, non-mergeability, and tail growth bounds concern the spatial structure of consecutive G-positions. The aggregate number of G-positions elsewhere is structurally irrelevant to whether pair 0 is G. Once the leftmost G-block containing pair 0 is consumed, the condition pair 0 = G is lost regardless of how many G-positions exist at other locations.

Re-pairing pathway containment. After division by 2^d, when d is odd, re-pairing reassigns pair boundaries (Table 5.2), which can place G at the new pair 0 through mechanical re-indexing rather than type redistribution. However, any G-block at pair 0 arising from re-pairing is subject to the full structural constraint set: finite lifetime (Lemmas 5.1(b), 5.6, 5.7), non-mergeability (Lemma 5.5), and the inseparable coupling of Step 1. Each such G-block is consumed in at most L' steps (its initial length), after which d≥2 and the trajectory contracts. Re-pairing provides no escape from the consumption process; it merely initiates a new finite episode, each terminated by contraction.

Step 4: Four-phase closure. The one-step map $T(n)=(3n+1)/2^d$ decomposes into four phases, each operating on the same closed set of four pair types {(0,0),(0,1),(1,0),(1,1)} with three GPK classes {G,P,K}:

1. **Scan** (carry propagation): A deterministic finite-state transducer with 4 states, governed by Table 5.1 (16 cases). The DFST permutation property ensures that each state maps the four input types to a permutation of the four output types.
2. **Division exponent determination**: The value d is a deterministic function of the scan output pair sequence, given by the formula in §5.7.5 (the number of leading P(0,1) pairs and the type of the first non-P(0,1) pair).

3. **Trailing zero removal** ($\div 2^d$): Removes $\lfloor d/2 \rfloor$ trailing K(0,0) pairs and, when d is odd, one additional bit that triggers re-pairing. The removed pairs are all K(0,0) (trailing zeros); the remaining pair types are preserved.
4. **Re-pairing** (when d is odd): Governed by Table 5.2 (16 cases). New pair types are determined by adjacent old pairs through the formula $\text{new\_m6}' = \text{m4}_A$, $\text{new\_m4}' = \text{m6}_B$.

No phase introduces pair types or GPK classes outside the established sets. The associative composition rule G∘X=G, K∘X=K, P∘X=X [6, 7, 8] is preserved throughout. The output n' has a new pair sequence from the same four types, and its GPK is determined by the same algebraic identity (Theorem 5.1). Both finite tables (16 + 16 cases) are exhaustive enumerations derived from the identity 3n+1=2n+n+1 and the exchange principle of §6, requiring no assumptions beyond elementary arithmetic.

Step 5: Scale invariance. The per-pair determination of the m4-stage GPK (Theorem 5.1) is an algebraic identity, not a finite verification. It holds for all $n \in \mathbb{N}$, at every bit-length. The 16-pattern scan table (Table 5.1) and the 16-pattern re-pairing table (Table 5.2) are exhaustive: they cover every possible input state. Lemmas 5.1–5.8 are logical consequences of these tables and therefore hold universally. A divergent trajectory passes through numbers of unbounded bit-length; at every such number, the same inseparable coupling applies. For $x \geq 5$, per-pair determination fails (Theorem 5.3), and the above coupling is not guaranteed.

Conclusion. G-dominance requires (1,1)-pairs; (1,1)-pairs unconditionally generate carry; carry eliminates the leftmost G-block's leading position and destroys the P-bridges connecting G-blocks. G-blocks are consumed left-to-right within this closed algebraic structure, at every scale. No configuration within this closed system can maintain G-dominance indefinitely. □

Remark 5.3 (Structural impossibility of divergence). Sustained trajectory growth requires d=1 at most steps (since $\log_2 3 \approx 1.585$ bits are added and d bits removed per step). By Lemma 5.8, d=1 requires pair 0 to be G(1,1). Pair 0, as the leftmost pair, is always in or to the left of the leftmost G-block. By Lemma 5.1(b), the leftmost G-block's leading G is eliminated at each scan step; by Lemma 5.6, its net length change is ≤0. The leftmost G-block containing pair 0 is therefore consumed in finite time, after which pair 0 is no longer G and d≥2. New G-blocks created by P→G transitions (Lemma 5.3) or by re-pairing (Table 5.2) also enter the left-to-right consumption process (Proposition 5.4, dimensional separation and re-pairing pathway containment). Therefore, the condition required for sustained growth (d=1 indefinitely) is structurally unattainable.
Corollary 5.3 provides an independent confirmation: L consecutive d=1 steps require $n \equiv 2^{L+1} - 1 \pmod{2^{L+1}}$, bounding the maximum streak by $\lfloor \log_2 n \rfloor$. This arithmetic constraint, derived purely from modular arithmetic, is entirely independent of the GPK analysis yet yields the same conclusion.

This is not a probabilistic argument. Each step in the chain—d=1⇔ pair 0 = G (Lemma 5.8), leftmost G-block consumption (Lemma 5.1(b)), bridge destruction (Lemma 5.1(c)), P→G bounded generation (Lemma 5.3), non-mergeability (Lemma 5.5)—is a deterministic consequence of the 16-pattern exhaustive case analysis, derived from the algebraic identity 3n+1=2n+n+1. The arithmetic bound of Corollary 5.3 serves as a cross-verification from an entirely different method.

Remark 5.4 (P→G feedback containment). The P→G transitions (Cases 5, 10, 11) can temporarily increase the G-count in one step (e.g., n=10495: G-ratio 50% → 71%). This fluctuation is structurally contained: (i) all newly created G-blocks enter the left-to-right consumption process (Lemma 5.1(b), 5.7); (ii) P→G generation consumes P-positions, reducing the fuel for future P→G; (iii) G-blocks cannot merge (Lemma 5.5), preventing accumulation; (iv) the same inseparable coupling (Proposition 5.4) applies to every new G-block at every scale. Bounded fluctuations within a closed, scale-invariant system do not constitute divergence.

Remark 5.5 (Contrast with x≥5). For $x \geq 5$, the m4-stage GPK is not per-pair determined (Theorem 5.3). The DFST state space grows exponentially with the shift parameter $s = \log_2(x-1)$: the reference distance spans $\lfloor s/2 \rfloor$ pairs, requiring tracking of distant pair information. This enlarged state space admits configurations where carry structure depends on inter-pair correlations that escape the structural constraints of Lemmas 5.1–5.8. This is consistent with the known existence of non-trivial cycles for 5n+1 [2]: the carry structure of 5n+1 admits stable configurations that the 2-bit observation space cannot resolve or constrain.

5.8 Numerical Verification

Verification of Theorem B. For the 50 odd numbers $n=1,3,5,\dots,99$ and large values $n=127,255,511,1023,4095,8191,65535,99991,999999$, the coincidence of the m4-stage $G_{out}[i]$ with $m2(n)_i$, and $P_{out}[i]$ with $m7(n)_i$ for 3n+1 was verified computationally. Full agreement for all numbers.

Verification of Theorem C. In addition to the exhaustive 784-way matching for n=31, the same matching was performed for n=7,13,19,27,43,83. For n=27,43, some predicate combinations coincidentally match (due to the arrangement of pair values), but for n=31, complete non-coincidence holds. Theorem C is an existence statement; a single counterexample suffices.

5.9 Summary of This Section

1. **Theorem B (5.1):** The m4-stage GPK of 3n+1 coincides with m2(AND)/m7(XOR)/m9(NOR) of n. Reading the pair type of n determines carry generation, propagation, and absorption.

2. **Theorem C (5.2):** For $x \geq 5$, the m4-stage GPK is not 16-predicate representable. The per-pair independence barrier makes inter-pair correlations inherently inexpressible. Supplemented by exhaustive 784-way verification for n=31.

3. **Classification Theorem (5.3):** m4-stage GPK closes within intra-pair predicates only for x=3. The map 3n+1 is the unique Collatz-type map whose carry structure is "transparent" in the 16-predicate space.

4. **Proposition 5.4 (Self-annihilation of G-dominance):** Within the closed GPK system of 3n+1, the activation condition for carry generation (G-dominance) and the activation result (carry-induced destruction of G-blocks) are inseparably coupled. The complete one-step map decomposes into four phases (scan, d-determination, trailing zero removal, re-pairing), each structurally closed within the same four pair types. The 16-pattern scan transition

table (Table 5.1) and 16-pattern re-pairing transition table (Table 5.2) yield Lemmas 5.1–5.8, which collectively establish that no configuration can maintain G-dominance indefinitely. The DFST permutation property confirms that no phase structurally biases pair type distribution. Combined with Remark 5.3, this establishes that divergence of 3n+1 trajectories is structurally impossible.

6. Division by 2 as Exchange

After computing xn+1, repeated division by 2 is required to reach the next odd number. This section shows that this operation is described as a role exchange between m4 and m6 within our framework.

6.1 Trailing Zeros and 2-adic Valuation

Definition 6.1. The 2-adic valuation $v_2(m)$ of a positive integer m is the exponent of the largest power of 2 dividing m:

$$v_2(m)=max\{d\in N:2^d \mid m\}$$

In binary, $v_2(m)$ equals the number of trailing zeros.

Setting $d=v_2(xn+1)$, the next odd number is $n'=(xn+1)/2^d$.

6.2 Bit-Level Operation of ÷2

In binary, m/2 removes the least significant bit (a right shift by 1). When m is even, the LSB is 0 and no information is lost. Dividing by 2^d removes d trailing zero bits—all carrying zero information.

Proposition 6.1. The binary representation of $m/2^d$ (where $2^d \mid m$) is obtained by removing the d trailing zeros from the binary representation of m. No information is lost.

6.3 Effect on Pair Structure

Consider the conversion from the pair representation of m=xn+1 to that of $n'=m/2^d$.

Let the binary representation of m be $\beta(m)=s_1s_2\ldots s_{2l}$ (even-digit padded). Since the trailing d bits are all zeros:

$$\beta(m)=s_1s_2\ldots s_{2l-d}0\ldots0$$

Removing the trailing d bits yields the bit string of $m/2^d$:

$$\beta(m/2^d)=s_1s_2\ldots s_{2l-d}$$

This string is then even-digit padded and grouped into pairs. **The parity of d determines how pair boundaries shift.**

When d is even: Removing d=2q bits removes q complete pairs. Pair boundaries are preserved. The remaining pair sequence coincides with the first l−q pairs of m.

$$P(m/2^d)=((s_1,s_2),(s_3,s_4),\ldots,(s_{2l-d-1},s_{2l-d}))$$

The m4/m6 correspondence is maintained.

When d is odd: Removing d=2q+1 bits removes q pairs plus 1 bit. The remaining bit string has odd length, requiring a leading-zero pad:

$$\beta(m/2^d)=0\ s_1\ s_2\ \ldots\ s_{2l-d}$$

The padding causes the original pair (s_1,s_2) to be regrouped as $(0,s_1)$ and (s_2,s_3). **This corresponds to swapping the roles of m4 and m6.**

6.4 The m4↔m6 Exchange Theorem

Theorem 6.1 (Exchange principle of ÷2). In the conversion from the pair sequence of m to that of $m/2^d$:

- When d is even: the m4/m6 correspondence is **maintained**.
- When d is odd: the m4/m6 correspondence is **exchanged**.

More precisely, letting the MSB-side pair sequence of m be $(\alpha_0,\beta_0),(\alpha_1,\beta_1),\ldots$, the re-pairing after removing the trailing d bits gives:

$$d\ \text{even: } m4(n')_j=\alpha_j,\ m6(n')_j=\beta_j$$

$$d\ \text{odd: } m4(n')_j\sim\beta_j,\ m6(n')_j\sim\alpha_{j+1}$$

where ~ denotes correspondence up to padding and boundary effects.

Derivation. When d is even, the d removed bits constitute an even number of pairs, preserving pair boundaries. When d is odd, the remaining bit string has odd length; the leading-zero padding shifts all bits by one position within their pairs. Bits formerly in left positions move to right positions and vice versa. □

6.5 Worked Examples

Example 6.1 (d=3, odd: exchange). 5×27+1=136=10001000_2. Pairs: ((1,0),(0,0),(1,0),(0,0)). m4 =(1,0,1,0), m6 = (0,0,0,0).

$v_2(136)=3$. 136/8=17=10001_2. Padded: 010001. Pairs: ((0,1),(0,0),(0,1)). m4' =(0,0,0), m6' =(1,0,1).

d=3 (odd) → exchange occurs. The non-zero pattern of m4 of 136 corresponds to m6' of 17.

Example 6.2 (d=1, odd: exchange). $3\times27+1=82=01010010_2$. Pairs: ((0,1),(0,1),(0,0),(1,0)). m4 =(0,0,0,1), m6 = (1,1,0,0).

$v_2(82)=1$. $82/2=41=101001_2$. Pairs: ((1,0),(1,0),(0,1)). m4' =(1,1,0), m6' =(0,0,1).

d=1 (odd) → exchange.

Example 6.3 (d=2, even: maintained). $n=7=111_2$. Even-digit padding: 0111_2. Pairs: ((0,1),(1,1)). m4 =(0,1), m6 =(1,1).

$5\times7+1=36=100100_2$. Pairs: ((1,0),(0,1),(0,0)).

$v_2(36)=2$ (even). Division process:

```
36 ÷ 2 = 18    (100100 → 10010)    ← pair boundaries shift by 1 bit
18 ÷ 2 = 9     (10010 → 1001)      ← pair boundaries return to original
```

$n'=9=1001_2$. Pairs: ((1,0),(0,1)). m4' =(1,0), m6' =(0,1).

d=2 (even) → two right shifts restore pair boundaries to original positions. m4 remains m4, m6 remains m6. **No exchange.**

Item	Example 6.1 (n=27)	Example 6.3 (n=7)
5n+1	136	36
d	3 (odd)	2 (even)
Pair boundary	shifted	restored
m4↔m6	**Exchanged**	**Maintained**

6.6 Information Preservation Principle

Proposition 6.2. In one step $T(n)=(xn+1)/2^d$ of a Collatz-type map, the bit information of the input n is transformed as follows:
1. **Pair addition** (§3–4): New m4/m6 are generated via bit references and carry propagation. The bit count increases by at most a constant.

2. **Trailing-zero removal**: Bits carrying zero information are removed. No information is lost.

3. **Re-pairing**: Depending on the parity of d, the m4↔m6 correspondence is maintained or exchanged.

In particular, ÷2 is **positional relocation** of information (swapping left/right roles within pairs), not **destruction** of information.

Remark 6.1. The increase and decrease of digit count is directly related to convergence/divergence of the map. The result xn+1 is larger than n (bit count may increase), but $\div 2^d$ reduces the bit count. If d is sufficiently large, the digit count decreases and the trajectory "shrinks." For general x, the balance of this digit-count variation remains an open question. For x=3, the structural impossibility of sustained growth is established in §5.7 (Proposition 5.4 and Remark 5.3): the condition d=1 required for net bit growth cannot be maintained indefinitely.

6.7 The Map T in m4/m6 Representation

Integrating the above, the odd-to-odd map $T{:}n \mapsto n'$ is described as a transformation on (m4, m6) space:

$$T{:}(m4(n),\, m6(n)) \longrightarrow (m4(n'),\, m6(n'))$$

This transformation consists of three phases:

Phase 1 (Pair addition):

$$(m4(n),\, m6(n)) \rightarrow (m4(xn+1),\, m6(xn+1))$$

Phase 2 (Tail removal):

$$(m4(xn+1),\, m6(xn+1)) \rightarrow \text{shortened pair sequence}$$

Phase 3 (Exchange decision):

$$\text{shortened pair sequence} \rightarrow \begin{cases} (m4',\, m6') & d \text{ even: maintained} \\ (m6',\, m4') & d \text{ odd: exchanged} \end{cases}$$

6.8 Determination of d

The trailing-zero count d can also be read directly from the pair addition result.

> **Proposition 6.3.** In the pair addition result (LSB order), d equals the number of trailing zero bits. Counting q consecutive (0,0) pairs from the LSB, then checking the right bit of the next pair: if it is 0, then d=2q+1; if it is 1, then d=2q.

6.9 Summary of This Section

1. Repeated division by 2 removes trailing zero bits (carrying zero information); no information is lost.

2. Re-pairing after removal: the parity of d determines whether the m4/m6 correspondence is maintained (even) or exchanged (odd).

3. The map $T{:}n \mapsto n'$ is described as a closed three-phase transformation on (m4, m6) space: pair addition → tail removal → exchange decision.

4. The value of d itself is readable directly from the pair addition result, requiring no additional computation.

7. Methodological Note

An earlier version of this paper included numerical trajectory data as an appendix, with GPK statistics computed across Collatz trajectories. The verification tool employed a stopping-time algorithm—the standard optimisation in which the trajectory of n is terminated once $T^k(n)$, on the grounds that all smaller integers have already been verified for convergence. While this early termination is correct for convergence verification, it truncates the trajectory before reaching 1, and the GPK distribution statistics collected from such truncated trajectories carry a systematic bias: the tail segment of the trajectory (from the stopping-time threshold down to 1) is absent, distorting the measured G/P/K ratios. This error was identified during the process of improving the verification tool.

In the course of re-examining the numerical results, it became clear that no numerical data is required. Every claim in this paper—the per-pair determination of the m4-stage GPK (Theorem 5.1), the uniqueness of x=3 (Theorem 5.3), the self-annihilation of G-dominance (Proposition 5.4), and the structural impossibility of divergence (Remark 5.3)—follows from the algebraic identity 3n+1=2n+n+1 and the closure of the GPK operator space.

These are not finite verifications extrapolated to general claims. The identity 3n+1=2n+n+1 holds simultaneously for all $n \in N$ by algebraic necessity. The per-pair determination derived from this identity (Theorem 5.1) is therefore not a pattern observed in data but a structural property that admits no exceptions at any scale. The scale-invariance argument of Proposition 5.4 requires no numerical support: it follows from the fact that the per-pair determination is an established algebraic identity, not a finite verification. For this reason, the numerical data previously contained therein has been integrated into Section 9.

8. Cycle Analysis via m4/m6

As an application of the unified algorithm, we analyze the known cycles of 5n+1 in m4/m6 space. We draw structural comparisons with 3n+1 and demonstrate the perspective that m4/m6 representation provides for cycle research.

8.1 Known Cycles of 5n+1

The following non-trivial cycles of 5n+1 are known [2]:

Cycle A (2 elements): 1→3→1

Cycle B (3 elements): 13→33→83→13

Cycle C (3 elements): 27→17→43→27

8.2 m4/m6 Trajectory of Cycle A

Step	n	Binary	m4	m6	5n+1	d	Exchange
0	1	01	(0)	(1)	6	1	⇄
1	3	11	(1)	(1)	16	4	—
→	1	01	(0)	(1)			

Observation. m6 is fixed at (1) throughout. Only m4 oscillates 0→1→0. The values of d alternate between 1 and 4, and exchange occurs every other step.

8.3 m4/m6 Trajectory of Cycle B

Step	n	Binary	m4	m6	5n+1	d	Exchange
0	13	1101	(1,0)	(1,1)	66	1	⇄
1	33	100001	(1,0,0)	(0,0,1)	166	1	⇄
2	83	01010011	(0,0,0,1)	(1,1,0,1)	416	5	⇄
→	13	1101	(1,0)	(1,1)			

Observation. The pair count varies as 2→3→4→2. From 13 to 33, the pair count increases; the large d=5 at 83→13 sharply reduces it back. All values of d are odd, causing m4↔m6 exchange at every step.

8.4 m4/m6 Trajectory of Cycle C

Step	n	Binary	m4	m6	5n+1	d	Exchange
0	27	011011	(0,1,1)	(1,0,1)	136	3	⇄
1	17	010001	(0,0,0)	(1,0,1)	86	1	⇄
2	43	101011	(1,1,1)	(0,0,1)	216	3	⇄
→	27	011011	(0,1,1)	(1,0,1)			

Observation. The pair count remains fixed at k=3 throughout. The values of d alternate as 3, 1, 3, with exchange at every step.

Between 27 and 43, the (m4, m6) values at pair position 0 are swapped: (0,1) versus (1,0), while positions 1 and 2 are unchanged. However, when all three cycle elements including 17 are considered, each position exhibits a different pattern. Whether this position-0 swap reflects a structural property of the cycle or is an artifact of selecting a particular pair of elements remains unclear; the mechanism of m4/m6 pattern formation within cycles is outside the scope of this paper.

8.5 Structural Features Common to the Cycles

(1) Periodicity of exchange. In all three cycles, steps with odd d appear, triggering m4↔m6 exchange. In Cycles B and C, exchange occurs at every step.

(2) Pair-count variation patterns. Cycle C has fixed pair count (k=3); Cycle B has variable pair count (k=2→3→4→2). In variable-count cycles, a "compression step" with large d (e.g., d=5 in Cycle B) sharply reduces the pair count.

8.6 Comparison with the 3n+1 Cycle

The only known cycle of 3n+1 is the trivial cycle 1→1 (3×1+1=4, d=2). The Collatz conjecture asserts this is the unique cycle.

	n	m4	m6	3n+1	d	Exchange	n'
	1	(0)	(1)	4	2	—	1

d=2 (even), so no exchange occurs. m4 =(0) and m6 =(1) form a fixed point.

Structural comparison:

- 5n+1: Multiple cycles, variable pair counts, frequent m4↔m6 exchange.
- 3n+1: Unique cycle (conjectured), d=2 fixed point, no exchange.

The fact that the trivial cycle of 3n+1 is a fixed point in m4/m6 space is consistent with the "structural privilege" of §5. That m4-stage GPK can be read directly as m2/m7 (Theorem 5.1) means the map transforms the pair structure of n "transparently," providing a structural reason why complex cycles may be difficult to form.

8.7 Observation Resolution: Why 5n+1 Admits Cycles

The structural difference between 3n+1 and 5n+1 can be reformulated as a question of *observation resolution*. For a Collatz-type map with parameter $x=2^s+1$, the multiplication xn+1 involves a shift of s bits. The m4-stage carry at pair position i depends on the input at positions within distance $\lfloor s/2 \rfloor$ pairs. We say that the carry structure *closes at resolution w* if a window of w bits suffices to determine the carry classification at each position from the input alone.

For 3n+1 (s=1), the shift is 1 bit, which is less than the pair width of 2 bits. The m4-stage inputs at each pair position are (b_i,a_i)—both bits from the same pair (Proposition 3.4). The carry classification is therefore per-pair determined: it closes at resolution 2. This is the content of Theorem 5.1.

For 5n+1 (s=2), the shift equals the pair width. The m4-stage inputs become (a_{i-1},a_i), spanning two adjacent pairs (Proposition 3.3). The carry classification depends on inter-pair correlations that the 2-bit observation space cannot express (Theorem 5.2). The structural closure fails, and the GPK system does not constrain trajectory behavior in the same way.

This suggests a geometric condition for closure: **the carry structure of xn+1 closes at 2-bit resolution if and only if the shift s is strictly less than the pair width**. For s≥2, a wider observation window may restore closure. The conjectured minimum resolutions are:

Parameter x	Shift s	Minimum window (conjectured)	Status
3	1	2 bits	**Closed** (Theorem 5.1)
5	2	4 bits?	Open
9	3	6 bits?	Open
2^s+1	s	2s bits?	Open

The existence of non-trivial cycles for 5n+1 (Cycles B and C above) is consistent with this picture: the 2-bit observation space lacks the resolution to capture the carry mechanism that would constrain trajectories. Whether a 4-bit observation space provides structural closure for 5n+1—and whether the known cycles become structurally visible within that space—is the central open question connecting cycle analysis to the observation framework.

8.8 Formulation of Cycle Conditions

> **Definition 8.1.** A sequence of odd numbers $n_0,n_1,\dots,n_{p-1}$ forms a cycle of period p if $T(n_i)=n_{i+1 \bmod p}$ holds for all i.

In m4/m6 space, this translates to the following conditions.

Proposition 8.1 (m4/m6 cycle conditions). Necessary conditions for $\{n_0,\ldots,n_{p-1}\}$ to form a cycle of 5n+1:

(i) Pair-count condition: $\sum_{i=0}^{p-1}(\delta_i-d_i)=0$, where δ_i is the bit increase from addition at step i and d_i is the trailing-zero count. The cycle returns to its starting bit length.

(ii) Exchange condition: The total d-sum must be such that after p steps, the m4/m6 correspondence returns to its original state.

(iii) Bit-pattern condition: Composing p steps of the scan must reproduce the original m4/m6 pattern.

Conditions (i) and (ii) follow from the definition of cycles and the exchange principle of §6. Condition (iii) is the explicit composition of scans; its closed-form description is a topic for future work.

Verification. Cycle C (27→17→43→27): d=(3,1,3). $\sum d_i=7$ (odd), which appears to violate condition (ii).

However, bit removal such as d=3 involves re-arrangement of pair boundaries, and the determination of exchange depends not on the simple sum of dmod2 but on the consistency of the entire bit string at each step. Condition (ii) requires a more precise formulation that accounts for boundary shifts during re-pairing. This refinement is left for future work.

8.9 Summary of This Section

1. The three known cycles of 5n+1 have been completely described in m4/m6 space, including the m4/m6 transitions, GPK classifications, d values, and exchange occurrences at each step.

2. Common structural features have been extracted: periodicity of exchange, pair-count variation patterns, and m4/m6 symmetric relationships between cycle elements.

3. The trivial cycle of 3n+1 (n=1) has been confirmed as a fixed point in m4/m6 space, and structural comparisons with 5n+1 have been drawn.

4. The structural difference between 3n+1 and 5n+1 has been reformulated as observation resolution: 2-bit resolution suffices for 3n+1 (shift $s=1<2$) but not for 5n+1 (shift $s=2$). Whether higher-resolution windows restore closure for $x\geq5$ is an open question (§8.7).

5. Cycle conditions in m4/m6 space have been formulated. While refinement remains for future work, the pair-count and exchange conditions provide necessary conditions.

9. Discussion

9.1 Relation to Prior Work

Relation to the Syracuse function. The Syracuse function $T(n)=(3n+1)/2^{v_2(3n+1)}$ [1] elides even steps and defines the odd-to-odd map in a single expression. However, it does not penetrate the internal structure of 3n+1, and the value of v_2 can only be known after computation. Our m4/m6 scan yields the same result as T(n), but in the process obtains the carry structure (GPK classification) of each pair as a by-product. The Syracuse function is a *definition* of the map; the m4/m6 scan is a *decomposition* of the map.

Relation to binary analysis. Binary-based analysis of Collatz maps by Wirsching [3] and Lagarias [1] treats 3n+1 as 2n+n+1 and tracks bit-string changes. Our approach extends this direction but differs in the following respects:

- Decomposition into 2-bit pairs and the introduction of the 16-predicate system uses the *pair type* (4 kinds) rather than individual bits as the basic unit. This allows carry behavior to be described as a function of pair type.
- The choice of the m4/m6 basis ensures that the map is described as a closed transformation on (m4, m6) space.
- The completeness and per-pair independence of the 16 predicates provides tools for establishing both positive results (Theorem 5.1) and negative results (Theorem 5.2): "what is expressible and what is not."

Relation to carry-lookahead adders. The GPK decomposition of §4 is based on the same principle as carry-lookahead technology introduced by Bedrij [6]. Our contribution is to connect this technology to the carry structure of Collatz-type maps and to show that the GPK classification is interpretable as pair predicates (specifically m2/m7). This bridges a standard technique from computer science to a problem in number theory.

9.2 Novelty of This Paper

(1) The m4/m6 pair-projection framework. We decomposed natural numbers into 2-bit pairs and defined a predicate system with m4/m6 as basis. We established completeness of the 16 predicates, reconstruction from the basis, and per-pair independence.

(2) Decomposition of xn+1 into bit scanning. We showed that the odd-to-odd transition of xn+1 is computable by an m4/m6 bit scan. The multiplication xn is decomposed into the shift of $(x-1)n$ as reference index offsets; the scan stage involves only intra-pair 1-bit additions.

(3) Derivation of the structural privilege of 3n+1. We established that for x=3, the m4-stage GPK coincides with m2/m7 (Theorem 5.1), and that for $x\geq5$, this coincidence is impossible in principle (Theorem 5.2). The Classification Theorem (5.3) shows that x=3 is the unique parameter for which the m4-stage carry structure closes within the 16-predicate space. This m4-stage closure enables the complete scan dynamics to be captured by a 16-pattern finite-state transducer whose input alphabet is derived from the predicate pair types.

(4) Exchange principle of ÷2. We showed that the ÷2 operation is described as an m4↔m6 role exchange (Theorem 6.1).

(5) Unified algorithm. We formulated an algorithm that accommodates arbitrary xn+1 by swapping only the reference pattern (Algorithm 4.1 with Table 3.1).

(6) First GPK statistics of Collatz-type maps. While stopping-time distributions and trajectory statistics for Collatz-type maps have been extensively studied, per-step GPK ratios and carry chain length histograms have not appeared in the literature. The measurements reported in §9.3 constitute, to our knowledge, the first empirical GPK characterization of xn+1 maps.

9.3 Computational Verification

Using our implementation of Algorithm 4.1, we measured GPK distributions for x=3,5,9 using full-path tracking. For x=3, all odd $n \leq 10^9$ (500,000,000 numbers) were tracked to n=1. For x≥5, trajectories do not generally converge; each trajectory was tracked for a fixed number of steps (max_steps), accumulating GPK over the entire computation. The results are shown in Table 9.1.

Methodological note. GPK statistics measure the carry structure of the xn+1 addition in each odd-to-odd step (Syracuse function). The division by 2^d involves no addition and produces no GPK. For x=3, full-path tracking to n=1 captures the complete trajectory including the descent phase. For x≥5, trajectories enter non-trivial cycles or diverge; the max_steps cap provides a uniform computation budget. The total odd-to-odd steps aggregated are approximately 3.5×10^{10} for both 3n+1 and 5n+1, ensuring comparable sample sizes. The structural arguments of §5.7, which derive all conclusions from finite case analysis of Tables 5.1 and 5.2, do not depend on these numerical values.

Table 9.1. GPK distribution comparison for Collatz-type maps $T(n)=(xn+1)/2^d$ (full-path tracking).

	3n+1 (s=1)	**5n+1 (s=2)**	**9n+1 (s=3)**
Range	$n \leq 10^9$	$n \leq 6.99 \times 10^7$	$n \leq 6.99 \times 10^7$
Odd numbers checked	500,000,000	34,969,256	34,970,000
Method	full path to n=1	max 1000 steps	max 500 steps
All reach n=1	**Yes**	No	No
Total odd-to-odd steps	3.50×10^{10}	3.50×10^{10}	1.75×10^{10}
G (Generate) %	**38.52**	37.16	37.32
P (Propagate) %	**29.89**	25.26	25.16
K (Kill) %	**31.59**	37.58	37.52
G/K ratio	**1.22**	0.99	1.00
Carry chain peak	2–3	7–8	9–10
Max carry chain length	23	71	69

Observation 1: G/K asymmetry is unique to x=3. For x=3, G/K=1.22 is clearly asymmetric, whereas for x=5 and x=9, G/K≈0.99–1.00 is nearly symmetric. By Theorem B, the GPK for x=3 coincides with the intra-pair

predicates of n (G=m2, P=m7, K=m9), so the bias in bit patterns of odd numbers is directly reflected in the GPK distribution. For x≥5, GPK depends on inter-pair correlations (Theorem C), mixing the input structure and homogenizing toward G≈K.

Observation 2: P≈25% is common for x≥5. For x=5, P=25.26%; for x=9, P=25.16%—both near 1/4, and converging closer to 1/4 as s increases. This is not coincidental. For x≥5, Propagate in the serial composition of m4-stage and m6-stage arises independently: $P \approx P_{m4} \cdot P_{m6} \approx 1/2 \times 1/2 = 1/4$. For x=3, P=29.89% is higher because the m4-stage input is the pair itself (a_i, b_i), creating correlation with the m6-stage. This elevated P is a direct signature of structural closure (Theorem B).

Observation 3: Carry chain peak shifts rightward with s. The carry chain length histogram peaks at 2–3 for x=3 (s=1), 7–8 for x=5 (s=2), and 9–10 for x=9 (s=3); see Figure 9.1 (log scale) and Figure 9.2 (linear scale). For x=3, the histogram decays exponentially from the peak; for x≥5, the histogram is bell-shaped with a longer tail. Notably, carry chains of length ℓ=0 are entirely absent for x=3 (the histogram begins at ℓ=1), while they occur for both x=5 and x=9. This is a direct numerical confirmation of the structural closure: for x=3, every Generate position is followed by at least one Propagate before the next Kill, because the intra-pair predicates structurally prevent G-K adjacency. For x≥5, Kill placement depends on inter-pair correlations (Theorem C), allowing both isolated carries (ℓ=0) and occasional long chains. Note that in the tail region (ℓ≳55), event counts drop to single digits against trillions of total pairs; the fluctuations visible in Figure 9.1 reflect the limits of available computation time rather than structural features.

Figure 9.1. Carry chain length distribution for *xn*+1 mappings (log scale).

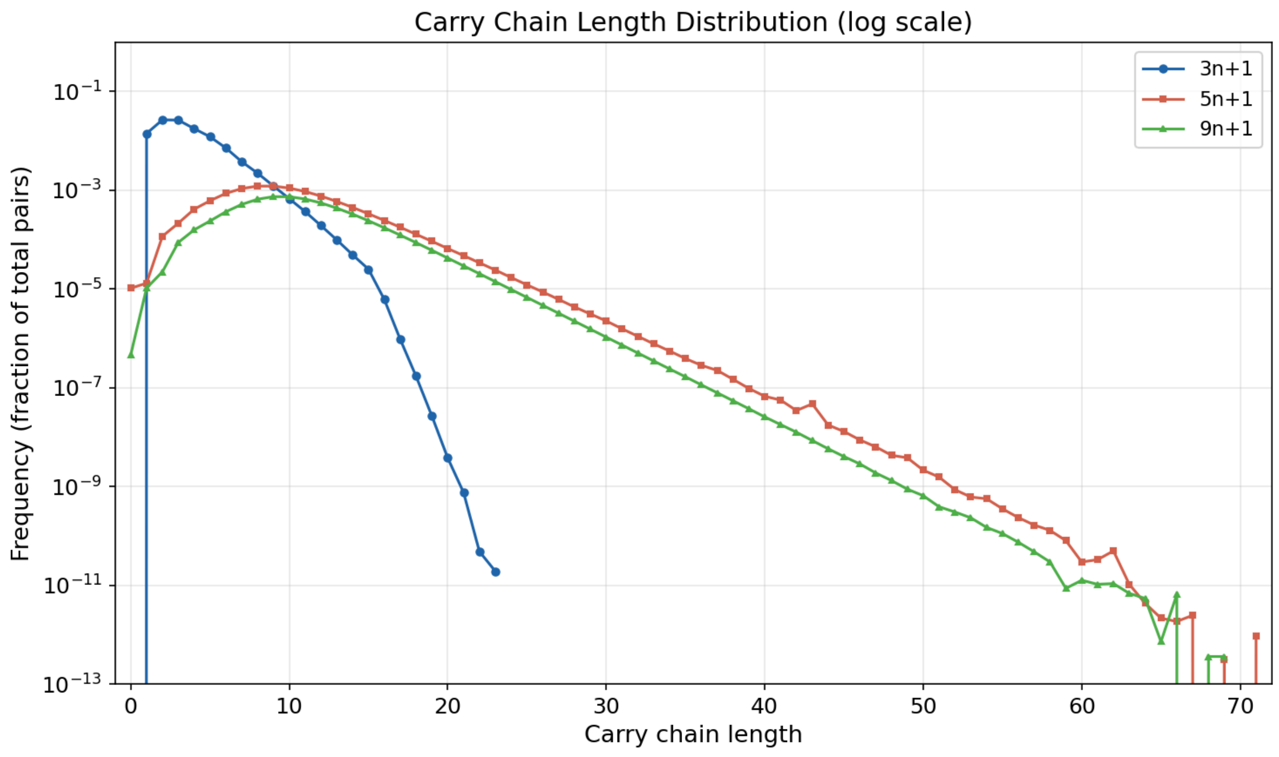
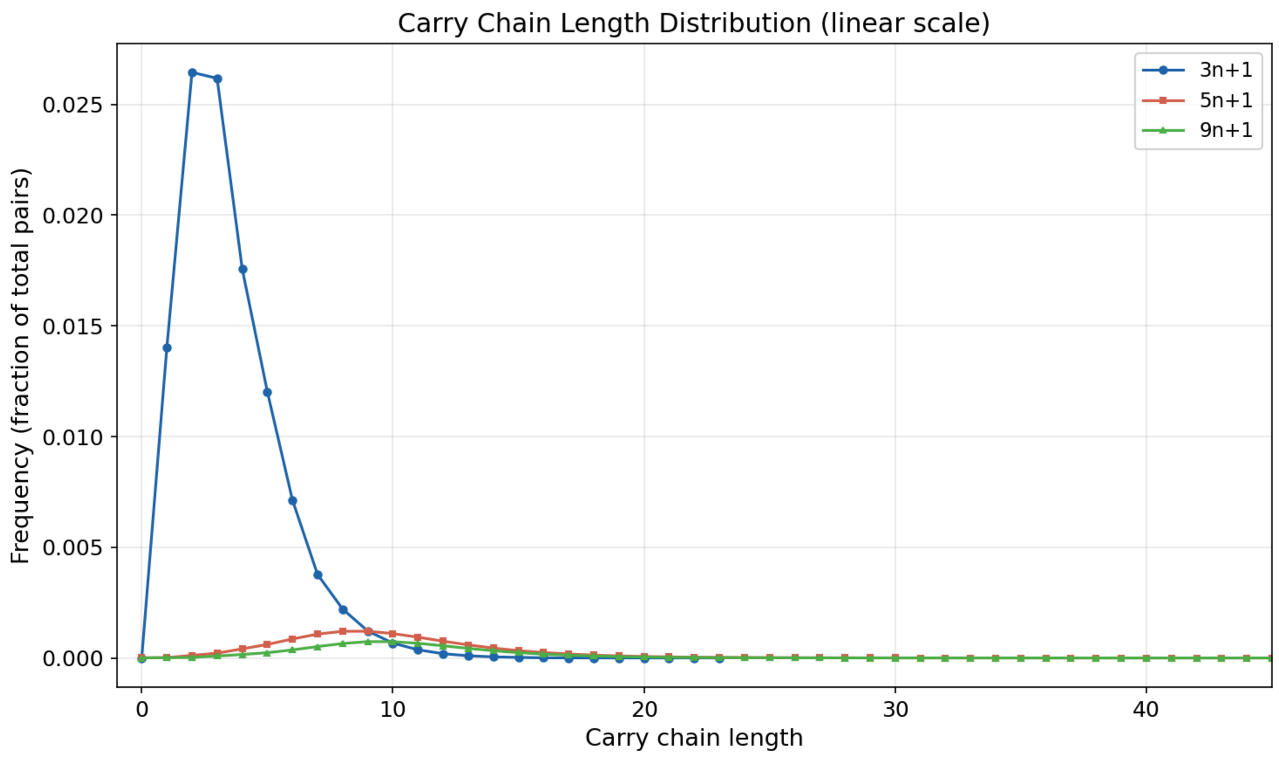


Figure 9.2. Carry chain length distribution for *xn*+1 mappings (linear scale).



9.4 Limitations of This Paper

Limitations that should be stated frankly:

(1) Divergence vs. convergence. Proposition 5.4, supported by Lemmas 5.1–5.8 derived from the exhaustive 16 + 16 case analysis (Tables 5.1 and 5.2), establishes that G-dominance self-annihilates through the inseparable coupling of activation condition and activation result within the closed GPK system. The one-step map decomposes into four phases (scan, d-determination, trailing zero removal, re-pairing), each structurally closed within the same four pair types, as confirmed by the complete enumeration of Tables 5.1 and 5.2. The DFST permutation property (§5.7.1) further shows that the scan preserves the multiplicity of pair types across DFST states. Since the closed, self-annihilating model for x=3 structurally admits only the 1-convergent cycle, understanding the mechanism of non-trivial cycle formation necessitates further analysis of other x parameters (§8). (As a supplementary observation, the expected division exponent under uniform pair-type distribution is $E[d]=2>\log_2 3$, suggesting net contraction (§5.7.5); this probabilistic estimate is independent of the structural argument of Proposition 5.4.)

(2) Incompleteness of cycle conditions. The cycle conditions formulated in §8 are necessary conditions only; sufficient conditions have not been obtained. A complete cycle analysis requires understanding the p-step composition of m4/m6 patterns, and may benefit from the structural analysis of 5n+1 cycles (§8.1–8.5) as a comparative framework. This is the principal remaining open problem.

(3) Cases where x−1 is not a power of 2. Extension to x=7,11,13,… is in principle possible but adds complexity as additional addition stages arise. This paper does not treat these cases.

9.5 The Role of "Tool" versus "Result"

The present paper establishes the structural closure of 3n+1 within the 2-bit pair predicate system and derives its consequences. The m4-stage carry structure is fully determined by intra-pair predicates (Theorem 5.1), and this closure is unique to x=3 (Theorem 5.3). The m4-stage closure determines the input alphabet (four pair types) of a deterministic finite-state transducer that captures the complete scan dynamics, including the m6-stage inter-pair contributions. Within this closed finite-state system, Proposition 5.4 establishes that the mechanism deterministically prevents trajectory divergence.

9.6 Future Directions

(1) m4/m6 characterization of cycles. Refine the cycle conditions of §8 to derive conditions for non-existence/existence of non-trivial cycles from periodicity of m4/m6 patterns. The structural analysis of 5n+1 cycles (§8.1–8.5), particularly the m4↔m6 symmetric relationships observed in Cycle C. To elucidate the mechanism of non-trivial cycle formation, further observation and structural analysis of different x values (x≥5) is necessary.

(2) Dynamical system on m4/m6 space. Formulate $T:(m4,m6)\to(m4',m6')$ as a dynamical system and study attractor structure. Showing that the fixed point (m4,m6)=((0),(1)) is a global attractor would be equivalent to the Collatz conjecture.

(3) Pattern analysis of GPK sequences. Analyze how the m2/m7/m9 bit string of n is transformed by one step of the scan under 3n+1. In particular, how the positions of m9 (Kill, pair (0,0)) evolve along the trajectory governs the reach of carry propagation.

(4) Higher-resolution observation spaces. The observation resolution framework (§8.7) reformulates the structural difference between 3n+1 and x≥5 as a geometric condition: the shift s versus the observation window width. A natural conjecture is that a 2s-bit observation space might provide structural closure for $x=2^s+1$ (e.g., 4-bit pairs for 5n+1); however, this remains conjectural and has not been investigated in this paper. Whether closure at higher resolution, if it exists, would yield the same structural consequences as the 2-bit closure for 3n+1 is an open question. The m4↔m6 symmetric exchange observed in Cycle C of 5n+1 (§8.4) may provide structural constraints visible within such a higher-resolution space.

9.7 Summary of This Section

This paper has decomposed Collatz-type maps in the framework of a 2-bit pair predicate system, established three foundational theorems (m4-stage coincidence, 16-predicate limitation, Classification Theorem), and derived a structural consequence: within the closed GPK system unique to x=3, Generate-dominant configurations self-annihilate through the carry mechanism they produce (Proposition 5.4), establishing that no trajectory of 3n+1 can diverge. The principal remaining open question is the construction mechanism of non-trivial cycles (§8, §9.6).

10. Conclusion

This paper introduced a predicate system based on 2-bit pair decomposition of natural numbers and performed a structural decomposition of Collatz-type maps $T(n)=(xn+1)/2^d$. We summarize the main results.

10.1 Summary of Main Results

Result 1: Establishment of the pair predicate system (§2). We decomposed the binary representation of n into 2-bit pairs and defined a system of 16 Boolean predicates with the left-bit projection m4(LEFT) and right-bit projection m6(RIGHT) as basis. We established completeness, reconstruction from the basis, and per-pair independence.

Result 2: Reduction of xn+1 to bit scanning (§3, §4). In the decomposition $xn+1=(x-1)n+n+1$, the left shift of $(x-1)n$ determines reference patterns on the pair structure. Addition at each pair position has a two-stage structure (m6 and m4), and carries are determined by a single $O(k)$ scan from the LSB. This yields Algorithm 4.1, which accommodates different parameters x by simply swapping reference patterns (Table 3.1).

Result 3: The structural privilege of 3n+1 (§5). This is the core result of the paper.

- **Theorem B (Theorem 5.1):** For x=3, the m4-stage Generate coincides with m2(AND), Propagate with m7(XOR), and Kill with m9(NOR), at every pair position. Carry generation, propagation, and absorption are determined merely by reading the pair type of n.
- **Theorem C (Theorem 5.2):** For $x \geq 5$, the m4-stage GPK cannot be expressed by any Boolean composition of the 16 predicates. This is due to the principled incompatibility between the per-pair independence of the 16 predicates and the inter-position correlations demanded by the map.
- **Classification Theorem (Theorem 5.3):** The m4-stage GPK closes within intra-pair predicates only for x=3. The map 3n+1 is the unique Collatz-type map whose carry structure is "transparent" in the 16-predicate space.
- **Proposition 5.4 (Self-annihilation of G-dominance):** Within the closed GPK system, Generate-dominant configurations deterministically self-annihilate through the carry propagation they produce. The one-step map decomposes into four phases (scan, d-determination, trailing zero removal, re-pairing), each structurally closed within the same four pair types. This is derived via exhaustive finite case analysis: a 16-pattern scan transition table (Table 5.1) and a 16-pattern re-pairing transition table (Table 5.2) yield Lemmas 5.1–5.8, which collectively establish the inseparable coupling of activation condition (G-dominance) and activation result (G-block destruction). The DFST permutation property confirms that no phase structurally biases pair type distribution. Combined with Remark 5.3, this establishes that divergence of 3n+1 trajectories is structurally impossible.

Result 4: Exchange principle of ÷2 (§6). We showed that the ÷2 operation is described as an exchange of the roles of m4 and m6 (Theorem 6.1). The parity of d determines whether exchange occurs. Information is preserved; ÷2 is positional relocation.

10.2 Significance

The results of this paper provide a new perspective on Collatz-type maps.

First, **structural decomposition of multiplication**. The arithmetic operation xn+1 has been decomposed into bit reference patterns and carry scanning. The value of x is absorbed as a shift of reference indices and does not appear in the scan stage. While shift+add is equivalent to multiplication at the binary level, our method transparently describes the internal mechanism at the pair level, making the carry structure (GPK) directly observable.

Second, **identification of the singularity of 3n+1**. To the question "why 3n+1?" this paper presents a structural answer. x=3 is the unique parameter for which the shift amount is less than the pair width, causing the addition to fold back within the pair. As a result, the carry structure is described directly by the intra-pair predicates m2/m7/m9 of n. This "transparency" cannot hold for x≥5 in principle.

Third, **explicit statement of limitations**. The per-pair independence of the 16 predicates is both the strength and the limitation of this framework. Inter-pair correlations—information that carry propagation of Collatz-type maps essentially requires—lie outside the 16 predicates. Describing them requires either extending the framework (introducing inter-pair predicates) or employing dynamic operations (scanning). This paper adopted the latter.

10.3 Outlook

Proposition 5.4 establishes that no trajectory of 3n+1 can diverge. The principal remaining question is:

- Refining cycle conditions to establish the construction mechanism of non-trivial cycles (§8, §9.6(1)). The m4/m6 framework provides the structural tools (§8.8), and the comparative analysis of 5n+1 cycles (§8.1–8.5) offers concrete structural constraints.

This is the gap between structural impossibility of divergence and the full Collatz conjecture (convergence of all trajectories to 1).

10.4 Closing

The Collatz conjecture has resisted resolution for over 80 years despite its elementary formulation. Within the framework developed in this paper, the full conjecture is framed by the distinction between the structural closure of x=3 and the cycle formation of x≥5. This suggests that the problem belongs to a broader structural inquiry regarding the presence or absence of non-trivial cycles across the xn+1 family.

The author has not proved anything. The 2-bit pair decomposition is a standard grouping of binary digits. The GPK classification is a half-century-old technique from carry-lookahead adder design. The algebraic identity 3n+1=2n+n+1 is elementary. Every derivation in this paper consists of substituting known values into known structures and recording what follows. If the structural descriptions presented here have any value, it belongs to the generations of researchers across number theory, computer arithmetic, dynamical systems, and combinatorics whose accumulated work made these observations possible. This paper merely directed an existing lens at an existing problem, revealing structures that may pave the way for further insights.

References

[1] J. C. Lagarias, "The 3x+1 problem and its generalizations," *American Mathematical Monthly*, vol. 92, no. 1, pp. 3–23, 1985.

[2] R. K. Guy, *Unsolved Problems in Number Theory*, 3rd ed. New York: Springer, 2004.

[3] G. J. Wirsching, *The Dynamical System Generated by the 3n+1 Function*, Lecture Notes in Mathematics, vol. 1681. Berlin: Springer, 1998.

[4] R. Terras, "A stopping time problem on the positive integers," *Acta Arithmetica*, vol. 30, pp. 241–252, 1976.

[5] C. J. Everett, "Iteration of the number-theoretic function f(2n)=n, f(2n+1)=3n+2," *Advances in Mathematics*, vol. 25, pp. 42–45, 1977.

[6] O. J. Bedrij, "Carry-select adder," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 3, pp. 340–346, 1962.

[7] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, 1973.

[8] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, 1982.

Appendix (Overview)

The complete version of this paper includes the following appendices.

Appendix A: Complete Definition Table of the 16 Predicates. Definitions of m1–m16, truth tables, expressions in m4/m6, and complement pair correspondences are listed.

Appendix B: Complete Computation Trace for n=27. For each step of the 5n+1 cycle 27→17→43→27, the reference bits, local GPK, carry values, and output bits at every pair position are enumerated. A similar trace is given for the 3n+1 trajectory of n=27 (until reaching n=1).

Appendix C: Formal Definition of GPK Tree Composition. Complete derivation of associativity (Proposition 4.3), pseudocode for the binary tree composition algorithm, and complexity analysis.

Appendix D: General Derivation of Reference Patterns. General formula for reference bits $(r_i^{(4)},r_i^{(6)})$ as a function of shift amount s. Discussion of extension to cases where $x-1$ is not a power of 2.

Appendix E: Verification Code. Python implementation of the Unified Algorithm (Algorithm 4.1 with Table 3.1). Verification scripts confirming agreement with arithmetic computation for 3n+1 and 5n+1. Exhaustive matching scripts for all 784 predicate combinations (verification of Theorem C).