The Complexity of Factoring Sums of Cycles

Nuutti Kuosa
ORCID: 0009-0003-4770-6289

27 November 2026

Abstract

A sum of directed cycles represents a finite dynamical system whose transition map is a permutation. Its decomposition into a direct product of two nontrivial systems is a classical algorithmic problem, also studied as conjugated tensor decomposition of a permutation. We establish its decision and counting complexity. For explicit transition tables, deciding nontrivial factorization is NP-complete, and counting unordered nontrivial factor pairs up to isomorphism is #P-complete under parsimonious reductions. The reduction classifies every nontrivial factor pair: for suitable sums A, B of odd-length cycles, the nontrivial pairs of $B + C_2A$ are exactly $\{A, X + C_2\}$ with $AX = B$, where C_2 is a two-cycle. Both factors are irreducible, and the target $B + C_2A$ has $|B| + 2|A|$ states. We give a self-contained parsimonious reduction from 3-SAT to the required division instances, including the decoding of arbitrary quotients and a polynomial bound on the explicit state space. For binary cycle-list encodings, a sparse-witness argument gives NP-completeness as well. We also give an elementary proof of the output-length obstruction to #P membership for counting all binary factor pairs, a consequence of earlier bounds on the number of quotients.

hexagon:2611.01191v1

1 Introduction

The direct product of two finite dynamical systems describes their synchronous evolution. Recovering factors from a given system is therefore a natural decomposition problem: can the observed transition function be expressed as two smaller functions evolving independently? Even when every state is periodic, the answer is not determined by the lengths of individual cycles. Different pairs of cycles can contribute to the same cycle length in the product, and their multiplicities must be compatible simultaneously.

Write C_d for a directed cycle of length d. Disjoint union is addition, and direct product satisfies

$$C_aC_b = \gcd(a, b)C_{\operatorname{lcm}(a,b)}.$$

Indeed, a pair of states first returns to its starting position after $\operatorname{lcm}(a, b)$ steps. The ab pairs of states therefore split into $ab/\operatorname{lcm}(a, b) = \gcd(a, b)$ cycles of that length. For example, $C_2C_3 = C_6$, whereas $C_2C_2 = 2C_2$. Thus a system consisting only of cycles is a finite sum $\sum_d a_dC_d$ with nonnegative integer coefficients. The multiplicative unit is C_1. A factorization is nontrivial when both factors have at least two states. We write $|S|$ for the number of states of a system S. Throughout the paper, systems and factors are identified up to isomorphism; for sums of cycles, this means equality of all cycle multiplicities.

There are two different questions. In *division*, both A and B are given and one asks whether $AX = B$ has a solution. In *reducibility*, only T is given and one asks whether $T = UV$ has a nontrivial factorization. A reduction from division cannot simply exhibit one possible factorization: it must also rule out every unintended pair U, V. This reverse implication is the central issue addressed here.

1.1 Results

Our main theorem concerns the explicit representation, in which the input lists the successor of every state. The binary representation lists distinct cycle lengths and their positive multiplicities, both in binary.

Theorem 1.1 (Main result)**.** *Deciding whether a nonempty sum of cycles is reducible is NP-complete in both the explicit and binary representations. Deciding irreducibility is coNP-complete, with the unit treated separately. In the explicit representation, counting unordered nontrivial factor pairs up to isomorphism is #P-complete under parsimonious reductions.*

The hardness statements already hold on a family with the following simultaneous properties:

1. *every cycle length is squarefree and greater than one;*
2. *the gcd of the cycle lengths and the gcd of the positive multiplicities are both one;*
3. *every nontrivial factor pair consists of two distinct irreducible systems;*
4. *every such pair contains the same irreducible factor A;*
5. *all even cycle lengths have multiplicity one, and A is recovered by halving exactly those lengths.*

In particular, the difficulty remains when a prescribed candidate factor can be read directly from the input. The unknown part must still satisfy all multiplicity constraints.

1.2 Proof strategy

The reduction passes through the following descriptions of the same solution set:

$$\begin{aligned}&\text{satisfying assignments of a Boolean formula}\\ &\quad\longleftrightarrow \text{feasible integer tables}\\ &\quad\longleftrightarrow \text{quotients } X \text{ satisfying } AX = B\\ &\quad\longleftrightarrow \text{nontrivial factor pairs of } T = B + C_2 A.\end{aligned}$$

Every arrow is a bijection. In particular, decoding must recover an assignment from an arbitrary factor pair, not just from the pairs we intended to construct. This is what preserves the number of solutions.

The last arrow is the shortest part, so we prove it first. For suitable odd-length cycle sums A, B, every nontrivial factor pair of T is exactly $\{A, X + C_2\}$ with $AX = B$. The marker 2 makes one factor recognizable and increases the state count only to $|B| + 2|A|$.

The longer task is to construct those division instances. Table entries record Boolean values, and bounds on weighted sums of entries express copying and clause constraints. An algebraic construction turns the slack in these bounds into cycle multiplicities. Its converse recovers the table from every quotient. We then place the Boolean constraints in a grid whose addresses encode polynomially bounded cycle lengths. The geometric details ensure that a constraint reaches exactly its intended entries. These two parts are proved in Sections 4 and 5; Section 6 assembles them into the explicit theorem.

For binary inputs, a sparse-witness argument proves NP membership. Counting all factor pairs has a different complexity: earlier quotient-count bounds imply an output too long for #P. Section 7 gives a direct proof of this obstruction.

1.3 Context and scope

Egner, Püschel and Beth [16] study the same permutation factorization problem and give an enumeration algorithm and large-output bounds; Glasby's polynomial tensor-factorization results allow a broader class of factors [17, 18]. To our knowledge, the decision and counting complexity classification proved here is new. It settles the explicit-successor-table formulation of the irreducibility question raised in the 2026 workshop abstracts, while reaction-rule inputs and general polynomial equations require separate results [12, 11]. Section 9 gives the detailed attributions and scope comparison, including the binary cycle-list result.

1.4 Organization

Section 2 fixes the representation and counting conventions. Section 3 proves the factor-pair classification, before the longer source reduction is introduced. Sections 4 and 5 construct and decode the division instances. Section 6 proves the explicit complexity results, and Section 7 treats binary inputs. Section 8 gives matrix and fixed-point-sequence interpretations of the explicit classification. Section 9 compares the results with earlier work and specifies which open questions they resolve. The proofs of the stated theorems and corollaries are contained in the article.

2 Preliminaries and input conventions

A finite dynamical system is a finite set S with a map $f : S \to S$. Its direct product with (R, g) has state set $S \times R$ and transition $(s, r) \mapsto (f(s), g(r))$. We work with nonempty systems. A permutation system is one in which f is bijective; it is uniquely specified, up to isomorphism, by its cycle sum

$$A = \sum_{d \geq 1} a_d C_d, \qquad \operatorname{supp} A = \{d : a_d > 0\}, \qquad |A| = \sum_d d a_d.$$

The support is finite. We write $[C_d]A = a_d$, and use 0 for the empty sum when it occurs in an intermediate expression. Coefficientwise inequalities are written $A \leq B$. The mass $|A|$ is the number of states, and satisfies $|A + B| = |A| + |B|$ and $|AB| = |A||B|$.

An *irreducible* system is a nonunit whose every factorization has a unit factor. Thus C_1 is neither reducible nor irreducible. Irreducibility is different from the Euclidean prime property $A \mid BC \Rightarrow A \mid B$ or $A \mid C$; no assertion about that property is made here. A factor pair is an unordered pair with multiplicity, so a square factorization $T = U^2$ is counted once. We count isomorphism classes of pairs, not vertex labellings or choices of an isomorphism.

Lemma 2.1 (Basic constraints on factors)**.** *If $XY = T \neq 0$ is an equality of cycle sums, then every length in X or Y divides a length in T, and every positive multiplicity in either factor is at most the largest multiplicity in T. If a nonempty direct product of arbitrary finite dynamical systems is a permutation system, both factors are permutation systems.*

Proof. A cycle of length a in one factor and a cycle of length b in the other make a positive contribution at length $\operatorname{lcm}(a, b)$. There is no cancellation, proving the divisibility claim. That contribution is at least either factor multiplicity, proving the second claim. For the last assertion, injectivity of the product map implies injectivity of each component map by fixing one state of the other nonempty factor. An injective map on a finite set is bijective. □

An explicit input is a successor table on N labelled states, of bit length $O(N \log N)$. The cycle list can be computed by following each orbit once. Conversely a cycle list of mass N can be expanded in $O(N \log N)$ bit operations. In the binary model the input is a sorted list of pairs (d, a_d) with positive binary integers; write L for its total bit length, including delimiters. The mass can then be exponential in L.

For nonempty inputs, define $\mathrm{DIV}(A, B)$ by the existence of a cycle sum X with $AX = B$, and $\mathrm{RED}(T)$ by the existence of $UV = T$ with $|U|, |V| \geq 2$. The function $\#\mathrm{PAIR}(T)$ counts

such unordered pairs. A parsimonious reduction preserves the number of solutions exactly. All reductions below are deterministic polynomial-time reductions in the stated input model.

We use the standard parsimonious #P-completeness of counting satisfying assignments of 3-CNF formulas. The parsimonious convention is important: in the circuit-to-3-CNF encoding, an auxiliary variable records the value of each gate and the clauses enforce that value in both directions. Each assignment to the original input bits therefore has exactly one extension to the gate variables, and the output variable is forced to accept. Clauses of size at most three suffice. We retain unused input variables explicitly, since their two choices must also be counted.

3 Forcing every factor pair

We first isolate a short algebraic argument. It shows why a prescribed divisor can be forced without solving an additional factorization problem. A cycle sum is called p-free when none of its cycle lengths is divisible by the prime p.

Think of $U + C_pV$ as two layers: the lengths of U avoid p, while those of C_pV contain it. If both factors used a p-divisible length, their product would contribute at least p cycles at such a length, by the gcd in the product rule. Keeping all marked multiplicities below p prevents this. Hence only one factor can carry the marker, and comparing the two layers exposes a common divisor of U and V. The next lemma makes this argument exact.

Lemma 3.1 (Two-layer classification)**.** *Let U, V be nonempty p-free cycle sums, where p is prime and every multiplicity of V is less than p. Every unordered nontrivial factor pair of $U + C_pV$ has a unique orientation*

$$U + C_pV = J(E + C_pF), \qquad JE = U, \quad JF = V, \quad |J| \geq 2,$$

where J, E, F are nonempty and p-free. Conversely, every such triple gives a nontrivial factor pair.

Proof. All lengths in the target have p-adic exponent zero or one. By Lemma 2.1, factor lengths have the same restriction. If both factors contained a cycle with length divisible by p, their product would contribute at least p to a multiplicity at a p-divisible length. This contradicts the strict bound on the multiplicities of V. At least one factor contains such a cycle, because $C_pV \neq 0$. Consequently exactly one factor does. Denote the other by J and split the marked factor uniquely as $E + C_pF$, with all three sums p-free.

The unmarked part of the product is $JE = U$, so $E \neq 0$. The marked part gives $C_pJF = C_pV$ and $F \neq 0$. On p-free sums, multiplication by C_p simply replaces C_d by C_{pd} without changing its coefficient, and is injective. Hence $JF = V$. The marked factor has at least two states, so the remaining nontriviality condition is $|J| \geq 2$. Multiplication proves the converse. The marked factor is determined by the target factor pair, which also proves uniqueness of the orientation and of the triple. □

Theorem 3.2 (Prescribed irreducible factor)**.** *Let A be irreducible and $B \neq 0$. Let p be a prime dividing no cycle length of A or B, and suppose $p > \max_d[C_d]A$. Then*

$$X \longmapsto \{A, X + C_p\}$$

is a bijection from all solutions of $AX = B$ to all unordered nontrivial factor pairs of $T = B + C_pA$. Both members of every pair are distinct and irreducible, and $|T| = |B| + p|A|$.

Proof. Apply Lemma 3.1 with $U = B$ and $V = A$. An arbitrary nontrivial pair gives $JE = B$ and $JF = A$, with J a nonunit. Irreducibility of A forces $J = A$ and $F = C_1$. Thus the other factor is $E + C_p$, where $AE = B$. Conversely, $A(X + C_p) = B + C_pA$ for every quotient X. The map is injective, since the marked factor identifies the orientation and removing its single p-cycle recovers X.

By Lemma 2.1, X is p-free. Apply the two-layer classification to $X + C_pC_1$: a nontrivial factorization would require a nonunit dividing C_1, which is impossible. Hence $X + C_p$ is irreducible. It differs from A because it contains a p-divisible length. The mass identity follows from additivity and multiplicativity of mass. □

The strict coefficient bound is substantive. For example, $p = 2$, $A = C_3 + C_5 + 2C_{15}$ and $B = C_1$ give

$$B + C_2A = (C_1 + C_6)(C_1 + C_{10}), \qquad A \nmid B.$$

Here A is irreducible: its mass is 38, so a nontrivial factorization would have a two-state factor. Neither $2C_1$ nor C_2 can divide it, because its coefficients and its lengths are not all even, respectively. Thus freshness of p and irreducibility alone do not suffice.

Lemma 3.3 (A convenient irreducible family)**.** *If $A = \sum_{i=1}^{k} C_{a_i}$, where $k \geq 2$ and the distinct lengths a_i are pairwise coprime, then A is irreducible.*

Proof. Suppose $A = UV$. Multiplicity one in the target forces all factor multiplicities to be one, and distinct cycle pairs cannot produce the same target length. If U contains two lengths, multiplying them by any length b of V gives two distinct target lengths divisible by b. If $b > 1$, these target lengths have a common divisor greater than one, contrary to pairwise coprimality. Thus every length of V is 1. Its multiplicity is also one, as observed above, so $V = C_1$. Otherwise U is one cycle, and V must contain at least two lengths because A does. The symmetric argument gives $U = C_1$. □

Corollary 3.4 (A constant marker)**.** *Suppose all lengths of A, B are odd and A satisfies Lemma 3.3. Then every nontrivial factor pair of $T = B + C_2A$ is exactly $\{A, X + C_2\}$ for a solution of $AX = B$. Moreover $|T| = |B| + 2|A|$, and the even part of T consists of one copy of each C_{2a} with $a \in \operatorname{supp} A$.*

Proof. Theorem 3.2 applies because 2 is fresh and every coefficient of A is one. □

Example. Let $A = C_3 + C_5$ and $B = 15C_3 + 15C_5 + 8C_{15}$. The only quotient lengths can be $1, 3, 5, 15$, by Lemma 2.1. Equating coefficients in $AX = B$ yields

$$x_1 + 3x_3 = 15, \qquad x_1 + 5x_5 = 15, \qquad x_3 + x_5 + 8x_{15} = 8.$$

The nonnegative integer solutions are $X = 15C_1 + C_{15}$ and $X = 5C_3 + 3C_5$. Therefore the 256-state system $T = B + C_6 + C_{10}$ has exactly two nontrivial pairs, namely $\{A, 15C_1 + C_2 + C_{15}\}$ and $\{A, C_2 + 5C_3 + 3C_5\}$. The theorem excludes every other pair, not merely every other pair containing an assumed divisor.

The rest of the proof constructs division instances with precisely this odd, coefficient-one, pairwise-coprime divisor structure. The marker argument will then apply without any factorization search.

4 Encoding interval constraints as division

We first turn interval constraints on an integer table into a division equation; the next section constructs the tables that encode Boolean formulas.

The algebraic idea is to make the product forget the entries of the table while the quotient remembers them. Each interval supplies two complementary slack quantities. We place them at two related cycle lengths so that their total contribution to the product is fixed. The main work is then the converse: showing that an arbitrary quotient has exactly this form, so that no extra solutions escape the encoding.

We begin with the algebraic construction from the explicit-division reduction [8], giving its full proof. Let $H_1, \dots, H_D$ be pairwise disjoint sets of primes, where $D \geq 2$ and $|H_j| \geq 2$, and choose a prime ω outside all these sets. Write

$$Q_j = \prod_{p \in H_j} p, \qquad \mathcal{V} = \prod_{j=1}^{D} H_j, \qquad \mathcal{E} = \prod_{j=1}^{D} (\{1\} \cup H_j).$$

Elements of $\mathcal{V}$ are *full addresses*. An element e of $\mathcal{E}$ is a *partial address*: its coordinates equal to 1 are unspecified. For $e \in \mathcal{E}$, let $d_e = \prod_j e_j$ and let $s(e)$ be the number of unspecified coordinates. We write $v \succeq e$ when the full address v agrees with every specified coordinate of e.

Assign each full address a capacity $\kappa_v \in \mathbb{N}_0$. For a table $x = (x_v)_{v \in \mathcal{V}}$, define the weighted marginal

$$f_e(x) = \sum_{v \succeq e} \frac{d_v}{d_e} x_v, \qquad W_e = f_e(\kappa).$$

Choose integer intervals $[L_e, R_e] \subseteq [0, W_e]$, with $[L_v, R_v] = [0, \kappa_v]$ at full addresses. A table is *feasible* if $L_e \leq f_e(x) \leq R_e$ for every $e \in \mathcal{E}$. Thus the full-address constraints include the bounds $0 \leq x_v \leq \kappa_v$. An address is *active* when its capacity is positive. In the Boolean construction below all capacities are zero or one, so active entries are precisely the entries allowed to carry a bit.

The notation used throughout the interval construction is collected here.

Symbol	Meaning
D	Number of coordinate groups
$H_j,\ Q_j$	Prime labels of coordinate j, and their product
$\mathcal{V},\ \mathcal{E}$	Full addresses and partial addresses
$v,\ e,\ d_e$	A full address, a partial address, and the product of its specified primes
$\kappa_v,\ f_e,\ W_e$	Entry capacity, weighted marginal, and its capacity bound
$L_e,\ R_e$	Lower and upper endpoints of the marginal interval

For example, take $D = 2$, $H_1 = \{5, 7\}$ and $H_2 = \{11, 13\}$. The partial address $e = (1, 11)$ describes the two-entry marginal

$$f_e(x) = 5x_{(5,11)} + 7x_{(7,11)}.$$

With unit capacities, requiring $5 \leq f_e(x) \leq 7$ makes the two integer entries complementary: exactly one is 1. The construction below stores the remaining slack as a cycle multiplicity, so that this inequality becomes nonnegativity of a quotient.

Lemma 4.1 (Interval realization)**.** *Suppose that some capacity is positive and that a feasible rational table exists. One can construct a nonempty cycle sum B such that, for*

$$A = C_\omega + \sum_{j=1}^{D} C_{Q_j},$$

the feasible integer tables are in bijection with the isomorphism classes of solutions to $AX = B$. All cycle lengths in A and B are squarefree. The lengths in A are pairwise coprime and each has coefficient one. If $U_e = R_e - L_e$, then

$$|B| = |A|\,\omega \sum_{e \in \mathcal{E}} d_e U_e \leq |A|\,\omega\, 2^D \sum_{v \in \mathcal{V}} d_v \kappa_v. \tag{1}$$

Proof. There are two reasons for the shape of the quotient below. First, nonnegativity of z_e and $U_e - z_e$ expresses membership in the interval of width U_e. Second, the sign of the dependence on

x alternates whenever one coordinate becomes unspecified. This will cancel the identity relating a marginal to its refinements. Temporarily allowing signed rational coefficients, set

$$z_e(x) = \begin{cases} f_e(x) - L_e, & s(e) \text{ is even,} \\ R_e - f_e(x), & s(e) \text{ is odd,} \end{cases}$$

and define

$$Y(x) = \sum_{e \in \mathcal{E}} (z_e(x) C_{\omega d_e} + \omega(U_e - z_e(x)) C_{d_e}). \tag{2}$$

The lengths d_e and ωd_e appearing here are all distinct. Consequently, $Y(x)$ has nonnegative coefficients exactly when x is feasible. We compute $B = AY(0)$ using signed integer arithmetic. Four observations establish the claim.

The product is independent of x. Fix a coordinate j and fix the other coordinates of a partial address e whose jth coordinate is unspecified ($e_j = 1$). Write $e[j \leftarrow p]$ for its refinement with $p \in H_j$. Then

$$f_e(x) = \sum_{p \in H_j} p f_{e[j \leftarrow p]}(x).$$

For the difference between any two tables, the alternating definition of z gives

$$\Delta z_e + \sum_{p \in H_j} p \, \Delta z_{e[j \leftarrow p]} = 0.$$

The required coefficient collection follows directly from the cycle rule:

$$\begin{aligned} C_{Q_j} C_{d_e} &= C_{Q_j d_e}, & C_{Q_j} C_{d_{e[j \leftarrow p]}} &= p C_{Q_j d_e}, \\ C_{Q_j} C_{\omega d_e} &= C_{\omega Q_j d_e}, & C_{Q_j} C_{\omega d_{e[j \leftarrow p]}} &= p C_{\omega Q_j d_e}. \end{aligned}$$

Indeed, d_e is coprime to Q_j, while $d_{e[j \leftarrow p]} = p d_e$ has gcd p with Q_j; the extra prime ω changes neither gcd. Thus the coefficients collect at a common length with weights $1, p$ in each layer of (2). Hence $C_{Q_j} Y(x)$ is independent of x. The remaining summand of A gives

$$C_\omega Y(x) = \sum_{e \in \mathcal{E}} \omega U_e C_{\omega d_e},$$

which is also independent of x. Therefore $AY(x) = B$ for every rational table x.

The cycles C_{Q_j} therefore enforce compatibility between marginals, while C_ω pairs the two slacks at each address. Neither part reveals the chosen table in the product. This explains why the same target B can have one quotient for each feasible table.

In the example above, with $\omega = 3$ and $[L_e, R_e] = [5, 7]$, we have $z_e = 7 - f_e(x)$ and $U_e = 2$. The identity

$$z_e + 5x_{(5,11)} + 7x_{(7,11)} = 7$$

is the fixed coefficient produced by $C_{35} Y(x)$ at length $3 \cdot 35 \cdot 11$. The guard at length 33 has coefficient $3z_e + 3(2 - z_e) = 6$. These are the two cancellations used in general.

In particular, $Y(0)$ shows that B has integer coefficients, whereas a feasible rational table shows that those coefficients are nonnegative. A positive capacity gives $U_v = \kappa_v > 0$ at a full address, so B is nonempty. The product formula also shows that all its lengths divide the squarefree integer $\omega \prod_j Q_j$.

Every quotient has the prescribed support. A length in B contains at least two primes from at most one group H_j. Indeed, $C_\omega Y(0)$ uses at most one prime from each group, while $C_{Q_j} Y(0)$ uses the entire group H_j and at most one prime from each other group. Cancellation in the signed calculation can remove lengths but cannot create additional ones.

Let $AX = B$, with X a genuine nonnegative cycle sum, and let b be a length in X. The positive contribution from $C_\omega C_b$ implies $b \mid \omega \prod_j Q_j$. If b contained two primes from a group H_j,

choose $k \neq j$. Then $C_{Q_k} C_b$ would contribute positively at a length containing at least two primes from each of H_j and H_k, which is impossible. Thus every length in X is d_e or ωd_e for some $e \in \mathcal{E}$.

The table can be recovered uniquely. For this support, the contributions of the different cycles of A have disjoint supports: a length in $C_\omega X$ contains no full prime group, while a length in $C_{Q_j} X$ contains exactly the full group H_j. The assumption $|H_j| \geq 2$ is used here. These contributions can therefore be read separately from B. Put $z'_e = [C_{\omega d_e}]X$. Comparing the $C_\omega X$ layer gives

$$[C_{d_e}]X = \omega(U_e - z'_e), \qquad 0 \leq z'_e \leq U_e. \tag{3}$$

At a full address set $x_v = z'_v$, an integer in $[0, \kappa_v]$. If $e_j = 1$, only the summand C_{Q_j} of A can contribute to the coefficient at $\omega Q_j d_e$. Within X, the possible lengths are exactly ωd_e and $\omega d_{e[j \leftarrow p]}$ for $p \in H_j$, contributing with weights $1, p$ as above. Comparing that coefficient gives the recursion

$$z'_e = [C_{\omega Q_j d_e}]B - \sum_{p \in H_j} p\, z'_{e[j \leftarrow p]}. \tag{4}$$

Here $e[j \leftarrow p]$ replaces the unspecified jth coordinate by p. The coefficients $z_e(x)$ satisfy the same recursion because $AY(x) = B$. They agree with z'_e at full addresses, where $z_v(x) = x_v$. Induction on $s(e)$ proves equality at every partial address. Equation (3) then recovers the remaining coefficients and yields $X = Y(x)$. The nonnegativity of X implies feasibility of x. Conversely, every feasible integer table gives a quotient $Y(x)$, and its coefficients at the lengths ωd_v recover the table. This proves the bijection.

The number of states has the stated bound. The two terms associated with e in (2) have total mass $\omega d_e U_e$, independently of $z_e(x)$. Multiplicativity of mass gives the equality in (1). For the inequality, use $U_e \leq W_e$. In $\sum_e d_e W_e$, each full address v contributes $d_v \kappa_v$ once for each of its 2^D partial addresses. □

The rational feasibility assumption is essential to this construction: it certifies that the signed calculation produces a nonnegative target even when no integer solution exists. Our Boolean encoding will always have the rational solution $x_v = 1/2$ at active addresses.

Here a *coordinate line* consists of the full addresses extending a partial address with precisely one unspecified coordinate. The weights in its marginal are the primes labeling that coordinate's active values. Only particularly simple intervals are needed. On a coordinate line with exactly two active addresses and weights p, q, the interval

$$\min(p, q) \leq px + qy \leq \max(p, q) \tag{5}$$

holds for bits x, y exactly when $x + y = 1$. On a line with three active addresses and weights p, q, r, the interval

$$\min(p, q, r) \leq px + qy + rz \leq p + q + r \tag{6}$$

holds exactly when $x \vee y \vee z$ is true. Both intervals admit the all-halves assignment.

5 A polynomial-size encoding of Boolean formulas

We next realize all variable-consistency and clause constraints as the lines just described. We use three copies of the routing grid, called *planes*. A *test* imposes a clause condition or consistency between positive and negative literals, using selected planes. A *signal* is the route of one literal occurrence from one test, on one plane. Its initial point is a *test vertex*. A *copying edge* joins consecutive points of a signal route or of an output-block connection; it changes one coordinate and imposes complementarity of the two bits. These edges form the *copying graph*; the tests themselves are not its edges. A *marked line* is a coordinate line selected for a copying edge or a Boolean test. The active points will be the route and block-connection vertices, each with capacity one.

Merely giving each occurrence a separate coordinate would make the cycle lengths too large. Instead, we use a routing network to place occurrences in a small-dimensional grid. The routing must satisfy two conditions simultaneously: different literals must remain separate, and every marked line must contain exactly its intended active addresses.

It helps to keep the desired copying graph in mind before constructing its coordinates. Each occurrence of a literal will have a path from its clause or sign test to a block shared by all occurrences of that literal on the same plane. These paths and blocks must form one connected component for each literal and plane, with no accidental connections between components. Along each edge the bit is complemented; a coloring will make all test vertices of a component carry the same bit. The copying part of the geometric construction realizes this graph by coordinate lines with two active points.

Theorem 5.1 (Parsimonious explicit division)**.** *Given a Boolean 3-CNF formula with n explicitly specified variables and m clauses, one can construct in polynomial time nonempty cycle sums A, B whose quotients $AX = B$ are in bijection with the satisfying assignments. The lengths in A are pairwise coprime, odd, and squarefree, and all its coefficients are one. There are at least two such lengths, each greater than one. Every length in B is odd, squarefree, and greater than one. For $n + m \to \infty$, the construction satisfies*

$$|A| = (n+m)^{o(1)}, \qquad |A| + |B| \le (n+m)^{3+o(1)}.$$

In particular, the output can be written as explicit successor tables in polynomial time.

We prove the theorem in five steps: route arbitrary permutations, embed those routes in a grid, place the Boolean constraints, verify the bijection, and bound the resulting system sizes. All choices of orderings, matchings and output permutations are made deterministically from the formula.

5.1 Routing a permutation by changing one digit at a time

A stage changing one coordinate may permute its symbols differently for each fixed choice of the other coordinates. Think of the words agreeing outside that coordinate as one switch. The lemma says that a short sequence of such switches can send each input word to any prescribed distinct output word.

Lemma 5.2 (A multivalued permutation network)**.** *For integers $u \ge 2$ and $g \ge 1$, every permutation of $\{0, \dots, u-1\}^g$ is a composition of $2g-1$ permutations, each changing just one coordinate. The coordinate sequence is $1, 2, \dots, g, \dots, 2, 1$, and the composition can be found in time polynomial in u^g.*

Proof. The case $g = 1$ is immediate. For $g > 1$, group input and output words by their last $g-1$ coordinates. Make a bipartite multigraph whose edges are the input words: an edge joins its input group to the group of its image. This graph is u-regular. It has a perfect matching, since for every set of input vertices, counting its incident edges gives Hall's condition. Repeatedly removing a perfect matching partitions the edges into u matchings, found by polynomial-time augmenting-path algorithms.

Color the matchings with $0, \dots, u-1$. The first stage changes the first digit of each word to the color of its edge. This is a permutation because each input group has one edge of every color. For each fixed color, its perfect matching pairs every input group with exactly one output group and uses every output group once. Thus the suffixes define a permutation on $g-1$ digits; route it recursively in $2g-3$ stages. The last stage changes the first digit to the required output digit. It is a permutation because each output group also has one edge of every color. All recursive branches use the same coordinate sequence and can be performed in parallel. This gives $2g-1$ stages. □

This is the usual recursive principle of a Beneš network [6]. The proof specifies the multivalued version needed here and, in particular, its constructive running-time bound.

5.2 A grid in which copying lines have two endpoints

The network specifies which signals should move, but its edges are not yet suitable marginal constraints. A coordinate line could contain several signals, and imposing one inequality on that line would mix their bits. We address three different issues: a checksum limits the number of address points on a line; lanes separate signals moving at the same stage; and time labels separate successive stages without requiring a large alphabet. The plane coordinate will later let three literal values meet at a clause test.

Fix $u \geq 3$ and $g \geq 1$, with $u^g \geq m + 6n$ so that every test will receive its own word. The size estimate below specifies these parameters. We use $u \geq 3$ as a uniform convention. The local routing and separation arguments also work for $u = 2$; the polynomial size estimate later uses the specified Λ-dependent choice of u. A word $a \in \{0, \ldots, u-1\}^g$ has checksum $\sigma(a) = \sum_i a_i \pmod{u}$. Use the two address types

$$S(a) = (a, \sigma(a)), \qquad I(a) = (a, \sigma(a) + 1 \pmod{u}),$$

called stable and intermediate addresses. Every coordinate line in this $(g+1)$-dimensional address space contains at most two addresses of these types. For a checksum line the word is fixed and there are two possible checksums. For a line varying one word digit, the other digits and the checksum determine at most one stable and one intermediate address.

Let $\mathbf{e}_i$ be the ith unit vector. Increasing digit i modulo u gives a two-edge path

$$S(a),\ I(a),\ S(a + \mathbf{e}_i).$$

Thus a change of one digit can be routed by at most $u - 1$ such increments, using at most $2u - 1$ vertices and an even number of edges. All intermediate values are taken modulo u. Stable and intermediate addresses remain distinct even when a digit wraps around; their types, rather than the parity of their numeric coordinates, will define the coloring below.

For example, with $u = 3$, incrementing the first digit of the word $(2, 0)$ uses the addresses $S(2,0) = (2,0,2)$, $I(2,0) = (2,0,0)$ and $S(0,0) = (0,0,0)$. The first edge changes only the checksum and the second only the word digit, even though both changes wrap around.

We add a lane coordinate. Lane 0 is the main lane, and $1, \ldots, u$ are buffer lanes. Each stage of Lemma 5.2 receives three time layers: entrance, routing, and exit. Suppose that the stage changes digit i and maps a to b. Its signal follows this path:

1. In the entrance layer, move from $S(a)$ in the main lane to $S(a)$ in buffer lane $a_i + 1$.
2. Move to the routing layer in that buffer lane, then follow the address path from $S(a)$ to $S(b)$.
3. Move to the exit layer, still in the same buffer lane, and then move to $S(b)$ in the main lane.

Writing $\ell = a_i + 1$ for the buffer lane, the stage has the form

$$\begin{array}{ccccc} \text{entrance} & & \text{routing} & & \text{exit} \\ S(a)_0 & & & & S(b)_0 \\ \downarrow & & & & \uparrow \\ S(a)_\ell & \longrightarrow & S(a)_\ell \longrightarrow S(b)_\ell & \longrightarrow & S(b)_\ell \end{array}$$

Subscripts indicate lanes. The middle path from $S(a)_\ell$ to $S(b)_\ell$ consists of the two-edge S, I, S increments above. Connect the main-lane exit of one stage to the main-lane entrance of the next. Within a group of words agreeing outside digit i, different signals receive different buffer lanes. Different groups have an unchanged word coordinate separating their paths. At entrance and exit, the stage permutation guarantees a unique signal at each stable address. Consequently, distinct signals do not meet inside a stage. Distinct stages have distinct time labels. Each main-lane

connector joins the unique exit at $S(b)$ to the next stage's unique entrance at $S(b)$, continuing the same signal without joining different signals.

The time layers must themselves use small coordinate alphabets. Let

$$P = ((0,0),(0,1),(1,1),(1,2),\ldots,(u-1,u-1)).$$

Every coordinate line contains at most two points of P. A reflected grid traversal of P^k visits all its points exactly once, with consecutive points changing a single coordinate. More explicitly, traverse the first copy of the previous-dimensional path forward, the next backward, and continue alternating. Choose

$$h = 3(2g-1), \qquad k = \lceil \log_{2u-1} h \rceil,$$

and use the first h points of this traversal for the time layers. Each time line still contains at most two used points. The time encoding uses $2k$ coordinates, each with alphabet size u. Indeed, fixing all but one time coordinate fixes every other copy of P and one coordinate in the remaining copy, where only two points can lie on that line. Passing to the first h points cannot increase this number. Two points on a time line need not be consecutive in the traversal; that line is marked only if a copying edge actually uses it.

This last distinction is useful throughout the construction. We do not require every pair of collinear active points to be connected. We choose the copying edges first, and mark only their lines. The two-point property guarantees that each selected line imposes exactly the edge constraint we intended.

Finally add a plane coordinate with three values. Copying paths stay in one plane; the Boolean tests will join selected planes. The full grid therefore has

$$D = g + 1 + 2k + 2 \tag{7}$$

coordinates: the word and checksum, the time coordinates, the lane, and the plane. Every alphabet has size between 2 and $u+1$. The roles of these coordinates are summarized below. The numeric symbols are grid labels; they will next be replaced by distinct primes.

Coordinates	Number	Purpose
Word	g	Identify a signal's address
Checksum	1	Isolate address-line endpoints
Time	$2k$	Separate entrance, routing and exit layers
Lane	1	Separate simultaneous routes inside a stage
Plane	1	Place the three literals of a clause

Assign each coordinate its own disjoint group of primes, one prime per alphabet symbol. This identifies grid points with the full addresses used in Lemma 4.1.

5.3 Placing literals and clauses

Assume first that $n \geq 1$ and every clause has between one and three literals. Pad a shorter clause to three by repeating its last literal. This preserves both truth and the number of satisfying assignments.

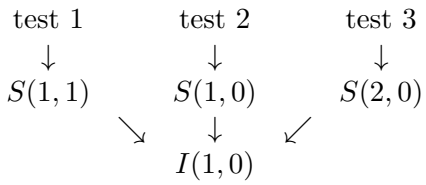
There are $m + 6n$ tests. Each clause gives a test requesting its three literals on planes $1, 2, 3$. The remaining tests synchronize a variable's positive and negative literals across the three planes. Using three planes lets even repeated occurrences of one literal in a clause occupy distinct points. It also gives a simple way to synchronize signs: complementarity between positive and negative occurrences on different planes will force all three positive values to agree and all three negative values to be their complements. For each variable x and each ordered pair (j, ℓ) of distinct planes, add a *sign test* requesting x on plane j and $\neg x$ on plane ℓ; the third plane is inactive. Assign a distinct input word to every test and use that same word on all its participating planes. The input

vertices are stable addresses at the first time layer in the main lane. In particular, unused variables still receive their six sign tests; their two possible truth values are preserved in the solution count.

On each plane separately, group the requested occurrences by literal. Choose an output permutation placing each literal's occurrences in a consecutive block of a reflected Hamilton path of the word grid $\{0, \dots, u-1\}^g$. To fix its orientation, start with $0, 1, \dots, u-1$ in the last coordinate; add the remaining coordinates one at a time on the left. Traverse successive copies of the previous path alternately forward and backward. Consecutive copies then meet at equal suffixes, so every step changes one digit by one. Complete the output assignment to a permutation by assigning unused input words to unused output words. These additional words are only a device for applying Lemma 5.2: their signals are never activated. Route the real occurrences using the network and grid construction.

At the last time layer in the main lane, join consecutive stable addresses of each literal block by two-edge checksum paths. The Hamilton path changes a digit by 1 in the ordinary, noncyclic grid. A forward step from a to $a + \mathbf{e}_i$ uses $S(a), I(a), S(a + \mathbf{e}_i)$; a backward step uses the same path in reverse. Different blocks have disjoint stable addresses. An intermediate address $I(a)$ can be shared by two such connections only when both word-grid edges are incident to a. If both connections are within blocks, their common stable endpoint puts them in the same block. Thus the added paths never join different literals. At this last time layer the signal routes contain only stable endpoints in the main lane. The new block paths therefore meet a signal route only through their intended stable endpoints; the buffer lanes cannot create an additional intersection with a main-lane block path.

For a concrete block turn, take $u = 3$, $g = 2$ and three consecutive words $(1,1), (1,0), (2,0)$ on the reflected path, with the second digit varying first. If they belong to the same literal block, both connections use $I(1,0)$. The following schematic shows how three test vertices of that literal are joined; each vertical arrow above a stable address represents an entire signal route.



The bottom three edges lie in the final layer and main lane of one plane. Here $I(1,0) = (1,0,2)$, while the three stable addresses are $(1,1,2), (1,0,1), (2,0,2)$; each edge changes exactly one coordinate. The shared intermediate point joins occurrences of the same literal. It is one vertex of the copying graph, not two independent choices. The edge from $S(1,0)$ to $I(1,0)$ is used by both block connections and imposes a single complementarity constraint.

Give capacity one to all vertices of real signal routes and block connections, and capacity zero to every other full address. Mark each coordinate line used by a copying edge and impose its complementarity interval (5). At the input time and main lane, mark each test line in the plane direction. For a sign test use the complementarity interval; for a clause test use the disjunction interval (6). Give every other partial address its entire interval $[0, W_e]$.

We verify that marking a line imposes exactly the intended constraint:

1. A line in a word or checksum coordinate contains at most two stable/intermediate addresses. The endpoints of a marked edge are active, so these are exactly its active addresses.

2. A time-coordinate line contains at most two points of the chosen subset of P^k. Again, a marked edge supplies both.

3. Lane lines are marked only at entrance and exit. There is a unique signal at the relevant stable address, giving precisely its main-lane and buffer-lane copies. Routing layers may have more occupied lanes on the same line, but no lane line is marked there. The output-block connections add only intermediate addresses in the main lane, so they do not add points to the marked lane lines at stable addresses.

4. A test line is at the input time in the main lane. Its active points are exactly the requested planes, because unused signals were never activated.

Different types of copying edges free different coordinates. If a copying line is encountered twice, it has the same two endpoints and receives the same interval. Test lines free the plane coordinate, which copying edges never change. Hence the interval specification is consistent.

5.4 Recovering the Boolean assignment

The preceding separation arguments identify the connected components; we must still check that complementing along paths is consistent. In particular, different paths to the same vertex must prescribe the same value. A single coloring that changes on every copying edge proves this at once and also identifies the values seen by the tests.

For an active point, let ϵ be 0 for a stable address and 1 for an intermediate address. Write its time coordinates as $t = (t_1, \dots, t_{2k})$ and its lane as b. Define

$$\chi = \epsilon + \sum_{r=1}^{k} (t_{2r} - t_{2r-1}) + \mathbf{1}_{b \neq 0} \pmod 2. \tag{8}$$

Every copying edge changes this color. An address edge changes stable/intermediate type, a time edge moves between a diagonal and an off-diagonal point in one copy of P, and a lane edge moves between the main lane and a buffer lane. Every test vertex has color zero, because the traversal starts at the all-zero time point and tests use stable main-lane addresses.

Here the copying graph includes only copying edges, not the sign or clause tests. It has exactly one connected component for each literal on each plane. All occurrences of that literal are joined at the output, and each real signal connects its input to its output. No connection to a different literal is introduced, for three reasons:

1. Within a stage, buffer lanes separate signals whose words agree outside the changed digit, and an unchanged digit separates different groups (Section 5.2).

2. Between stages, distinct time labels separate the interiors, and each main-lane connector continues the unique signal at its stable address (Section 5.2).

3. At the output, distinct literal blocks have disjoint stable endpoints. A shared intermediate point can occur only within one block, and block paths meet routes only at their intended endpoints (Section 5.3).

Every variable has both signs represented on every plane because of the six sign tests. A bit at any one test vertex therefore extends uniquely to its entire component: at a vertex of color χ, use that bit XOR χ. All complementarity constraints on copying edges are then satisfied.

For a fixed variable, write π_j and ρ_j for the test values of its positive and negative literals on plane j. The six sign tests impose $\pi_j + \rho_\ell = 1$ whenever $j \neq \ell$. Their graph is $K_{3,3}$ with a perfect matching removed, hence a connected six-cycle. These constraints force

$$\pi_1 = \pi_2 = \pi_3, \qquad \rho_1 = \rho_2 = \rho_3 = 1 - \pi_1.$$

Thus a choice of the original variable determines every copied bit uniquely. The clause lines accept precisely the assignments satisfying the formula. We have obtained a bijection between satisfying assignments and feasible integer tables.

A one-variable instance. Take one variable x and no clauses. There are six sign tests, one for each ordered pair of distinct planes, and each test requests two occurrences. Thus there are twelve signals. We may take $u = 3$, $g = 2$, since nine input words suffice for the six tests. These are exactly the choices in the size construction below: $\Lambda = 6$ gives $\lambda = 3$, $u = \max(3, \lceil 3/2 \rceil) = 3$

and $g = 2$. On each plane the two occurrences of x are joined in one copying component, and the two occurrences of $\neg x$ in another. There are six copying components altogether, each containing two test vertices.

After replacing each copying component by its common test value, the sign-test constraints have the following shape:

$$\begin{array}{ccccc} \pi_1 & \longleftrightarrow & \rho_2 & \longleftrightarrow & \pi_3 \\ \updownarrow & & & & \updownarrow \\ \rho_3 & \longleftrightarrow & \pi_2 & \longleftrightarrow & \rho_1 \end{array}$$

Each drawn edge requires its two values to sum to one. This is the graph of sign constraints between the six components, not the copying graph itself. There are exactly two choices:

x	(π_1, π_2, π_3)	(ρ_1, ρ_2, ρ_3)
0	$(0, 0, 0)$	$(1, 1, 1)$
1	$(1, 1, 1)$	$(0, 0, 0)$.

The coloring extends either row uniquely to every active vertex, and all inactive entries are zero. Hence this instance has exactly two feasible integer tables and, by the interval lemma, exactly two quotients. It illustrates why even an unused input variable contributes its intended factor of two to the solution count.

Regardless of satisfiability, set every active entry to $1/2$ and every inactive entry to zero. A two-point line then has weighted sum $(p + q)/2$, inside its prescribed interval. A three-point line has sum $(p + q + r)/2$, at least its smallest weight and at most the sum of all weights. Unmarked constraints hold automatically. This is a feasible rational table. Lemma 4.1 therefore produces a nonnegative integer target B even for an unsatisfiable formula, and its decoding direction gives the required bijection between satisfying assignments and quotients.

For completeness, an input containing an empty clause is mapped to $A = C_3 + C_5$, $B = C_3$, which has no quotient by its mass. A formula with no variables has either zero or one satisfying assignment; use the same zero instance or $A = B = C_3 + C_5$, respectively. The latter has the unique quotient C_1, again by mass. No free auxiliary Boolean variable is introduced. These constant instances satisfy the stated restrictions on lengths and coefficients.

5.5 Bounding the explicit output size

It remains to choose the alphabet size so that both the routing grid and its prime labels are small enough. A polynomial bound suffices for the hardness result; the following choice also gives the stated exponent $3 + o(1)$.

An active grid point is not a single state of the final dynamical system: its prime labels determine a cycle length, which can be much larger. We must bound both the number of active points and these lengths before applying the mass formula. There is a tradeoff in the alphabet size. Smaller alphabets require more coordinates, and hence more prime factors in an address; larger alphabets make the products Q_j in the divisor larger. The parameters below balance these costs.

Write Λ for the formula-size parameter, reserving R_e for an interval endpoint. Set

$$\Lambda = \max(m + 6n, 2n, 3), \qquad \lambda = \min\{s \geq 3 : s^s \geq \Lambda\},$$

and choose

$$u = \max\left(3, \left\lceil \frac{\lambda}{\lceil \log_2 \lambda \rceil} \right\rceil\right), \qquad g = \min\{s \geq 1 : u^s \geq \Lambda\}.$$

There are enough input words for all tests. As $\Lambda \to \infty$,

$$\log \Lambda \sim \lambda \log \lambda, \qquad \log u \sim \log \lambda, \qquad g \sim \lambda, \qquad u = o(\lambda).$$

Since $h = O(g)$, the time parameter $k = \lceil \log_{2u-1} h \rceil$ stays bounded. Consequently $D = g + O(1)$, and all the coordinate groups together require $O(gu)$ primes. Choose $\omega = 3$ and assign successive

primes starting at 5 to the groups. The standard bound $p_r = O(r \log r)$ for the rth prime [7] implies that the largest prime p_* used satisfies

$$\log p_* \le (2 + o(1)) \log \lambda.$$

Every group has at most $u + 1$ members, so

$$|A| \le (D + 1) p_*^{u+1} = \Lambda^{o(1)}.$$

For clarity, the logarithm of this bound is $O(\log \lambda + u \log \lambda) = O(\lambda) = o(\log \Lambda)$. A full address has

$$\log d_v \le D \log p_* \le (2 + o(1)) \lambda \log \lambda = (2 + o(1)) \log \Lambda,$$

so $d_v \le \Lambda^{2+o(1)}$. There are $3m + 12n$ real signals. Each uses $O(gu)$ vertices, and the output-block connections add only a linear number in the number of signals. Thus the set $\mathcal{V}^+$ of active addresses has size $\Lambda^{1+o(1)}$. Since $2^D = \Lambda^{o(1)}$, the mass bound from Lemma 4.1 gives

$$|B| \le |A| \, \omega \, 2^D \sum_{v \in \mathcal{V}^+} d_v \le \Lambda^{3+o(1)}.$$

The exponent 3 is the sum $1 + 2$: the number of active addresses costs $\Lambda^{1+o(1)}$, and the largest address length costs at most $\Lambda^{2+o(1)}$. The remaining factors contribute only $\Lambda^{o(1)}$. The second exponent comes from having $D \sim \lambda$ coordinates while $\log p_* \le (2 + o(1)) \log \lambda$; the latter bound follows from using $O(gu)$ prime labels, with $gu = \lambda^{2+o(1)}$, and the bound for the rth prime. These estimates explain the upper bound; no matching lower bound is needed here. All selected primes are odd, so the lengths in both A and B are odd. Squarefreeness and the remaining restrictions on A follow directly from the interval construction. Since the rational all-halves witness gives $B = AY(x)$ with nonnegative coefficients, every length in B is a multiple of a length of A, and is therefore greater than one.

Finally, these are bounds for an effective polynomial-time reduction. The full word set has size $u^g < \Lambda u = \Lambda^{1+o(1)}$, so the matchings and all network stages can be found in polynomial time. The required primes are obtained by sieving. Only partial addresses below an active full address can have $W_e > 0$, and there are at most $|\mathcal{V}^+| 2^D = \Lambda^{1+o(1)}$ such addresses. We enumerate these addresses, compute their intervals, form $Y(0)$, and multiply by A.

The signed intermediate calculation also has polynomial bit complexity. The size of the final nonnegative target alone would not establish this: large positive and negative contributions might cancel. We therefore bound their absolute masses before any collection. For a signed cycle sum $Z = \sum_d z_d C_d$, write $\|Z\|_{\text{mass}} = \sum_d d|z_d|$. Since $z_e(0)$ is either $-L_e$ or R_e, the two terms at address e in (2) give

$$\|Y(0)\|_{\text{mass}} = \omega \sum_e d_e (L_e + R_e) \le 2\omega \, 2^D \sum_{v \in \mathcal{V}^+} d_v.$$

The sum of absolute masses of all uncollected terms in $AY(0)$ is $|A| \, \|Y(0)\|_{\text{mass}} \le \Lambda^{3+o(1)}$, by $\gcd(a, b) \operatorname{lcm}(a, b) = ab$. This bounds every partial coefficient sum as well as the final coefficients, even before cancellation. More precisely, at length ℓ the absolute value of any partial coefficient sum is at most $\ell^{-1} |A| \, \|Y(0)\|_{\text{mass}}$, by the triangle inequality. Thus all lengths and coefficients actually constructed have $O(\log \Lambda)$ bits; the parameter ceilings can be computed exactly by integer arithmetic. The resulting cycle list can be expanded into successor tables in $O((|A| + |B|) \log(|A| + |B|))$ bit operations. This completes the proof of Theorem 5.1.

6 Explicit decision and counting complexity

We now combine the construction with the factor classification. The two reverse implications play different roles: the interval decoder recovers a Boolean assignment from an arbitrary quotient, while the marker lemma recovers that quotient from an arbitrary factor pair.

Proof of Theorem 1.1 for explicit inputs. Take a 3-CNF formula with n explicitly listed variables and m clauses. Sections 4 and 5 construct A, B with a bijection between satisfying assignments and all quotients of $AX = B$. All lengths are odd and squarefree, and $A = C_3 + \sum_j C_{Q_j}$ has pairwise coprime lengths, each of multiplicity one. Form $T = B + C_2 A$. By Corollary 3.4, its nontrivial factor pairs are in bijection with those quotients. The construction has polynomially many states and can be expanded to a successor table in polynomial bit time. This proves NP-hardness and parsimonious #P-hardness.

For NP membership, a witness consists of two cycle lists of masses at least two whose product is T. By multiplicativity of mass, each factor has at most $|T|$ states. Thus its coefficient vector on lengths $1, \dots, |T|$ has polynomial size. The gcd/lcm rule verifies the product in polynomial time. For counting, the encoding must be canonical: neither vertex labellings nor the order of the two factors may create additional accepting branches. For #P membership, set $N = |T|$ and encode each of the N coefficients with exactly $\lceil \log_2(N+1) \rceil$ bits. Require every entry to be at most N and the first vector to be lexicographically at most the second. Reject if either mass is less than two or their product differs from T. Each unordered factor pair then has exactly one accepting encoding, including a repeated pair. No test of irreducibility is needed in this verifier.

We verify the additional restrictions. Irreducibility and distinctness of the two factors follow from Theorem 3.2. A factorization into three nonunits would, by grouping two factors, give a factor pair with a reducible member, so none exists. To check the target's lengths, we use the feasible rational table, which exists even when the formula has no satisfying assignment. The interval lemma gives $B = AY(x)$ with $Y(x)$ nonnegative. Every length occurring in this product is therefore a multiple of a length of A; this reasoning does not assume an integer quotient exists. In particular no length is one. The new factor 2 preserves squarefreeness. The even lengths have coefficient one, so the gcd of all multiplicities is one. Their length gcd is 2, while B is nonempty and has only odd lengths; hence the gcd of all target lengths is one. The even part recovers A by halving its lengths.

The exceptional source formulas handled outside the routing construction can use $A = C_3 + C_5$, with $B = A$ for a single satisfying assignment and $B = C_3$ for no satisfying assignments. Their marked targets satisfy the same restrictions. Finally, on nonunits irreducibility is the complement of reducibility, giving coNP-completeness. The unit is recognized directly and is defined to be neither reducible nor irreducible. □

Corollary 6.1. *Reducibility of arbitrary nonempty explicitly represented finite dynamical systems is NP-complete, and irreducibility is coNP-complete.*

Proof. Hardness holds already for permutation systems, and by Lemma 2.1 a factorization of such a system cannot acquire nonpermutation factors. For NP membership on arbitrary inputs, guess two successor tables and a bijection from their Cartesian product to the input states. Check that both factor sizes exceed one, their product is the input size, and the bijection commutes with the transition functions. The tables, bijection and verification all have polynomial size and time. Complementation gives the irreducibility assertion, again separating the unit. □

The hard construction uses a growing number of lengths and unbounded multiplicities. These hardness results do not by themselves determine the complexity when support patterns or multiplicities are restricted.

7 Binary encodings: short witnesses and long counts

Compressing a successor table into a cycle list preserves the hardness reduction. Membership in NP requires a separate argument: a factor might have too many distinct lengths to be listed within a polynomial bound. The following elementary support lemma shows that at least one short pair always exists.

Lemma 7.1 (A known nonnegative support bound)**.** *Let $b \in \mathbb{N}_0^k$ be a nonnegative integer combination of nonzero vectors in $\mathbb{N}_0^k$. Then b has such a representation using at most $\sum_{i=1}^k \log_2(b_i+1)$ distinct vectors.*

Proof. Choose a representation with minimum support size s. The 2^s subset sums of its distinct used vectors lie in the box $\prod_i\{0,\dots,b_i\}$, because every used vector has coefficient at least one. If two subset sums coincide, cancel their common vectors. The remaining two subsets are nonempty and disjoint, because all vectors are nonnegative and nonzero. Subtract the smallest coefficient on the first subset from all its coefficients, and add that amount to those on the second subset. The represented vector and nonnegativity are preserved, but at least one used vector disappears. This contradicts minimality. Thus $2^s \le \prod_i(b_i+1)$. □

This is the subset-sum argument for nonnegative integer cones in [14]; the exact box-counting argument also appears in Shmonin's dissertation [15, Theorem 5.1 and its proof]. The proof is included to specify the right-hand-side bound used below.

Theorem 7.2 (Short factor witnesses)**.** *If $T = \sum_{i=1}^k b_i C_{d_i}$ is reducible, it has a nontrivial pair whose two factors each use at most $\sum_i \log_2(b_i+1)$ distinct lengths. Consequently binary reducibility is in NP.*

Proof. Start from any nontrivial factorization $T = XY$ and fix Y. For each length a used in X, let $v(a)$ be the coefficient vector of C_aY on the support of T. Positivity ensures that no other output length occurs. Writing $X = \sum_a x_a C_a$, the factorization is now the vector equation

$$(b_1,\dots,b_k) = \sum_a x_a v(a).$$

The support lemma replaces this nonnegative integer combination by one using at most the stated number of distinct columns. Interpreting the new column coefficients as cycle multiplicities gives $X'Y = T$ with the stated support bound. Distinct lengths cannot give identical columns: the mass of the column for a is $a|Y|$, which determines a because Y is nonempty. Mass multiplicativity gives $|X'| = |T|/|Y| = |X| \ge 2$. Now fix this already sparse X' and apply the same argument to Y. This changes only the second factor, giving $X'Y' = T$ with both support bounds. The mass of Y' also remains at least two. Thus neither sparsification can replace a nontrivial factor by the unit. This is an existence argument for a short certificate; the verifier does not need an algorithm for finding a minimum-support representation.

By Lemma 2.1, every factor length is at most $\max_i d_i$ and every factor coefficient is at most $\max_i b_i$. Their binary descriptions therefore use $O(L^2)$ bits in total: there are $O(L)$ entries and each length and coefficient uses $O(L)$ bits. Their product can be verified by polynomially many gcd/lcm operations on polynomial-bit integers. □

The explicit hardness reduction also produces a binary list in polynomial time. Theorem 7.2 therefore completes the binary NP-completeness assertion in Theorem 1.1; the corresponding irreducibility assertion follows by complementation as before.

Proposition 7.3 (Counting all binary factor pairs is not in #P)**.** *For binary cycle lists of bit length L, the function* #PAIR *does not belong to #P.*

This is already a consequence of the quotient bounds of Bridoux et al. [1, Appendix B, Lemmas 13–14]. We include a direct subset construction as an alternative proof.

Proof. Let M be the product of the first k primes and put $n = \sigma(M) = \sum_{d\mid M} d$. The input $T = nC_M$ has bit length $O(k\log k)$: indeed $\log M = O(k\log k)$ and $\sigma(M) = \prod_{p\mid M}(p+1) \le 2^k M$. For every subset S of the divisors of M greater than one, define

$$X_S = \sum_{d\in S} C_d + \left(n - \sum_{d\in S} d\right) C_1.$$

This is a nonnegative cycle sum of mass n, and $C_M X_S = nC_M$ because $C_M C_d = dC_M$ whenever $d \mid M$. Different subsets give different X_S. Moreover $n > M$, so the unordered pairs $\{C_M, X_S\}$ are distinct and nontrivial. There are 2^{2^k-1} such pairs. Their count requires at least 2^k bits, exceeding every polynomial in L along this family. A #P function has at most $2^{\mathrm{poly}(L)}$ accepting branches and hence a polynomial-bit output. This proves the assertion. □

The support theorem chooses a short witness for existence; it does not encode all witnesses. This is why short decision certificates coexist with counts too large for #P. In the explicit representation, the mass itself is part of the input size, so the same output-length obstruction does not arise.

8 Equivalent factorization problems

The explicit classification has two useful interpretations. These are transfers of Theorem 1.1; the underlying factorization operations are already studied in the cited literature.

Corollary 8.1 (Permutation matrices)**.** *Given an explicit permutation matrix P, deciding whether $QPQ^{\mathrm{T}} = A \otimes B$ for permutation matrices Q, A, B with $\dim A, \dim B > 1$ is NP-complete. Counting unordered nontrivial pairs of factor conjugacy classes is #P-complete under parsimonious reductions.*

Proof. Permutation similarity relabels states, and the Kronecker product gives synchronous evolution. A successor table and its permutation matrix are polynomially interconvertible. Cycle types identify the factor conjugacy classes, preserving the unordered-pair count. Neither conjugating matrices Q nor different labellings of a factor are counted. □

This is the problem of Egner et al. [16]. The unknown common row and column permutation is essential: independent row and column permutations could turn any permutation matrix into the identity.

For a cycle sum $T = \sum_d a_d C_d$, let

$$F_T(n) = \sum_{d \mid n} d a_d.$$

Puri and Ward [21] study pointwise factorization of *realizable* sequences, those recording the finite numbers of fixed points of iterates of a map. Here $F_{UV} = F_U F_V$ pointwise. Let μ denote the Möbius function. To specify a finite input, give $N \geq 1$ and the full vector of nonnegative integers $f(1), \ldots, f(N)$, all in binary, satisfying

$$a_d = \frac{1}{d} \sum_{e \mid d} \mu(d/e) f(e) \in \mathbb{N}_0 \quad (1 \leq d \leq N), \qquad \sum_{d=1}^{N} d a_d = N.$$

Set $a_d = 0$ for $d > N$ and define the whole sequence by the fixed-point formula. Thus the input specifies an extension, not just an observed prefix. Invalid encodings are rejected.

Corollary 8.2 (Realizable sequences)**.** *Under this encoding, deciding nontrivial pointwise factorization into realizable sequences is NP-complete. Counting unordered nontrivial pairs of factor sequences is parsimoniously #P-complete. The unit is the constant sequence* 1*.*

Proof. Möbius inversion makes the encoding polynomially equivalent to an explicit permutation and identifies its isomorphism class. It remains to exclude additional factor sequences from infinite realizations. Suppose $F_T = GH$, where G, H are realizable. Since F_T is nonzero, choose m with $H(m) > 0$. Fixed-point counts cannot decrease when the iterate index is replaced by a multiple. For $r = \mathrm{lcm}(n, m)$,

$$G(n) \leq G(r) \leq G(r)H(r) = F_T(r) \leq N.$$

The same argument bounds $H(n)$ by N for every n. If g_d counts the length-d orbits in a realization of G, then $dg_d \le G(d) \le N$; thus $g_d = 0$ for $d > N$. Both sequences therefore have finite periodic realizations, say U, V. At a common multiple of all cycle lengths in T, U, V, the identity $F_T = F_U F_V$ gives $|U||V| = N$. Möbius inversion then gives $UV = T$ up to isomorphism. It also identifies each factor sequence with a unique cycle sum, preserving every unordered nontrivial pair. Theorem 1.1 now applies. Aperiodic points are invisible to the sequences and are not counted. □

9 Related work and scope

9.1 Relation to earlier work

Egner, Püschel and Beth [16] study precisely this factorization problem as decomposition of a permutation into a conjugated tensor product. Their permutation input is polynomially equivalent to our explicit successor-table representation. They pass to cycle types, use the same gcd/lcm multiplication, and give an algorithm enumerating the decompositions. Their Sections 4 and 6 describe potentially exponential searches, and Section 7 gives superpolynomial factorization counts, obstructing polynomial total-time enumeration. These are output-size complexity lower bounds. Indeed, their large-output family $p^r C_{p^r}$, for prime p and $r \ge 1$, is immediately reducible as $(p^r C_1)C_{p^r}$. The abstract's qualitative claim of fast indecomposability testing gives no polynomial worst-case guarantee. These results do not give a decision or counting complexity classification. To our knowledge, the classification proved here is new; it concerns an existing problem and follows from a parsimonious reduction controlling every factor pair. Püschel's dissertation also treats outer tensor decomposition of transitive monomial representations over a direct product of acting groups [19, Section 1.3, pp. 33–35]. Those hypotheses differ from arbitrary single-permutation factorization. For example, $C_2C_2 = 2C_2$ is a valid state-space factorization, but the cyclic group generated by C_2C_2 has order two. In contrast, the outer tensor product of the two transitive actions is the regular action of the order-four group $(\mathbb{Z}/2\mathbb{Z})^2$ on their Cartesian product. Moreover, $2C_2$ has two orbits and is therefore intransitive.

Glasby [17, 18] studies tensor products of polynomials; Section 9 of the earlier report gives existence and nonuniqueness results for inverse tensor factorization, and uniqueness theorems for ordinarily irreducible polynomials over finite fields. Its nonuniqueness example already includes the permutation identity $C_2(3C_1) = C_2(C_1 + C_2) = 3C_2$ [17, p. 17].

To specify the polynomial operation, write $f(x) = c_f \prod_{i=1}^{r}(x - \alpha_i)$ and $g(x) = c_g \prod_{j=1}^{s}(x - \beta_j)$, listing roots with multiplicity. For positive-degree polynomials with nonzero constant terms, their *tensor product* is

$$(f \otimes g)(x) = c_f^s c_g^r \prod_{i=1}^{r} \prod_{j=1}^{s} (x - \alpha_i \beta_j).$$

For monic polynomials the leading coefficients are both one, so the operation simply takes all pairwise products of roots. The characteristic polynomial $f_T(x) = \prod_d (x^d - 1)^{t_d}$ of a cycle sum $T = \sum_d t_d C_d$ satisfies $f_{UV} = f_U \otimes f_V$ over $\mathbb{Q}$: the permutation matrix of the product system is the Kronecker product of the two permutation matrices, whose eigenvalues are the pairwise products of their eigenvalues.

The tensor identity is $x - 1$. Two polynomials f, g are *tensor associates* if $g = h \otimes f$ for a tensor unit h. Over $\mathbb{Q}$, these units are $h = \varepsilon(x - \eta)$ with $\varepsilon, \eta \in \mathbb{Q}^*$ [17, Lemma 10]. Over $\mathbb{Q}$ (and hence over $\mathbb{Z}$), tensor association of permutation characteristic polynomials coincides with equality and therefore with permutation isomorphism. Indeed, multiplying a monic degree-r polynomial by this tensor unit scales every root by η and gives leading coefficient ε^r. If both characteristic polynomials are monic, this coefficient must be one. Since 1 is a root of every nonempty permutation system's characteristic polynomial, η must be a rational root of unity, hence $\eta = \pm 1$. The positive case changes nothing. In the negative case, let a be the original number of cycles and b the number of even cycles. The transformed multiplicities at $1, -1$ are

b, a. For a permutation characteristic polynomial the first is at least the second, so $b \geq a$, forcing all cycles to be even. Negating their roots leaves each root multiset unchanged. Finally, the multiplicity of a primitive mth root is $\sum_{k\geq 1} t_{km}$, which recovers the cycle counts by descending subtraction.

As already noted by Egner et al. [16], each polynomial factor in our problem must itself be the characteristic polynomial of a permutation. Every nonunit permutation characteristic polynomial has the proper ordinary factor $x - 1$ and is therefore not ordinarily irreducible. Thus the report's finite-field uniqueness theorems do not apply directly. Unrestricted polynomial tensor factorization can admit additional factors. For example, Glasby's cyclotomic formula [17, p. 15, equation (5), case $m = n$], with $p = 3$ and $m = n = 1$, gives

$$(x^2 + x + 1) \otimes (x^2 + x + 1) = (x - 1)(x^3 - 1) = f_{C_1+C_3}(x).$$

To check this directly, let ζ be a primitive cube root of unity. The pairwise products of ζ, ζ^2 are $\zeta^2, 1, 1, \zeta$, giving $(x - 1)^2(x^2 + x + 1) = (x - 1)(x^3 - 1)$. The polynomial $x^2 + x + 1$ is not a permutation characteristic polynomial, since it has no root 1. Yet $C_1 + C_3$ has no nontrivial permutation factorization: its factors would both have two states, and their products are only $4C_1$ or $2C_2$. Thus admissible polynomial factors must be restricted to permutation characteristic polynomials.

The algebra of sums of cycles belongs to the classical Burnside and necklace ring framework and also appears in Novotný's treatment of monounary algebras [20]. Its gcd/lcm multiplication and fixed-point coordinates are classical [13]; we use the positive integer part of this algebra.

Naquin and Gadouleau [2] study factorization in the semiring of finite dynamical systems. Bridoux, Crespelle, Phan and Richard [1] observe coNP membership of irreducibility and give division algorithms. Their general bound is polynomial when the divisor is fixed, but its exponent depends on that divisor; it is not a uniform polynomial bound when A is part of the input. Their Section 6 and Appendix C also give an $e^{O(\sqrt{N})}$ irreducibility algorithm for N-state cycle sums. This subexponential upper bound is consistent with the NP-completeness proved here. Crespelle, Phan and Richard [5] give further factorization and root algorithms, while Porreca and Rolland [3] develop polynomial algorithms for injective and pseudo-injective equations. The latter work explicitly lists the complexity of reducibility among its open problems [3, Section 7, p. 20]. Algorithms for those special equations do not decide unrestricted reducibility.

Earlier hardness results for polynomial equations over dynamical systems allow several unknowns [9, 10, 11]. They do not by themselves give the one-unknown division instances required here. The explicit division construction originates in the preceding manuscript [8]. We reproduce its interval-realization argument and give the routing construction in full, so that no hardness claim in this paper depends on an omitted proof from that manuscript.

Fresh-prime arguments for irreducibility are also known: Appendix B of Bridoux et al. [1] uses a fresh prime in a cycle of multiplicity one. Our contribution at this step is the classification of *all* factor pairs, the bijection with the quotients of a prescribed division problem, and the constant marker 2 with linear state overhead on the hard family. The binary witness and counting arguments in Section 7 build on a known nonnegative integer-cone support bound and earlier large quotient families; their precise sources are given there.

9.2 Open questions addressed

The workshop abstracts of Ascone and Porreca [12, p. 2] and Ascone, Porreca and Rolland [11, p. 2] explicitly ask for the complexity of testing irreducibility. Theorem 1.1 and Corollary 6.1 settle the explicit-successor-table formulation of this question, both for permutation systems and for arbitrary finite dynamical systems: reducibility is NP-complete and irreducibility is coNP-complete. Theorem 1.1 also classifies the binary decision problem for permutation systems. The one-unknown division theorem reproduced from [8] shows that one unknown already suffices for NP-hardness of

a linear equation when its cycle-sum coefficients are part of the explicit input. It also answers the explicit-input division question posed by Porreca and Rolland [4, pp. 2, 15].

The reaction-system abstract reports NP-hardness of irreducibility for systems given by reaction rules. Such a description on s entities represents 2^s states. It is a different compressed input from a binary cycle list, so our coNP upper bounds do not automatically apply. Ascone's CANA slides leave tighter general upper bounds open, while giving logarithmic-space algorithms for the bijective RS$(1,0)$ and RS$(0,1)$ subclasses [22, PDF pp. 48, 89, 91]. Our results do not resolve that succinct reaction-rule question. Likewise, nontrivial factorization $UV = T$ does not decide general quadratic or cubic equations with unknowns on both sides.

10 Conclusions

The parsimonious correspondence between division quotients and all nontrivial factor pairs proves NP-completeness of the classical factorization problem for explicit and binary cycle-list inputs, and parsimonious #P-completeness of unordered nontrivial factor-pair counting for explicit permutation inputs. Hardness persists even when the common irreducible factor can be read directly from the target.

The classification settles the explicit-input formulation of the irreducibility question identified in Section 9.2; the reaction-rule and general-equation questions there require separate results. Several further questions concern the boundary of this hardness. Which restricted patterns of cycle lengths admit polynomial algorithms? How should output-sensitive enumeration be measured in the binary representation, where the number of pairs can itself have a very long binary expansion? Finally, counting complete lists of irreducible factors is a different problem from counting two-factor decompositions.

Code availability. The verification scripts and reproduction instructions are available in the versioned code repository. They provide finite checks of the construction and examples; the proofs above do not depend on these computations.

Use of AI. OpenAI Codex and Anthropic Claude were used for mathematical development, literature comparison, manuscript preparation, and computational checking. The contributor directed the work but does not claim to understand the proofs or to have independently verified them. The AI-assisted review passes and finite computational checks do not constitute human peer review or formal verification.

License. This work is made available under CC0 1.0 Universal.

References

[1] F. Bridoux, C. Crespelle, T. H. D. Phan and A. Richard. *Dividing sum of cycles in the semiring of functional digraphs*. Natural Computing 25 (2026), article 11. doi:10.1007/s11047-026-10065-w. Version used: arXiv:2504.11943v2.

[2] E. Naquin and M. Gadouleau. *Factorisation in the semiring of finite dynamical systems*. Theoretical Computer Science 998 (2024), 114509. doi:10.1016/j.tcs.2024.114509.

[3] A. E. Porreca and M. Rolland. *Injective and pseudo-injective polynomial equations: From permutations to dynamical systems*. arXiv:2604.04065v2, 9 April 2026.

[4] A. E. Porreca and M. Rolland. *Solving "pseudo-injective" polynomial equations over finite dynamical systems*. arXiv:2504.06986v2, 2 September 2025.

[5] C. Crespelle, T. H. D. Phan and A. Richard. *Tensor Product of Functional Graphs: Algorithms for Factorization and Polynomial Root-Finding Restricted to the Cycle Part*. Manuscript (2025), HAL-05392325v1. doi:10.2139/ssrn.5316210.

[6] V. E. Beneš. *Optimal Rearrangeable Multistage Connecting Networks*. Bell System Technical Journal 43(4) (1964), 1641–1656. doi:10.1002/j.1538-7305.1964.tb04103.x.

[7] J. B. Rosser and L. Schoenfeld. *Approximate formulas for some functions of prime numbers*. Illinois Journal of Mathematics 6(1) (1962), 64–94. doi:10.1215/ijm/1255631807.

[8] N. Kuosa. *Division of Explicit Finite Dynamical Systems Is NP-Complete*. Preprint, Hexagon 2611.00886v1, 23 November 2026. `https://hexagonmath.org/2611.00886v1`.

[9] A. Dennunzio, V. Dorigatti, E. Formenti, L. Manzoni and A. E. Porreca. *Polynomial equations over finite, discrete-time dynamical systems*. ACRI 2018, LNCS 11115, 298–306. doi:10.1007/978-3-319-99813-8_27.

[10] A. E. Porreca. *Composing behaviours in the semiring of dynamical systems*. CANA presentation, 6 October 2020, slides 91–94. Author presentation.

[11] R. Ascone, A. E. Porreca and M. Rolland. *Polynomial equations over bijective dynamical systems and their semiring extensions*. Extended abstract, Workshop on Cellular Automata and Boolean Automata Networks at UCNC 2026, Trieste, 22 June 2026, 2 pp. Version v1, 28 May 2026. doi:10.5281/zenodo.20431158. Author's PDF.

[12] R. Ascone and A. E. Porreca. *Irreducible dynamics in reaction systems*. Extended abstract, 6th International Workshop on Reaction Systems at UCNC 2026, Trieste, 23 June 2026, 3 pp. Version 1, deposited 16 April 2026. doi:10.5281/zenodo.19609954. Author's PDF.

[13] A. W. M. Dress and C. Siebeneicher. *The Burnside ring of the infinite cyclic group and its relations to the necklace algebra, λ-rings, and the universal ring of Witt vectors*. Advances in Mathematics 78 (1989), 1–41. doi:10.1016/0001-8708(89)90027-3.

[14] F. Eisenbrand and G. Shmonin. *Carathéodory bounds for integer cones*. Operations Research Letters 34(5) (2006), 564–568. doi:10.1016/j.orl.2005.09.008.

[15] G. Shmonin. *Parameterised Integer Programming, Integer Cones, and Related Problems*. Dissertation, Universität Paderborn, 2007. Theorem 5.1 and its proof, p. 52. Author dissertation PDF.

[16] S. Egner, M. Püschel and T. Beth. *Decomposing a permutation into a conjugated tensor product*. Proceedings of ISSAC 1997, 101–108. doi:10.1145/258726.258761. Author-hosted abstract. Complete author version (PostScript).

[17] S. P. Glasby. *Tensor products of polynomials*. University of Sydney, Research Report 95-4, dated 1 June 1994, revised January 1995. Section 9, pp. 15–20. Institutional record and full text.

[18] S. P. Glasby. *On the tensor product of polynomials over a ring*. Journal of the Australian Mathematical Society 71(3) (2001), 307–324. doi:10.1017/S1446788700002950. Author full text.

[19] M. Püschel. *Constructive Representation Theory and Fast Discrete Signal Transforms*. Technical Report Drexel-MCS-1999-1, Drexel University, June 1999. English translation of the 1998 Karlsruhe dissertation, with a corrective English preface. Section 1.3, Theorem 1.29 and Algorithm 1.32, pp. 33–35. Author full text (PostScript).

[20] J. Novotný. *Cardinal arithmetic of a certain class of monounary algebras*. Časopis pro pěstování matematiky 111(4) (1986), 384–393. doi:10.21136/CPM.1986.118286.

[21] Y. Puri and T. Ward. *Arithmetic and growth of periodic orbits*. Journal of Integer Sequences 4 (2001), Article 01.2.1. Journal full text.

[22] R. Ascone. *Reaction Systems and Irreducible Dynamics*. CANA seminar, LIS, Marseille, 4 June 2026. Presentation, 93 PDF pages including overlays; PDF pp. 12, 48, 89, 91. Version retrieved 26 November 2026. CANA slides.