Unconstrained Detection of Productive Autocatalytic Cores Is NP-Complete

18 September 2026

Abstract

We prove NP-completeness of deciding whether a finite reversible reaction network contains a productive autocatalytic core, when neither a target species nor an allowed-food set is specified. The input is a binary encoding of the two literal nonnegative integer complex matrices, and each reaction identity carries a signed real flow. The proof first characterizes existence of a core by a nonsingular square restriction with literal two-sided participation. A weighted incidence construction then reduces prescribed two-path directed linkage to this unrestricted search: incidence saturation controls every possible selected support, and a boundary determinant distinguishes the desired pairing from the crossed pairing. An explicit constant-size switch construction reduces SAT to the required linkage instances. Polynomial certificates and a total polynomial-time binary transducer complete the classification. The terminal theorem, including actual machine-level membership and reduction bounds, is verified in Lean.

1 The decision problem

Selecting the internal species is a substantive part of detecting autocatalysis. A network can fail to amplify one chosen set while containing a different productive core. We study the existential question over all nonempty species and reaction supports.

Kosc et al. [1] define productive motifs and their inclusion-minimal cores. Their complexity theorem imposes a target species and an allowed-food set; the published article explicitly leaves the unconstrained variant open. We resolve that question for finite reversible sources with integer complexes. Andersen et al. [2] consider a different resource-production formulation of autocatalysis. Recent algorithms for enumerating autocatalytic subsystems in large networks [3] also address practical instances; worst-case decision complexity does not preclude effective enumeration on particular networks.

Definition 1. *A* source *consists of finite sets E of entities and R of reaction identities, and matrices $L, P \in \mathbb{N}^{E\times R}$. Its net matrix is $N = P - L$, interpreted over $\mathbb{R}$. A candidate is a pair of nonempty subsets (X, S) of E and R. It is* admissible *if, for every $r \in S$,*

$$\exists e \in X\ L_{er} > 0, \qquad \exists e \in X\ P_{er} > 0. \tag{1}$$

It is productive *if there is $v \in \mathbb{R}^R$, zero outside S, such that*

$$\sum_{r\in R} N_{er} v_r > 0 \qquad (e \in X). \tag{2}$$

A motif *is an admissible productive candidate. A* productive autocatalytic core *(PAC) is a motif minimal under componentwise inclusion: there is no motif (X', S') with $X' \subseteq X$, $S' \subseteq S$, and at least one inclusion strict.*

Flows may be negative: reversing a reaction changes the sign of its flow, rather than introducing a second independently selectable identity. Condition (1) uses the literal matrices even when an entity occurs on both sides. Cancellation in N must not change admissibility. Entities outside X have no production constraint. The input contains no distinguished target and no list of allowed external entities.

We use a canonical dense binary encoding of $|E|$, $|R|$, and the entries of L and P in reaction-major order. Natural-number fields use binary digits with delimiters; malformed strings are rejected. Precise conventions and the formal interfaces appear in Appendix B. Let UPAC be the language of well-formed sources containing a PAC.

Theorem 2. *The language* UPAC *is NP-complete under polynomial-time many-one reductions.*

The theorem concerns structural possibility in this literal-source model. It does not impose conserved elemental compositions, elementary reaction arity, concentration bounds, or a kinetic law. In particular, a structural YES certificate alone does not establish a dynamically realizable pathway. For network screening, the definitions specify exactly what a certificate checks: literal participation and strictly positive internal production.

2 Square witnesses and membership in NP

Finiteness immediately implies that every motif contains a PAC. The useful stronger fact is that a PAC has a square nonsingular net restriction.

Lemma 3. *A finite source contains a PAC if and only if it has nonempty subsets $X \subseteq E$, $S \subseteq R$ such that $|X| = |S|$, every reaction in S satisfies (1), and $N_{X,S}$ is nonsingular.*

Proof. Let (X, S) be a PAC and choose a productive flow v. Every v_r, $r \in S$, is nonzero, since otherwise deleting r preserves a motif. Orient each selected reaction in the direction of its flow, and write $w_r = |v_r| > 0$. The following observation gives $|X| \leq |S|$: each $e \in X$ is the sole internal reactant of some oriented selected reaction.

Indeed, suppose this fails for e. Remove e, and delete each reaction whose only internal product is e. Every remaining reaction still has an internal reactant, by the supposition, and an internal product, by the rule for deletion. For any remaining entity, a deleted reaction had nonpositive net contribution. Thus removing those contributions preserves its strictly positive production. The remaining entity set is nonempty: every original reaction had a reactant other than e. Strict production on this set also forces the remaining reaction set to be nonempty. We have a proper motif, a contradiction. Reactions witnessing the observation for distinct entities are distinct, proving the claimed cardinality inequality.

Next, the columns of $A = N_{X,S}$ are linearly independent. Otherwise choose $z \neq 0$ with $Az = 0$, and choose r with $z_r \neq 0$. The modified flow $v - (v_r/z_r)z$ has the same production and is zero at r. Deleting r again yields a proper motif; the selected reaction set cannot become empty because production remains strictly positive. Therefore $|S| \leq |X|$, and equality follows. The square matrix A is nonsingular.

Conversely, if the stated square restriction exists, solve $Av = \mathbf{1}$ and extend v by zero outside S. This is a motif. A minimal motif below it exists by finiteness, and is a PAC. □

This argument allows literal catalysts. Only in a source with disjoint reactant and product supports does admissibility become the assertion that every selected reaction column of N has both signs.

Lemma 4. UPAC $\in$ NP.

Proof. For a YES instance choose the square restriction in Lemma 3. Let its order be q and let $H \geq 1$ bound the absolute values of its integer entries. With $d = \det A \neq 0$, the integer vector

$$z = \operatorname{sgn}(d)\operatorname{adj}(A)\mathbf{1} \quad \text{satisfies} \quad Az = |d|\mathbf{1} > 0.$$

The determinant expansion bounds every cofactor by $(q-1)!H^{q-1}$, including the usual empty-minor convention when $q = 1$. Hence $|z_r| \leq q!H^{q-1}$, so each signed entry has $O(q\log(q+1) + q\log(H+1))$ bits. In a nonempty dense input of length n, $q \leq n$ and $\log(H+1) = O(n)$, giving a polynomial-size certificate.

The certificate supplies X, S and this integer flow, extended by zero. A verifier parses the source, checks both literal sides for every selected reaction, and performs exact integer matrix–vector multiplication and strict comparisons on the selected entities. All operand lengths and loop counts are polynomial in n. Accepting such a certificate establishes a motif and therefore PAC existence; the verifier need not test minimality. Malformed sources are rejected. The formal proof implements the verifier as a nondeterministic machine and proves its polynomial time bound. □

3 From directed linkage to an arbitrary selected support

Consider a directed graph with distinct terminals a_0, a_1, b_0, b_1. We ask for vertex-disjoint directed paths $a_0 \rightsquigarrow b_0$ and $a_1 \rightsquigarrow b_1$. We may delete arcs entering the sources, arcs leaving the sinks, and loops: none is needed by a pair of simple disjoint paths. The remaining vertices are called internal. Parallel arcs, if present, retain distinct identities.

Construct an integer matrix M with one column for each arc and one row for each internal vertex, together with two boundary rows s, t. Both sources are identified with row s, and both sinks with row t. For an arc $e : u \to w$, only its tail row and head row are nonzero. Define

$$\alpha(a_0) = -1, \quad \alpha(a_1) = 1, \quad \alpha(u) = -1 \quad (u \text{ internal}),$$

$$\beta(b_0) = -1, \quad \beta(b_1) = 1, \quad \beta(w) = 1 \quad (w \text{ internal}).$$

Let $f(a_1) = f(b_1) = 2$, with $f = 1$ at all other endpoints. Set

$$M_{\mathrm{tail}(e),e} = \alpha(u), \qquad M_{\mathrm{head}(e),e} = \beta(w)f(u)f(w). \tag{3}$$

The two rows are distinct, and every nonzero coefficient has magnitude at most four. Turn M into a source by taking arcs as entities, matrix rows as reactions, and

$$L_{er} = \max\{-M_{re}, 0\}, \qquad P_{er} = \max\{M_{re}, 0\}. \tag{4}$$

Thus the net matrix is M^{T} and the literal sides are disjoint.

Lemma 5. *The source* (4) *contains a PAC if and only if the graph has vertex-disjoint paths* $a_0 \rightsquigarrow b_0$ *and* $a_1 \rightsquigarrow b_1$.

Proof. First consider any square nonsingular witness supplied by Lemma 3, transposed to a $q \times q$ restriction of M. Every selected row is mixed, so it has at least two selected nonzero entries. Every selected column has at most two nonzero entries. Counting incidences gives

$$2q \;\leq\; \#\{(r,e) : M_{re} \neq 0 \text{ in the restriction}\} \;\leq\; 2q. \tag{5}$$

Equality forces both endpoints of every selected arc to be selected, and exactly two selected arcs, of opposite signs, at every selected row. At an internal vertex these are one entering and one leaving arc. At s they are one arc from each of a_0, a_1; at t they are one arc into each of b_0, b_1.

If neither boundary row is selected, the finite selected graph is a union of directed cycles on internal vertices. Each such cycle has ordinary $-1, +1$ incidence coefficients and makes the restriction singular. If s is selected, following either outgoing path through the unique outgoing arc at every internal vertex cannot enter a cycle: entry would violate uniqueness of the incoming arc. It must reach a sink. The reverse argument applies if t is selected. Thus nonsingularity forces both boundary rows, two vertex-disjoint source-to-sink paths, and possibly separate internal cycles. Those cycles again give a singular block and are excluded.

There are two possible pairings of the paths. Eliminate the internal rows along each path, using their nonzero incidence pivots. Apart from nonzero row or column scaling, the remaining boundary blocks are

$$B_{\text{desired}} = \begin{pmatrix} -1 & 1 \\ -1 & 4 \end{pmatrix}, \qquad B_{\text{crossed}} = \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix}. \tag{6}$$

For example, the factor two leaving a_1 propagates along its path, and the factor two entering b_1 multiplies it once more. This also covers a direct source-to-sink arc by (3). The determinants are -3 and 0. Consequently a nonsingular selection must use the desired pairing.

Conversely, take a desired pair of vertex-disjoint simple paths. Select their arcs, their internal vertices, and s, t. Each path has one more arc than internal vertices, so the selection is square. Every selected row is mixed. Internal elimination leaves B_{desired}, so the selected matrix is nonsingular. Lemma 3 supplies a PAC. □

The quantifier over all restrictions is essential here. Equation (5) prevents a witness from using only one endpoint of an arc, branching at an internal vertex, or retaining a boundary-free productive component. No promise that a candidate includes the full intended gadget is required.

4 An explicit reduction from SAT to linkage

We give the graph construction used by the formal reduction, so the hardness argument does not rely on an assumed linkage-completeness theorem. Paths in this section are simple. A CNF formula has c clauses, o literal occurrences, and V variable slots, including any unused indices. Index occurrences by $0 \le i < o$, retaining repeated literals as separate occurrences, and put $K = o + 1$.

4.1 The switch

Use the fixed directed graph on vertices $0, \ldots, 19$ listed in Appendix A. Its input ports are $0, 1, 2, 3$ and its output ports are $4, 5, 6, 7$. It has the following three properties.

Lemma 6. *For the fixed switch:*

1. *If disjoint paths connect an input a to 4 and 1 to an output b, then $a = 0$ and $b = 5$.*
2. *Given disjoint control paths $0 \rightsquigarrow 4$ and $1 \rightsquigarrow 5$, every input-to-output path disjoint from both controls is either $2 \rightsquigarrow 6$ or $3 \rightsquigarrow 7$. Both kinds cannot exist for the same controls, even without requiring the two residual paths to be disjoint from one another.*

3. *There are disjoint controls with an available* $2 \rightsquigarrow 6$ *path, and there are disjoint controls with an available* $3 \rightsquigarrow 7$ *path.*

Proof. Appendix A specifies the complete finite check of the first two assertions, together with the completeness argument for its path enumeration. The third assertion has explicit witnesses in the same appendix. All three assertions are proved in Lean; the finite checks are kernel-reduced decisions, not external numerical tests. □

4.2 Stack and formula wiring

Take K switches, indexed by $i = 0, \ldots, K-1$, writing (i, p) for port p at level i. Join

$$(i, 5) \longrightarrow (i+1, 1), \qquad (i+1, 4) \longrightarrow (i, 0) \qquad (0 \le i < K-1). \tag{7}$$

The last switch is a dummy level with no literal occurrence. Introduce variable gates $u_0, \ldots, u_V$, rail vertices $h_{j,b,k}$ for $0 \le j < V$, $b \in \{0,1\}$, $0 \le k \le K$, and clause gates $d_0, \ldots, d_c$. Add the following arcs, and no others beyond switch and stack arcs.

First, $(K-1, 5) \to u_0$, and for each variable j add $u_j \to h_{j,b,0}$ for both choices of b. Between consecutive positions on a rail, use $h_{j,b,i} \to h_{j,b,i+1}$ unless occurrence i is a literal of variable j that is false when $j = b$. In that case replace the step by

$$h_{j,b,i} \longrightarrow (i, 3), \qquad (i, 7) \longrightarrow h_{j,b,i+1}. \tag{8}$$

The dummy position always uses the direct rail step. Add $h_{j,b,K} \to u_{j+1}$, and $u_V \to d_0$. Finally, for each occurrence i in clause ℓ, add

$$d_\ell \longrightarrow (i, 2), \qquad (i, 6) \longrightarrow d_{\ell+1}. \tag{9}$$

The prescribed linkage is

$$(0, 1) \rightsquigarrow d_c, \qquad (K-1, 0) \rightsquigarrow (0, 4). \tag{10}$$

The first path is denoted P and the second Q. The dummy level makes $K > 0$ for every formula; the four terminals remain distinct in boundary cases.

Lemma 7. *The graph above has the linkage* (10) *if and only if the CNF formula is satisfiable.*

Proof. Suppose first that an assignment satisfies the formula. At a true occurrence choose switch controls leaving channel $2 \rightsquigarrow 6$ available; at a false occurrence choose controls leaving $3 \rightsquigarrow 7$ available. Either choice suffices at the dummy level. The path Q follows the upward controls $0 \rightsquigarrow 4$, and the initial part of P follows the downward controls $1 \rightsquigarrow 5$. After the stack, P chooses the assignment's rail for each variable. Exactly the false occurrences require a detour through $3 \rightsquigarrow 7$. Then, in each clause, choose a true occurrence and traverse its available $2 \rightsquigarrow 6$ channel. Distinct occurrences are separate switches. A true occurrence is not used on a false-literal rail detour, so the clause and rail portions do not conflict. These choices give the two disjoint paths.

For the converse, let P, Q be any disjoint paths with the required endpoints. We first force the controls at every level, without assuming that the paths respect their intended roles. At level zero, P contains port 1 and Q contains port 4 by their endpoints. Extract the segment of P from that 1 to its next exit from the switch, and the segment of Q from its last entrance into the switch to that 4. Entrances use input ports and exits use output ports. The two segments are disjoint, so Lemma 6(1) forces P to exit at 5 and Q to enter at 0. The unique continuation from $(i, 5)$ is $(i+1, 1)$, and the unique predecessor of $(i, 0)$ is $(i+1, 4)$, except at the stack

boundaries. Induction therefore forces the downward control in P and the upward control in Q at every level. Simplicity gives the corresponding order of their visits.

After P leaves $(K-1,5)$, its remaining suffix is disjoint from its own control prefix and from Q. Any later input-to-output passage through a switch must consequently be a residual channel. By Lemma 6(2), it preserves the rail or clause role: it cannot enter at 3 and leave at 6, or enter at 2 and leave at 7. The external wiring now forces this suffix to pass the variable gates in order, choose one rail for each variable, and subsequently pass the clause gates in order. Its rail choices define an assignment. Its passage between d_ℓ and $d_{\ell+1}$ chooses an occurrence in clause ℓ and uses that occurrence's $2 \rightsquigarrow 6$ channel. If the chosen literal were false under the assignment, the earlier rail passage would have used the same switch's $3 \rightsquigarrow 7$ channel. This contradicts residual exclusivity for the fixed controls. Every chosen clause literal is therefore true.

An empty clause has no connecting occurrence and prevents the clause passage. With no clauses, the clause portion is empty and the construction accepts. Thus the argument includes these boundary cases. □

5 Binary execution and completion of the proof

For the SAT encoding used by the formalization, occurrence count, clause count, and the largest variable index plus one are bounded by a polynomial in the input length n (indeed by linear bounds in the chosen encoding). The stack and rail construction therefore has polynomially many vertices and arcs. Equation (3) has bounded coefficients, and the dense source (4) has polynomial encoding length. Its entries can be generated by bounded loops over the explicit vertex and arc families.

To obtain a language reduction on *all* binary strings, parse the input CNF. On a valid input, write the source constructed above. On a malformed input, write the source for the fixed unsatisfiable formula consisting of one empty clause. Lemmas 5 and 7 show that this fixed output is a NO instance and that the resulting total function F satisfies

$$x \in \mathsf{SAT} \iff F(x) \in \mathsf{UPAC} \qquad \text{for every binary string } x. \tag{11}$$

For completeness, the formal execution proof goes beyond bounding output size. A parser prepares a validity flag and payload; a selector returns either the decoded formula's encoding or the fixed unsatisfiable encoding. The valid-input writer starts on that ordinary input with blank work tapes, writes the headers, and traverses both literal matrix sides in canonical order. Its loop invariants and a polynomial bound on its actual transitions are proved. A composition theorem combines these runs, including copying and tape-boundary overhead, into a total transducer computing F in FP. No well-formed-input premise remains in its final interface.

Proof of Theorem 2. Lemma 4 gives membership in NP. The total polynomial-time map F, with equivalence (11), reduces SAT to UPAC. SAT hardness follows from the Cook–Levin theorem; the formal development proves and imports that theorem rather than postulating SAT hardness. Polynomial-time reduction closure yields NP-hardness, and hence NP-completeness. □

A The fixed switch and its complete check

This appendix specifies the only finite gadget check used in the proof. The vertex set is $\{0, \dots, 19\}$ and the complete arc set is

$$(0,9),(0,15),(1,17),(1,18),(2,8),(3,12),$$
$$(8,5),(8,9),(9,10),(9,11),(10,8),(10,11),$$
$$(11,6),(11,19),(12,13),(13,5),(13,14),(14,12),$$
$$(14,15),(15,7),(15,16),(16,4),(16,10),(17,4),$$
$$(17,18),(18,16),(18,19),(19,14),(19,17).$$

Let $\mathcal{E}(k, H, a)$ enumerate simple path lists starting at a, with at most k vertices and avoiding the previously visited set H. Its recursive definition is

$$\mathcal{E}(k,H,a) = \begin{cases} \varnothing, & k = 0 \text{ or } a \in H, \\ \{[a]\} \ \cup \bigcup_{a \to b} \{a :: p : p \in \mathcal{E}(k-1, H \cup \{a\}, b)\}, & \text{otherwise.} \end{cases}$$

Every simple path has at most 20 vertices. Induction on its length shows it belongs to $\mathcal{E}(20, \varnothing, a)$. Filtering these lists by last vertex therefore gives all paths between any two specified ports. The first two assertions of Lemma 6 are finite universal tests over these lists, using disjointness of vertex sets. In particular, for each disjoint pair of controls, the exclusivity test checks that it is not the case that a residual $2 \rightsquigarrow 6$ path exists *and* a residual $3 \rightsquigarrow 7$ path exists. This stronger statement is what the formula converse consumes.

In Lean, `enumerate_complete` proves coverage of the enumeration, and `path_mem_routes` transfers arbitrary simple paths into it. The Boolean identities `controlCheck_true`, `residualCheck_true`, and `exclusiveCheck_true` are proved by `decide`. Their logical consequences are `control_ports`, `residual_channel`, and `residual_exclusive`. Thus the finite reduction is part of the kernel proof, with no unproved search-coverage assumption.

The constructive switch modes are displayed below. Within each column the three paths are pairwise vertex-disjoint; every consecutive pair is an arc in the displayed graph.

Role	Channel $2 \rightsquigarrow 6$ available	Channel $3 \rightsquigarrow 7$ available
$0 \rightsquigarrow 4$	$0, 15, 16, 4$	$0, 9, 11, 19, 17, 4$
$1 \rightsquigarrow 5$	$1, 17, 18, 19, 14, 12, 13, 5$	$1, 18, 16, 10, 8, 5$
Residual	$2, 8, 9, 10, 11, 6$	$3, 12, 13, 14, 15, 7$

B Formal theorem and reproducibility

The mathematical source files are under `proofs/UnconstrainedPACDetection/`. The terminal declaration is

`UnconstrainedPACDetection.complexity_root`
of type
`Complexity.NPComplete`
`UnconstrainedPACDetection.BinaryPACVerifier.language`

It has no hypotheses. Its reported axioms are exactly `propext`, `Classical.choice`, and `Quot.sound`. There are no admitted lemmas or assumed mathematical hardness axioms in this interface. Strict verification treats warnings as errors.

The source encoding writes the two dimensions followed by all entries of the left matrix and then all entries of the right matrix, with reaction index outermost and entity index innermost. A natural-number field uses its canonical least-significant-bit-first binary digits, each prefixed by 1, followed by a terminating 0. Canonical decoding rejects invalid representations and incorrect field counts. The formal language accepts precisely decoded sources with a PAC. In particular, a net matrix is not silently substituted for the pair of literal matrices.

Mathematical obligation	Principal formal component
Core and square equivalence	`CoreWitness`, `SquareMinor`
Linkage in both directions	`LinkagePACForward`, `LinkagePACConverse`
SAT linkage equivalence	`FormulaLinkage.formula_paths_iff_sat`
All-string semantic reduction	`FormulaPACEncoding.compile_correct`
Actual total FP transducer	`FormulaTotalWriter.compile_mem_FP`
Actual NP membership	`VerifierNPPolynomial.in_NP`
Final classification	`complexity_root` in `ComplexityRoot.lean`

The frozen toolchain is `leanprover/lean4:v4.30.0`, with dependencies fixed by `mathlib4_project/lake-manifest.json`. The strict receipt is `complexity_root.verify.json`, stored in the workspace's `campaign_2026_09_05/` directory. It records the source and transitive dependency hashes, the authenticated declaration types, and the axiom report. Publication verification rechecks those hashes; it is distinct from claiming a fresh clean build or a performance benchmark.

From the repository root, the verification bridge can be invoked as follows (the long declaration is one command argument):

```
.venv/Scripts/python.exe scripts/verify_proof.py
  proofs/UnconstrainedPACDetection/ComplexityRoot.lean
  -timeout 600
  -declaration UnconstrainedPACDetection.complexity_root
```

The bridge runs Lean in the pinned project and checks the dependency closure. The publication directory includes the editable source, build script, and evidence audit. The complete proof is provided in the repository accompanying this manuscript; no external archival DOI or submission is asserted.

References

[1] T. Kosc, D. Kuperberg, E. Rajon, and S. Charlat. Thermodynamic consistency of autocatalytic cycles. *Proceedings of the National Academy of Sciences*, 122(18):e2421274122, 2025. doi:10.1073/pnas.2421274122.

[2] J. L. Andersen, C. Flamm, D. Merkle, and P. F. Stadler. Maximizing output and recognizing autocatalysis in chemical reaction networks is NP-complete. *Journal of Systems Chemistry*, 3:1, 2012. doi:10.1186/1759-2208-3-1.

[3] R. Golnik, T. Gatter, P. F. Stadler, and N. Vassena. Enumeration of autocatalytic subsystems in large chemical reaction networks. *Journal of Chemical Theory and Computation*, 22(10):4888–4907, 2026. doi:10.1021/acs.jctc.5c01979.