Predecessor Existence for Explicit Finite Game-of-Life Boards with a Permanently Dead Boundary is NP-Complete

R. Rumana

July 2026

Abstract

We prove that the following decision problem is NP-complete: given a rectangular Conway Game-of-Life board presented as an explicit matrix of bits, under the convention that every cell outside the rectangle is permanently dead, decide whether the board has a one-step predecessor.

Membership in NP is immediate as verification is a single forward time step, which can be calculated for any board by applying the Game of Life rules individually to each cell. The hardness argument when considering the finite dead-boundary must address two additional features as compared to the infinite grid. First, the instance is the dense matrix, so the reduction must produce a board of area polynomial in the formula and must write that board out explicitly. Second, the permanently dead exterior is a constraint on the predecessor rather than a neutral convention, so no general truncation of an infinite-grid instance is available. Nonempty 3-CNF is compiled into a bounded-fanout NOT/OR circuit; each variable is realized by a closed two-valued source loop rather than by an input on the boundary. The circuit is routed deterministically over the published 450×450 composite tiles of Salo and Törmä; and the tile grid is surrounded by a complete blank macrotile frame. The output is an explicit dense board of area $O(S^2)$ in the formula size S.

Each composite target realizes exactly its declared Boolean relation as shown in Salo and Törmä's Theorem 1. Because the reduction depends on that single local statement, the accompanying software also derives it inside the repository: exact reconstruction of all eleven composites from their published 90×90 and 180×90 constituents, finite relational composition, forward simulation of every satisfying witness, and 122 LRAT proofs checked by an independently compiled `cake_lpr`.

1 Introduction

Conway's Game of Life [1] is the cellular automaton on the square grid in which a cell's next state depends only on its own state and on the number of live cells among its eight Moore neighbors. Writing $s \in \{0, 1\}$ for the current state and $m \in \{0, \ldots, 8\}$ for the live-neighbor count, the local rule is

$$F(s, m) = 1 \quad \Longleftrightarrow \quad m = 3 \text{ or } (s = 1 \text{ and } m = 2).$$

The forward map is computable in linear time. The most basic question about the backward map is whether a given target lies in the image of the global update at all, that is, whether it has a predecessor.

This paper settles the complexity of that question in the setting of a finite rectangular board stored as a matrix of bits, with all cells outside the rectangle permanently dead. The resulting language is shown to be NP-complete for one Life step.

1.1 Two distinct reductions

Two reductions are associated with "reverse Life and SAT," and they establish different things.

(i) A reduction from reverse Life to SAT is an algorithm for predecessor search. It yields an NP verifier and is the appropriate tool for finding predecessors.

(ii) A reduction from SAT to reverse Life establishes NP-hardness.

A SAT encoding of reverse Life therefore does not by itself establish hardness. Bain's encoding [2] is direction (i), and Proposition 2.5 below records the version used by the solver in this repository. The construction of Sections 3–8 is direction (ii). The two are developed independently and share no implementation.

1.2 The finite dead-boundary case

Salo and Törmä established structural and computability results for Life preimages, and published the associated gadget assets and verification code [3, 4]. Their setting is the infinite grid. The present work uses theirs both as literature context and as the source of the composite tiles.

The finite dead-boundary statement does not follow from the infinite-grid one without further argument. Two obstacles must be addressed.

Encoding. On $\mathbb{Z}^2$ a construction may be specified by any finite gadget assembly, however sparse. In the finite-board language the instance is the mn-bit matrix itself, and this has consequences in both directions. Under a succinct encoding, such as run-length encoding or a generating program, a board can be exponentially larger than its description, the natural predecessor witness is then no longer polynomial in the input length, and membership in NP ceases to be immediate. Under the dense encoding membership is immediate, but the reduction must write its board out in polynomial time; a construction of merely finite area does not suffice, since the area must be polynomial in the formula and the output must actually be emitted.

Boundary semantics. A permanently dead exterior is a constraint on the predecessor: cells outside the rectangle must be dead in the predecessor as well. A gadget that reaches the board edge is therefore subject to that constraint and may be either over-constrained, so that a satisfiable formula maps to a board with no predecessor, or under-constrained, if the absent outside neighbors admit unintended predecessors. Nor is the natural shortcut available: one cannot in general take a finite-support infinite-grid target, enclose it in a rectangle with a constant-width collar, and conclude that the finite-board problem has the same answer, because such an argument must control both the predecessor outside the rectangle and the births contributed by the remainder of the plane. Any padding argument must therefore be made against the specific structure of the construction.

The reduction below addresses the first obstacle with a board of area $O(S^2)$ and a streaming writer, and the second by excluding computational structure from a neighborhood of the boundary. Specifically: each variable is realized by a closed loop requiring no external input (Lemma 5.2); the routed tile circuit is verified to carry no wire on its outer boundary (Lemma 6.1); and the board is framed by a complete blank macrotile, 450 cells thick, on all four sides, so that every active tile together with its radius-one predecessor halo lies strictly interior (Lemma 7.1).

2 Problem statement and encodings

2.1 The update map on a finite board

For positive integers m, n, a board is a matrix $P \in \{0,1\}^{m\times n}$. Extend P to all of $\mathbb{Z}^2$ by assigning zero to every cell outside $[0, n-1] \times [0, m-1]$, apply the Life rule to that extension, and retain only the original rectangle. Explicitly, $F_{m,n}(P) \in \{0,1\}^{m\times n}$ is given by

$$F_{m,n}(P)_{y,x} = \begin{cases} 1, & P_{y,x} = 0 \text{ and exactly three neighbors are live,} \\ 1, & P_{y,x} = 1 \text{ and exactly two or three neighbors are live,} \\ 0, & \text{otherwise,} \end{cases}$$

where out-of-board neighbors count as dead. After each update the exterior is again dead, so $F_{m,n}$ maps the board to itself and may be iterated.

Definition 2.1 (Explicit dead-boundary predecessor language)**.**

$$\mathsf{DB\text{-}PREIMAGE} = \{\langle B\rangle : B \in \{0,1\}^{m\times n} \textit{ for some } m, n, \quad \exists P \in \{0,1\}^{m\times n},\ F_{m,n}(P) = B\}\,.$$

The encoding $\langle B\rangle$ records m, n, and all mn board bits.

The choice of encoding is substantive rather than notational. A run-length encoding of a sparse board can be exponentially shorter than the corresponding dense matrix, so the two conventions define distinct languages. The companion tool can emit RLE, which is the appropriate format for inspecting a board in a Life viewer, but RLE is not the encoding in which the results below are stated.

2.2 Membership in NP

Proposition 2.2 (One step)**.** $\mathsf{DB\text{-}PREIMAGE} \in \mathsf{NP}$.

Proof. Guess $P \in \{0,1\}^{m\times n}$. Each target cell is determined by its own state and its at most eight in-board neighbors, with zero substituted for every out-of-board neighbor, so computing $F_{m,n}(P)$ and comparing it with B takes $O(mn)$ time. The witness has mn bits, which is linear in the input length. □

Proposition 2.3 (Fixed number of steps)**.** *For every fixed $k \geq 1$, the language*

$$\mathsf{DB\text{-}PREIMAGE}_k = \left\{\langle B\rangle : \exists P \in \{0,1\}^{m\times n},\ F^k_{m,n}(P) = B\right\}$$

lies in NP.

Proof. Guess P and simulate k updates, resetting the exterior to dead after each one. Verification takes $O(kmn)$ time, which is $O(mn)$ for fixed k. □

Remark 2.4. *The same unrolling remains polynomial when k is part of the input in unary. That extension is useful to the solver but is not used below: the hardness construction applies to one step only, and no claim is made for fixed $k > 1$.*

2.3 The solver direction

Direction (i) of Section 1.1 is recorded here for completeness and is not used subsequently. For a board of width n and height m and k simulated steps, introduce variables

$$x_{y,x,t} \in \{0,1\}, \qquad 0 \le y < m, \quad 0 \le x < n, \quad 0 \le t \le k,$$

fix the final layer to the target by unit clauses, and for each cell and each $t < k$ constrain $x_{y,x,t+1}$ to equal the Life rule applied to the cell's time-t neighborhood. A Life neighborhood has nine Boolean inputs including its center, so one may enumerate the at most 2^9 local predecessor patterns and forbid each one inconsistent with the desired next-state literal.

Proposition 2.5 (Direct SAT encoding)**.** *For fixed k, and also for k given in unary, the k-step finite dead-boundary predecessor problem reduces to CNF-SAT in time and output size $O(kmn)$, with a constant factor determined by the local truth-table encoding.*

Proof. There are $mn(k+1)$ state variables, the target contributes mn unit clauses, and each of the kmn cell-time transitions contributes a constant-size clause family because the neighborhood has fixed size. Out-of-board predecessor literals are replaced by zero. The resulting CNF is satisfiable exactly when its time-zero layer is a predecessor that evolves to the target in k steps. □

This direction accounts for the use of both CaDiCaL and ParKissat as backends in the solver. It is logically independent of Sections 3–8.

3 Structure of the hardness reduction

Hardness is proved by reduction from 3-SAT restricted to nonempty formulas containing no empty clause, which is NP-complete. The implementation accepts arbitrary DIMACS CNF and is total, returning fixed boards on the two degenerate inputs (Section 4.1).

For a formula φ the reduction proceeds in four stages,

$$\varphi \longmapsto C_\varphi \longmapsto \mathcal{T}_\varphi \longmapsto B_\varphi,$$

where C_φ is a bounded-fanout Boolean circuit over NOT, binary OR, binary splitter, and a constant-one sink; $\mathcal{T}_\varphi$ is a closed edge-matched grid of abstract tiles; and B_φ is the explicit finite Life board obtained by substituting a 450×450 pattern for each tile and adding a blank frame.

The corresponding proof obligations, and the results discharging them, are:

(1) the circuit is satisfiable exactly when φ is (Lemma 4.1);

(2) the closed variable motifs realize free, independent Boolean choices (Lemma 5.2);

(3) routing preserves every net, closes every port, and leaves no wire on the tile-grid boundary (Lemma 6.1);

(4) composite substitution identifies abstract tile assignments with Life predecessors (Lemma 5.1);

(5) the blank frame isolates the construction from the finite board boundary (Lemma 7.1); and

(6) the board area and the construction time are polynomial in $|\varphi|$ (Section 7.2).

Obligations (2), (3), and (5) are the points at which the dead boundary is addressed, and are where this construction differs most from an infinite-grid argument.

4 From CNF to a bounded-fanout circuit

Let V denote the number of variables of φ, C the number of clauses, and L the total number of literal occurrences. Each variable contributes one input node. A positive literal uses that node directly and a negative literal passes through NOT. Each nonempty clause, of any length, is collapsed by a balanced binary OR tree, so unit clauses and long clauses require no separate treatment. The clause values are conjoined by De Morgan's law: each clause output is negated, the negations are combined by a balanced OR tree, and the result is negated. If the circuit output is itself an input node, two NOT gates are inserted to materialize an identity, so that the physical output is always an ordinary gate port and the downstream geometry has no special case. Two features of the compilation are used later and are stated separately.

Fanout is explicit. The tile system provides no wire with two consumers; a signal reaching two destinations must be split by a gadget whose relation says so. Every physical output port is therefore required to have at most one consumer, and a logical value with $r > 1$ consumers is passed through a balanced tree of $r - 1$ binary splitters producing r distinct physical outputs. A splitter's relation is equality among its three ports, so this transformation is logically inert.

Satisfaction is enforced by a sink. The circuit output is connected to the west port of a constant-one tile, whose relation admits only the value 1. Consistency with the tile relations is therefore equivalent to satisfaction of φ, with no separate evaluation step: under a falsifying assignment the constant-one tile has no admissible row.

Lemma 4.1 (Bounded-fanout compilation)**.** *For every nonempty CNF formula with no empty clause, the compiler constructs in polynomial time a circuit over NOT, binary OR, binary splitter, and constant-one such that*

(i) every physical port has fanin and fanout at most one;

(ii) the constant-one tile is connected to the circuit output;

(iii) the circuit admits a consistent Boolean assignment exactly when φ is satisfiable; and

(iv) with $S = V + C + L + 1$, the number M of source and logical macros and the number E of physical nets satisfy

$$M \leq 8S, \qquad E \leq 12S.$$

Proof. Balanced OR trees and De Morgan's law preserve the value of the formula, and the two-NOT identity preserves a direct variable output. A splitter constrains its input and its two outputs to be equal, so replacing a fanout-r signal by a splitter tree preserves every signal value while making each physical port single-use; conversely, any consistent assignment to a splitter tree assigns a single value to all of its ports. The constant-one tile admits exactly the row $W = 1$, so a consistent global assignment exists if and only if some input assignment drives the output to 1, that is, if and only if φ is satisfiable. For (iv), the numbers of literal gates, OR-tree internal nodes, splitter leaves, and splitters are each linear in $V + C + L$, and summing them with generous constants gives the displayed bounds, which are deliberately loose. The implementation records the actual values of M and E and checks both inequalities. □

4.1 Degenerate DIMACS inputs

The compiler is total on syntactically valid DIMACS:

- the empty conjunction maps to the 1×1 dead board, a fixed yes-instance;
- any formula containing an empty clause maps to the 1×1 live board, a fixed no-instance;
- all remaining inputs, including unit clauses, repeated variables, duplicate clauses, and clauses of arbitrary length, use the ordinary compiler.

Both fixed cases are correct because a 1×1 board has no in-board neighbors and therefore always evolves to dead: the dead 1×1 board has a predecessor and the live one does not.

5 The composite tile system

5.1 Tiles, relations, and the edge-mask condition

The abstract alphabet $\mathcal{A}$ consists of a blank tile together with eleven wired tiles: horizontal and vertical wire, the four turns, NOT, OR, splitter, crossing, and constant-one. Each side of a tile either carries one Boolean port or is unwired. In the port orders shown, the relations are

$$\begin{array}{lll} H(E,W): & E = W, & V(N,S): N = S, \\ \mathrm{turn}(a,b): & a = b, & \mathrm{NOT}(E,W): E = \neg W, \\ \mathrm{OR}(E,N,S): & E = N \vee S, & \mathrm{split}(E,N,S): E = N = S, \\ \mathrm{cross}(E,N,W,S): & E = W,\ N = S, & \mathrm{one}(W): W = 1. \end{array}$$

Blank has no ports. Turns are named for the two sides they connect, so NE joins north and east, and similarly for the others. The compatibility condition between neighbors is that two adjacent sides are either both unwired or both wired, and in the latter case carry the same Boolean value. This edge condition is checked mechanically on all four sides of every placed tile of a completed grid, including the sides facing the exterior of the grid.

5.2 The 450×450 composites

Each of the eleven wired tiles is realized by a concrete 450×450 Life target. These patterns are vendored from Salo and Törmä's `gol-backward-comp` repository [4]. The published RLEs are cropped to their bounding boxes, so a parser reads each file's `#CXRLE Pos` declaration and re-seats the pattern on the common logical frame

$$[-180, 269] \times [-180, 269],$$

which has exactly 450×450 cells. Blank is the all-dead pattern on the same frame. The parser supports run counts and the standard `b`, `o`, `$`, and `!` tokens, and every vendored file is pinned by SHA-256 (Appendix C).

The property of these patterns on which the reduction relies is local and two-sided.

Lemma 5.1 (Composite substitution)**.** *Let X be a finite edge-matched configuration over $\mathcal{A}$ carrying no wire on its outer boundary, and substitute the normalized 450×450 target for each tile. Then the substituted target has a Life predecessor if and only if the abstract tile relations of X admit a consistent Boolean assignment. Moreover:*

(i) (completeness) *every allowed local row has a local witness whose predecessor border is zero away from its wire ports; and*

(ii) (soundness) *every local predecessor, with no assumption on its border, induces one of the declared rows on every active tile.*

The two halves require different strengths, and the distinction is used repeatedly below. Completeness requires witnesses that can be glued, hence the zero border away from ports. Soundness must assume nothing about the border, since in the completed construction the surrounding cells are determined by an arbitrary predecessor. The same asymmetry appears in the machine-checked query catalog of Section 9.3.

Lemma 5.1 is Salo and Törmä's Theorem 1 [3, Thm. 1], and for the purposes of Theorem 8.2 that citation constitutes the proof. Section 9 describes the independent derivation from the published 90×90 and 180×90 constituents.

5.3 Closed variable sources

The standard way to supply inputs to such a circuit is to admit wires from the edge of the pattern, which is harmless on the infinite grid and is what an earlier version of this construction did. On a finite board with a permanently dead exterior the boundary condition constrains such inputs, so inputs are eliminated: each variable is realized by a self-contained motif with one output port and no other wired side.

Place a splitter at the right center of the following 3×3 motif, whose center position is blank:

SE	H	SW
V	□	split
NE	H	NW

The splitter's north and south ports are joined to each other by the identity loop formed by the two turns above it, the two turns below it, the two horizontal wires, and the vertical wire in the left column. Its east port is the variable output, which the ordinary east adapter converts into a north-facing riser.

Lemma 5.2 (Two-valued source)**.** *The source motif has no wired side other than its east output, and its consistent assignments are exactly the all-zero and the all-one assignment.*

Proof. Every turn and every straight wire of the motif imposes equality between its two ports, and the splitter forces its north, east, and south values to agree. Composing these equalities around the loop shows that all signals of the motif, including the east output, take one common value. Both 0 and 1 satisfy every relation involved and there is no third Boolean value, so the consistent assignments are exactly those two. That all sides other than the east output are unwired follows by inspection of the nine edge masks. □

One motif is instantiated per variable, ahead of every logical macro, and each occupies its own slot in the layout of Section 6. Since the motifs are closed and slot-disjoint, variable choices are neither exposed to the board boundary nor coupled to one another, and the construction supplies exactly V free bits.

6 Deterministic band-and-trunk routing

The router performs no search and has no formula-dependent failure modes; the position of every tile is a closed-form function of two integers.

Each of the M source or logical macros is assigned a 12-column slot. In zero-based coordinates, macro i is placed at

$$x_i = 12i + 6, \qquad y_\star = E + 9,$$

so all macros share one horizontal band. Every used macro port is converted by a short adapter into a distinct north-facing riser:

macro port	riser column
W	$x_i - 1$
N	x_i
E	$x_i + 1$
S	$x_i + 2$

The west and east adapters are single turns. The south adapter is an NE–H–NW dogleg placed below the macro, which accounts for the two-column offset of the south riser. A variable source exposes only its east riser.

Net j, in deterministic net-ID order, is assigned trunk row $j + 2$, and is routed by placing an SE turn at the left endpoint's riser column, an SW turn at the right endpoint's riser column, horizontal wire between them, and vertical wire in each riser column from immediately below the trunk to the adapter it attaches to. When a horizontal trunk is placed on a column already carrying a vertical wire, the two straight tiles are replaced by a single crossing; every other overlay is rejected and aborts compilation.

The unframed tile grid has exact dimensions

$$W = 12M + 12, \qquad H = E + 13,$$

and the following fixed row structure:

rows	contents
0–1	unwired
2–$(E + 1)$	one horizontal trunk per net
$(E + 2)$–$(E + 7)$	unwired
$(E + 8)$–$(E + 10)$	the macro band and its adapters
$(E + 11)$–$(E + 12)$	unwired

Macro i occupies only columns $x_i - 2$ through $x_i + 2$, so columns 0 through 3 and columns $12M - 3$ through $12M + 11$ carry no tiles.

Lemma 6.1 (Routing and closure)**.** *The router runs in polynomial time, uses only tiles of $\mathcal{A}$, preserves every logical net, and produces a closed edge-matched tile circuit. No wire reaches the boundary of the $W \times H$ tile grid.*

Proof. Distinct slots give distinct riser columns and distinct net IDs give distinct trunk rows, so no two vertical routes share a column and no two horizontal routes share a row. The only possible collision is therefore perpendicular, between the horizontal trunk of one net and the vertical riser of an earlier one, and such an intersection is represented exactly by the crossing tile, whose edge

mask is the union of the horizontal and vertical masks and whose relation is the conjunction of the two independent equalities. Trunk rows lie strictly above the macro band and every adapter lies strictly below the last trunk, so a vertical riser meets no macro tile other than its own; the vertical clearance $j + 3 \leq (\text{attachment row}) - 1$ is checked for every net. The margins tabulated above leave unwired rows at the top and bottom and unwired columns on both sides, so no tile with a wired side is adjacent to the grid boundary. A final validation pass compares every adjacent pair of edge masks and every outer edge against "unwired," visiting at most $WH = O(ME)$ positions. □

Two consequences of this geometry are used below. First, the layout is formula-independent, so the polynomial area bound of Section 7.2 follows directly from M and E. Second, the hypothesis of Lemma 5.1 that no wire lies on the outer boundary is a consequence of the coordinate formulas rather than a property established only by the final check.

7 Substitution onto a finite dead-boundary board

7.1 Stamping, framing, and explicit output

The target is represented internally as a sorted set of live coordinates together with its width and height. A nonblank tile at grid position (u, v) is stamped at cell offset $(450u, 450v)$; blank tiles are implicit and contribute no live cells. One complete blank macrotile is then added on each of the four sides, giving exact board dimensions

$$n = 450(W + 2), \qquad m = 450(H + 2).$$

The sparse representation is what makes compilation, hashing, and RLE export practical, but the theorem concerns the dense encoding, so the writer streams the $m \times n$ matrix row by row to disk without materializing it in memory. For the regression fixture of Appendix A the board already has roughly 8.6×10^8 cells, which is why streaming is required for the explicit instance to be writable at all.

Lemma 7.1 (Dead-boundary isolation)**.** *The framed finite board has a predecessor under permanently dead exterior conditions if and only if its closed abstract tile circuit admits a consistent Boolean assignment.*

Proof. Completeness. Let a consistent tile assignment be given. For every active tile choose the zero-bordered local witness supplied by Lemma 5.1(i). At each matched interface the two adjacent witnesses agree because the corresponding port values agree, and at every other predecessor-border cell both witnesses assign zero. Assign zero throughout the blank frame and outside the board. The result is a finite predecessor with dead exterior, and it evolves to the stamped target: within each active tile by the choice of witness, and throughout the frame because a zero neighborhood yields a dead cell.

Soundness. Let P be any predecessor of the finite board. Since the frame is 450 cells thick, every active composite tile together with its radius-one predecessor halo lies strictly inside the board. Restrict P to each such halo. The restriction is an unrestricted local preimage of that tile's target, no assumption on its border being required and the behavior of P elsewhere in the frame being irrelevant, so Lemma 5.1(ii) applies and the restriction induces one of the tile's declared rows. Two adjacent active tiles read their shared interface from the same predecessor cells, so the port values they induce agree. Since the tile circuit is closed, no port requires a value from outside the circuit, and the induced rows therefore assemble into a single globally consistent tile assignment. □

Remark 7.2. *Lemma 7.1 is not a claim that an arbitrary finite-support infinite-grid target can be transplanted onto a finite board by the addition of a constant-width collar. Such a statement would require control of both the predecessor outside the chosen rectangle and the births contributed by the remainder of the plane, and is not proved here. The proof above instead uses four specific structural facts: composite witnesses with zero borders, local soundness assuming nothing about the border, closure of the circuit, and a frame as thick as one macrotile.*

7.2 Polynomial size

With $S = V + C + L + 1$, Lemma 4.1 gives

$$\begin{aligned} W &= 12M + 12 \le 96S + 12, \\ H &= E + 13 \le 12S + 13, \\ mn &= 450^2(W+2)(H+2) = O(S^2). \end{aligned}$$

Logical compilation is linear up to balanced-tree bookkeeping. Routing and sparse stamping visit $O(WH)$ macrotiles and at most $450^2(W+2)(H+2)$ cell positions. Streaming the dense matrix takes $\Theta(mn)$ time, which is linear in the length of the instance being produced. The constant 450^2 is large but does not affect the polynomial bound.

8 Correctness and the main theorem

Lemma 8.1 (Reduction equivalence)**.** *For every DIMACS formula φ, the compiled finite board B_φ has a one-step predecessor with dead exterior if and only if φ is satisfiable.*

Proof. The two fixed 1×1 boards of Section 4.1 have the stated truth values. For every other formula the lemmas compose: Lemma 4.1 identifies satisfying assignments of φ with consistent assignments of the closed logical circuit; Lemma 5.2 shows that each variable contributes exactly one free Boolean choice and no exposed port; Lemma 6.1 shows that the physical tile circuit realizes the same nets and leaves no port open; and Lemma 7.1 identifies consistent tile assignments with predecessors of the framed finite board. Satisfaction is enforced within the tile circuit by the constant-one sink, which admits only output value 1. □

Theorem 8.2 (One-step finite-board NP-completeness)**.** DB-PREIMAGE *is NP-complete under polynomial-time many-one reductions.*

Proof. Membership is Proposition 2.2. For hardness, restrict Lemma 8.1 to nonempty 3-CNF formulas without empty clauses, which form an NP-complete language. By Section 7.2 the board B_φ is computable, and writable in its dense encoding, in time polynomial in $|\varphi|$, and by Lemma 8.1 it belongs to DB-PREIMAGE exactly when φ is satisfiable. Hence DB-PREIMAGE is NP-hard, and therefore NP-complete. □

Remark 8.3 (Scope)**.** *Theorem 8.2 concerns one Life step on explicitly encoded rectangular boards with permanently dead exterior. Proposition 2.3 gives membership in NP for every fixed k. The hardness construction is not extended past $k = 1$, and a compact RLE is not used as the complexity-theoretic input encoding. Both restrictions are essential to the statement.*

9 Independent certification of the local claims

9.1 Two justifications for Lemma 5.1

Every step of the argument above is elementary except Lemma 5.1, on which the construction depends. Two justifications are available:

(i) Salo and Törmä's composite construction [3, Thm. 1]; or

(ii) the repository's derivation from the smaller published constituents, described below.

Justification (i) suffices for Theorem 8.2 and requires no computation. Justification (ii) reduces the reliance of the result on a single imported local theorem and makes the associated finite computations auditable. Its own trusted base is delimited in Section 11.

9.2 Exact decomposition of the composites

Each of the eleven large targets is accompanied by an explicit placement manifest recording its published 90×90 center tile, the required transformed 180×90 charged connectors, their translations, and their rotations or reflections. The checker reconstructs the target from the manifest and requires cell-for-cell equality with the vendored large RLE on the common 450×450 frame, rather than equality up to translation or up to live-cell count.

A separate structural pass verifies the semantics of the assembly:

- every transformed constituent lies within the common frame;
- every internal port is paired with its partner at an identical seam and wire anchor;
- only the declared large-tile ports remain external;
- every external wire is reached by a charged connector;
- every phase transformation returns to neutral alignment;
- the finite product of the transformed constituent relations, projected to the outer frame, equals the declared large-tile relation exactly; and
- every allowed outer row admits a compatible tuple of constituent rows, so that zero-bordered stitching is possible.

All eleven manifests pass both the exact reconstruction and the semantic composition.

9.3 Primitive SAT and UNSAT queries

The primitive catalog contains 28 published specifications: the basic gadgets, the 90×90 tiles, and all nine 180×90 connector phase pairs. For each specification the generator emits deterministic DIMACS for four query classes, with the following outcomes in the completed run:

query class	SAT	UNSAT
allowed row, unrestricted	62	0
allowed row, zero-bordered	62	0
forbidden row, unrestricted	0	84
unnamed charging state	0	38
total	124	122

The separation between unrestricted and zero-bordered queries corresponds to the two halves of Lemma 5.1. Soundness asserts that a forbidden row is impossible for *every* local predecessor, so the corresponding queries must not assume a dead predecessor border; assuming one would exclude precisely the predecessors at issue. Completeness asserts the existence of a witness that can be stitched to its neighbors, so those queries additionally require a zero border away from the ports. Charging queries are unrestricted for the same reason as soundness: an unnamed or non-operating boundary state must be excluded outright rather than under assumptions favorable to completeness.

Every SAT model obtained for an allowed row is converted back into predecessor cells and checked by direct forward Life simulation, so no solver "SAT" answer is used without independent confirmation. Every forbidden-row and charging query receives an LRAT proof from CaDiCaL `rel-2.1.3`, and each proof must be accepted by an independently compiled CakeML checker, `cake_lpr` [5, 6].

9.4 Division of labor among tools

ParKissat remains in use for parallel model search and in the practical reverse solver. Its current Rust wrapper does not expose a checker-compatible LRAT stream, so it cannot serve as the final authority for an UNSAT claim on which a theorem depends. The certification path therefore uses CaDiCaL to produce the SAT outcomes and the LRAT proofs, direct Rust forward simulation to validate each allowed-row witness, and `cake_lpr` to check every UNSAT proof independently. This division concerns proof production and is not a limitation on ParKissat's parallel search. The sweep's own parallelism is at the outer level: independent queries are solved concurrently, and `PROOF_JOBS` sets the worker count.

9.5 The artifact manifest

The completed run comprises 246 queries, 124 SAT and 122 UNSAT. Its manifest records the exact CaDiCaL, `cake_lpr`, and asset commits; the complete set of specification and query identities; SHA-256 hashes of every CNF, LRAT proof, outcome, and checker log; the hashes of all published assets; and the hash of the composite decomposition manifest. Validation recomputes the required catalog and rejects a bundle containing missing, duplicated, renamed, or unexpected queries.

This validation step is not redundant. An earlier filename normalization erased unary minus signs, which collided distinct connector phase pairs and left the catalog incomplete while appearing complete. Preserving negative phase names in the query identities exposed the omitted cases and increased the catalog from 206 to 246 queries.

10 Recomputable reduction certificates

Each compiled formula also yields a `CompositeReductionCertificate`. Unlike the bookkeeping certificate of the earlier pipeline, it contains sufficient structure to be recomputed rather than merely inspected. It records:

- the formula and its hash;
- the pinned asset commit, the individual asset hashes, and the manifest hash;
- the logical macro and net counts;
- the complete tile circuit and its hash;

- the tile-grid dimensions, per-kind tile counts, and nonblank count;
- the closed-edge, source-loop, output-enforcer, and blank-frame checks;
- the exact board dimensions and a hash of all live coordinates; and
- the actual value and the coarse upper bound for every quantity appearing in Section 7.2.

The `verify()` entry point reloads the vendored assets, validates and hashes the embedded tile circuit, recompiles the embedded formula, restamps the board, and compares the certificate against that deterministic recomputation. A mutated formula, asset, route, tile, dimension, or board is rejected. The two fixed 1×1 edge cases yield certificates through the same interface, so no case is left unverified.

11 Trust model and limitations

The result is a machine-supported finite-board NP-completeness proof, in the following sense.

- The compilation, routing, size, isolation, and equivalence arguments are ordinary mathematical proofs. The executable checks mirror them and do not replace them.
- The published Salo–Törmä theorem is an independent justification for the local composite semantics, and on a fresh checkout it is the operative justification.
- Direct forward simulation removes the SAT solver from the trusted base for allowed-row witnesses.
- LRAT checking removes CaDiCaL from the trusted base for UNSAT conclusions.
- The DIMACS generator, the transcription of the intended port relations into code, the decomposition checker, and the `cake_lpr` executable itself remain within the local verification boundary unless separately formalized.

The final item warrants emphasis. Proof logging certifies the generated CNFs; it does not establish that those CNFs encode the intended Life statement. That link is established by source review, by direct simulation on satisfying models, by exact pattern reconstruction, and by the independent published theorem, and not by the fact that a solver reported UNSAT. Stating this boundary explicitly is more informative than treating a successful solver call as a proof certificate.

The generated proof bundle is large and is deliberately not committed. Committed instead are the deterministic generator, the pinned tool versions, the hashes, and the validation commands. A fresh checkout may cite the published composite theorem immediately; reproducing the internal derivation requires regenerating and re-checking the bundle.

12 Discussion

The construction is conceptually smaller than its predecessor although its boards are substantially larger. Once the tiles are pre-composed, the global compiler requires only two facts about Life: eleven Boolean relations and a rule for matching edges. The phase management is confined to the one-time certification of the composite assets. The resulting separation of scales is the principal structural feature of this formulation:

(i) the theorem-facing compiler manipulates closed 450×450 Boolean tiles and never inspects an individual cell; and

(ii) the certification layer establishes, once, that each large tile has the semantics the compiler assumes.

The blank frame is conservative in the same sense. A one-cell halo would suffice for radius-one locality if every relevant predecessor cell were already controlled, and a tighter argument along those lines is presumably available. A full 450-cell frame instead makes Lemma 7.1 immediate at the same scale as the substitution it protects, and keeps every active halo far from the finite boundary. The constant is large and does not affect the complexity bound.

The engineering separation between sparse construction and streamed output follows from the choice of encoding. Retaining only live coordinates is what makes compilation, hashing, and visualization practical; streaming the dense matrix is what makes the theorem instance writable without holding hundreds of millions of dead cells in memory. Both are consequences of taking the explicit encoding literally, and neither affects the mathematical content.

13 Conclusion

For explicitly encoded rectangular boards with a permanently dead exterior,

$$\mathsf{DB\text{-}PREIMAGE} = \{\langle B \rangle : \exists P,\ F_{m,n}(P) = B\}$$

is NP-complete for one Life step. The reduction compiles nonempty 3-SAT into a closed bounded-fanout circuit, routes it deterministically over pinned 450×450 composite tiles, and isolates it with a complete blank frame. For every fixed k, membership in NP follows by direct simulation; hardness for $k > 1$ is outside the present claim.

A Reproduction commands

From the repository root, the default test suite:

```
cargo test --workspace
```

Exact and semantic reconstruction of all eleven composites:

```
cargo run --release -p rev_gol_proof --example check_composites
```

The completed run reports:

```
composite_decomposition_valid=true exact=11/11 semantic=11/11
```

Every allowed published row, in both unrestricted and zero-bordered modes, with direct simulation of each returned predecessor:

```
cargo test --release -p rev_gol_proof \
  all_published_allowed_rows -- --ignored --nocapture
```

Bootstrap the pinned tools, generate every query, solve them, check all LRAT proofs, finalize the hashes, and validate the manifest:

```
crates/rev_gol_proof/scripts/proof_artifacts.sh all
```

Four query workers are used by default; `PROOF_JOBS` overrides that count, for example

```
PROOF_JOBS=8 crates/rev_gol_proof/scripts/proof_artifacts.sh all
```

The completed manifest reports:

```
proof_artifacts_complete=true queries=246 sat=124 unsat=122
outcomes_verified=246
```

Compile the small regression formula with the default composite pipeline:

```
cargo run --release -p rev_gol_proof --example compile_dimacs -- \
  --input crates/rev_gol_proof/tests/fixtures/small_example.cnf \
  --check-exhaustive \
  --certificate-json target/small-example-certificate.json \
  --output-rle target/small-example.rle
```

Adding

```
  --output-grid target/small-example.txt
```

streams the explicit dense theorem instance; the RLE output is supplementary. For this fixture the compiler reports 12 logical macros, 12 nets, a 156×25 unframed tile grid, and a 71100×12150 finite board.

The legacy path remains callable explicitly:

```
cargo run -p rev_gol_proof --example compile_dimacs -- \
  --pipeline legacy --input <FILE> [legacy options]
```

B Proof artifact and tool pins

The proof script pins:

- CaDiCaL `rel-2.1.3`, commit `f13d74439a5b5c963ac5b02d05ce93a8098018b8`;
- `cake_lpr`, commit `a36874a8b750b43fe4b385b8ddbf5b033e46a3fa`; and
- the composite assets, commit `2ff235848fb9d62ad4ffbbe1ada37d544d63252b`.

The run reported here used `rustc 1.96.0-nightly (362211dc2 2026-03-24)`, `cargo 1.96.0-nightly (e84cb639e 2026-03-21)`, and `git 2.54.0`. These are reproduction metadata rather than mathematical premises.

C Pinned composite asset hashes

file	SHA-256
cross_large.rle	932b7e6aabe9fe76798cb64219eb8d90dc2e71eb80f551b0dd35a02f00169729
hor-wire_large.rle	beb4799d302c2ee63b2e875b38872c3c26f1d073b381fdefa9c164c07b77990d
ne-turn_large.rle	c81424b2939b0c4972dda018f5ae7a21241b6e1d9f6b77faba80bf100bf8584a
not_large.rle	151345346a3af02fa5117120ad24dc541c53d54b952c1e13839ac3706a393fd3
nw-turn_large.rle	3484ef0aa6c63b3a1b5624d111df7562e9c7ae381ef24ded2146db0bbb4fdd23
one_large.rle	11e26c85920c412f821789323ebfed30ac0d15ac06ca22f2c5f4748f5d41da8b
or_large.rle	d73e77c5f552813f2832652d9aa234bba434844ef31481cdcacba462e16675c6
se-turn_large.rle	462d947ea7f53c8dde833fb3c9e2daa0529fb99ed60122b69306f39201fe142e
split_large.rle	065d9c8eaa1d4b27573341cf8c7327b9e6f3b6c69a84a3593d4062ad2a888d8b
sw-turn_large.rle	22e1c62282a7dc67daa4343403086e90fb64d965540e27cc64d10d53a973b59a
ver-wire_large.rle	8e525a3aa3b1628c90b3965443fc8f6ad374e9833eb1304151c2da75cd7690f7

D Artifact map

path	role
`crates/rev_gol_proof/src/composite.rs`	logical compiler, tile router, RLE normalization, board stamping, certificates
`crates/rev_gol_proof/src/composite_certificate.rs`	large-instance manifests, topology, relation composition, exact checks
`crates/rev_gol_proof/src/verifier.rs`	unrestricted soundness, zero-bordered completeness, witness simulation
`crates/rev_gol_proof/src/proof_artifacts.rs`	deterministic query generation and proof manifest validation
`crates/rev_gol_proof/scripts/proof_artifacts.sh`	pinned solver/checker bootstrap and bounded parallel proof sweep
`crates/rev_gol_proof/examples/compile_composite.rs`	command-line default and legacy command-line interface
`crates/rev_gol_proof/vendor/published/large_150x150`	pinned large 150×150 source RLEs
`docs/composite_reduction.tex`	compact formal theorem chain

References

[1] Martin Gardner. Mathematical Games: The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223(4):120–123, 1970.

[2] Stuart Bain. Time-Reversal in Conway's Life as SAT. In *AI 2007: Advances in Artificial Intelligence*, Lecture Notes in Computer Science 4830, pages 614–618. Springer, 2007. doi:10.1007/978-3-540-76928-6_63.

[3] Ville Salo and Ilkka Törmä. Structure and computability of preimages in the Game of Life. *Theoretical Computer Science*, 1042:115237, 2025. doi:10.1016/j.tcs.2025.115237. Preprint: arXiv:2308.10198.

[4] Ilkka Törmä. `gol-backward-comp`: auxiliary files for the Game of Life preimage construction. GitHub repository, pinned at commit `2ff235848fb9d62ad4ffbbe1ada37d544d63252b`. https://github.com/ilkka-torma/gol-backward-comp.

[5] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In *Computer Aided Verification (CAV 2024)*, Lecture Notes in Computer Science 14681, pages 133–152. Springer, 2024. The proof artifact uses release `rel-2.1.3`. GitHub repository, pinned at commit `f13d74439a5b5c963ac5b02d05ce93a8098018b8`. https://github.com/arminbiere/cadical.

[6] The CakeML Project. `cake_lpr`: LPR proof checking with CakeML. GitHub repository, pinned at commit `a36874a8b750b43fe4b385b8ddbf5b033e46a3fa`. https://github.com/tanyongkiam/cake_lpr.

[7] R. Rumana. `game_of_life_reverse`: reverse-Life solver and proof artifact. GitHub repository, July 2026. https://github.com/rrumana/game_of_life_reverse.