Machine-Checked Exact-Integer Certificates for Normal and Misère Grid NoGo, Coins-are-Lava, and Subcubic Bipartite Graph NoGo PSPACE-Completeness

Lluis Eriksson*

November 25, 2026

Abstract

Kyle Burke, Alfie Davies, and Craig Tennenhouse (`arXiv:2611.33127`, November 24, 2026) resolved the open problem of Burke and Hearn (2019), proving that NoGo is PSPACE-complete on positions reachable from an empty $n \times n$ square grid under both normal play and misère play (Theorem 1.1) via direct square-grid Bounded Two-Player Constraint Logic (B2CL) reductions, and proving that Graph NoGo remains PSPACE-complete under normal play on connected bipartite subcubic ($\Delta \leq 3$) graphs (Theorem 1.2) via a new Coins-are-Lava edge-removal game (Lemma 4.1). We present a machine-checked, zero-floating-point formalization and exact combinatorial game-tree auditor for the finite budget, truth-table, reachability, rope-majority, and parity-win core of these results. First, an 18-theorem zero-dependency Lean 4 kernel (`NoGoPSPACE.lean`, Lean 4.35.0; 0 `sorry`, 0 warnings, 0 `Classical.choice`, 9 zero-axiom theorems) formalizes the B2CL move budget $x = a + 2b + c + d + 3e$, White reserve hole capacity, empty-board alternating parity, game-tree depth bounds, normal-vs.-misère terminal and wire-signal duality, exact AND/OR/SPLIT/CHOICE truth tables, Lemma 4.1 clause/formula majorities, odd-parity cut shifts, rope-width separation $2tw + 1 < (2t + 3)w$, and subcubic bipartiteness. Second, `NOGO-PSPACE-CERT-1.0` audits four exact modes—empty-board reachability replay, normal/misère grid minimax, Coins-are-Lava minimax, and connected bipartite subcubic Graph NoGo—replaying 16 canonical certificates in 0.060–6.495 ms and rejecting 60/60 live adversarial forgeries.

1 Introduction

NoGo is the two-player combinatorial placement game derived from Go by forbidding both captures (including self-captures) and passes [3, 6]. Following the PSPACE-completeness of planar Graph NoGo [3] and Dots & Boxes [1], Burke, Davies, and Tennenhouse [2] settled two major open questions:

1. *Square-grid normal & misère play (Theorem 1.1):* Realizing B2CL gadgets (Goal, Variable, Or, And/Split, Choice) directly on $n \times n$ grids with insulated White wires, frozen liberties, and a checkerboard White reserve yields PSPACE-completeness on positions reachable from the empty grid under both normal and misère play.
2. *Subcubic bipartite graphs (Theorem 1.2 & Lemma 4.1):* Introducing Coins-are-Lava (a Strings-and-Coins [5] variant forbidding isolating any coin), proving its PSPACE-completeness from Schaefer's positive-CNF game via odd rope widths $N = 2t + 3$, and reducing to connected bipartite graphs of maximum degree $\Delta \leq 3$.

Contributions and Scope. We formally verify and audit the finite combinatorial, algebraic, truth-table, parity, and exact minimax core of [2]. We explicitly scope our Lean 4 kernel away from full Turing-machine PSPACE reductions, planar grid layout routing, and asymptotic polynomial-size circuit constructions:

1. **Zero-dependency Lean 4 kernel (§3):** 18 constructive theorems in `NoGoPSPACE.lean` (Lean 4.35.0 [4]), 0 `sorry`, 0 warnings, 0 `Classical.choice`, and **9 zero-axiom**

*Independent researcher; manuscript prepared with transparent AI assistance via Google Antigravity. Source code, Lean 4 proofs, and exact certificates: https://github.com/lluiseriksson/nogo-pspace-certificates.

theorems.

2. **Pure-stdlib exact game verifier (§4):** `NOGO-PSPACE-CERT-1.0`, auditing grid reachability, normal/misère minimax, Coins-are-Lava, and subcubic bipartite Graph NoGo with zero floats.
3. **Empirical atlas & red-team audit (§5):** 16 canonical certificates (0.060–6.495 ms), exact normal-vs.-misère winner flips on grids ($2 \times 2, 2 \times 4$) and subcubic bipartite graphs ($K_{3,3}, Q_3, L_8$), rope-separation scaling, and 60/60 rejected live adversarial forgeries.

2 Budget, Rope-Width, and Parity Invariants

In Section 2 of [2], a B2CL circuit with a Variable, b Or, c And, d Split, and e Choice gadgets prescribes

$$x = a + 2b + c + d + 3e \tag{1}$$

Black moves; a checkerboard White reserve with $x + 1 - a$ non-adjacent holes yields $x - a$ waiting moves alongside White's a Variable plays, so Black wins going first iff Black claims the extra Goal move ($x + 1 > x$). In Section 4 (Lemma 4.1), a positive-CNF formula with c clauses and n variables is encoded into $2c-1$ rows (each k-literal clause padded by $k-1$ copies of 1 to $2k-1$ entries) joined by ropes of widths $1, N, N^2, N^3, N^4$ with odd $N = 2t + 3$. At every width w, at most $2tw+1$ lower-level cuts occur before completion, strictly below Nw:

$$2tw + 1 < (2t + 3)w = Nw, \tag{2}$$

and from an odd initial string total $M = 2k + 1$, a False chosen entry leaves an odd count $R_{\mathrm{F}} = 2(c + n + 1) + 1$ (even cuts, White wins), whereas a True entry allows one extra cut to ground g, leaving $R_{\mathrm{T}} = 2(c + n + 1)$ (odd cuts, Black wins).

3 Zero-Dependency Lean 4 Kernel

We formalize 18 constructive theorems in `lean/NoGoPSPACE.lean` (Lean 4.35.0, zero external dependencies), audited via `lean/AuditAxioms.lean` with 0 warnings, 0 `sorry`, 0 `Classical.choice`, and 0 `Lean.ofReduceBool` (Table 1).

Proposition 1 (Constructive NoGo PSPACE Kernel)**.**

Table 1: Complete 18-theorem Lean 4.35.0 kernel in `NoGoPSPACE.lean` (0 `sorry`, 0 warnings, 0 `Classical.choice`, 9 zero-axiom theorems).

#	Lean 4 Theorem Identifier	Axiom Footprint
1	`normal_play_move_budget_exact`	`[propext]`
2	`white_reserve_hole_capacity`	`[propext, Quot.sound]`
3	`empty_board_reachability_parity`	`[propext]`
4	`nogo_game_tree_depth_bound`	`[propext, Quot.sound]`
5	`normal_vs_misere_terminal_duality`	none
6	`normal_vs_misere_wire_inversion`	none
7	`b2cl_and_gate_truth_table`	none
8	`b2cl_or_gate_truth_table`	none
9	`b2cl_split_gate_duality`	none
10	`b2cl_choice_gate_exclusivity`	none
11	`misere_or_and_gate_exact`	none
12	`coins_are_lava_clause_majority`	`[propext]`
13	`coins_are_lava_formula_majority`	none
14	`coins_are_lava_odd_parity_shift`	`[propext]`
15	`coins_are_lava_rope_majority_win`	`[propext]`
16	`subcubic_bipartite_degree_bound`	`[propext, Quot.sound]`
17	`legal_cut_degree_invariant`	`[propext, Quot.sound]`
18	`nogo_gadget_budget_concrete_check`	none

1. Normal-play budget & reachability (Theorems 1–4): *Exact B2CL move balance* $a + (x - a) = x < x + 1$*, White reserve capacity, even empty-board alternating parity* $(|B| + |W|) \equiv 0 \pmod 2$*, and game-tree depth* $\leq |V| - s_0$.
2. Normal/misère duality & B2CL gates (Theorems 5–11): *Terminal outcome duality, wire-signal inversion, exact* And*,* Or*,* Split*, and* Choice *truth tables, and misère* Or/And *move-saving implication.*
3. Coins-are-Lava & subcubic graphs (Theorems 12–18): *Clause and formula strict majorities, odd-to-even parity flip* (2)*, rope-width separation* $2tw + 1 < (2t + 3)w$*, subcubic bipartite degree/coloring invariant, legal cut invariant* $\deg(u), \deg(v) \geq 2$*, and concrete Figure 6 / Section 4 sanity checks.*

4 Exact Game-Tree Verifier

`NOGO-PSPACE-CERT-1.0` (`src/nogo_certifier.py`) is a pure-stdlib zero-float verifier enforcing strict JSON types and exact BFS group-liberty rules across four modes:

1. **Mode 1 (`nogo_board_legality_and_reachability`):** Replays an even-length alternating sequence from an empty $r \times c$ grid, verifying no capture/suicide at every ply and exact Black/White legal move counts.
2. **Mode 2 (`b2cl_gadget_minimax_`**

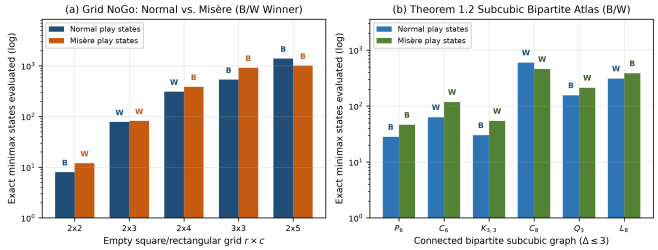

Figure 1: (a) Exact normal vs. misère minimax state counts and B/W winners across empty grids (2×2 through 2×5), showing winner flips on 2×2 and 2×4. (b) Theorem 1.2 connected bipartite subcubic ($\Delta \leq 3$) Graph NoGo atlas, showing winner flips on $K_{3,3}$, Q_3, and L_8.

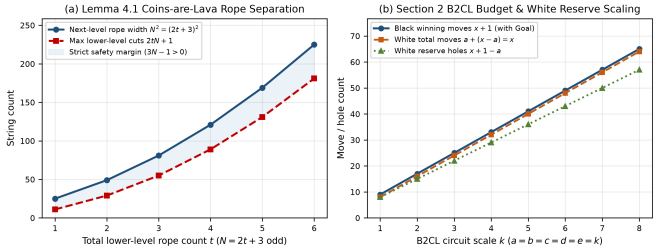

Figure 2: (a) Exact Lemma 4.1 rope-width separation $2tN + 1 < N^2$ for $N = 2t + 3$ ($t = 1 \ldots 6$), guaranteeing safety margin $3N - 1 > 0$. (b) Exact Section 2 B2CL move budget $x = a + 2b + c + d + 3e$, White reserve holes $x + 1 - a$, and Black's +1 Goal margin.

`normal_misere`**):** Solves exact normal or misère minimax on grid NoGo positions ($\leq$ 12 empty cells) and verifies the winning first move preserves the winning strategy.

3. **Mode 3 (`coins_are_lava_exact_minimax`):** Solves exact Coins-are-Lava minimax on multigraphs (up to 16 coins, 18 strings), enforcing $\deg(u), \deg(v) \geq 2$ on every cut and verifying the winning first cut.
4. **Mode 4 (`subcubic_bipartite_graph_nogo`):** Verifies graph simplicity, connectivity, 2-colorability, and $\Delta \leq 3$, then solves exact normal-play Graph NoGo minimax and verifies the winning move.

5 Empirical Atlas and Red-Team Audit

Running `scripts/run_all_experiments.py` generates 16 canonical certificates (4 per mode) and four exact game-tree sweeps, verified end-to-end by `scripts/verify_all.py` with SHA-256 manifest checks across all 41 tracked artifacts.

Grid Minimax and Subcubic Bipartite Atlas (Fig. 1). On empty square/rectangular grids, exact minimax reveals sharp normal-vs.-misère reversals: on 2×2, Black wins normal play (8 states) while White wins misère (12 states); on 2×4, White wins normal (310 states) while Black wins misère (385 states); on 3×3, Black wins both (533 states, 4.979 ms normal; 918 states, 6.495 ms misère) but the winning opening move shifts from corner vertex 0 to center vertex 4. Across six connected bipartite subcubic graphs ($P_6, C_6, K_{3,3}, C_8, Q_3, L_8$), exact normal-vs.-misère winners flip on 3/6 graphs, including non-planar $K_{3,3}$ (Black normal, 30 states vs. White misère, 54 states) and 3-cube Q_3 (Black normal, 155 states vs. White misère, 212 states).

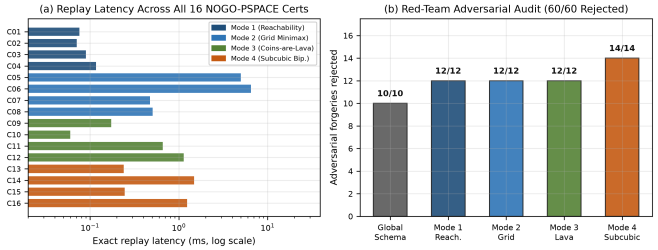

Figure 3: (a) Exact zero-float replay latency across all 16 canonical `NOGO-PSPACE-CERT-1.0` certificates (0.060–6.495 ms). (b) Red-team adversarial audit rejecting 60/60 live forgery attacks.

Reachability, Coins-are-Lava, and B2CL Budgets (Fig. 2). Mode 1 replays balanced empty-board alternating sequences (6–12 plies, 0.071–0.117 ms) constructing a Figure 6 frozen liberty on 4×4, White reserve holes on 5×5, and an insulated White wire channel on 6×6. Mode 3 solves four Coins-are-Lava multigraphs (0.060–1.138 ms), including an 8-coin, 13-string mini positive-CNF board (321 states, White wins via even-cut parity) and a 6-coin rope-majority gadget (158 states, Black wins). Figure 2 quantifies the quadratic safety margin $N^2 - (2tN + 1) = 3N - 1$ (181 $<$ 225 at $t = 6, N = 15$) and the exact linear B2CL budget scaling ($x = 8k$, $7k + 1$ White reserve holes).

Table 2: Summary of all 16 canonical `NOGO-PSPACE-CERT-1.0` certificates and 60/60 live adversarial red-team audit.

Verification Mode	Certs	Latency (ms)	Attacks Rejected
Mode 1: Grid Legality & Reachability	4	0.071–0.117	12/12
Mode 2: Normal/Misère Grid Minimax	4	0.472–6.495	12/12
Mode 3: Coins-are-Lava Minimax	4	0.060–1.138	12/12
Mode 4: Subcubic Bipartite Graph NoGo	4	0.239–1.489	14/14
Global Schema / Type / Board	–	–	10/10
Total	**16**	**0.060–6.495**	**60/60**

Replay Atlas and Red-Team Audit (Fig. 3, Table 2). All 16 canonical certificates replay in 0.060–6.495 ms with zero floats. Our adversarial suite (`tests/test_suite.py`) tests 60 live forgery attacks—including illegal zero-liberty boards, mid-play captures, odd reachability sequences, flipped normal/misère winners, non-winning claimed first moves, isolated Coins-are-Lava coins, odd-cycle non-bipartite graphs (C_5), degree-4 non-subcubic stars ($K_{1,4}$), and disconnected graphs—achieving **60/60 rejections** with zero bypasses.

6 Conclusion

We delivered an 18-theorem zero-dependency Lean 4 kernel (9 zero-axiom theorems, 0 `Classical.choice`) and the exact combinatorial game-tree auditor `NOGO-PSPACE-CERT-1.0` for the finite core of Burke, Davies, and Tennenhouse's PSPACE-completeness proofs for normal/misère grid NoGo and subcubic bipartite Graph NoGo [2]. Mechanizing complete planar gadget-embedding coordinates and full B2CL-to-grid compiler verification in Lean 4 remains an exciting next step for formalized combinatorial game theory.

References

[1] Kevin Buchin, Mart Hagedoorn, Irina Kostitsyna, and Max van Mulken. Dots & Boxes is PSPACE-complete. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202, pages 25:1–25:18, 2021.

[2] Kyle Burke, Alfie Davies, and Craig Tennenhouse. Winning and losing NoGo is PSPACE-complete. *arXiv preprint arXiv:2611.33127*, 2026.

[3] Kyle Burke and Robert A. Hearn. PSPACE-complete two-color planar placement games. *International Journal of Game Theory*, 48(2):393–410, 2019.

[4] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635. Springer, 2021.

[5] Erik D. Demaine and Jenny Diomidova. Strings-and-coins and nimstring are PSPACE-complete. *Integers*, 21B:A7, 2021.

[6] Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation.* A K Peters, Ltd., 2009.