3×3 Nexus Is a First-Player Win

Ian Joiner
Haapavesi Institute

Claude
AI system developed by Anthropic

November 2026

Abstract

Nexus is a two-player, perfect-information game that combines Go-style placement and capture on an $n \times n$ grid, tic-tac-toe-style line completion, and a debt mechanism: every capture creates debt, and a player who completes a line while owing more than the opponent must pay the difference by removing their own stones, or lose by bankruptcy. We weakly solve the 3×3 game with the default debt parameters (two units per captured stone, four per stone removed in forced unblocking) under positional superko: the first player (Black) wins. Every first move wins. From a corner or an edge Black forces a win within 9 turns, and from the centre within 18; exhaustive search shows that no opening forces a win within 8 turns, and the centre opening not within 17. The proof is a finite strategy tree, extracted by an exact depth-limited search over complete move histories, and every tree was checked by replaying all of its lines through two separate implementations of the rules: the game's reference implementation, and the TypeScript engine of the Peace Cat Games website, with its draw rule. The search is small (about 2.2×10^5 nodes and 0.02 s for the main result), but the winning line already uses the debt rule: Black completes a line while in debt and pays for it with two stones that lie off the line. Varying the prices over $0 \le p_1, p_2 \le 20$ and some larger values, Black still wins at every pair, but the first moves do not. At every pair tested, for $p_1 \ge 2$ the centre opening wins exactly when $p_2 \ge 3p_1 - 2$, so the default prices lie on its boundary; for $p_1 \ge 3$ the corner opening loses exactly when $p_2 \ge 4p_1 - 2$; and the edge opening is undecided when $p_2 \ge 5p_1 - 4$. On 4×4 and 5×5 neither player can force a win within 20 and 16 turns. We also compare Nexus with Go and Enhanced Go in state-space and game-tree size, and show that debt cycles let a game of Nexus last forever.

1 Introduction

A game is *weakly solved* when the game-theoretic value of its initial position is known together with a strategy that attains that value against any opposition [1, 13]. Small games have long served as test beds for solving methods, from tic-tac-toe to small-board Go [14] and, at the other end of the scale, checkers [10].

Nexus [3] is played on the intersections of an $n \times n$ grid. Stones are placed and captured as in Go; a player wins by owning a complete row, column or main diagonal, as in tic-tac-toe; and the two are tied together by debt. Capturing is not free: each captured stone adds to the capturer's debt, and when a player completes a line, the debts are audited. A player who owes no more than the opponent wins at once; otherwise they must remove their own stones to pay the difference, keep the win only if a complete line survives, and lose outright if they cannot pay. A player with no legal placement must first remove opponent stones, at a higher price per stone. Positional superko forbids repeating a position.

This paper settles the smallest interesting case, $n = 3$ with the default prices. We state the rules precisely, including the points where the written rules are silent (Section 2); describe an exact

search that respects the history dependence created by superko (Section 3); explain how the result was verified, by two checkers built on two separate implementations of the rules (Section 4); and give the result with its principal variations (Section 5). We then vary the prices (Section 6): Black wins at every pair we tested, but which first moves win changes along straight lines in the (p_1, p_2) plane, and the default prices lie exactly on one of them. On 4×4 and 5×5 we obtain only lower bounds (Section 7). Finally we compare Nexus with Go and with Enhanced Go in state-space size, game-tree size and computational complexity, and show that debt cycles allow a game of Nexus to go on forever (Section 8).

2 Rules

Two players, Black (who moves first) and White, alternate on the 9 intersections of a 3×3 grid, which starts empty. Each player P carries a debt $D_P \geq 0$, initially 0. Two parameters fix the prices: p_1, the debt per captured stone, and p_2, the debt per stone removed in forced unblocking. The defaults, studied here, are $p_1 = 2$ and $p_2 = 4$.

Placement and capture. Groups and liberties are as in Go (orthogonal adjacency). After a stone is placed, every adjacent opponent group without liberties is removed; if k stones are removed, the mover's debt increases by kp_1. A placement that leaves the mover's own group without liberties is illegal (suicide) unless it captured.

Lines and the audit. The $2n + 2 = 8$ lines are the three rows, the three columns and the two main diagonals. If, after a placement and its captures, the mover P owns a complete line, an audit follows with $net = D_P - D_Q$, where Q is the opponent. If $net \leq 0$, P wins. If P has fewer than net stones, P loses by bankruptcy. Otherwise P chooses and removes exactly net of their own stones; if a complete line of P's survives, P wins, and otherwise both debts are reset to 0 and play passes to Q.

Forced unblocking. If the player to move has no legal placement, they must first remove the minimum number k of opponent stones that creates a legal placement, adding kp_2 to their debt, and then place a stone normally. If no removal creates a legal placement, they lose.

Superko. A move is illegal if it recreates any earlier pair (board, $D_{\text{Black}} - D_{\text{White}}$).

2.1 Interpretations

The written rules [3] leave some points open. Where they are silent we follow the reference implementation, `nexus/game.py` in [3], which also served as the arbiter in verification (Section 4). We call one complete action of a player a *turn*: an optional forced unblocking, a placement, and an optional audit removal.

I1. The superko history is the set of keys (board, $D_{\text{Black}} - D_{\text{White}}$); the side to move is not part of the key.

I2. A key is recorded at the end of every completed, non-final turn: after captures and after any audit removal and debt reset. The empty starting board is not recorded, nor is the position immediately after an unblocking.

I3. A placement is illegal if the position immediately after it and its captures, before any audit, is already recorded. That position's debt difference includes capture debt and any unblocking debt paid in the same turn. Audit removals are never tested against the history.

I4. Points forbidden by superko count as unavailable when deciding whether a player has a legal placement.

I5. When several minimum unblocking sets exist, the mover chooses among them. Minimality is judged with the debts before the p_2 charge; the placement that follows is judged with the debts after it.

I6. One case is undefined in the reference implementation: minimum unblocking sets exist, but after the p_2 charge superko leaves no legal placement. We score it as a loss for the mover. It never occurred in any search, in any strategy tree, or in any of the 58,856 differential-test positions.

I7. There is no pass move.

I8. Unending play would be scored a draw. This never matters: the result is a finite strategy tree, so every game in which Black follows it ends.

I9. Every distinct choice of the opponent (unblocking set, placement, audit removal set) is a separate branch that the strategy must answer.

The draw rule. The game's designer has since added a draw rule, which the Peace Cat Games site plays [7]: a game is drawn after $2n^2$ consecutive turns without a capture (18 on 3×3), or when both players agree. The reference implementation predates it. A draw by agreement needs the loser's opponent to agree, so it cannot spoil a winning strategy; the quiet-turn draw could. Every strategy tree in this paper was therefore also checked under the site's rules, draw rule included (Section 4), and every one passed.

3 Method

Representation. A position is two 9-bit masks (one per colour), the two debts, the side to move, and the superko history. A turn is a triple (S, m, R): the unblocking set S (possibly empty), the placed point m, and the audit removal set R (possibly empty). The move generator enumerates every legal triple, so all of a player's choices, including which stones to pay with in an audit, become explicit branches.

History dependence. Under superko the legality of a move depends on every position seen so far, not only on the current one. A value computed for a position reached by one path can therefore be wrong for the same position reached by another, which is the graph-history interaction problem [4]. We avoid it rather than repair it: every superko test uses the exact history of the real path from the empty board.

Search. The proof search is a plain depth-limited minimax over exact histories, with three outcomes: *win* within the limit, *loss* within the limit, or *unknown*. A depth limit can turn a result into *unknown* but can never produce a wrong win or loss, so the search is sound. The limit is raised (iterative deepening) until the root is proved. A transposition table keyed on (board, both debts,

side to move) is unsound under superko because it ignores the history, so we use it only to order moves, never to decide values.

Strategy extraction. From the root, the extractor replays the game with full history and no table shortcuts. At each Black node it keeps one turn proved winning by exact search, choosing the shortest win. At each White node it keeps every legal turn. The output is a finite tree whose leaves are finished games; finiteness is what guarantees termination when Black follows it (interpretation I8).

4 Verification

Differential testing. The solver's move generator, written in Rust for speed, was compared with the reference implementation. On 58,856 positions drawn from random playouts and from random boards seeded with ko patterns, the complete list of legal turns and their outcomes agreed exactly, with no mismatches. The positions included 10,694 with forced unblocking, 5,444 audits with a choice of removal, 15,499 positions in which superko removed a move, and 2,183 positions with no move.

Independent checking. A separate checker, which uses only the reference implementation and none of the solver's code, replays every line of a strategy tree from a fresh game, so the superko history is the reference implementation's own. It requires that:

- at each Black node there is exactly one turn, legal in the reference implementation;
- at each White node the children are exactly the set of turns the reference implementation allows, with none missing, extra or duplicated;
- every leaf is a finished game won by Black, and no internal node is a finished game.

All eleven strategy trees (the main strategy, a copy produced with table-based ordering, and one tree for each of the nine first moves) passed. The main strategy checks in 0.07 s and the centre-opening tree, the largest, in 0.9 s.

A second checker, on the site's engine. The Peace Cat Games site plays Nexus with its own engine, a TypeScript port of the reference implementation [7]. The port was tested for parity on 172 recorded games, and it differs from the original in three ways: it refuses a payment that names the same stone twice (the original accepted one and removed it once), a turn that fails part-way changes nothing (the original could leave an unblocking removal on the board), and it has the draw rule above. A second checker, `tools/check_strategy_ts.mjs`, applies the same three requirements as the first, but takes every legal turn and every outcome from the site's engine: its turn status (placement, unblocking with its options, or blocked), its preview of a placement after a given unblocking, and its function that plays a whole turn and applies the draw rule. It shares no code with the solver or with the first checker. All eleven trees pass it, with the draw rule on. The longest stretch without a capture in any line is 7 turns in the corner and edge trees and 9 in the centre tree, half the limit. Each checker rejects all six deliberately corrupted trees we gave it: an illegal winning turn, a duplicated reply, a missing reply, a winning turn replaced by a non-winning one, the wrong winner in the header, and a tree relabelled with other prices. With the quiet-turn limit lowered to 5 for the test, the site checker rejects the centre tree as a draw.

5 Results

Theorem 1. *In 3×3 Nexus with $p_1 = 2$, $p_2 = 4$ and positional superko, under interpretations I1–I9, Black (moving first) wins. Every first move wins.*

The same strategies win under the Peace Cat Games rules, draw rule included, since every tree also passes the second checker.

Table 1 lists, for each first move, the shortest win Black can force and the size of the verified strategy tree. A *turn* is one player's complete action, so a 9-turn game has five Black turns and four White turns. Exhaustive search shows that no first move forces a win within 8 turns, and that the centre opening does not force a win within 17. These lower bounds rest on the solver's exact search alone; the strategy trees, which give the upper bounds, are independently checked.

First move	Forced win within	Tree nodes	Leaves	Longest game
$(0,0)$ corner (main strategy)	9	198	79	9
$(0,1)$ edge	9	202	76	9
$(0,2)$ corner	9	206	82	9
$(1,0)$ edge	9	196	74	9
$(1,1)$ centre	18	2743	1139	18
$(1,2)$ edge	9	229	85	9
$(2,0)$ corner	9	240	94	9
$(2,1)$ edge	9	247	82	9
$(2,2)$ corner	9	240	95	9

Table 1: Each first move of Black, with the shortest forced win (in turns) and the size of its verified strategy tree. Points are (row, column), with row 0 at the top. Tree nodes count turns.

The trees for symmetric openings differ in size because the extractor keeps the first shortest win it proves, not a canonical one.

5.1 The main line

Figure 1 shows the principal variation of the main strategy: White's longest resistance against the corner opening.

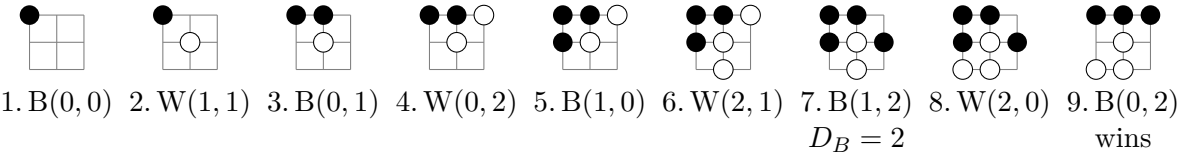


Figure 1: Principal variation of the main strategy, showing the position after each turn (Black filled, White open). At turn 7 Black captures the White stone at $(0,2)$ and owes 2. At turn 8 White cannot retake $(0,2)$, because the point is suicide for White. At turn 9 Black completes the top row with $net = 2$, pays with the two stones at $(1,0)$ and $(1,2)$, which lie off the row, and wins.

This short line already uses each of the game's mechanisms. Black's capture at turn 7 is what makes $(0,2)$ unplayable for White. It also puts Black in debt, so Black's completing move triggers a real audit. Black can pay because it has two stones off the winning row. Against Black's corner

opening, White's second moves $(1,1)$, $(1,2)$ and $(2,1)$ hold out for 8 turns; every other second move loses within 6.

5.2 The centre opening

The centre is Black's slowest first move: 18 turns. Its principal variation includes a forced unblocking by White (turn 8, raising White's debt to 4). At turn 9 Black completes a line, but the audit costs Black two stones from that line, so the line breaks and the debts reset to 0. Later White's debt rises to 8 through captures, and the game ends at turn 18 when White completes a line it cannot pay for and loses by bankruptcy. The full line is given in the artifact (`results/pv_center_opening.txt`).

5.3 Cost

On one thread of an Intel i7-12700H, the exact search proving the root result to depth 9 visits 224,253 nodes in 0.02 s, with 1.8 MB peak memory. The exact search for the centre opening visits 35,293,441 nodes in 2.1 s. Differential testing takes about two minutes.

6 Other debt prices

The prices p_1 (per captured stone) and p_2 (per stone removed in forced unblocking) are parameters of the game, and the Peace Cat Games site lets players choose each from 1 to 20. We repeated the 3×3 analysis for every pair of prices from $\{0, 1, \ldots, 20, 32, 100, 1000\}$ (576 pairs) and for 112 further pairs on the boundary lines described below: 688 pairs in all.

Method. The rules are unchanged by the eight symmetries of the square, so Black has three distinct first moves: a corner, an edge and the centre. For each pair of prices and each first move, two runs of the two-valued search of Section 7 (with its sound transposition table) raced each other, one asking whether Black forces a win and one whether White does, each limited to 60 turns and 120 s. When one run proved a win, the other was stopped. Every proved win was extracted as a strategy tree, and every one of the 1,773 trees passed both checkers of Section 4: the reference implementation, and the site's engine with its draw rule on. The value of the empty board follows from the first moves: Black wins if some first move wins, and White wins only if every first move loses. The undefined case of interpretation I6 never occurred. The longest stretch without a capture in any line of any tree is 9 turns, half the site's limit, so the draw rule never comes close to mattering. Every decided case was settled within 5×10^5 search nodes (median 716); the time went to the undecided cases, each of which used its full 120 s per side. The scripts are `tools/sweep_params.py`, `check_params.sh` and `summarize_params.py`, and `results/params/summary.csv` lists every run.

Theorem 2. *For every pair of prices tested ($0 \le p_1, p_2 \le 20$, and pairs with a price of 32, 100 or 1000), Black, moving first, wins 3×3 Nexus. Each claim is a strategy tree checked by both checkers.*

The value of the empty board never changes, but almost everything else does (Figure 2).

Cheap captures. With $p_1 \le 1$ every first move wins, and the corner and the edge win within 7 turns (Black's fourth stone) but not within 6, whatever p_2 is.

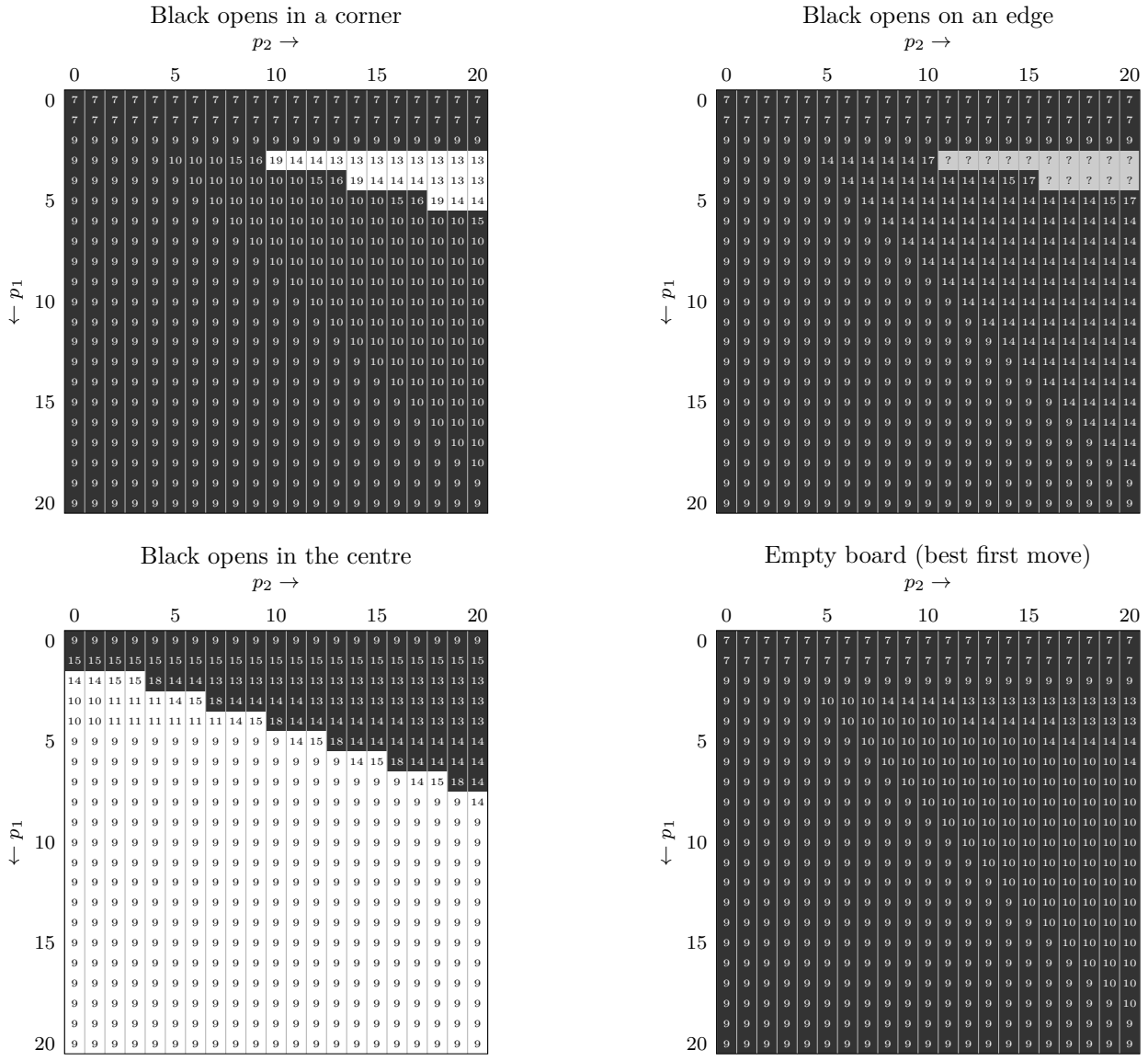


Figure 2: Value of 3×3 Nexus for $0 \leq p_1, p_2 \leq 20$ (rows p_1, columns p_2). A filled cell is a Black win, an open cell a White win, a grey cell with "?" undecided; the number is the shortest forced win in turns from the empty board (so the first move counts as turn 1). The fourth panel is the empty board: Black's best first move. The default prices are row 2, column 4. Values for the prices 32, 100 and 1000 are in Table 2.

The centre. For $p_1 \geq 2$ the centre opening wins exactly when $p_2 \geq 3p_1 - 2$, and loses to White otherwise. On the line $p_2 = 3p_1 - 2$ itself the centre wins in exactly 18 turns, as at the defaults, and one step below it ($p_2 = 3p_1 - 3$) White wins in 15, for every p_1 we tried from 2 to 1000. The default prices lie exactly on this line: with $p_2 = 3$ instead of 4, the centre opening would lose. The principal variation of Section 5 shows why the line has this slope. At turn 9 Black captures three White stones and completes the diagonal, while White has paid for one unblocking, so Black owes $net = 3p_1 - p_2$ and holds four stones. With $net \leq 1$ Black keeps the diagonal and wins at once; at the defaults ($net = 2$) Black pays two stones, the debts reset, and Black still wins at turn 18. The searches put the boundary of the centre opening exactly where net in this line passes from 2 to 3, although away from the defaults White's best defence need not follow this line.

The corner and the edge. For $p_1 \leq 2$ both win for Black at every p_2. For $p_1 \geq 3$ the corner opening loses exactly when $p_2 \geq 4p_1 - 2$; just before the line it is a slow Black win (16 turns at $p_2 = 4p_1 - 3$), and on it a slow White win (19 turns), again for every p_1 from 3 to 1000. For $p_1 \geq 3$ the edge opening is a Black win for $p_2 \leq 5p_1 - 5$ (in 17 turns on that line), and for $p_2 \geq 5p_1 - 4$ it is undecided: Black has no forced win within 28 to 61 turns and White none within 25 or 26. For both, with $p_1 \geq 3$, a smaller step appears at $p_2 = p_1 + 2$, where the shortest win grows from 9 turns to 10 (corner) or 14 (edge). When the unblocking price is large compared with the capture price, the only first move known to win is the centre, which then wins in 13 or 14 turns.

Boundaries. Beyond the grid we tested the pairs on both sides of each line: $p_2 = 3p_1 - 3$ and $3p_1 - 2$ for the centre, $4p_1 - 3$ and $4p_1 - 2$ for the corner, $5p_1 - 5$ and $5p_1 - 4$ for the edge, and $p_1 + 1$ and $p_1 + 2$ for the step, for every p_1 from 2 or 3 to 20 and for $p_1 = 32$, 100 and (except for the edge) 1000. All 1,849 runs agree with the lines, with no exception. The lines also explain Theorem 2: wherever the corner loses ($p_2 \geq 4p_1 - 2$), the centre wins, since $4p_1 - 2 \geq 3p_1 - 2$. We have not proved that the lines continue beyond the prices tested.

What the undecided cases suggest. In most undecided edge openings with $p_1 \geq 5$, Black's search completes all 60 turns while visiting fewer than 10^5 nodes, so White has an inexpensive way to stop Black from winning within any depth we searched; but White's own search finds no win within 25 turns. An expensive unblocking leaves the player who paid for it unable to afford a line: once their debt exceeds the opponent's by more than the stones they own, completing a line is bankruptcy, and only the opponent's debt can close the gap. A natural guess is that these openings are draws by unending play (interpretation I8), or by the quiet-turn rule on the site. Proving a draw would require a different kind of certificate from ours: a strategy that never loses, which, because play can be infinite (Theorem 4), is not a finite tree.

7 Larger boards: 4×4 and 5×5

We applied the same method to $n = 4$ and $n = 5$, with the same prices ($p_1 = 2$, $p_2 = 4$) and interpretations I1–I9. On an $n \times n$ board the $2n + 2$ lines are the n rows, the n columns and the two main diagonals, each of length n. A mover who completes a line while owing $net > 0$ therefore keeps the win exactly when $net \leq$ (own stones) $- n$. Black can first complete a line at turn $2n - 1$ and White at turn $2n$. Each board size was given a compute budget of about 45 minutes per side.

Result in brief. Neither board size was solved. Within the budget, exact search shows that on 4×4 Black cannot force a win within 20 turns and White cannot within 23, so neither player can

	corner, $p_2 =$			edge, $p_2 =$			centre, $p_2 =$		
p_1	32	100	1000	32	100	1000	32	100	1000
3	W13	W13	W13	?	?	?	B13	B13	B13
4	W13	W13	W13	?	?	?	B13	B13	B13
5	W13	W13	W13	?	?	?	B13	B13	B13
6	W13	W13	W13	?	?	?	B13	B13	B13
7	W14	W13	W13	?	?	?	B13	B13	B13
8	W14	W13	W13	B14	?	?	B14	B13	B13
9	B15	W13	W13	B14	?	?	B14	B13	B13
10	B10	W13	W13	B14	?	?	B14	B13	B13
11	B10	W13	W13	B14	?	?	B14	B13	B13
12	B10	W13	W13	B14	?	?	W14	B13	B13
13	B10	W13	W13	B14	?	?	W9	B13	B13
14	B10	W13	W13	B14	?	?	W9	B13	B13
15	B10	W13	W13	B14	?	?	W9	B13	B13
16	B10	W13	W13	B14	?	?	W9	B13	B13
17	B10	W13	W13	B14	?	?	W9	B13	B13
18	B10	W13	W13	B14	?	?	W9	B13	B13
19	B10	W13	W13	B14	?	?	W9	B13	B13
20	B10	W13	W13	B14	?	?	W9	B13	B13
32	B9	B10	W13	B9	B14	?	W9	B14	B13
100	B9	B9	W13	B9	B9	?	W9	W9	B13
1000	B9	B9	B9	B9	B9	B9	W9	W9	W9

Table 2: The large prices. Bd and Wd: Black or White forces a win within d turns from the empty board; ?: undecided. Rows $p_1 = 0$, 1 and 2 are omitted: every entry there is a Black win (within 7, 7 and 9 turns from a corner or an edge; 9, 15 and 13 from the centre). For $p_1 \in \{32, 100, 1000\}$ and $p_2 \leq 20$, every corner and edge opening is B9 and every centre opening W9.

within 20; on 5×5 neither player can force a win within 16 turns (Table 3). No forced win was found for either side at any depth searched, so there was no strategy tree to verify.

7.1 Engine and differential test

The move generator now handles any $n \leq 5$, storing each colour as an n^2-bit mask and each superko key in 128 bits. Its output on 3×3 is unchanged: the generalized solver regenerates all eleven 3×3 strategy trees byte for byte. The differential tester and the independent checker take the board size as a parameter. We repeated the differential test against the reference implementation on each size, with positions from random playouts and random boards seeded with superko bait, as in Section 4:

n	positions	mismatches	no move	unblocking	audit with choice	superko-limited
3	58,856	0	2,183	10,694	5,444	15,499
4	54,704	0	509	5,031	5,006	8,930
5	1,227	0	8	96	97	179

The 5×5 sample is small because the reference implementation is slow there: it re-enumerates the minimum unblocking sets for every turn it validates, and on a crowded 5×5 board there are many candidate sets. We ran 32 shards of one playout and ten random boards each, with a 25-minute limit per shard; 30 finished, and two were stopped by the limit.

7.2 Search

To get deeper, we changed the search in three ways. Each change keeps it sound.

Two-valued search. For each side X separately, the solver decides "X can force a win within d turns". This is an AND/OR search with exact histories. At X's nodes it stops at the first winning turn, and at the opponent's nodes it stops at the first turn after which X has no forced win. The three-valued search of Section 3 has no such cut-off: at a node whose value is still unknown it must examine every child. On 3×3 the new search proves the root, a Black win within 9 turns but not within 8, in 2,940 nodes; without the table of the next paragraph it needs 3,906. The three-valued search needed 224,253. It also reproduces the centre-opening result: a win within 18 turns but not within 17. The trees it extracts pass both checkers. A "no" at depth d for both sides means that neither player has a strategy that wins by turn d against every defence.

A sound transposition table. Section 3 used a transposition table only for move ordering, because a value computed with one history can be wrong with another. The move generator, however, depends on the history in only one way: it asks whether certain superko keys are present. Each key it asks about is the board right after a placement and its captures. When a subtree has been searched, let θ be the smallest number of stones on any board whose key was looked up anywhere in it, including inside any table entries the search reused.

Theorem 3. *Suppose the search of position P with history H returns v for "X wins within r turns", with threshold θ. Let H' be another history at P that contains exactly the same keys with at least θ stones as H. Then v is also the answer to "X wins within r turns" at P with history H'. Hence, under H', X wins within every $r' \geq r$ turns if v is a win, and within no $r' \leq r$ turns if it is not.*

Proof. Compare the two searches step by step. At a node below P, the history is $H \cup A$ or $H' \cup A$, where A is the set of keys recorded along the path from P, the same in both. Every key k that the first search looks up has at least θ stones. So $k \in H \cup A$ exactly when $k \in H' \cup A$, and both searches generate the same turns, outcomes and children. A table entry reused inside the subtree was valid for $H \cup A$ under its own threshold, which is at least θ, so it is valid for $H' \cup A$ too. So the search under H' would make exactly the same decisions and return v, and the exact search is correct. The statement about other depths holds because a win within r turns is a win within any $r' \geq r$ turns, and a win within $r' \leq r$ turns would be a win within r. □

Each table entry stores, along with the position (board, both debts, side to move) and the value, the threshold θ and a 128-bit order-independent hash of the history keys with at least θ stones. An entry is reused only if the current history has the same hash. Hash equality stands in for set equality, so a hash collision could cause an error; this is the only point at which soundness is probabilistic. The hash treats the history as a multiset, which is stricter than set equality, so duplicate keys can only cause misses. The rule works because, in play without captures, two move orders that reach the same position differ only in their early positions, which have few stones, while the subtree looks up only keys with more stones. In the runs of Table 3, between 6% and 18% of the nodes visited were answered from the table.

As a check, the solver was run from 620 random start positions, which mostly contained captures, audits or unblockings: 150 on 3×3 (to depth 12), 100 on 4×4 (depth 10) and 60 on 5×5 (depth 8), plus a shallower set of 120. Each position was searched for both sides, once with table reuse and once with the table used only for ordering, i.e. with pure exact search. The per-depth answers agreed in every case, including the 284 runs that found a forced win.

Horizon pruning. Two cut-offs need no history at all. (A) If X is to move with one turn left, X can win only by completing a line now, which requires a line in which X lacks at most one stone. (B) If the opponent is to move with two turns left and X lacks at least two stones in every line, then the opponent's turn cannot add X stones, so X cannot complete a line on the last turn. X then wins only if every turn of the opponent loses at once. Both follow because an X stone can appear on a point only through X's own placement.

Strategy extraction is unchanged: the strategy is replayed with full histories, keeping one proved turn at X's nodes and every turn at the opponent's nodes, and the result goes to the checker.

7.3 Results

Board	Side	no forced win within	nodes searched	time	table entries	peak memory
3×3	Black	8 (wins within 9)	2,940	< 0.1 s	1,493	2 MB
4×4	Black	20	1,781,986,915	2,099 s	6,000,000	590 MB
4×4	White	23	977,224,693	1,690 s	5,192,855	590 MB
5×5	Black	16	1,976,084,372	2,309 s	4,000,000	590 MB
5×5	White	16	151,950,175	134 s	4,000,000	590 MB

Table 3: Deepest depth (in turns from the empty board) at which exact search completed and found no forced win for the given side. Node counts and times cover every iteration up to and including that depth. The next depth was cut off by the 45-minute limit. The table was capped at 6 million entries on 4×4 and 4 million on 5×5; once it is full, no new entries are stored, which costs speed but not soundness. One thread of an Intel i7-12700H per run.

The node count of each iteration grows unevenly with the depth. On 4×4 the ratios between successive iterations alternate between about 1–2 and about 4.5–5.7. Adding a turn for the side trying to win is expensive, but adding a final turn for the defender is cheap, because the horizon cut-offs answer most such leaves without generating any turns. On 5×5 the expensive steps cost factors of 6–13. Each step of two turns therefore costs roughly 5–9 times as much on 4×4 and 15–40 times as much on 5×5, and the next depth on each board would have needed several more hours. For example, the 5×5 White search spent 2,566 s and 2.1×10^9 nodes on depth 17 without finishing it. The ill-defined case of interpretation I6 did not occur in any of these searches.

What the bounds mean. Table 3 gives lower bounds only. No forced win exists for either player within 20 turns on 4×4 (23 for White alone) or within 16 turns on 5×5. These numbers are well beyond the earliest possible line completions (turns 7–8 on 4×4 and 9–10 on 5×5). Neither the bounds nor the absence of a forced win says which side, if either, wins. Like the minimality figures for 3×3, these lower bounds rest on the solver's exact search together with Theorem 3. The differential tests and the cross-checks above support them, but no independent checker confirms them.

8 Complexity of Nexus, Go and Enhanced Go

This section compares Nexus with two games that share its board and its capture rule: Go, and Enhanced Go, a Go variant invented by the first author [6]. We compare state-space size, game-tree size (in the sense of Allis [1]) and the computational complexity of the generalized $n \times n$

games. Everything measured here was computed by the scripts in `nexus/solver/complexity/` of the artifact. Where a number comes from the literature, we say so.

8.1 Rules

All three games are played by Black (first) and White on the intersections of an $n \times n$ grid, which starts empty. Stones form groups by orthogonal adjacency. A placement captures every adjacent opponent group left without liberties, and suicide is illegal. The games differ in their turns, their goals and their repetition rules (Table 4).

Go. Unless stated otherwise, *Go* means standard Go with positional superko (a turn may not recreate any earlier board, the empty starting board included), passing, two consecutive passes ending the game, and area scoring. For game-theoretic purposes we score the final board as it stands (every stone counts as alive), so the game is a finite two-player game with a numerical outcome; komi does not affect any count below.

Enhanced Go. Enhanced Go is Go with a second kind of turn. The rules below are those of the reference implementation [6] (rules text `content.tsx`, engine `engine/*.ts`).

- *Ordinary turn:* place one stone, exactly as in Go. *Pass:* always legal.
- *Special turn:* for some $k \geq 2$, remove exactly k^2 of your own stones and place k stones. The player must own at least k^2 stones at the start of the turn, and only those stones can pay. Removal comes first, and the new stones may go on points just vacated. All k stones land at once; then every opponent group without liberties is captured; then the turn is refused if any of the mover's groups has no liberties.
- *Superko* compares whole boards only (not the side to move); the starting board counts, and a special turn that recreates an earlier board is refused. A pass never violates superko.
- Two consecutive passes end play and start a scoring phase in which the players agree on dead stones (area scoring, komi 7.5). Either player may instead resume play. For game-theoretic purposes we treat two passes as the end of the game and score the board as it stands, as for Go.

Our counts use a reimplementation of these rules in Rust (`src/bin/complexity.rs`). On 660 positions sampled from Enhanced Go playouts on 3×3, 4×4 and 5×5 boards, it agrees exactly with the reference TypeScript engine, which we drove directly with Node.js, on the number of legal ordinary turns, the number of legal special turns and the number of distinct successor boards. The sample contained 186,440 legal special turns, and superko removed at least one turn in 12 of the positions. Because ordinary turns are Go moves, this also checks our Go move generator.

Nexus. The rules are those of Section 2, with interpretations I1–I9, $p_1 = 2$ and $p_2 = 4$; on an $n \times n$ board the $2n + 2$ lines are the rows, the columns and the two main diagonals. There is no pass. Superko compares the board together with the debt difference $D_{\text{Black}} - D_{\text{White}}$.

The length bound for Go and Enhanced Go holds because every turn other than a pass creates a board never seen before in the game, and there are at most $L(n)$ boards; at most one pass separates two such turns.

	Go	Enhanced Go	Nexus
Turn	place one stone, or pass	place one stone; or remove k^2 own stones and place k ($k \geq 2$); or pass	place one stone, after a forced unblocking if no placement is legal; then possibly an audit removal
Goal	larger area	larger area	own a full line, subject to the debt audit
End	two passes	two passes	a line win, a bankruptcy, or no move
Repetition	board may not recur	board may not recur	(board, debt difference) may not recur
Length	finite: at most $2L(n) + 1$ turns	finite: at most $2L(n) + 1$ turns	unbounded; infinite play exists (Proposition 4)

Table 4: The three games side by side. $L(n)$ is the number of legal Go boards (Table 5).

8.2 State-space complexity

What a state is. In all three games the position that determines the future is not just the board. Under positional superko the set of boards already seen restricts every later move, so the full state is (board, side to move, set of earlier boards, and for Go and Enhanced Go whether the last turn was a pass). In Nexus the full state is (board, debt difference, side to move, set of earlier (board, debt difference) keys). The individual debts do not matter: every rule that uses them (the audit's $net = D_P - D_Q$, the bankruptcy test, the reset and the superko key) uses only the difference. Since p_1 and p_2 are even, the difference is always even. Following common practice we count boards (and, for Nexus, give the count per value of the debt difference); the history component multiplies these counts by an amount that is not usefully bounded.

Boards. A board is *Go-legal* if every group has a liberty. Every board reachable in any of the three games is Go-legal, because every turn ends with captures followed by a liberty check of the mover's groups, and removals only add liberties. For Go the converse also holds: the boards that can arise in a game of Go are exactly the Go-legal boards [11]. We confirmed this for $n \leq 4$ by breadth-first search from the empty board over moves of either colour. Enhanced Go therefore has exactly the Go-legal boards too: its ordinary turns and passes are Go moves, and its special turns also produce only Go-legal boards. A Nexus board at the start of a turn is Go-legal and has no monochrome full line: a player's line can only be created by that player's placement, and the audit that follows either ends the game or breaks every full line of the mover.

Asymptotics. For Go, $L(m,n) \sim 0.8506 \cdot 0.96554^{m+n} \cdot 2.97573^{mn}$ [11, 12], so $\log_{10} L(n) = 0.4736\, n^2 + O(n)$, slightly below $\log_{10} 3^{n^2} = 0.4771\, n^2$. Enhanced Go has the same boards. For Nexus boards, a simple construction gives a matching lower bound: leave every odd-numbered row empty and fill each even-numbered row (rows $0, 2, 4, \dots$) with anything except a monochrome full row. Every stone then has an empty neighbour, and every column and diagonal crosses an empty row, so

$$(3^n - 2)^{\lceil n/2 \rceil} \;\leq\; N(n) \;\leq\; L(n) \;\leq\; 3^{n^2},$$

and $\log N(n) = \Theta(n^2)$. The same construction, with any pattern allowed in the filled rows, gives $L(n) \geq 3^{n\lceil n/2 \rceil}$.

	$n=3$	$n=4$	$n=5$
All boards, 3^{n^2}	19,683	43,046,721	847,288,609,443
Go-legal boards $L(n)$ (Go and Enhanced Go)	12,675	24,318,165	414,295,148,741
source	exact, ours	exact, ours	[11]; ours: MC
Nexus boards $N(n)$ (no full line)	7,649	19,458,797	$(3.78316 \pm 0.00009) \times 10^{11}$
source	exact, ours	exact, ours	MC, ours
$\log_{10} L(n)$	4.10	7.39	11.62
$\log_{10} N(n)$	3.88	7.29	11.58

Table 5: Board counts. "Exact, ours" is a complete enumeration of all 3^{n^2} boards (`complexity legal`). For $n = 5$ we cite the exact count of legal Go boards from Tromp's page [11]; our Monte Carlo estimate from 2×10^9 uniformly random boards, $(4.14302 \pm 0.00009) \times 10^{11}$ (one standard error), agrees with it within one standard error. $N(n)$ is an upper bound on the number of Nexus boards at the start of a turn; we did not determine which of these boards are reachable.

The debt difference, and why Nexus has infinitely many states. The board count is only part of the Nexus state; the debt difference is the other part, and it is not bounded.

Theorem 4. *Consider any stretch of Nexus play since the last debt reset (or since the start), with $p_1 = 2$ and $p_2 = 4$. Let U_B and U_W be the numbers of Black and White stones removed by forced unblocking in that stretch. Then*

$$\left|\,(D_{\text{Black}} - D_{\text{White}}) - 2(U_W - U_B)\,\right| \;\le\; 2(2n^2 + 1).$$

In particular, play without forced unblocking keeps the debt difference bounded. With unblocking it is unbounded: on the 3×3 board there is a legal game that, after 7 opening turns, repeats a 2-turn cycle forever, and each repetition lowers the difference by 2.

Proof. Both debts are 0 at the start of the stretch, and the stretch contains no audit removal, since every audit either ends the game or resets the debts. Let P_B, P_W be the placements, C_B, C_W the stones of each colour captured, and s_B, s_W (with initial values s_B^0, s_W^0) the stone counts. Then $D_{\text{Black}} - D_{\text{White}} = 2(C_W - C_B) + 4(U_W - U_B)$, and counting stones gives $C_W + U_W = s_W^0 - s_W + P_W$ and $C_B + U_B = s_B^0 - s_B + P_B$. Substituting,

$$D_{\text{Black}} - D_{\text{White}} = 2\big[(s_B - s_W) - (s_B^0 - s_W^0) + (P_W - P_B)\big] + 2(U_W - U_B),$$

and the bracket is at most $n^2 + n^2 + 1$ in absolute value. For the second claim, the opening $(0,0)$, $(0,1)$, $(0,2)$, $(2,0)$, $(1,1)$, $(2,2)$, $(2,1)$ (players alternating, Black first; Black's $(1,1)$ captures $(0,1)$) leaves Black on $(0,0), (0,2), (1,1), (2,1)$ and White on $(2,0), (2,2)$, with White to move. Every empty point is suicide for White, so White must unblock: White removes the Black stone at $(0,0)$ (paying 4) and plays $(0,1)$. Black replays $(0,0)$, capturing $(0,1)$ (paying 2), and the board and side to move are back where they started, with the difference lowered by 2. Because the difference never repeats, superko never forbids the cycle. The cycle was found by a search over the graph of boards (`complexity cycle 3`). We replayed the opening and 200 repetitions through the reference implementation, which accepted every turn and ended with debts $D_{\text{Black}} = 402$ and $D_{\text{White}} = 800$ (`complexity/verify_cycle.py`). □

So even without its history, the Nexus state space {(board, $D_{\text{Black}} - D_{\text{White}}$, side)} reachable from the empty 3×3 board is infinite, and superko on (board, debt difference) does not make the

game finite: the cycle above can be played forever. This is why interpretation I8 (unending play is a draw) is needed to give the game a value at all. In contrast, Go and Enhanced Go are finite games. We constructed the cycle only for $n = 3$; the theorem shows that on any board only forced unblocking can drive the difference without bound.

8.3 Game-tree complexity

Following Allis [1], we estimate the game-tree complexity as $\bar{b}^{\,\bar{d}}$, where $\bar{b}$ is the average number of legal turns and $\bar{d}$ the average game length, both measured over random playouts from the empty board (`complexity playout`). Because the branching factor of Enhanced Go varies over several orders of magnitude within a game, the arithmetic mean $\bar{b}$ is dominated by a few late positions. We therefore also report the mean over playouts of $\sum_t \log_{10} b_t$, the logarithm of the product of the branching factors met along a game. Playout policies:

- *Go:* uniformly random among legal moves that do not fill a one-point eye of the mover's own (an empty point whose neighbours are all the mover's stones); pass only when no such move exists. Branching b_t counts all legal moves plus the pass.
- *Enhanced Go:* the Go policy; with probability q, a uniformly random legal special turn instead, when one exists. Branching b_t counts all legal ordinary moves, all legal special turns (each choice of removal set and placement set is a separate turn) and the pass. We also count *distinct successor boards*, which merges special turns that give the same board.
- *Nexus:* uniformly random among all turns (unblocking set, placement, audit removal set), the same branching the solver uses; b_t is the number of turns. Every playout ended in fewer than 2,000 turns (the cap).

Game	n	games	$\bar{d}$ (turns)	$\bar{b}$	$\log_{10} \bar{b}^{\,\bar{d}}$	mean $\sum_t \log_{10} b_t$
Go	3	20,000	15.1	4.84	10.3	9.1
	4	20,000	26.8	7.50	23.5	20.8
	5	20,000	40.9	10.9	42.4	37.8
Enhanced Go, $q = 0$	3	5,000	15.1	42.0 (18.7)	24.5 (19.2)	14.0 (13.1)
	4	2,000	27.0	4,199 (795)	97.9 (78.4)	50.2 (45.4)
	5	240	40.8	1,890,856 (148,223)	256.1 (211.0)	125.7 (113.5)
Enhanced Go, $q = 0.1$	3	5,000	17.8	40.3 (18.4)	28.6 (22.6)	16.5 (15.5)
	4	2,000	36.7	3,825 (755)	131.6 (105.7)	70.1 (63.6)
	5	120	79.1	1,618,337 (120,359)	491.1 (401.8)	258.7 (234.0)
Nexus	3	20,000	10.0	5.77	7.6	7.0
	4	20,000	19.9	10.7	20.4	18.0
	5	20,000	33.2	32.7	50.2	35.4

Table 6: Random-playout estimates of game-tree complexity. For Enhanced Go, values in parentheses count distinct successor boards instead of turns. Lengths include passes; no playout reached the cap ($20n^2$ turns for Go and Enhanced Go). In Nexus, 75–80% of random games end with a winning line and the rest with a bankruptcy; none ended with a player unable to move.

Reading the table. On these boards Nexus and Go have game trees of comparable size, and Enhanced Go's is far larger. Random Nexus games are shorter than random Go games, because

a random line completion ends the game while Go continues until the board is filled. But Nexus branches more on 5×5, because an audit with a large debt lets the mover choose any *net* of their stones to remove; this is why its arithmetic mean exceeds the number of points. For optimal play the 3×3 game is much shorter still: the main strategy wins in at most 9 turns. Enhanced Go plays about as long as Go (with $q = 0$ the playouts are Go playouts), but once a player owns $m \geq 4$ stones there are $\sum_{k \geq 2,\, k^2 \leq m} \binom{m}{k^2}\binom{e+k^2}{k}$ candidate special turns, where e is the number of empty points. That makes the mean branching factor nearly three orders of magnitude larger than Go's on 4×4 and about five on 5×5. Occasional special turns ($q = 0.1$) also lengthen the game, because they remove stones and reopen the board.

Asymptotics. In Go a position has at most $n^2 + 1$ legal moves, and if games last $\Theta(n^2)$ turns, as random playouts do, then $\log(\text{game-tree size}) = \Theta(n^2 \log n)$. In Enhanced Go a player with m stones has up to $\binom{m}{k^2}$ removal sets for each k; choosing k^2 within $O(\sqrt{m})$ of $m/2$ gives $2^{\Theta(m)}$ of them, and m can be $\Theta(n^2)$. So a single turn can have $2^{\Theta(n^2)}$ options, and games of $\Theta(n^2)$ turns give $\log(\text{game-tree size})$ up to $\Theta(n^4)$. This is a heuristic count of candidate turns, not a proof that a constant fraction of them is legal. Nexus branches like Go apart from audits and unblockings, but its game length is unbounded (Theorem 4), so its full game tree is infinite. Only depth-bounded trees, like the ones the solver searches, are finite.

8.4 Computational complexity

For each game, consider the decision problem: given an $n \times n$ position together with its history (for Nexus also the debts), does the player to move have a winning strategy (for Go and Enhanced Go, one that wins by a given margin)? We separate what is known or provable from what we conjecture.

Go (known results). Lichtenstein and Sipser proved that Go is PSPACE-hard [5]. Their positions involve no ko, so the bound holds under any ko rule. Robson proved that Go with the Japanese ko rule is EXPTIME-complete [8]. Under positional superko, Robson's EXPTIME-hardness proof does not carry over, because it relies on repeating positions in ko fights. Superko Go is in EXPSPACE: a game lasts at most $2L(n) + 1$, that is $2^{O(n^2)}$, turns, so an alternating machine can play it in exponential time, and alternating exponential time equals EXPSPACE [2]. Robson also showed that some games with a no-repetition rule are EXPSPACE-complete [9]. As far as we know, whether superko Go is EXPTIME-hard or EXPSPACE-hard is still open, so its known bounds are PSPACE-hard and in EXPSPACE.

Enhanced Go (provable here). Enhanced Go is a finite game with at most $2L(n) + 1$ turns (Table 4). Every turn has a description of polynomial size (a removal set and a placement set), and its legality can be checked in time polynomial in n and the length of the history. So the same argument puts Enhanced Go in EXPSPACE, and its depth-bounded version ("can the mover win within k turns", k in unary) in PSPACE by depth-first search. The number of legal turns can be exponential in n^2, but a depth-first search enumerates them one at a time in polynomial space.

Enhanced Go (conjecture). We conjecture that Enhanced Go is at least PSPACE-hard. Go's hardness does not transfer automatically: the reductions use large positions in which both players own many stones, so special turns are available, and placing two stones at once can break a capture race or a ladder. A proof would need gadgets that make every special turn a losing deviation, for

example by giving each player only a few stones outside the gadgets, but the gadgets themselves contain many stones. We have not attempted such a proof.

Nexus (provable here).

1. *Winning is semi-decidable.* Each position has finitely many turns. A winning strategy for the mover is a tree in which the mover's nodes have one child and the opponent's nodes have every child, and every branch ends in a win of the mover. This tree is finitely branching and has no infinite branch, so by Kőnig's lemma it is finite and its depth is bounded. Hence "the mover wins" is equivalent to "the mover wins within D turns for some D", which can be checked for $D = 1, 2, \ldots$ in turn. The search in Section 3 is exactly this procedure, and Section 7 applies it to larger boards.

2. *The depth-bounded problem is in PSPACE.* Deciding whether the mover wins within k turns, with k in unary, takes polynomial space. A depth-first search keeps one turn, one position and the history per level. Unblocking sets, minimality and audit removal sets can each be enumerated in polynomial space.

3. *The unbounded problem has no known complexity bound.* Because play can be infinite (Theorem 4) and the state includes an unbounded integer and an unbounded history, the finite-game arguments used for Go do not apply. We do not know a computable bound on the depth D in item 1 in terms of the position, so we do not know whether the problem is decidable.

Nexus (conjectures). We conjecture that the depth-bounded problem is PSPACE-complete for k polynomial in n, and that the unbounded problem is decidable. The win condition (a complete row, column or main diagonal) is too rigid for the standard reductions from generalized geography or from k-in-a-row games, and the debt audit changes the outcome of every capture race. Hardness would need new gadgets. For deciding the unbounded problem, Theorem 4 suggests an approach: away from forced unblocking, which needs a nearly full board, the debt difference is bounded, so the infinite part of the state space is confined to positions where unblocking is possible.

Summary. Measured on 3×3 to 5×5 boards, Nexus has about as many boards as Go (Table 5; the ratio $N(n)/L(n)$ is 0.60, 0.80 and 0.91) and a game tree of comparable size (Table 6). Enhanced Go has the same boards as Go but a game tree that is larger by tens to hundreds of orders of magnitude. Qualitatively the order is reversed: Go and Enhanced Go are finite games, in EXPSPACE, while Nexus admits infinite play through debt cycles, so even the decidability of its generalized decision problem is open.

9 Discussion and limitations

Rules. The result holds for the rules as stated together with interpretations I1–I9, and the reference implementation decides any remaining ambiguity. Two interpretations deserve attention from anyone who adopts different rules: the exact superko bookkeeping (I1–I4) and the mover's freedom to choose among several minimum unblocking sets (I5). The undefined case (I6) never arises in any tree, so its scoring does not affect the result.

Scope. The 3×3 game is solved at the default prices and at every other pair we tested, but only Black's best first move is known everywhere; the edge opening remains undecided when unblocking is expensive (Section 6), and the boundary lines are observed, not proved, beyond the tested range. On 4×4 and 5×5 we have lower bounds only. There the debt rule should matter more, because the lines are longer and a player has more stones to pay with.

Trust. The upper bounds (the strategy trees) are checked by two checkers independent of the solver and of each other, one built on the reference implementation and one on the site's engine. The lower bounds (the minimality figures, the undecided cases and the larger-board bounds) rest on the solver's exact search; the differential tests support it, but no independent checker confirms it. While building the checker we also found a call, in an unused method of the reference implementation, to a function that does not exist; it plays no part in the result.

Reproducibility

The solver and the independent checker are public at `https://github.com/haapavesi-institute/nexus-solver` (Apache License 2.0). That repository, like the artifact accompanying this paper, contains the solver (Rust), the differential tester and the independent checker (Python), a snapshot of the reference implementation, all strategy trees (as text and as JSON, with the board and debts at every node), the principal variations, and the solver and checker logs. Running `nexus/solver/run_all.sh` rebuilds the solver, repeats the differential test, regenerates every tree byte for byte and re-runs the checker. The repository also holds the second checker, the price sweep with all 1,773 of its trees and both checkers' logs, the mutation test, the larger-board searches and the complexity measurements. The second checker needs the Peace Cat Games engine (`games/nexus/engine` in the site's repository), which is not public at the time of writing.

Author contributions

I.J. posed the problems, chose the rules and interpretations, and reviewed the work and the manuscript. I.J. is responsible for the final text. Claude, an AI system developed by Anthropic, working as an agent, wrote the solver, the prover, the differential tester and both checkers, ran every experiment reported here, and drafted the manuscript.

References

[1] Louis Victor Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, University of Limburg, Maastricht, The Netherlands, 1994.

[2] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[3] Ian Joiner. Nexus: Rules and reference implementation. Software repository, `https://github.com/peace-cat-games/nexus`, 2025. Rules in `RULES.md`; reference engine `nexus/game.py`. A snapshot is included in this paper's artifact and in `https://github.com/haapavesi-institute/nexus-solver`.

[4] Akihiro Kishimoto and Martin Müller. A general solution to the graph history interaction problem. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 644–649, 2004.

[5] David Lichtenstein and Michael Sipser. Go is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, 1980.

[6] Peace Cat Games. Enhanced Go: Rules and rules engine. Web page and software, `https://www.peacecatgames.com/games/enhanced-go`; engine in `games/enhanced-go` of the site's repository, 2026. Ruleset approved by the inventor, Ian Joiner, on 5 November 2026; rules text in `content.tsx`, engine in `engine/*.ts`.

[7] Peace Cat Games. Nexus: Rules and browser engine. Web page and software, `https://www.peacecatgames.com/games/nexus`, 2026. TypeScript port of the reference engine, `games/nexus/engine` in the site's repository; accessed 15 November 2026.

[8] J. M. Robson. The complexity of Go. In *Information Processing 83: Proceedings of the IFIP 9th World Computer Congress*, pages 413–417, 1983.

[9] J. M. Robson. Combinatorial games with exponential space complete decision problems. In *Mathematical Foundations of Computer Science 1984*, volume 176 of *Lecture Notes in Computer Science*, pages 498–506. Springer, 1984.

[10] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007.

[11] John Tromp. Number of legal Go positions. Web page, `https://tromp.github.io/go/legal.html`, 2016. Accessed 12 November 2026.

[12] John Tromp and Gunnar Farnebäck. Combinatorics of Go. In *Computers and Games (CG 2006)*, volume 4630 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2007.

[13] H. Jaap van den Herik, Jos W. H. M. Uiterwijk, and Jack van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1–2):277–311, 2002.

[14] Erik C. D. van der Werf, H. Jaap van den Herik, and Jos W. H. M. Uiterwijk. Solving Go on small boards. *ICGA Journal*, 26(2):92–107, 2003.