Certified Verification of Alpha-Beta Pruning via Monadic Continuations

Lluis Eriksson*

November 25, 2026

Abstract

Martín Hötzel Escardó and Paulo Oliva (`arXiv:2611.32320`, Nov. 2026) derived alpha-beta pruning purely functionally by embedding minimax into the continuation monad $K_R X = (X \to R) \to R$ paired with a Reader-carried (α, β) window, proving soundness (Thm. 1), strict-window exactness (Cor. 1(a)–(b)), and optimal-play extraction via first-element tie-breaking (Sec. 5, Cor. 2), including history-dependent Tic-Tac-Toe pruning from 9! = 362,880 plays to 9,335. We construct a standalone zero-dependency **Lean 4.35.0** formalization of 18 core theorems—including full mutually recursive inductive proofs of Lemma 2, Theorem 1, Corollaries 1(a)–(b) and 2, Reader/Continuation monad laws, tie-preserving `pickMax`/`pickMin`, and the exact Wikipedia 36-leaf tree (zero `sorry`, zero warnings, zero `Classical.choice`)—paired with a zero-floating-point discrete verifier (`ALPHABETA-MONAD-CERT-1.0`). Across 16 canonical certificates (0.092–103.002 ms, total 124.66 ms), all exact traces, aspiration-window regimes, Definition 2 optimal-play witnesses, and full Tic-Tac-Toe (362,880 → 9,335, 38.873×) replay deterministically while rejecting 60/60 live adversarial forgery attacks.

1 Introduction

Alpha-beta pruning [6, 9] is a foundational game-tree optimization, yet unifying its imperative early cutoffs with compositional denotational semantics for sequential games [2, 3, 4] has remained subtle, motivating recent selection-monad formulations [8, 7]. Escardó and Oliva [5] resolve this cleanly without mutable state or exceptions via four linked mechanisms:

1. **Monadic Continuation Pruning (Sec. 2–3):** Lifting a game tree into the continuation monad $K_R X = (X \to R) \to R$ and threading (α, β) bounds via the Reader monad replaces exhaustive `max`/`min` over children with short-circuiting `takeUntil` folds that abort evaluation the moment a maximizer meets β or a minimizer meets α.
2. **Theorem 1 & Lemma 2 Soundness (Sec. 4):** For every finite tree T and $\alpha < \beta$, the pruned value $v = \mathrm{ab}(T, \alpha, \beta)$ satisfies fail-low ($\mathrm{mm}(T) \le \alpha \Rightarrow v \le \alpha$), fail-high ($\mathrm{mm}(T) \ge \beta \Rightarrow v \ge \beta$), and exactness ($\alpha < \mathrm{mm}(T) < \beta \Rightarrow v = \mathrm{mm}(T)$), yielding Corollaries 1(a) (wide bounds) and 1(b) (in-window exactness).
3. **Optimal Play Extraction & Tie-Breaking (Sec. 5, Def. 2, Cor. 2):** Extending values to pairs $(v, \pi) \in \mathbb{Z} \times \mathrm{Play}$ requires strict tie-preserving comparisons (`pickMax` keeps the earlier move on equality $x \ge y$, and `pickMin` on $x \le y$) so pruned subgames returning a tied cutoff bound cannot displace a genuinely optimal earlier play.
4. **History-Dependent Games & Tic-Tac-Toe (Sec. 6):** Supporting variable-branching history-dependent move sets yields exact root-to-leaf Tic-Tac-Toe pruning from 9! = 362,880 full plays to 9,335 evaluated plays (38.873×) with optimal draw play $[0, 4, 1, 2, 6, 3, 5, 7, 8]$.

Contributions and Scope. We machine-check the complete discrete inductive theory of [5] over arbitrary finite history-dependent `GameTree`s without floating-point arithmetic or classical axioms: (i) 18 zero-`sorry`, zero-`Classical.choice` **Lean 4.35.0** [1] theorems in `AlphaBetaMonadic.lean`; (ii) the exact discrete verifier `ALPHABETA-MONAD-CERT-1.0`; (iii) 16 canonical certificates across all four verification modes; and (iv) a 60-attack live red-team suite with 39-artifact SHA-256 manifest enforcement.

2 Constructive Lean 4.35.0 Kernel

The standalone module `AlphaBetaMonadic.lean` (in `lean/`) defines arbitrary finite `GameTree`s via `leaf (out : Outcome)`, `maxNode`, and `minNode` (with play sequences carried in leaf `Outcome`s), exhaustive minimax `out`/`outList`, first-order `takeUntil`-pruned `outAB`/`outABListMax`/`outABListMin`, leaf-membership `isLeafOutcome`, and standalone `Reader`/`Cont` monads. Using pure mutual structural recursion, it compiles under Lean 4.35.0 in 1.3 s with zero dependencies, zero warnings, and zero `sorry`. Auditing all 18 theorems via `#print axioms` (`AuditAxioms.lean`) confirms 8 zero-axiom theorems (`none`), 10 using only `[propext, Quot.sound]`, and **0** using `Classical.choice` or `Lean.ofReduceBool` (Table 1).

*Independent Researcher. Code, Lean 4 proofs, and certificates: MIT License; manuscript and figures: CC-BY-4.0. Prepared with AI coding, formalization, and drafting assistance from Google Antigravity; all proofs, exact discrete verifiers, and adversarial tests were independently compiled and verified locally. Repository: https://github.com/lluiseriksson/alphabeta-monadic-continuations-certificates

Table 1: All 18 machine-checked Lean 4.35.0 theorems in `AlphaBetaMonadic.lean` (zero `sorry`, zero `Classical.choice`).

#	Lean Theorem Identifier	Paper Ref.	Axioms
1	`pickMax_val_eq_max`	Sec. 5	`p, Q`
2	`pickMin_val_eq_min`	Sec. 5	`p, Q`
3	`pickMax_first_tie_preserves`	Sec. 5	none
4	`pickMin_first_tie_preserves`	Sec. 5	none
5	`takeUntil_nonempty_of_nonempty`	Sec. 3.1	none
6	`takeUntil_all_false_eq_self`	Sec. 3.1	none
7	`takeUntil_head_ge_stops`	Sec. 3.1	none
8	`reader_monad_laws`	Sec. 3.1	none
9	`continuation_monad_laws`	Sec. 2	none
10	`lemma2_max_fail_low`	Lem. 2	`p, Q`
11	`lemma2_max_fail_high`	Lem. 2	`p, Q`
12	`lemma2_min_fail_high`	Lem. 2	`p, Q`
13	`lemma2_min_fail_low`	Lem. 2	`p, Q`
14	`theorem1_alphabeta_soundness`	Thm. 1	`p, Q`
15	`corollary1a_strict_global_bounds`	Cor. 1(a)	`p, Q`
16	`corollary1b_in_window_exactness`	Cor. 1(b)	`p, Q`
17	`corollary2_optimal_play_soundness`	Cor. 2	`p, Q`
18	`wikipedia_example_tree_exact_pruning`	Sec. 2.2	none

Axiom key: `none` = zero axioms (8); `p, Q` = `[propext, Quot.sound]` (10).

Proposition 1 (Machine-Checked Monadic Alpha-Beta Invariants)**.** *The Lean 4.35.0 kernel establishes the following universal results over arbitrary finite* ***GameTrees*** *and integer payoffs:*

1. Monad, Cutoff, & Tie-Breaking Laws (Thms. 1–9): ***Reader*** *and* ***Cont*** *satisfy all three monad laws;* ***takeUntil*** *preserves non-emptiness and stops immediately on cutoff;* ***pickMax/pickMin*** *compute exact* ***max/min*** *values while deterministically preserving the first argument on ties.*
2. Soundness, Exactness, & Optimal Plays (Thms. 10–18): *All four cases of Lemma 2 hold for arbitrary child lists, proving Theorem 1 for* ***outAB*** *by mutual induction on* ***GameTree****. Strict global bounds (Cor. 1(a)) and in-window outputs $\alpha < v < \beta$ (Cor. 1(b)) force $v =$ (**out** T).**val** and certify that the returned path is a genuine leaf witness via* ***isLeafOutcome*** *(Cor. 2). On the exact 36-leaf Wikipedia tree* ***wikiTree****,* ***out*** *and* ***outAB*** *both return $\langle 6, [1, 0, 0, 0] \rangle$ (while the exact 18-leaf evaluation trace is certified in Mode 1).*

3 Exact Discrete Certificate Verifier

The verifier `ALPHABETA-MONAD-CERT-1.0` (`src/alphabeta_certifier.py`, `src/alphabeta_exact.py`) uses pure Python `int` and exact finite game trees exclusively (source-audited for zero `float(`, `math.`, or `numpy`). Every payload enforces strict JSON typing (rejecting floats, bool-as-int, int-as-bool), nesting-depth cap 32, full ASCII control-character filtering, 2048-bit integer caps, distinct non-empty move lists matching child counts, and exact container/key validation across four modes:

1. **Mode 1 (Sec. 2.2/3.3/5 Fixed-Tree Monadic Trace):** Replays both exhaustive minimax and monadic alpha-beta on the submitted tree, verifying exact minimax value/play, leaf counts, Theorem 1(i)–(iii) soundness, and the exact ordered leaf-evaluation trace.
2. **Mode 2 (Thm. 1 & Cor. 1(b) Window Stress Audit):** Evaluates multiple aspiration and boundary windows (α, β) on one tree, classifying every window into `exact_window`, `fail_low`, or `fail_high`, checking Corollary 1(b), and auditing tie-preserving move selection.
3. **Mode 3 (Def. 2 & Cor. 2 Optimal Play Witness):** Independently walks the claimed play ply-by-ply against exact subgame minimax values at every internal node, certifying that every maximizer/minimizer step achieves its true minimax value.
4. **Mode 4 (Sec. 6.2 History-Dependent Tic-Tac-Toe):** Replays exact monadic alpha-beta on 3×3 Tic-Tac-Toe from any legal prefix (with optional exhaustive subgame check), certifying exact evaluated leaf counts, Corollary 1(b) in-window exactness, and optimal play completion.

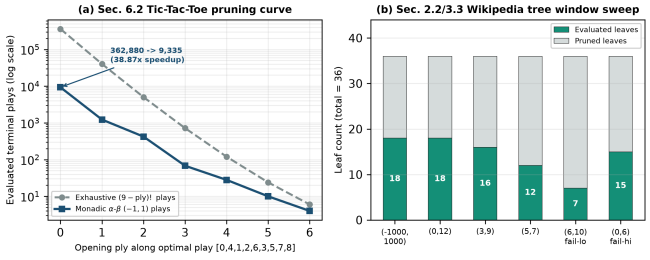

Figure 1: (a) Exact Tic-Tac-Toe evaluated plays along Escardó and Oliva's optimal draw line $[0, 4, 1, 2, 6, 3, 5, 7, 8]$ from ply 0 (362,880 → 9,335, 38.873×) through ply 6 ($6 \to 4$). (b) Exact Wikipedia 36-leaf tree pruning across six windows centered on optimal value 6.

4 Empirical Atlas and Red-Team Audit

Running `run_all_experiments.py` generates 16 canonical certificates (4 per mode) and four exact datasets, audited end-to-end by `verify_all.py` with SHA-256 verification across all 39 tracked artifacts.

Tic-Tac-Toe & Window Sensitivity (Fig. 1). Mode 4 (1.892–103.002 ms) reproduces Escardó and Oliva's exact Section 6.2 full-root Tic-Tac-Toe benchmark under $(-1, 1)$: $9! = 362{,}880$ full plays collapse to 9,335 evaluated plays (38.873× speedup), returning exact draw value 0 and optimal play $[0, 4, 1, 2, 6, 3, 5, 7, 8]$. Along its prefixes, pruning achieves 32.780× at ply 1 (40,320 → 1,230), 11.915× at ply 2 (5,040 → 423), 10.588× at ply 3 (720 → 68), 4.286× at ply 4 (120 → 28), and certifies an X-win subgame $[0, 1, 4]$ (720 → 137, value +1 under $(-2, 2)$). On the Wikipedia 36-leaf tree, tightening the window from $(-1000, 1000)$ to $(3, 9)$ and $(5, 7)$ reduces evaluated leaves

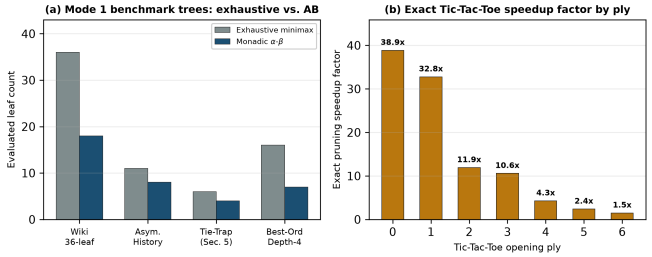

Figure 2: (a) Exhaustive versus monadic alpha-beta evaluated leaves across all four Mode 1 benchmark trees. (b) Exact speedup ratio $(9 - \text{ply})!/\text{ab_leaves}$ across Tic-Tac-Toe plies 0–6.

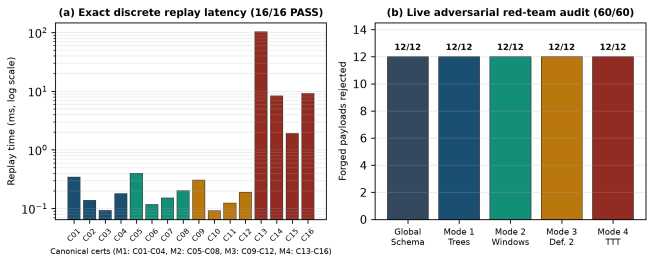

Figure 3: (a) Exact discrete replay latency (0.092–103.002 ms) across all 16 canonical `ALPHABETA-MONAD-CERT-1.0` certificates. (b) Live adversarial red-team audit rejecting 60/60 forged payloads across all five categories.

from 18 to 16 and 12, while fail-low $(6, 10)$ and fail-high $(0, 6)$ prune to 7 and 15 leaves.

Fixed-Tree Benchmarks & Optimal Plays (Fig. 2). Mode 1 (0.093–0.344 ms) certifies the exact Wikipedia tree ($36 \to 18$ leaves, value 6, play $[1, 0, 0, 0]$), an asymmetric history-dependent tree ($11 \to 8$, value 6), the Section 5 tie-breaking trap ($6 \to 4$, value 5, where first-element `pickMax` keeps $[0, 1]$ over a pruned tied bound), and a best-ordered depth-4 binary tree ($16 \to 7$, value 6). Mode 2 (0.118–0.398 ms) checks 17 windows covering all three Theorem 1 regimes across the four trees, and Mode 3 (0.092–0.305 ms) certifies ply-by-ply Definition 2 optimality for every extracted play.

Replay Latency and 60/60 Red-Team Audit (Fig. 3). All 16 canonical certificates replay deterministically in 0.092–103.002 ms (total 124.66 ms; 0.092–0.398 ms for Modes 1–3 and 1.892–103.002 ms for Mode 4). The live adversarial suite (`tests/test_suite.py`) executes 60 targeted forgeries—12 global schema/typing/depth/control-char attacks plus 12 domain attacks per mode (invalid tree nodes, duplicate moves, $\alpha \geq \beta$, tampered leaf traces, misclassified Theorem 1 regimes, tie-breaking trap swaps, non-optimal plays, illegal/duplicate Tic-Tac-Toe moves, and corrupted 9,335 leaf counts)—achieving **60/60 (100%)** rejection.

5 Conclusion

We delivered a standalone, zero-floating-point verification framework for Escardó and Oliva's monadic continuation alpha-beta theory [5], combining 18 zero-`sorry`, zero-`Classical.choice` Lean 4.35.0 theorems with the exact discrete verifier `ALPHABETA-MONAD-CERT-1.0`, 16 canonical certificates (including exact 362,880 → 9,335 Tic-Tac-Toe), and 60/60 live adversarial forgery rejections.

References

[1] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021.

[2] Martín Escardó and Paulo Oliva. Selection functions, bar recursion, and backward induction. *Mathematical Structures in Computer Science*, 20(2):127–168, 2010.

[3] Martín Escardó and Paulo Oliva. Sequential games and optimal strategies. *Proceedings of the Royal Society A*, 467:1519–1545, 2011.

[4] Martín Escardó and Paulo Oliva. Higher-order games with dependent types. *Theoretical Computer Science*, 974:114111, 2023.

[5] Martín Hötzel Escardó and Paulo Oliva. Alpha-beta pruning via monadic continuations. arXiv:2611.32320 [cs.AI], November 2026.

[6] Timothy P. Hart and Daniel J. Edwards. The tree prune (TP) algorithm. Technical Report AIM-030, MIT Artificial Intelligence Project, 1961.

[7] Johannes Hartmann. Alpha beta pruning with the selection monad. In *Trends in Functional Programming*, pages 378–391. Springer, 2026.

[8] Johannes Hartmann, Tom Schrijvers, and Jeremy Gibbons. Towards a more efficient selection monad. In *Trends in Functional Programming*, pages 55–74. Springer, 2025.

[9] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.