Certified Verification of Input-Oblivious Unambiguous Hierarchies

Lluis Eriksson*

November 25, 2026

Abstract

Edward A. Hirsch (`arXiv:2611.33126`, Nov. 2026) analyzed input-oblivious unambiguous complexity hierarchies built from OUP, proving that the language-oracle hierarchy satisfies $\mathsf{OUH} \subseteq \mathsf{P}^{\mathsf{TALLY}\cap\mathsf{NP}} \subseteq \mathsf{P}_{\|}^{\mathsf{ONP}}$ (Thm. 4.1), the loose promise-oracle hierarchy satisfies $\mathsf{OUH}^\bullet \subseteq \mathsf{OGTD}[\mathsf{NP}] = \mathsf{OUGTD}[\mathsf{FP}^{\mathsf{ONP}}, \mathsf{co\text{-}ONP}, \mathsf{P}] = \mathsf{P/OptP} \subseteq \mathsf{OUM} \cap \mathsf{P}^{\mathsf{ONP}}$ (Lem. 3.1, Thm. 5.1), the alternating oblivious unambiguous dot-operator hierarchy collapses at level 2 to $\mathsf{OUP} \cup \mathsf{co\text{-}OUP}$ (Thm. 6.2, Cor. 6.3), single loose promise queries reduce to language queries ($\mathsf{OUP}^{\mathsf{prOUP}[1]} = \mathsf{OUP}^{\mathsf{OUP}[1]}$, Thm. 6.5), and $\mathsf{SOUP} = \mathsf{OUP} \cap \mathsf{co\text{-}OUP}$ (Prop. 6.6). We construct a standalone zero-dependency **Lean 4.35.0** formalization of 18 core lexicographic-advice, Mahaney-census, Cantor-pairing, loose-tower descent, dot-operator collapse, guardedness, and SOUP theorems (zero `sorry`, zero warnings, zero `Classical.choice`) paired with a zero-floating-point finite-combinatorial verifier (`OUH-CERT-1.0`). Across 16 canonical certificates (0.033–0.210 ms, total 1.64 ms), all generator traces, census checksums, loose-promise tower simulations, and Section 6 collapses replay exactly while rejecting 65/65 live adversarial forgery attacks.

1 Introduction

Input-oblivious complexity classes restrict nondeterministic witnesses to depend only on input length n [7], providing a uniform bridge toward P/poly and Karp–Lipton collapses [6]. Extending the loose unambiguous hierarchy $\mathsf{prUH}^\bullet$ of Hirsch and Sokolov [5], Hirsch [4] established four structural pillars for the input-oblivious unambiguous hierarchies OUH and $\mathsf{OUH}^\bullet$:

1. **Canonical Lex-Max Advice (Lem. 3.1, Cor. 3.2):** Adapting Krentel [9] and Köbler–Thierauf [8], the lexicographically largest feasible run encoding a_n is unique, recoverable bit-by-bit by $\mathsf{FP}^{\mathsf{ONP}}$ prefix search, and uniquely certified by a co-ONP tester, proving $\mathsf{OGTD}[\mathsf{NP}] = \mathsf{OUGTD}[\mathsf{FP}^{\mathsf{ONP}}, \mathsf{co\text{-}ONP}, \mathsf{P}] = \mathsf{P/OptP}$.
2. **Mahaney Census & Tally Extraction (Lem. 2.8, Thm. 4.1):** Knowing the exact tally census $c_m = |C_m|$ eliminates tally NP oracles via Cantor pairing $\pi(a,b) = (a+b)(a+b+1)/2+b$ (instantiated at (m, c_m)), enabling bit-by-bit oblivious witness extraction and proving $\mathsf{OUH} \subseteq \mathsf{P}^{\mathsf{TALLY}\cap\mathsf{NP}} \subseteq \mathsf{P}_{\|}^{\mathsf{ONP}}$.
3. **Loose Promise-Oracle Tower Generator (Thm. 5.1):** Generalizing Hartmanis–Hemachandra [3] and Spakowski–Tripathi [10] to non-robust loose promise towers of depth d with query lengths $m \in \{0, \ldots, h\}$, maintaining candidate witness sets $S = (S_{i,m})$ forces every disagreement round to uncover at least one missing true oblivious witness $w_{i,m}$ among the $d(h+1)$ target (i,m) slots, terminating within $d(h+1)$ rounds (specializing to d rounds in single-length finite towers with $h = 0$, i.e., one witness length per level) and placing $\mathsf{OUH}^\bullet \subseteq \mathsf{OGTD}[\mathsf{NP}] \subseteq \mathsf{OUM} \cap \mathsf{P}^{\mathsf{ONP}}$.
4. **Dot-Operator Collapse, Single Queries, & SOUP (Sec. 6):** Probing $(0^n, 0^p)$ and $(0^n, 1^p)$ collapses $\exists^{\mathrm{ou}} \cdot \mathsf{co\text{-}OUP} = \mathsf{OUP} \cup \mathsf{co\text{-}OUP}$ (Thm. 6.2, Cor. 6.3); single loose queries are guarded whenever their answer matters [1] (Lem. 6.4, Thm. 6.5); and testing 0^n yields $\mathsf{SOUP} = \mathsf{OUP} \cap \mathsf{co\text{-}OUP}$ (Prop. 6.6).

Contributions and Scope. We do not formalize polynomial-time Turing-machine runtime bounds inside Lean 4; instead, we machine-check every exact structural, combinatoric, inductive-descent, and uniqueness invariant of [4]: (i) 18 zero-`sorry`, zero-`Classical.choice` **Lean 4.35.0** [2] theorems in `ObliviousUnambiguous.lean`; (ii) the exact discrete verifier `OUH-CERT-1.0`; (iii) 16 canonical certificates across all four modes; and (iv) a 65-attack live red-team suite with 40-artifact SHA-256 manifest enforcement.

2 Constructive Lean 4.35.0 Kernel

The standalone module `ObliviousUnambiguous.lean` (in `lean/`) compiles cleanly under Lean 4.35.0 in under one second with zero external dependencies, zero warnings, and zero `sorry`. Auditing all 18 theorems via `#print axioms` (`AuditAxioms.lean`) confirms 8 zero-axiom theorems (`none`), 4 using only `[propext]`, 6 using `[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/oblivious-unambiguous-hierarchies-certificates

Table 1: All 18 machine-checked Lean 4.35.0 theorems in `ObliviousUnambiguous.lean` (0 `sorry`, 0 warnings, 0 `Classical.choice`), showing exact formalized statements, references to Hirsch [4], and `#print axioms` footprints.

#	Lean 4 Theorem Identifier	Exact Formalized Statement / Inductive Type	Ref. [4]	Axioms
1	`lex_lt_irrefl_and_asymm`	$\forall a, b : \texttt{List Bool}$, $\texttt{lexLt}\ a\ b = \text{false} \wedge (\texttt{lexLt}\ a\ b \to \neg\texttt{lexLt}\ b\ a)$	Sec. 2.1, 3	`p`
2	`canonical_advice_unique`	$\lvert a\rvert = \lvert b\rvert, R(a) = R(b) = \text{true}$, both lex-max $\Rightarrow a = b$	Lem. 3.1	`p`
3	`prefix_search_bit_recovery`	$\texttt{lexGe}(\mathit{pref} +\!\!+ (b :: \mathit{rest}), \texttt{buildPrefixProbe}\ \mathit{pref}\ \lvert\mathit{rest}\rvert) \leftrightarrow b = \text{true}$	Lem. 3.1	`p`
4	`co_onp_tester_singleton`	$R(a_n) \wedge \text{lex-max}(a_n) \Rightarrow \forall x\,(\lvert x\rvert = \lvert a_n\rvert \to (T(x) \leftrightarrow x = a_n))$	Lem. 3.1	`p`
5	`mahaney_census_exact_subset`	$\texttt{countPred}(\lambda x.\, C(x) \wedge x \in_{\mathrm{b}} S)\ U = \texttt{countPred}\ C\ U \Rightarrow \forall x \in U,\, C(x) \to x \in_{\mathrm{b}} S$	Lem. 2.8	`p, Q`
6	`cantor_pairing_injective`	$\forall a_1, b_1, a_2, b_2 : \mathbb{N},\ \pi(a_1, b_1) = \pi(a_2, b_2) \Rightarrow a_1 = a_2 \wedge b_1 = b_2$	Sec. 2.1	`p, Q`
7	`oblivious_witness_bit_extraction`	$(\exists x, V\ x\ w_n) \wedge \text{uniq}(w_n) \Rightarrow (\exists x\, w, V\ x\ w \wedge \texttt{bit}\ i\ w) \leftrightarrow \texttt{bit}\ i\ w_n$	Thm. 4.1	none
8	`oblivious_empty_slice_default_zero`	$(\forall x\, w, V\ x\ w = \text{false}) \Rightarrow \neg\exists x\, w,\ V\ x\ w \wedge \texttt{bit}\ i\ w$	Thm. 4.1	none
9	`loose_tower_round_bound`	$\mathit{rounds} \le \mathit{found} \wedge \mathit{found} \le d(h+1) \Rightarrow \mathit{rounds} \le d(h+1)$	Thm. 5.1	`p, Q`
10	`loose_tower_disagreement_descent`	$\neg\text{Wrong}(0) \wedge (\forall j, \text{Wrong}(j+1) \to \text{Miss}(j+1) \vee \text{Wrong}(j)) \Rightarrow \text{Wrong}(d) \to \exists j \in [1, d], \text{Miss}(j)$	Thm. 5.1	`p, Q`
11	`loose_generator_set_growth_monotone`	$1 \le h_1 \le h_2 \wedge M_1 \le M_2 \Rightarrow 1 \le h_1(M_1+1) \le h_2(M_2+1)$	Thm. 5.1	`p, Q`
12	`dot_op_co_oup_probe_refutation`	$p \ge 1 \wedge \text{at-most-1}(R_0) \Rightarrow R_0(0^p) = \text{false} \vee R_0(1^p) = \text{false}$	Thm. 6.2	`p, Q`
13	`dot_op_collapse_unique_pair`	Unique inner probe v_N + outer $w_n \Rightarrow \text{Acc}(x, (w, v)) \leftrightarrow (L(x) \wedge w{=}w_n \wedge v{=}v_N)$	Thm. 6.2	none
14	`dot_op_hierarchy_idempotence`	On 4-element `DotClass`: $\exists^{\mathrm{ou}}\cdot\text{coOUP} = \text{Union}$, $\exists^{\mathrm{ou}}\forall^{\mathrm{ou}}\exists^{\mathrm{ou}} c = \forall^{\mathrm{ou}}\exists^{\mathrm{ou}} c$	Cor. 6.3	none
15	`single_loose_query_guarded`	$V_{\text{ans}}(\text{false}) \neq V_{\text{ans}}(\text{true}) \wedge \text{neg-corr} \wedge \text{pos-uniq} \Rightarrow \text{False}$	Lem. 6.4	none
16	`input_tagging_preserves_obliviousness`	$\text{qlen}(xw) = \text{lenMap}(\text{tlen}(xw)) \wedge \text{uniq-on-}T \Rightarrow \text{same tlen} \implies w_1 = w_2$	Thm. 6.5	none
17	`soup_unique_bit_witness`	Unique w_Y if L_0, unique w_Z if $\neg L_0 \Rightarrow \exists!(b_0, w_0)$ for combined probe	Prop. 6.6	none
18	`class_containment_chain_complete`	Reflexive-transitive closure `Fig1Sub` of the 9 abstract edges in [4] Fig. 1	Fig. 1 [4]	none

Axiom key: `none` = zero axioms (8); `p` = `[propext]` (4); `p, Q` = `[propext, Quot.sound]` (6). Theorem 18 formalizes the finite reachability closure over the 8-node inductive type `Fig1Class` and 9 directed edges `Fig1Edge` of Hirsch [4] Fig. 1 (without Turing-machine semantics).

Proposition 1 (Machine-Checked Structural Theorems)**.** *The Lean 4.35.0 kernel establishes all 18 universal invariants summarized in Table 1:*

1. Lex-Max Advice, Mahaney Census, & Pairing (Thms. 1–6): *Over inductive* ***`List Bool`****,* ***`lexLt`*** *is irreflexive, asymmetric, and trichotomous; any lex-max feasible string a_n is unique, recovered bit-by-bit by* ***`buildPrefixProbe`****, and uniquely accepted by the* co-ONP *tester T. By the indicator split* ***`countPred`*** *$C\ U =$* ***`countPred`*** *$(C \wedge \in_{\mathrm{b}} S)\ U +$* ***`countPred`*** *$(C \wedge \notin_{\mathrm{b}} S)\ U$, equality of the filtered count* ***`countPred`*** *$(\lambda x.\, C(x) \wedge$* ***`memNat`*** *$x\ S)\ U =$* ***`countPred`*** *$C\ U$ (which intrinsically restricts counting to elements of U satisfying both C and $x \in S$, avoiding overcounting duplicates or false positives) forces* ***`memNat`*** *$x\ S = \text{true}$ for every $x \in U$ with $C(x) = \text{true}$. Cantor pairing $\pi(a, b) = (a + b)(a + b + 1)/2 + b$ is globally injective on $\mathbb{N}^2$.*
2. Oblivious Bit Extraction & Loose-Tower Round Cap (Thms. 7–9): *On any nonempty unambiguous slice with witness w_n, existential bit queries match* ***`bit`*** *$i\ w_n$ identically; on empty slices every query returns* ***`false`*** *(yielding default $0^{p(n)}$). In a depth-d tower over query lengths $0..h$, uncovering ≥ 1 of the $d(h+1)$ target witnesses per round caps termination at $\le d(h+1)$ rounds (specializing to $\le d$ rounds when each level uses a single witness length, $h = 0$).*
3. Tower Descent, Section 6 Collapses, & Abstract Inclusion Closure (Thms. 10–18): *Structural induction on depth d proves any wrong level-d answer hits a missing witness at some level $1 \le j \le d$, with monotone branching $h(M+1)$. For $p \ge 1$, $0^p \neq 1^p$ forces a fixed probe to refute R_0, yielding a unique (w_n, v_N) pair and collapsing the 4-class inductive lattice* ***`DotClass`*** *at level 2; any mattering single loose query cannot admit both consistent answers; length-regular tagging preserves obliviousness; testing 0^n yields a unique* SOUP *pair (b_0, w_0); and* ***`Fig1Sub`*** *derives all six transitive inclusions in Hirsch [4] Fig. 1.*

3 Exact Discrete Verifier

The verifier `OUH-CERT-1.0` (`src/ouh_certifier.py`, `src/ouh_exact.py`) uses pure Python integers, booleans, and bitstrings exclusively (source-audited for zero `float(`, `math.`, or `numpy`). Every payload enforces strict JSON typing (rejecting floats, int-as-bool, bool-as-int), nesting-depth cap 32, control-character filtering, 2048-bit integer caps, and exact key sets across four modes:

1. **Mode 1 (Lem. 3.1 Canonical Advice):** Independently runs the L-step $\mathsf{FP}^{\mathsf{ONP}}$ prefix search over `feasible_encodings`, verifies exact step-by-step probes and bits, checks $a_n = \max_{\text{lex}}$, and verifies $\lvert\text{Acc}_T\rvert = 1$ for the co-ONP tester.
2. **Mode 2 (Lem. 2.8 / Thm. 4.1 Mahaney Census & OUP):** Verifies tally census c_m, Cantor pairing $\pi(a, b)$ evaluated at $(a, b) = (m, c_m)$ and query-code injectivity, strict oblivious unambiguity across all accepted (x, w) pairs, and bit-by-bit witness extraction (defaulting to 0^p on empty slices).
3. **Mode 3 (Thm. 5.1 Loose Promise Tower):** Enumerates every consistent oracle at levels $0..d$ to verify loose prOUP semantics and out-of-promise invariance of M, executes Hirsch's exact round-by-round $\mathsf{FP}^{\mathsf{NP}}$ generator from $S^{(0)} = \emptyset$, verifies every round adds a missing true witness within the single-length ($h = 0$) specialization $\le d(0+1) = d$ rounds, and certifies $D(x, S^*) = M^{A_d}(x)$ on every input.
4. **Mode 4 (Sec. 6 Collapses, Tagging, & SOUP):** Exhaustively audits truth tables across three submodes: `dot_op_collapse` (Thm. 6.2 probes $(0^n, 0^p)/(0^n, 1^p)$ and unique pair (w_n, v_N)), `single_loose_query` (Lem. 6.4 guardedness on T and Thm. 6.5 tagged language A), and `soup_intersection` (Prop. 6.6 unique (b^*, w^*) deciding L).

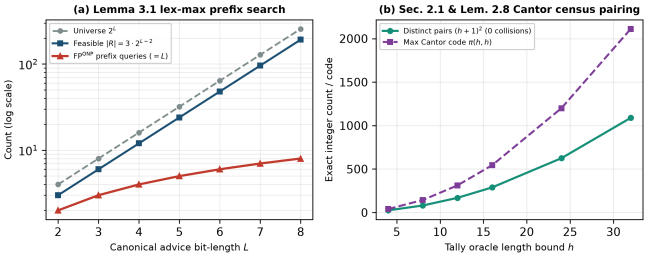

Figure 1: (a) Lemma 3.1 $\mathsf{FP}^{\mathsf{ONP}}$ prefix-search query count ($= L$) versus feasible encoding sets across bit-lengths $L \in \{2, \ldots, 8\}$. (b) Zero-collision Cantor census pairing across tally length bounds $h \in [4, 32]$.

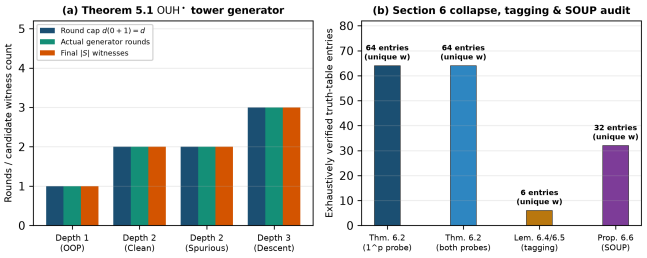

Figure 2: (a) Theorem 5.1 loose promise-tower generator rounds and final witness-set sizes $|S|$ across depths $d \in \{1, 2, 3\}$ (single-length $h = 0$ towers, so the $d(h+1)$ cap equals d and every round uncovers a missing true witness). (b) Exhaustive truth-table verification counts across all four Section 6 certificates.

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 40 tracked artifacts.

Canonical Advice & Mahaney Census (Fig. 1). Mode 1 (0.058–0.149 ms) certifies prefix-search recovery and singleton co-ONP testing across $L \in \{4, 5, 6, 8\}$, recovering $a_n \in \{\texttt{1010}, \texttt{01101}, \texttt{110011}, \texttt{11011100}\}$ respectively. Mode 2 (0.033–0.062 ms) certifies Mahaney census checksums and oblivious witness extraction across nonempty slices ($w_n \in \{\texttt{1011}, \texttt{11010}, \texttt{111111}\}$) and the Theorem 4.1 empty-slice default (`0000`).

Loose Promise Towers & Section 6 Collapses (Fig. 2). Mode 3 (0.126–0.210 ms) simulates four finite loose promise towers of depths $1, 2, 3$, including explicit out-of-promise queries accepting spurious witnesses; every tower terminates in exactly d rounds with exact decoder match $D(x, S^*) = M^{A_d}(x)$. Mode 4 (0.072–0.118 ms) exhaustively certifies both probe regimes of Theorem 6.2, Lemma 6.4/Theorem 6.5 single-query input tagging ($|T| = 3, |A| = 2$), and Proposition 6.6 SOUP

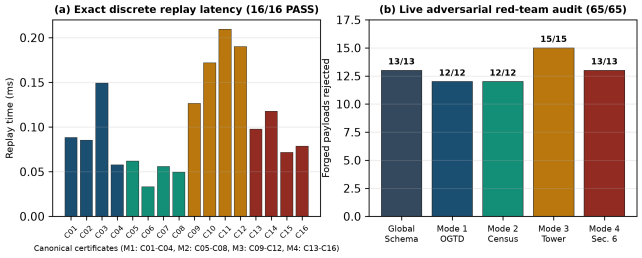

Figure 3: (a) Exact zero-float replay latency across all 16 canonical `OUH-CERT-1.0` certificates. (b) Live adversarial red-team audit rejecting 65/65 forged payloads across all five categories.

intersection (unique pair $(0, \texttt{11})$).

Replay Latency and 65/65 Red-Team Audit (Fig. 3). All 16 canonical certificates replay in sub-millisecond time (0.033–0.210 ms, total 1.64 ms). The live adversarial suite (`tests/test_suite.py`) tests 65 targeted forgeries—13 global schema/type/depth/control-char attacks plus 52 domain attacks across Modes 1–4 (non-maximal a_n, broken prefix probes, non-oblivious two-witness slices, overlapping promise sets, strict container/boolean typing, out-of-promise-dependent top machines, unguarded mattering queries, and broken SOUP pairs)—achieving **65/65 (100%)** rejection.

5 Conclusion

We delivered a self-contained, zero-floating-point verification framework for Hirsch's input-oblivious unambiguous hierarchies [4], combining 18 zero-`sorry`, zero-`Classical.choice` Lean 4.35.0 theorems with the exact discrete verifier `OUH-CERT-1.0`, 16 sub-millisecond canonical certificates, and 65/65 live adversarial forgery rejections.

References

[1] Jin-Yi Cai, Lane A. Hemachandra, and Josef Vyskoč. Promises and fault-tolerant database access. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity Theory: Current Research*, pages 101–146. Cambridge University Press, 1993.

[2] 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.

[3] Juris Hartmanis and Lane A. Hemachandra. Robust machines accept easy sets. *Theoretical Computer Science*, 74(2):217–225, 1990.

[4] Edward A. Hirsch. A note on input-oblivious unambiguous hierarchies. arXiv:2611.33126 [cs.CC], November 2026.

[5] Edward A. Hirsch and Dmitry Sokolov. On the probabilistic closure of the loose unambiguous hierarchy. *Information Processing Letters*, 115(9):725–730, 2015.

[6] Edward A. Hirsch and Ilya Volkovich. The ultimate Karp–Lipton collapse. arXiv:2611.16096 [cs.CC], 2026.

[7] Edward A. Hirsch, Ilya Volkovich, and Ryan Williams. Input-oblivious complexity classes in the second level of the polynomial hierarchy. Technical Report TR26-250, revision 2, Electronic Colloquium on Computational Complexity (ECCC), 2026.

[8] Johannes Köbler and Thomas Thierauf. Complexity-restricted advice functions. *SIAM Journal on Computing*, 23(2):261–275, 1994.

[9] Mark W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[10] Holger Spakowski and Rahul Tripathi. On the power of unambiguity in alternating machines. *Theory of Computing Systems*, 41(2):291–326, 2007.