EXECUTION IS EVIDENCE: CERTIFIED SEMANTIC PROGRAM IDENTIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A program can be identified by what it does even when its source syntax is unrecoverable. We study semantic program identification from partial executions with independently checkable correctness. First, a routing-and-witness condition gives an exact family with $d + r$ adaptive versus $d + 2^d r$ fixed queries, and a separate nonlinear repeated-operator language admits exact worst-case optima $6 < 9$, certified by a complete decision tree and an independently checked adversary DAG over all 10,974 behaviors. Second, for a fixed accumulator language we prove a width cutoff: two programs of length at most L agree at all bit widths iff they agree at width $2L + 1$. This turns large-width uniqueness into a cutoff-width symbolic refutation without enumerating the input domain or the behavioral quotient. On eight adapted 32-bit Hacker's Delight specifications, all 120 paired runs return independently checked programs. With the same cutoff reduction, adaptive witness queries use 3.125 replies and 0.304 seconds on average versus 4.375 replies and 0.335 seconds for structured fixed probes; holding the word and transcript fixed isolates a $4.47\times$ reduction in terminal certification time. Under the same per-call conflict budget, the cutoff also raises horizon-six certification coverage from $1/8$ at native width to $8/8$. Together, the results connect adaptive observation design with proof-carrying semantic identification under explicit, executable contracts.

1 INTRODUCTION

Many program induction problems contain unavoidable syntactic aliases: identities, repeated operators, algebraic cancellations, or task-restricted equivalences map different strings to the same transformation. When observations are executions, literal-string recovery can therefore ask for information that the interaction cannot reveal. We instead take the estimand to be the *program behavior*: the equivalence class of all legal programs that agree on the declared task domain.

This shift raises two questions. First, when can choosing the next execution from previous replies provably reduce the number of observations? Second, once a behavior has been isolated, can correctness be certified without explicitly materializing every behavior or every input? We give complementary answers.

For observation complexity, we separate *routing* from *simultaneous coverage*. A short adaptive transcript can reveal which local witness set matters, whereas a fixed design must cover all disjoint witness families at once. This yields an exact binary family with $q^*_{\mathrm{ad}} = d+r < d+2^d r = q^*_{\mathrm{fixed}}$. In a different nonlinear language containing repeated operators, noncommuting maps, a nonlinear Toffoli map, and a non-injective reset, complete enumeration gives 10,974 distinct behaviors. Independent upper and lower certificates establish the exact gap

$$q^*_{\mathrm{ad}} = 6 < 9 = q^*_{\mathrm{fixed}}.$$

For verification, we move beyond explicit behavior tables. A bounded bitvector program induces a symbolic execution relation. We prove that, for a fixed eight-operation accumulator bank, any two programs of length at most L that differ at some width already differ by width $2L + 1$. Consequently a checked refutation at width $2L + 1$ certifies equivalence at a much larger deployment width. This makes the encoded uniqueness problem depend on program length rather than deployment width for this language; solver runtime and proof length remain empirical quantities.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

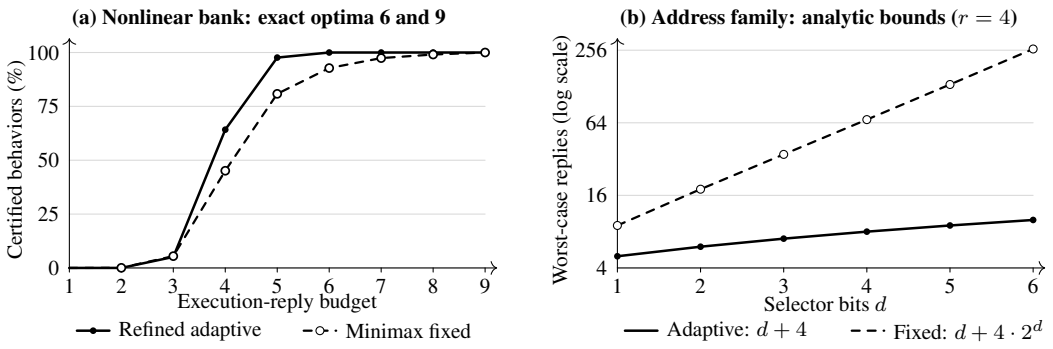


Figure 1: **Exact observation separations. (a)** Certified fraction of all 10,974 nonlinear behaviors versus query budget. The adaptive policy reaches all behaviors by six replies; every fixed design needs nine in the worst case. **(b)** The analytic address family at $r = 4$: adaptive cost is $d + 4$, while fixed cost is $d + 4 \cdot 2^d$. The right panel is theorem-derived, not an extrapolation of the finite bank.

The paper's contributions are: **(i) certified adaptive separations**, including a structural sufficient condition, an exact compositional family, and the independently checked nonlinear optimum $6 < 9$; **(ii) a language-uniform verification cutoff**, reducing full-width uniqueness to width $2L + 1$ for a bounded accumulator language; and **(iii) matched proof-carrying experiments** that measure queries, search, proof production, and independent checking on the same tasks, including identical-transcript ablations and longer-horizon failure cases.

2 Semantic Program Identification

A word $w = (w_1, \ldots, w_L)$ over primitives $\{S_1, \ldots, S_M\}$ induces $f_w = S_{w_L} \circ \cdots \circ S_{w_1}$. For task domain $\mathcal{X}_{\text{task}}$ and legal language $\mathcal{W}$, define

$$w \sim w' \iff f_w(x) = f_{w'}(x) \ \ \forall x \in \mathcal{X}_{\text{task}}, \qquad \mathcal{Q} = \mathcal{W}/\!\sim . \tag{1}$$

The target is $[W] \in \mathcal{Q}$. At round t, a policy chooses X_t from public metadata and the previous replies, then observes $Y_t = f_W(X_t)$.

Proposition 1 (Syntax indistinguishability)**.** *If $w \sim w'$ and public metadata are identical, every execution-only adaptive protocol has the same transcript distribution under w and w'. Under a uniform prior over a aliases of one behavior, exact source-string recovery is at most $1/a$.*

Conditioning on policy randomness, equivalent programs produce the same first reply and therefore the same later queries and replies by induction. For the complete transcript Z,

$$I(W; Z \mid U) = I([W]; Z \mid U), \qquad H(W \mid U) = H([W] \mid U) + H(W \mid [W], U).$$

The final term is alias entropy that execution cannot remove.

For K equiprobable semantic classes and reply alphabet size B, any zero-error protocol using at most q replies has at most B^q leaves, hence $q \geq \lceil \log_B K \rceil$. For variable-length protocols, $\mathbb{E}q \geq H([W] \mid U)/\log_2 B$. When a syntax prior is used, alias masses must be aggregated,

$$\pi(q \mid U) = \sum_{w:[w]=q} p(w \mid U), \qquad \pi_t(q) \propto \pi(q \mid U) \prod_{i \leq t} \mathbf{1}\{f_q(X_i) = Y_i\}. \tag{2}$$

The behavioral quotient is also the minimal exact execution state under the usual right-congruence condition:

Theorem 1 (Minimal exact behavioral state)**.** *Every exact execution summary h refines $\mathcal{Q}$, so $|h(\mathcal{W})| \geq |\mathcal{Q}|$. If*

$$w \sim w' \Longrightarrow \begin{cases} wj \in \mathcal{W} \iff w'j \in \mathcal{W}, \\ wj \sim w'j \quad \textit{whenever these extensions are legal}, \end{cases} \tag{3}$$

then $h_*(w) = [w]$ attains this bound, up to state renaming.

This fixes the semantic target; grammar legality may require additional control state. The full factorization proof, bounded-language caveat, and reuse corollaries are in Appendix A.

3 Certified Adaptive Separations

3.1 Routing versus simultaneous witness coverage

For distinct behaviors define $\Delta(f, g) = \{x : f(x) \neq g(x)\}$. A fixed design must hit every distinguishing set. Adaptivity can be cheaper when replies first reveal which witness family matters.

Theorem 2 (Routing and disjoint witness supports)**.** *Partition $\mathcal{Q}$ into fibers $\mathcal{Q}_s$. Suppose s is identified within a probes and fiber $\mathcal{Q}_s$ within b_s further probes. Let $U_0, \ldots, U_m$ be disjoint probe sets; in U_i, let τ_i be the minimum hitting-set size of a family of full distinguishing sets from legal behavior pairs. Then*

$$q^*_{\mathrm{ad}} \leq a + \max_s b_s, \qquad q^*_{\mathrm{fixed}} \geq \sum_{i=0}^{m} \tau_i. \tag{4}$$

The adaptive upper bound routes first and solves only the selected fiber. A fixed design must hit every witness family, and disjoint supports make those requirements additive.

Proposition 2 (Exact compositional family)**.** *For every $d, r \geq 1$, a finite transformation DSL has 2^{d+r} behaviors $(s, b) \in \{0,1\}^d \times \{0,1\}^r$ with selector probes c_i and private probes $p_{u,j}$ satisfying*

$$f_{s,b}(c_i) = s_i, \qquad f_{s,b}(p_{u,j}) = \mathbf{1}\{u = s\} b_j. \tag{5}$$

All replies are binary and

$$\boxed{q^*_{\mathrm{ad}} = d + r < d + 2^d r = q^*_{\mathrm{fixed}}.} \tag{6}$$

Under the uniform-class prior, entropy-greedy acquisition attains $d + r$.

Selectors reveal s; the r private probes at address s reveal b. Every selector and every private probe is the unique witness for a legal behavior pair, forcing the fixed cost. Appendix B gives the total-map realization, repeated-operator grammar, aliases, and checker.

3.2 An exact nonlinear separation

The second language acts on four-bit states. Its five primitives are NOT, controlled NOT, Toffoli, cyclic bit rotation, and reset of bit 0. We include every word of length zero through eight. The 488,281 legal strings induce 10,974 distinct full-domain transformations; 7,955 are nonlinear, 6,794 are non-injective, and 10,888 canonical representatives repeat an operator.

Proposition 3 (Certified finite-language separation)**.** *For zero-error identification of these 10,974 behaviors using exact 16-valued execution replies,*

$$\boxed{q^*_{\mathrm{ad}} = 6 < 9 = q^*_{\mathrm{fixed}}.} \tag{7}$$

A depth-six response-adaptive tree identifies every behavior and all 10,974 terminal transcripts pass an independent checker that rebuilds the language by raw syntax enumeration. For the adaptive lower bound, an independently verified adversary DAG checks 12,244 reachable obstruction nodes and 130,795 query obligations, excluding depth five. For the fixed lower bound, every eight-input subset has a colliding legal pair; the nine-input set $\{2, 3, 4, 6, 9, 10, 13, 14, 15\}$ is injective. The complete adaptive and fixed lower-bound artifacts and validators are in Appendices E and D.

The average-query comparison remains nontrivial under stronger baselines. Optimizing over all 2^{16} fixed subsets and all fixed orders gives mean 4.780 replies under a uniform behavior prior versus 4.331 for the refined adaptive tree, a 9.40% reduction. Under a uniform syntax prior, alias-aware entropy uses 4.401 replies versus the reoptimized fixed optimum 4.572. Appendix I gives the complete prior-sensitive and per-budget frontier.

The next query genuinely depends on the reply. The nonlinear tree is not a fixed order with variable stopping time. Its root queries $x = 5$; replies $0, 1, 2$ lead to next inputs $2, 3, 14$, respectively, and these successors are unique entropy maximizers in their corresponding posterior sets. Across the unmodified entropy tree, 568 internal nodes select different successor queries for different replies. Figure 2 shows three representative branches.

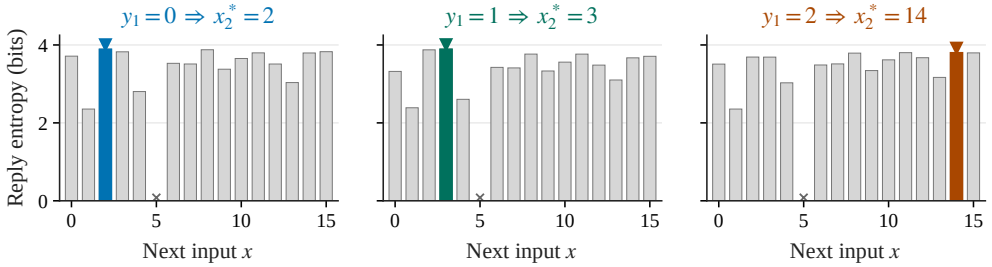


Figure 2: **Response-dependent acquisition in the nonlinear language.** After the common first query $x_1 = 5$, three different replies induce three different next queries. Bars are exact reply entropies over the remaining behavior set; arrows mark the selected unique maximizer. This distinguishes response adaptivity from a fixed query order with early stopping.

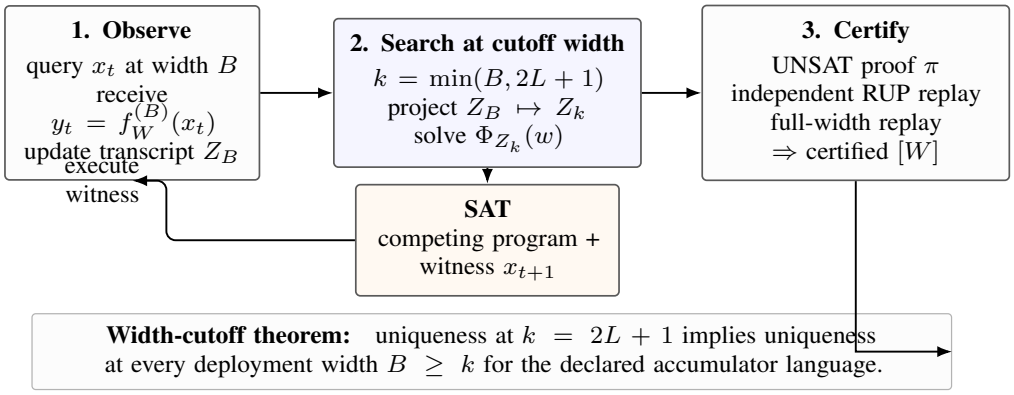


Figure 3: **Proof-carrying semantic identification.** Queries are executed at deployment width B. Search and terminal uniqueness can operate on the low $k = \min(B, 2L + 1)$ bits. SAT provides the next witness; UNSAT produces a refutation that is replayed independently, while the returned program is checked against the full-width transcript.

4 Proof-Carrying Identification Beyond Explicit Tables

Let $E_\theta(x)$ encode execution of a legal bounded program and let $C_Z(\theta) = \mathrm{Legal}(\theta) \wedge \bigwedge_{(x,y)\in Z} E_\theta(x) = y$. For a legal transcript-consistent representative w, define the disagreement miter

$$\Phi_Z(w) = C_Z(\theta) \wedge \big(E_\theta(x) \neq E_w(x)\big). \tag{8}$$

SAT returns an alternative program and a distinguishing input; UNSAT proves that every legal program consistent with the transcript has the same behavior. We export a propositional refutation and independently replay it instead of trusting the solver status.

Proposition 4 (Proof-carrying identification)**.** *If the encoding is exact, a legal transcript-consistent w together with a checked refutation of $\Phi_Z(w)$ certifies the unique behavior, conditional on authentic replies and target membership. The construction need not enumerate the input domain or $\mathcal{Q}$.*

Acquisition and checking loop. Starting from the current transcript Z, solve the disagreement miter. A SAT model contains both a competing legal program and an input on which it differs from the current representative; executing that input adds one new exact reply to Z. When the miter becomes UNSAT, the prover emits a clausal refutation. The checker reconstructs the public execution relation, replays the refutation, and re-executes the returned word on every observed input. Search may reuse the same execution constraints across rounds, while proof replay remains an independent final operation.

A language-uniform width cutoff. Consider an accumulator initialized to immutable input x, with primitives identity, NOT, negation, increment, decrement, and AND/OR/XOR with x. Arithmetic is modular at width B, repetitions are legal, and program length is at most L.

Theorem 3 (Linear width cutoff)**.** *Let $a(w)$ count negation, increment, and decrement occurrences. If two words u, v differ at any width, they have a distinguishing input of width at most $a(u)+a(v)+1$. Hence, for $|u|, |v| \le L$ and $B \ge k = 2L+1$,*

$$f_u^{(B)} = f_v^{(B)} \iff f_u^{(k)} = f_v^{(k)} \iff \forall b \ge 1,\ f_u^{(b)} = f_v^{(b)}. \tag{9}$$

Read bits from least to most significant. Each negation, increment, or decrement contributes one carry/borrow flag initialized to one; along the stream it can turn off but never turn on again. In the product transducer for u and v, every non-self transition decreases the number of live flags. A shortest distinguishing word therefore has at most $a(u) + a(v)$ such transitions before the mismatch. Low-bit projection commutes with every primitive, so the same witness remains valid at every larger width. Appendix M gives the complete transducer proof and certificate-binding theorem.

Set $k = \min(B, 2L+1)$. The version-space miter can use the low k bits of each actual B-bit reply, while the final representative is replayed against the full transcript. The symbolic formula is $O((|Z|+1)Lk)$ rather than $O((|Z|+1)LB)$ in width. This is the mechanism that makes large deployment widths inexpensive to certify for this language.

Corollary 1 (Deployment-width-independent verification encoding)**.** *For every deployment width $B \ge 2L+1$, the uniqueness miter for a transcript Z has $O((|Z|+1)L^2)$ variables and clauses after cutoff projection, independent of B. A checked UNSAT refutation of that miter, together with full-width replay of the returned legal program on Z, certifies the deployed B-bit behavior.*

The corollary bounds the encoded verification problem, not SAT runtime or proof length. Its value is that execution, search, and verification share one length-controlled semantic interface even when deployment width grows.

5 Evaluation: Queries, Proof Cost, and Scaling

We adapt the first eight unary Hacker's Delight specifications from the public SyGuS 2017 repository to the common accumulator bank above while retaining their 32-bit oracle semantics. Inference receives neither the reference expression nor a task identifier. Every reported success returns a legal program, a transcript, and a refutation replayed by an independent RUP checker; an evaluator-only Z3 encoding separately cross-checks the returned behavior.

Table 1: **Same 32-bit oracle and legal-program output; different proof widths.** Eight external specifications, three timing repeats; every row certifies 24/24. Total time includes encoding, search, proof production, serialization and independent checking, with no added observation delay. Cutoff rows conservatively include two narrow-proof replays. Proof size is measured in decimal kB.

Policy	Proof bits	Replies	Total (s)	Proof (kB)
Native witness	32	2.375	0.808	363.9
Native fixed edges	32	4.375	1.069	498.5
Cutoff witness	9	3.125	0.304	79.6
Cutoff fixed edges	9	4.375	0.335	111.6
Cutoff with rebuilding	9	3.125	0.304	71.5

At $L = 4$, all five policies certify all eight tasks in each of three randomized paired timing repeats (120 runs total). Cutoff witness acquisition uses 3.125 replies and 0.304 seconds on average, including proof production and checking. Structured fixed probes under the same cutoff use 4.375 replies and 0.335 seconds, so adaptivity reduces both observations (28.6%) and measured total cost (9.25%) under a common proof reduction. Native-width witness search uses fewer replies, 2.375, but 0.808 seconds because its refutations are larger. Incremental and restarted cutoff solvers have essentially identical mean time, showing that the dominant gain here comes from the width reduction rather than solver-history reuse.

A controlled terminal-proof ablation freezes the *same returned word and same full transcript* and changes only proof width. Across 48 runs, width nine reduces encoding–proof–checking time from 0.6705 to 0.1501 seconds ($4.47\times$) and proof size from 445.6 to 79.6 kB. This isolates the verification effect from acquisition.

The cutoff also changes the reachable certification regime. At $L = 6$ and the same per-call conflict cap, native-width search certifies 1/8 tasks whereas the 13-bit cutoff certifies 8/8. At $L = 8$, the 17-bit cutoff returns UNKNOWN on all eight tasks, so the present solver does not yet provide long-horizon scalability. Lifting the same eight expressions to 256-bit deployment while keeping $L = 4$ reuses the nine-bit proof hashes and certifies all tasks, demonstrating width transfer independently of the 32-bit setting. Table 2 summarizes the observed boundary; full task-level records, proof hashes, mutation tests, and development deviations are included in the appendix.

Table 2: **Certification boundary across program horizon and deployment width.** Counts are distinct specifications certified under the final protocol. For $L = 6$, native and cutoff use the same per-call conflict cap. A dash denotes a setting not rerun in the final paired protocol.

Deployment width B	Horizon L	Native width	Cutoff $k = \min(B, 2L + 1)$
32	4	8/8	8/8 (9 bits)
32	6	1/8	8/8 (13 bits)
32	8	–	0/8 (17 bits)
256	4	–	8/8 (9 bits)

External scope test on graph reasoning. To test the observation principle outside the bitvector DSL, we freeze the UltraQuery executor (Galkin et al., 2024) and form held-out conditional collisions whose candidates differ only through outcome content. Across nine $K = 24$ groups and 216 balanced tasks, exact symbolic membership trees have worst-case depth five in every group, attaining $\lceil \log_2 24 \rceil$. A neural information-gain policy recovers .801 of targets with five bits and .991 with eight, compared with .079 for eight random bits. Figure 4 shows one complete held-out tree; Appendix K contains the split and control details.

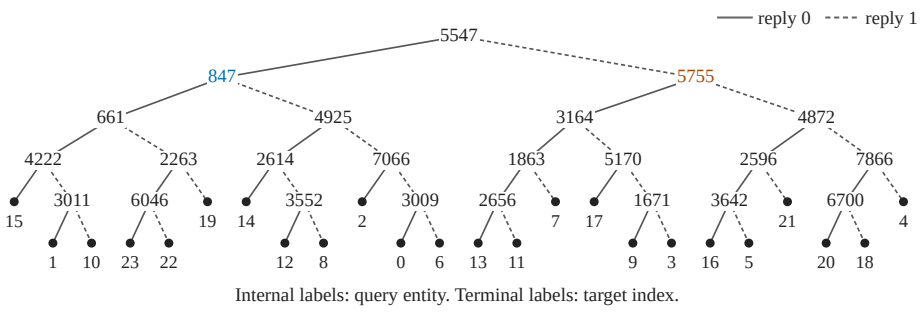


Figure 4: **A complete held-out $K = 24$ membership policy.** All 23 internal nodes and 24 leaves are shown. Solid and dashed edges denote the two reply bits. The exact symbolic tree has depth five, matching the information-theoretic lower bound for 24 behaviors under this probe set.

Finite-bank observation–cost frontier. For the 10,974-behavior nonlinear language, the adaptive tree uses fewer queries than every globally mean-optimal fixed order (4.331 versus 4.780), but a 16-query full-domain readout is computationally cheaper when oracle replies are local and essentially free. Figure 5 places these policies on the same 100%-certified behavior task. Thus the observation optimum and the zero-latency runtime optimum are different design points.

With external per-reply latency λ, write $C_m(Q, \lambda) = S_m + Q(\bar{t}_m + \lambda \bar{q}_m)$. For $Q = 1000$, the measured paired-repeat break-even latency is 0.463 ms per reply against the global mean-optimal fixed policy and 0.150 ms against full readout. Above those regimes, fewer observations repay the

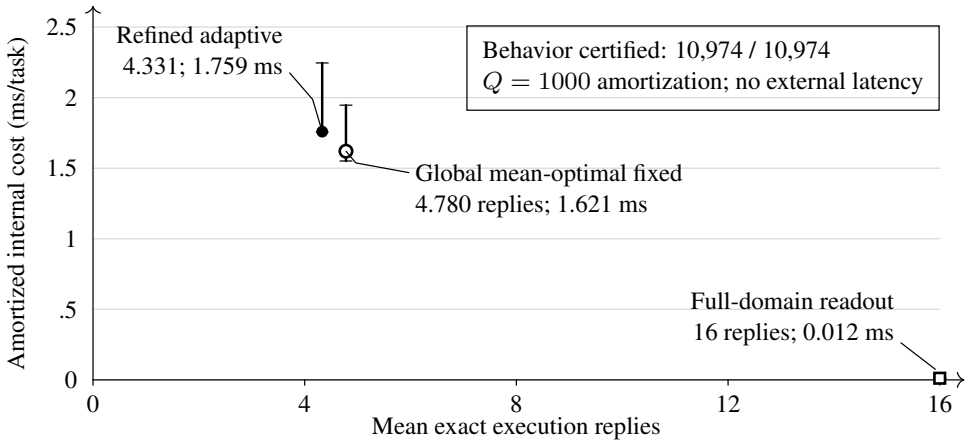


Figure 5: **Observation–internal-cost tradeoff at 100% certified coverage.** The horizontal axis is the exact mean execution-reply count; the vertical axis is amortized internal cost per task for $Q = 1000$, including setup and independent checking. Bars are observed ranges across three paired repeats. The displayed points are implemented policies, not an interpolation or a claimed globally optimal runtime frontier.

additional internal work; below them, full readout or a fixed design can be cheaper. Complete paired timings and output-contract checks are included in the appendix.

6 Related Work and Scope

Semantic normalization and observational equivalence are established tools in program synthesis (Smith & Albarghouthi, 2019; Feser et al., 2023). Active disambiguation, generalized binary search, and oracle-guided synthesis already use informative queries or counterexamples (Tiwari et al., 2020; Nowak, 2011; Golovin & Krause, 2011; Jha et al., 2010; Barnaby et al., 2026). Our observation contribution is the exact identification separation under an explicit program-behavior objective, including independently checkable upper and lower certificates; test cover and direct-sum principles themselves are classical (Berman et al., 2004; Jain et al., 2010). SemRE proves strong adaptivity phenomena for regular-expression membership rather than identification of the complete unknown behavior (Li et al., 2026).

Width-independent bitvector verification is also an active line of work (Niemetz et al., 2021; Bhat et al., 2025). Our verification result is narrower and more explicit: for one bounded accumulator language, the cutoff $2L + 1$ is uniform over the unknown program and directly plugs into execution-based identification and proof replay.

The cutoff relies on monotone carry/borrow state; it does not cover arbitrary shifts, multiplication, or free constants. The external study contains eight related specifications, and horizon eight remains unresolved under the current budget. The CNF compiler and RUP checker are tested and mutation-audited but not proof-assistant verified. These boundaries identify the next extension: broader transducer classes and long-horizon search while preserving independently checkable certificates.

7 Conclusion

Semantic program identification asks for the behavior that executions can actually reveal. Adaptivity can reduce the required observations exactly: a structural family gives $d + r$ versus $d + 2^d r$, and a nonlinear repeated-operator language gives a certified optimum $6 < 9$. A separate $2L + 1$ width cutoff turns large-width uniqueness into a bounded-width proof obligation for a fixed accumulator language. On external bit-manipulation specifications, that reduction lowers proof cost, enables full horizon-six certification under the tested budget, and composes with adaptive querying. The resulting

picture is one of proof-carrying identification: execution chooses what to observe, symbolic search finds what remains possible, and an independent checker certifies when only one behavior remains.

Reproducibility Statement

The appendix contains the complete finite-language upper and lower certificates, prior-sensitive fixed-design optimization, symbolic encodings, proof-replay code, all paired timing records, the width-cutoff proof, certificate-binding mutation tests, and retained UNKNOWN outcomes. Scripts reproduce every table in the main paper from archived records.

AI Use Statement

Generative AI tools were used to provide assistance with method implementation, result interpretation, manuscript editing, and exploratory technical drafting. All experimental results, mathematical claims, and proofs included in the final manuscript were independently reviewed and verified by the authors, who take full responsibility for the final content of this work.

References

Celeste Barnaby, Danny Ding, Osbert Bastani, and Işıl Dillig. Choose, don't label: Multiple-choice query synthesis for program disambiguation. *Proceedings of the ACM on Programming Languages*, 10(PLDI):805–829, 2026. doi: 10.1145/3808279.

Piotr Berman, Bhaskar DasGupta, and Ming-Yang Kao. Tight approximability results for test set problems in bioinformatics. In *Algorithm Theory – SWAT 2004*, volume 3111 of *Lecture Notes in Computer Science*, pp. 39–50. Springer, 2004. doi: 10.1007/978-3-540-27810-8_5.

Siddharth Bhat, Léo Stefanesco, Chris Hughes, and Tobias Grosser. Certified decision procedures for width-independent bitvector predicates. In *OOPSLA*, 2025. doi: 10.1145/3763148.

Jack Feser, Işıl Dillig, and Armando Solar-Lezama. Inductive program synthesis guided by observational program similarity. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA2): 912–940, 2023. doi: 10.1145/3622830.

Mikhail Galkin, Jincheng Zhou, Bruno Ribeiro, Jian Tang, and Zhaocheng Zhu. A foundation model for zero-shot logical query reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 54137–54160, 2024. doi: 10.52202/079017-1715.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011. doi: 10.1613/jair.3278.

Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity, 2010.

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, volume 1, pp. 215–224, 2010. doi: 10.1145/1806799.1806833.

Runzhou Li, Hongfei Fu, Qingkai Shi, and Peisen Yao. Sharp two-round adaptivity and round hierarchies for semantic regular expressions, 2026.

Aina Niemetz, Mathias Preiner, Andrew Reynolds, Yoni Zohar, Clark Barrett, and Cesare Tinelli. Towards satisfiability modulo parametric bit-vectors. *Journal of Automated Reasoning*, 65(7): 1001–1025, 2021. doi: 10.1007/s10817-021-09598-9. URL `https://theory.stanford.edu/~barrett/pubs/NPR+21c.pdf`.

Robert D. Nowak. The geometry of generalized binary search. *IEEE Transactions on Information Theory*, 57(12):7893–7906, 2011. doi: 10.1109/TIT.2011.2169298.

Calvin Smith and Aws Albarghouthi. Program synthesis with equivalence reduction. In *Verification, Model Checking, and Abstract Interpretation*, volume 11388 of *Lecture Notes in Computer Science*, pp. 24–47. Springer, 2019. doi: 10.1007/978-3-030-11245-5_2.

Ashish Tiwari, Arjun Radhakrishna, Sumit Gulwani, and Daniel Perelman. Information-theoretic user interaction: Significant inputs for program synthesis. arXiv:2006.12638, 2020.

Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pp. 422–429, 2014. doi: 10.1007/978-3-319-09284-3_31.

A Proofs for Semantic Identification

A.1 Execution-only syntax is unidentifiable within a semantic class

Fix a task input domain $\mathcal{X}_{\text{task}}$ and write $w \sim w'$ when $f_w(x) = f_{w'}(x)$ for every $x \in \mathcal{X}_{\text{task}}$. An adaptive execution protocol selects X_t from public information, private task-independent randomness R, and the previous transcript, then observes $Y_t = f_W(X_t)$. Suppose $w \sim w'$. Condition on the same realization of R. Before the first observation the policy selects the same X_1 under both words; semantic equivalence gives the same Y_1. Inductively, identical transcripts through step $t-1$ cause the policy to select the same X_t, and equivalence again gives the same Y_t. Thus the full transcript distributions are identical. Consequently no execution-only decoder can distinguish members of one semantic class. Under a uniform prior over a syntactic aliases in that class, every decoder has exact-syntax success at most $1/a$. This is an information statement, not a limitation of the proposed search algorithm.

A.2 Transcript counting bound on semantic classes

Let K_{sem} balanced semantic classes remain possible after conditioning on all public task fields. A depth-q protocol whose replies lie in an alphabet of size B has at most B^q transcripts after conditioning on private randomness. At most one semantic class can be decoded correctly per transcript, so

$$P_{\text{succ}} \le \min\{1, B^q/K_{\text{sem}}\}. \tag{10}$$

Zero-error identification therefore requires $q \ge \lceil \log_B K_{\text{sem}} \rceil$. If replies use a fixed-width binary code, the corresponding bit lower bound is $\lceil \log_2 K_{\text{sem}} \rceil$. The same argument applies to an adaptive policy because conditioning on its private randomness leaves a decision tree. Standard Fano inequalities give the nonzero-error counterpart; we use the exact counting statement because the new finite-state experiment is noiseless.

A.3 Minimal state, continuation, and posterior preservation

Let h be any exact summary, and let $q(w) = [w]$. If $h(w) = h(w')$, then $D(h(w), x) = D(h(w'), x)$ for all admissible x, so $q(w) = q(w')$. Consequently $q = r \circ h$ for a well-defined surjection from the reachable summary states onto the behavior classes. This factorization is stronger than the cardinality inequality alone: every exact summary partitions syntax at least as finely as the behavioral quotient. If the quotient is finite and the cardinalities agree, r is bijective. With compatible legal extensions, r also intertwines transitions: $r(T_j(h(w))) = [wj] = T_j^*(r(h(w)))$. The output decoder is similarly preserved. This establishes uniqueness up to state renaming under the stated exact summary contract.

The lower bound needs only exact decoding. Attainment as a legality-aware reusable state needs both lines of Eq. (3). For example, with $N^2 = I$ and length bound two, $\epsilon \sim NN$ but only ϵ can be extended. A pure behavioral quotient cannot represent that legality predicate. A product with remaining depth suffices for unrestricted bounded words; general grammars can need a residual grammar state. If the observable output itself is a projection of a richer hidden execution state, continuation congruence must also be proved rather than assumed.

For unrestricted words of length at most L, a shortest-representative BFS still discovers exactly all reachable behaviors. Induct on word length. Replace a prefix by its already discovered equivalent representative, whose length is no greater. Appending the same last primitive preserves behavior and stays within the bound. Expanding a node only when its shortest depth is below L is therefore complete. This is a reachability compiler; it is not a length-sensitive recognizer for arbitrary source histories. Each expanded state/primitive pair is composed once, at most $M|\mathcal{Q}_L|$ pairs. For regular-language constraints one can instead compile behavioral state jointly with grammar state, charging the larger reachable product.

For posterior preservation, write $\ell_Z(w) = P(Z \mid w, U)$. The observation contract implies $\ell_Z(w) = \ell_Z(q(w))$. Hence

$$\sum_{w:q(w)=q} P(w \mid Z, U) = \frac{\ell_Z(q) \sum_{w:q(w)=q} p(w \mid U)}{\sum_{q'} \ell_Z(q') \sum_{w:q(w)=q'} p(w \mid U)}.$$

This is precisely the update from the aggregated quotient prior. Predictive reply distributions agree, so their entropies and information-gain scores agree; behavior-only decisions then agree given the same tie breaking. A uniform prior over nodes is justified only when that is the declared prior or when the aggregated masses actually are uniform. For example, syntax masses $0.2, 0.3$ on two aliases and 0.5 on a second class induce class masses $0.5, 0.5$, whereas three uniform syntax masses induce $2/3, 1/3$.

A finite transition graph can be cyclic. The canonical-parent relation is a tree because parent depth strictly decreases, but generator transitions need not decrease depth. Certificate packaging returns the already materialized word, semantics, and transcript; it does not solve uniqueness for the checker. For the affine case this checker performs algebraic reconstruction. For the nonlinear case it uses independent exhaustive enumeration or the separate SMT encoding. Their initialization, checks, and solver work are not covered by the zero-new-composition packaging statement.

B Structural Separations: Proofs and Compositional Realizations

B.1 Routing versus simultaneous witness coverage

A fixed sequence with worst-case stopping time q must have an injective transcript map on its first q inputs. Otherwise two behaviors with identical prefix replies cause the same stopping decision and the same answer. Thus its first q inputs must meet every nonempty $\Delta(f, g)$. Conversely, any such hitting set is a valid fixed design. Repeated queries cannot improve this zero-error bound. This explains the connection to test cover (Berman et al., 2004); it is not a new characterization of nonadaptive testing.

For Theorem 2, concatenate the routing tree with the appropriate fiber tree. Even without exploiting information from routing in the second phase, worst-case depth is at most $a + \max_s b_s$. For a fixed design A and each i, its intersection $A \cap U_i$ must hit $\mathcal{E}_i$, because each witness pair agrees on every input outside U_i. Consequently $|A \cap U_i| \geq \tau_i$. The U_i are disjoint, giving $|A| \geq \sum_i \tau_i$. Empty witness families may be assigned hitting number zero. This is a sufficient condition; its failure does not imply that adaptivity cannot help. It also does not guarantee that entropy-greedy finds the routing tree for an arbitrary family. Proposition 2 supplies a family where that additional property is proved.

In the address family, take $U_0 = \{c_i : i \in [d]\}$ and $U_s = \{p_{s,j} : j \in [r]\}$. Singleton difference witnesses give $\tau_0 = d$ and $\tau_s = r$ for each of the 2^d addresses. The routing and local costs are $a = d$ and $b_s = r$. The resulting separation is therefore the explicit strict inequality $d + 2^d r > d + r$ for every $d, r \geq 1$.

B.2 A finite primitive bank and a bounded legal language

Use disjoint state symbols

$$X = \{\bot_0, \bot_1\} \sqcup \{c_i : i \in [d]\} \sqcup \{p_{u,j} : u \in \{0,1\}^d, j \in [r]\} \sqcup \{z_j : j \in [r]\}.$$

Every state in X, including workspace symbols z_j, is an admissible input. There are $2 + d + 2^d r + r$ states. For each address s, define a routing primitive A_s by

$$A_s(\bot_v) = \bot_v, \qquad A_s(c_i) = \bot_{s_i},$$

$$A_s(p_{u,j}) = \begin{cases} z_j, & u = s, \\ \bot_0, & u \neq s, \end{cases} \qquad A_s(z_j) = \bot_0.$$

For $j \in [r]$ and $v \in \{0,1\}$, the write primitive W_j^v maps z_j to $\bot_v$ and fixes every other state. All primitives are total maps $X \to X$; the bank has $2^d + 2r$ members. Numeric implementation uses sink labels $0, 1$, then selectors, private probes, and workspace states, in that order.

For any fixed repeat bound $R \geq 1$, the legal language is

$$\mathcal{W}_{d,r,R} = \left\{ A_s v_1 \cdots v_r : v_j \in \{W_j^0, W_j^1\}^{1:R} \right\}. \tag{11}$$

Here operators execute left to right and the exponent means lengths one through R. The first write in block j determines b_j; subsequent writes fix the sink already reached. Induction through the blocks

proves Eq. (5) for every legal word. It also proves $f(z_j) = \perp_0$ and $f(\perp_v) = \perp_v$, so workspace probes leak no additional information. Every final output lies in $\{\perp_0, \perp_1\}$.

The canonical representative is $A_s W_1^{b_1} \cdots W_r^{b_r}$. Distinct (s, b) give distinct full-domain behaviors, hence the quotient has exactly 2^{d+r} classes. The completed-program summary (s, b) therefore attains the minimum behavioral state count in Theorem 1; grammar and partial-execution control are separate. Each class has $(2^R - 1)^r$ aliases, and the language contains $2^{d+r}(2^R - 1)^r$ syntax strings. For $R \geq 2$, lengths vary from $r + 1$ to $rR + 1$ and repeated operators occur. Writes are idempotent, routing is non-injective, and routing does not commute with writing: on $p_{s,j}$, $W_j^1 \circ A_s$ returns $\perp_1$, whereas $A_s \circ W_j^1$ returns z_j. Thus this is a compositional transformation language with aliases and non-invertible semantics. Its grammar is intentionally restricted and needs control state beyond the behavioral summary; it is not the unrestricted length-L closure of the primitive bank.

B.3 Exact query bounds and greedy optimality

There are 2^{d+r} possible behaviors and every reply is binary. Thus every exact adaptive policy has worst-case depth at least $d + r$. Querying all c_i reveals s, after which querying $p_{s,1}, \ldots, p_{s,r}$ reveals b. This constructs an attaining tree, also executable in two batches: selectors then local payload. The statement concerns total queries, not latency or a complete hierarchy of round budgets.

For every private input $p_{s,j}$, the behaviors $(s, 0)$ and (s, e_j) differ only at that input on the *whole* domain. For each selector c_i, the behaviors $(0, 0)$ and $(e_i, 0)$ differ only at c_i. A fixed exact policy must query every one of these $d + 2^d r$ inputs; querying them all suffices. This proves both sides of Eq. (6), including early-stopping fixed policies.

For the uniform-class prior, suppose $k \geq 1$ selector bits remain unknown and no private input has yet been queried. Each remaining selector is balanced, with entropy one. A private input at a still possible address returns one with probability $2^{-(k+1)}$, giving strictly smaller entropy; inputs at excluded addresses and all workspace/sink inputs have entropy zero. Every entropy maximizer is therefore an unqueried selector. Induction proves that greedy acquisition first resolves the address, regardless of tie breaking. Once the address is known, every unqueried local payload bit is balanced and all other unused probes are constant. Greedy then asks exactly r local queries. Since every class has the same number of aliases at a fixed R, the uniform syntax prior also induces the uniform-class prior in this family.

If global payload probes g_j with $f_{s,b}(g_j) = b_j$ are added, all selector and global payload queries form a fixed design of size $d + r$, meeting the same counting lower bound. A total-map realization extends $A_s(g_j) = z_j$ for every s and lets writes fix g_j. Now private-payload witness pairs differ on both $p_{s,j}$ and g_j; the disjoint-support premise is lost. This demonstrates why the admissible query interface is part of the separation theorem.

B.4 Independent certification and finite cross-checks

A certificate contains a legal representative, its claimed full table, and the d selector plus r local replies. The checker separately parses Eq. (11), scalar-replays the public primitive definitions, and checks equality to the closed-form table. It verifies that the transcript contains all selector inputs and all payload inputs for the recovered address, with consistent replies. The grammar normalization proof establishes that every legal behavior is some (s, b); the transcript forces both coordinates uniquely. Thus checking does not depend on a proposer-supplied class list or search tree. This checker is specialized to the declared public schema; it does not certify an arbitrary replacement primitive bank or an arbitrary transcript format.

The code is `src/tango/structural_semantic_languages.py`; running `scripts/validate_structural_separation.py` with `PYTHONPATH=.:src` reproduces Table 3. Across its six parameter settings, 24,156 raw words, 1,876 distinct behaviors/certificates, and 503 mandatory-probe witnesses are checked. Every incomplete route-first transcript formed by removing one required reply, and every tested corrupted table, is rejected. Separate small decision-tree dynamic programs and exhaustive fixed-subset search check $(d, r) = (1, 1), (1, 2), (2, 1), (2, 2)$; a cross-address-probe control has equal adaptive and fixed optima. These finite checks supplement the analytic proof; they do not establish an all-parameter theorem by extrapolation.

Table 3: Finite cross-checks of the address-language theorem. The formulas hold for all positive d, r; rows are exhaustive checks of the stated bounded grammars, not samples used to infer the theorem. Every reported behavior has an independently checked certificate.

d	r	R	Behaviors	Raw words	Adaptive	Fixed
1	1	2	4	12	2	3
2	2	2	16	144	4	10
3	3	2	64	1728	6	27
4	4	2	256	20736	8	68
5	4	1	512	512	9	133
6	4	1	1024	1024	10	262

B.5 Routing the original nonlinear language

The address mechanism also lifts the exact $6 < 9$ artifact, without replacing its behavior class by independent payload bits. Let a finite base class $\mathcal{Q} \subseteq \{0, \dots, n-1\}^X$ contain the constant-zero behavior, with $n \geq 2$. For $d \geq 1$, define a routed class by

$$F_{s,f}(c_i) = s_i, \qquad F_{s,f}(p_{u,x}) = \begin{cases} f(x), & u = s, \\ 0, & u \neq s, \end{cases} \quad s, u \in \{0,1\}^d,\ f \in \mathcal{Q}. \tag{12}$$

All remaining admissible inputs have target-independent replies. The selector bits distinguish addresses even when f is zero; private queries at the selected address distinguish base behaviors. Thus there are exactly $2^d|\mathcal{Q}|$ behaviors.

Proposition 5 (Routed lifting of a finite base class)**.** *Writing $D = q^*_{\mathrm{ad}}(\mathcal{Q})$ and $N = q^*_{\mathrm{fixed}}(\mathcal{Q})$,*

$$q^*_{\mathrm{ad}}(\mathcal{R}_d(\mathcal{Q})) \leq d + D, \qquad q^*_{\mathrm{fixed}}(\mathcal{R}_d(\mathcal{Q})) = d + 2^d N. \tag{13}$$

A complete selector transcript plus a unique local base transcript independently certifies the routed behavior, provided base-language completeness is checked.

Proof. Ask the selectors and then run an optimal base policy at the resulting address, giving the adaptive upper bound. For each address s, a fixed design must separate every pair $(s, f), (s, g)$. All their differences lie in its private block, so that block needs at least N queries. In addition, the behaviors $(0, 0)$ and $(e_i, 0)$ differ only at selector c_i, making every selector mandatory. The blocks and selectors are disjoint, giving $d + 2^d N$. Asking all selectors and a base-optimal fixed design at every address attains that cost. Finally, a complete selector transcript forces s and a unique local transcript forces f; the normal-form construction ensures that these exhaust the legal behaviors. □

For the nonlinear bank, constant zero is behavior 4247, with a legal length-seven representative $(4, 3, 4, 3, 4, 3, 4)$ in zero-based primitive indices. This is four resets interleaved with three rotations. Its full 16-entry table is checked to be zero. Proposition 3 supplies $D = 6$ and $N = 9$, hence

$$q^*_{\mathrm{ad}}(\mathcal{R}_d(\mathcal{W}_8)) \leq d + 6 < d + 9 \cdot 2^d = q^*_{\mathrm{fixed}}(\mathcal{R}_d(\mathcal{W}_8)).$$

The fixed-to-adaptive ratio is therefore at least $(d + 9 \cdot 2^d)/(d + 6)$, which is unbounded as d grows. This is a proved lifting of the finite artifact, not a regression over increasing experimental horizons.

Total-map implementation without workspace leakage. For the concrete base, use protected output states $o_0, \dots, o_{15}$, selector states c_i, private inputs $p_{u,x}$, and workspace states $z_0, \dots, z_{15}$. The domain has $32 + d + 16 \cdot 2^d$ states, all queryable. Routing primitive A_s fixes protected outputs, maps c_i to o_{s_i}, maps $p_{s,x}$ to z_x, and maps every other private or workspace input to o_0. Lift base primitive S_j to $z_x \mapsto z_{S_j(x)}$, fixing every non-workspace state. A final projection P sends z_x to o_x and fixes all other states. The public bank has $2^d + 6$ total transformations. The grammar permits exactly $A_s w P$ for a base word w of length at most eight. Thus lengths range from 2 to 10 and repeated base operators and their aliases are retained. Protected outputs ensure that inactive and workspace queries stay target-independent through the entire composition. Eq. (12) follows by induction on w. In particular no global execution of f can be obtained by probing a workspace input.

A checker parses this grammar, checks the selector bits, and invokes the independent base checker on the selected block's word and transcript. The public routing/projection schema and the complete base-language audit establish coverage of every legal routed behavior. No enumeration of $10974 \cdot 2^d$ classes is needed for each new d. The implementation checks full transformation tables for four specified base behaviors at every address for $d = 1, 2, 3, 4$ (120 tables), including identity, reset, and constant zero. All 120 routed certificates pass the wrapper checker, and their ambiguous final-query prefixes are rejected; these are implementation checks of the original greedy certificates, not exhaustive routed-cohort experiments or timings of the refined policy. The improved all-d bound uses the independently checked six-query base tree and the same algebraic construction above.

Unlike the independent-bit family, this corollary uses a route-first policy followed by the refined six-query base tree. It does not assert that flat global entropy-greedy acquisition is optimal, or even follows that routing order, for the routed nonlinear class. The bank, domain, and legal-language interface grow with d; the result is not an unbounded-gap theorem for the original five-primitive bank alone.

B.6 Tagged composition and the nonlinear family

For finite component domains X_i and word languages $\mathcal{W}_i$, lift each primitive to the disjoint union $\bigsqcup_i X_i$: it acts on its own tagged component and is identity elsewhere. A global word is legal exactly when its projection onto component i belongs to $\mathcal{W}_i$. For prefix-closed bounded-word languages this admits every interleaving of legal component words. Its behavior is

$$F(i, x) = (i, f_i(x)), \qquad (f_1, \ldots, f_t) \in \prod_i \mathcal{Q}_i.$$

A probe executes one tagged input (i, x). Its tag is already known; the reply carries only the component's output information. The hypothesis set is the full Cartesian product, with no cross-component restriction.

Theorem 4 (Exact additivity under component-local execution)**.** *For the interface above,*

$$q^*_{\mathrm{ad}}(\mathcal{W}^{\oplus}) = \sum_i q^*_{\mathrm{ad}}(\mathcal{W}_i), \qquad q^*_{\mathrm{fixed}}(\mathcal{W}^{\oplus}) = \sum_i q^*_{\mathrm{fixed}}(\mathcal{W}_i). \tag{14}$$

The behavioral quotient has size $\prod_i |\mathcal{Q}_i|$. Global transcript uniqueness holds if and only if every projected component transcript is unique. Under a product prior, posterior inference also factorizes by component.

Proof. The upper bounds run the component policies in sequence. For the fixed lower bound, the queries assigned to each component must separate that entire component class; otherwise hold the other components fixed and use its colliding pair. Summing proves the fixed equality.

For the adaptive lower bound, let $D_i(V)$ be the minimum deterministic worst-case depth to identify a nonempty remaining component set V. For a nonsingleton V and any queried input x, some nonempty reply cell V_y obeys $D_i(V_y) \geq D_i(V) - 1$: otherwise asking x and following optimal subtrees would contradict the definition of $D_i(V)$. An adversary chooses such a cell whenever component i is queried. Consistent candidates always remain a product of nonempty component sets. Each query decreases the potential $\sum_i D_i(V_i)$ by at most one. Exact identification ends only when all sets are singleton and the potential is zero. At least $\sum_i D_i(\mathcal{Q}_i)$ queries are therefore needed. This is the usual deterministic direct-sum principle (Jain et al., 2010), specialized to the permitted execution queries.

Two tuples differ in behavior exactly when at least one component differs, proving the quotient count. Transcript consistency is the conjunction of the component consistency conditions, so the surviving set is the Cartesian product of the component surviving sets. It is singleton exactly when each factor is. With a product prior and local deterministic replies, multiplying likelihoods preserves posterior factorization. No independence claim is made for a correlated prior. □

For t copies of the four-bit nonlinear bank, the lifted bank has $5t$ primitives on $16t$ states. Impose a separate length bound of eight on each component projection, not merely a total-length bound of $8t$. The quotient has 10974^t behaviors and

$$q^*_{\mathrm{ad}}(\mathcal{W}_8^{\oplus t}) = t q^*_{\mathrm{ad}}(\mathcal{W}_8) = 6t < 9t = q^*_{\mathrm{fixed}}(\mathcal{W}_8^{\oplus t}).$$

Table 4: Exhaustive nonlinear identification over all 10,974 behaviors. Policies identify every behavior; except for the optimal-set reference, they stop at the first singleton. Independent checking certifies all active leaves. Each reply has 16 values (4 bits). Fixed-design trace diversity comes from stopping times, not reply branching.

Policy	Mean queries	Worst	Traces	Branch nodes
Response-adaptive IG	4.33	7	789	568
Optimal fixed set (exact)	–	9	–	0
Fixed joint-entropy design	4.86	11	10	0
Fixed index scan	9.05	15	14	0
Random fixed order (3 seeds)	6.56	12–13	10–12	0

Worst-case counting lower bound: 4 queries. Exhaustive subset checking gives an exact fixed-design optimum of 9; the active tree needs at most 7. The optimal fixed row uses the whole set without early stopping; the other policies stop at a singleton.

The exact base depth six is established by the independent obstruction proof in Appendix E; additivity therefore gives an equality for the tagged family. Each global certificate consists of legal component words and local transcripts. A checker can reuse one independently rebuilt base language and validate the t component certificates without enumerating 10974^t tuples. Behavioral summaries are tuples of base summaries; legality additionally tracks the per-component depth budgets. A product prior permits componentwise search and acquisition as well as certificate checking.

The implementation checks the lifted primitive maps and interleaved words for specified tuples at $t = 1, 2, 4, 8$. The complete product families are not enumerated; the all-t claims follow from the theorem and the independently checked base artifact. If a single query instead returns a whole vector of component outputs, parallel component execution invalidates the additive accounting. Composition by itself is therefore insufficient: the local probe interface and unrestricted product of behaviors are essential premises.

C Nonlinear Language and Independent Checking

Artifact navigation. The central result's records are in `results/semantic_revision_20260915/nonlinear/`: `public_language.npz` declares the bank and horizon, `behaviors.npz` stores complete tables, `query_tree.json` and `certificates.json` store the adaptive witnesses, and `nonadaptive_certificate.npz` supplies every eight-input collision and the separating nine-input set. Running `scripts/run_finite_semantic_revision.py` regenerates the nonlinear study; `scripts/finite_nonadaptive_lower_bound.py` computes the fixed optimum. With the project dependencies installed and `PYTHONPATH=.:src`, `scripts/validate_semantic_revision.py` checks the saved evidence independently.

For bits $b_i = (x \gg i)\ \&\ 1$, the five maps in bank order are

$$N(x) = x \oplus 1, \qquad C(x) = x \oplus b_1, \qquad F(x) = x \oplus (b_1 b_2),$$
$$R(x) = ((2x) \bmod 16) + b_3, \qquad E(x) = x\ \&\ 14.$$

All operations are exact integers on $0, \ldots, 15$. A program's semantic key is its complete 16-byte output table. The BFS compiler applies public primitive tables to discovered behavior tables and stores shortest words. It performs 21,240 table compositions for the length-eight language. The independent checker instead retains all raw syntax at each length, including aliases: $\sum_{\ell=0}^{8} 5^\ell = 488{,}281$ tables before independent deduplication. Equality of the two enumerated behavior sets is explicitly checked. This independent rebuild is necessary: filtering a proposer-supplied hypothesis list would not certify its completeness.

The shared checker initialization takes 2.85 seconds in the recorded run. Its concatenated raw syntax tables alone occupy 7,812,496 bytes, and its retained unique-table and alias-count arrays occupy 263,376 bytes. These are array payloads, not peak resident memory: enumeration retains layers and uses sorting temporaries. The proposer's behavior tables occupy 175,584 bytes, excluding Python representatives, containers, and policy nodes. The original entropy-greedy policy tree was

compiled once in 1.18 seconds in the earlier run. Thus table composition, policy compilation, checker initialization, and individual certificate checks are four distinct costs; reporting only the cheap final tree traversal would omit most cold-start work.

Every target certificate records its representative, reply transcript, and checker time. The original greedy query-count distribution is: 1 target at 2 replies, 559 at 3, 6,484 at 4, 3,674 at 5, 254 at 6, and 2 at 7. The refined six-query optimum and its separate lower-bound proof are in Appendix E. This does not contradict the four-reply lower bound, which concerns worst-case depth across all 10,974 classes. A query trace contains only the input indices, not replies; the explicit branching count includes a policy node only when two nonterminal reply children choose different next inputs. Leaf stopping alone is not counted as branching.

A second checker uses an independent bit-vector encoding of the primitives. It asks whether a legal word w' can satisfy

$$Z(w') = Z \quad \wedge \quad \bigvee_{x=0}^{15} f_{w'}(x) \neq f_{\hat{w}}(x). \tag{15}$$

All 12 targets selected by a fixed random seed return UNSAT with a 10-second per-check timeout (mean 3.37 seconds). Identity padding encodes lengths up to 8. This SMT cross-check covers a stated sample; exhaustive enumeration certifies the full cohort. UNSAT is a solver result, not a separately checked proof trace. Certification is relative to the declared length-bounded language, not all functions on 16 states.

The SMT checker uses four-bit state variables, a shared three-bit primitive choice at each of eight positions, and exact conditional bit-vector updates. Choice 5 is identity padding; removing padding gives a legal word of length at most eight, so it neither removes nor introduces a behavior. Sixteen copies of the state trajectory share the same operator choices. Transcript equations and a disjunction asserting difference from the proposed table encode Eq. (15). Representative legality and consistency are checked before asking the solver. Twelve targets are selected without replacement by NumPy seed 20260915. Z3 5.1.0 returns UNSAT for all twelve; no timeout or UNKNOWN is counted as certification. The SMT checks are a second encoding audit, not the source of the exhaustive 10,974-target certification claim.

D Exact Nonadaptive Query Lower Bound

For each pair of distinct behavior tables f, g, encode the nonempty difference set $D(f, g) = \{x : f(x) \neq g(x)\}$ as a 16-bit mask. A fixed probe set A separates all behaviors if and only if it intersects every such difference set. Our audit collects the 10,520 distinct difference masks from all behavior pairs. A Boolean subset transform marks every unqueried set containing one of these masks. We then check all 65,536 possible query subsets. The unique smallest separating set has nine inputs,

$$A_* = \{2, 3, 4, 6, 9, 10, 13, 14, 15\}.$$

Directly projecting the 10,974 tables onto these nine columns verifies that all rows remain distinct. No eight-input set separates all rows. For a fixed sequence with early stopping, worst-case depth at most eight would imply that its first eight inputs separate every behavior, which is impossible. Thus its worst case is at least nine. This is a finite-language exhaustive optimality result, not an asymptotic query-complexity theorem. The original entropy tree's depth seven already gives a strict separation on the same complete candidate set. The refined tree and adaptive obstruction proof strengthen this to exact optima six versus nine (Appendix E).

A checkable lower-bound certificate. The artifact includes one pair of distinct legal behaviors for each of the $\binom{16}{8} = 12{,}870$ eight-input subsets, such that both behaviors agree on all eight inputs. A separate validator checks all 12,870 pairs directly against the independently validated full-domain tables and verifies that every eight-set is covered. It also checks that A_* separates all tables. Therefore the nonadaptive lower bound can be checked without trusting the difference-mask optimizer or an optimality flag.

Strictly different acquisition optima. After root query $x = 5$, replies $0, 1, 2$ leave respectively 875, 664, and 779 behaviors. Their unique next-query entropy maximizers are respectively $2, 3, 14$, with

entropies 3.891372, 3.893245, and 3.803954 bits. The validator recomputes all 15 remaining query scores for each branch. The displayed response adaptivity therefore does not arise from arbitrary tie breaking. This is one-step information-gain optimality, not global optimality of entropy-greedy. The separate one-branch refinement attains the independently proved optimal depth six.

E Exact Adaptive Optimum and Independently Checked Obstructions

The nonlinear result is strengthened from the original entropy-greedy upper bound seven to the exact optimum six. This section distinguishes discovery of a better policy, exhaustive checking of its certificates, and the independent proof that five queries cannot suffice. None of these conclusions comes from an optimizer's success flag alone.

A one-branch policy repair. Start with the original entropy-greedy tree. After the transcript

$$((5,6),(3,2),(14,11),(9,14)),$$

exactly six behaviors remain. Change the next query from the entropy maximizer 11 to input 1, then resume entropy-greedy selection. Every other branch is unchanged. The resulting query histogram is

queries	2	3	4	5	6
behaviors	1	559	6484	3672	258.

All 10,974 transcripts pass the independent raw-syntax checker. The mean remains 4.3305084746: two former depth-seven targets move to depth six, while two former depth-five targets move to depth six. This optimizes the worst case, not every target's stopping time, the mean, or every partial-budget success rate. The recipe is frozen public policy information, not a hidden target-dependent rule. It was found post hoc by bounded exact lookahead visiting 63 states; we do not present that search as a new general synthesis algorithm.

The complete policy is reconstructed by `build_refined_tree` in `src/tango/fixed_design_audit.py`. The discovery audit is `scripts/audit_bounded_tree_repair.py`; its six-query run independently checks every final certificate. The original greedy tree and its seven-query results remain separate controls. An implementation shortcut stops entropy scoring as soon as a query is injective on the remaining candidates: this attains the absolute bound $\log_2 |V|$. Inputs are scanned in increasing order, so the shortcut preserves the declared smallest-index maximizing policy.

An adversary proof of the adaptive lower bound. For a nonempty behavior set V, let $P(V,k)$ mean that some decision tree of depth at most k identifies every element of V. For an input x and reply y, write $V_{x,y} = \{f \in V : f(x) = y\}$. The exact recurrence is

$$P(V,k) = \bigvee_{x} \bigwedge_{y : V_{x,y} \neq \varnothing} P(V_{x,y}, k-1), \qquad |V| > 1,\ k > 0. \tag{16}$$

Singleton states are feasible at every nonnegative budget. A nonsingleton state at budget zero is infeasible, as is $|V| > 16^k$ by transcript counting. A constant query can be omitted: deleting its unique edge from a successful tree strictly reduces its depth without changing the identified behavior. Thus the recurrence need only examine informative queries, but must examine *all* of them.

An infeasibility certificate for (V,k) supplies, for each informative x, a nonempty reply cell V_{x,y_x} already certified infeasible at $k-1$. The base cases above and induction on k prove that such a directed acyclic graph is an obstruction to every depth-k decision tree. Identical states may share proof nodes; this sharing changes proof size, not the logical obligations. The proof therefore concerns all adaptive policies on the declared interface, not just entropy-greedy, a particular root query, or a fixed candidate order.

Independent production and verification. `scripts/check_adaptive_depth_lower_bound.py` reads only the public bank and horizon. Its reference set is rebuilt by exhaustive raw enumeration of all 488,281 syntax strings, yielding 10,974 distinct behaviors. A memoized bit-set producer applies Eq. (16). A separate checker uses array row filtering to reconstruct every claimed

child, checks that the reply is possible, and checks every informative input's obstruction. It does not trust the producer's partitions, feasibility flags, or proposer enumeration. At budget five, it checks 12,244 reachable obstruction nodes and 130,795 query obligations. Its terminal checks include 7,737 cardinality obstructions and 99,815 depth-zero obstructions; these are visit counts, not distinct leaves. A deliberately corrupted root certificate is rejected.

The producer generates 18,538 obstruction nodes in total, including nodes not needed by the root proof, after visiting 80,210 feasibility states. The recorded producer and verification times are 3.42 and 1.57 seconds; these are proof-audit costs, not per-target identification times. The command's `--emit-proof` option emits the entire regenerated proof DAG. A timeout never certifies infeasibility. A second array-based bounded search also rejects budget five, but the exported lower-bound claim rests on the checked obstruction, not on agreement of two solver flags. Together with the six-query certificate this establishes $q^*_{\mathrm{ad}} = 6$.

The compact provenance record is `paper/revision_20260916/exact_adaptive_optimum.json`. The proof SHA-256 is `047717f5e73d29cfdc7d19145322cbf41be601a11049c155e3a4bf6d5c6b9dc9`. This is a computer-assisted finite-language proof, not a proof-assistant formalization. Authors must review the recurrence, implementation, and public language boundary independently.

F A Mean-Optimal Ordering of the Best Fixed Design

The original plot queried the unique minimum nine-input separating set in increasing numeric order. That order is not mean-optimal. For a subset $A \subseteq A_*$, define

$$a(A) = |\{f : \exists g \neq f,\ f|_A = g|_A\}|.$$

This is the number of targets that have not reached a singleton after querying A. If the cohort is uniform, the total stopping-depth sum of a fixed order is the sum of $a(A)$ over its prefix sets. Therefore the exact dynamic program

$$F(A) = a(A) + \min_{x \in A_* \setminus A} F(A \cup \{x\}), \qquad F(A_*) = 0 \tag{17}$$

minimizes this sum over all permutations of A_*. All arithmetic in the objective is integer-valued; lexicographic ties are resolved deterministically. There are $2^9 = 512$ states. The selected order is

$$(4, 9, 3, 14, 15, 6, 2, 10, 13), \qquad F(\varnothing) = 52,595.$$

Its mean is $52,595/10,974 = 4.7926918170$. The stopping histogram for budgets 2 through 9 is $(6, 596, 4349, 3918, 1312, 509, 182, 102)$. Since A_* is the unique minimum fixed set, this is also the best mean among worst-case-optimal fixed designs. We have not proved it minimizes the mean among longer fixed sequences. The code's dynamic program is cross-checked against all permutations on a separate small instance; invalid and nonseparating designs are rejected.

G Fixed-Bank Horizon Sensitivity

Table 5 keeps the original five primitives and all 16 probes unchanged, varying only the public maximum word length. Every fixed optimum is found by checking all 65,536 probe subsets. This is a post hoc sensitivity analysis of the same designed bank, not an external benchmark or an all-horizon theorem. The unmodified greedy policy already gives strict worst-case separations at every tested horizon from four through eight; its depths tie the fixed optimum at horizons one through three. Only the length-eight adaptive optimum is established by the separate obstruction proof.

H Same-Language Certified Deployment Accounting

The nonlinear timing study evaluates all 10,974 targets in each of three repeats. All methods share a public canonical behavior roster and the same independently rebuilt checker. Each repeat uses a seeded target permutation, paired across methods, and a seeded random method order. Reported uncertainty is the observed minimum–maximum of repeat-level means, not a confidence interval.

Table 5: **No routing extension is used in this horizon sweep.** Greedy depths are upper bounds, while the fixed column gives exact optima. At horizon eight, the additional independently checked refinement reduces adaptive depth to the exact optimum six.

Maximum length	Behaviors	Syntax strings	Greedy worst	Fixed optimum
1	6	6	3	3
2	21	31	4	4
3	64	156	4	4
4	190	781	4	5
5	552	3,906	5	6
6	1,563	19,531	6	7
7	4,248	97,656	6	8
8	10,974	488,281	7	9

Table 6: **Nonlinear timing medians and observed repeat ranges.** Setup includes graph compilation, tree construction, and independent checker initialization. Online time includes oracle-table lookup, traversal, certificate packaging, and checking. No external reply latency is assigned.

Policy	Setup (s)	Online (μs)	Full census (s)
Refined adaptive	2.115 [1.964,2.721]	91.2 [91.1,105.8]	3.115 [2.964,3.883]
Entropy-greedy	1.973 [1.957,2.593]	90.5 [89.1,90.8]	2.969 [2.951,3.570]
Frozen optimal fixed	1.854 [1.730,2.247]	96.1 [95.4,97.2]	2.921 [2.784,3.294]
Fixed with order DP	1.955 [1.807,2.327]	97.1 [97.0,101.4]	3.019 [2.921,3.393]
Fixed joint-entropy	3.086 [2.562,3.264]	111.2 [97.5,125.3]	4.461 [3.632,4.484]

Every method certifies every target in every repeat. Thus coverage and query counts are complete finite-population results, whereas timings depend on the host and its load. Python 3.10.20 and NumPy 2.2.6 are used on the revision host.

The frozen fixed comparator receives the optimized order without paying its scientific discovery cost. Symmetrically, the refined adaptive policy receives the frozen branch recipe. Both build their installed policy trees and pay that cost. The additional fixed-with-DP row charges the 512-state ordering search; the frozen row is the stronger deployment comparator. Discovery of the optimal nine-input set, the branch repair, and the worst-case proof are not hidden inside an online timing claim: they are separate public-language research/audit costs, excluded for both frozen policies. Timing hashes and repeat-level measurements are recorded in `paper/revision_20260916/matched_nonlinear_cost.json`.

The retained policy objects occupy 5,826,446 bytes for the refined policy, 5,826,447 for unmodified greedy, 6,676,542 for the fixed-order tree, and 6,992,059 for fixed joint entropy. The common roster occupies 1,360,120 bytes; the independent checker's retained arrays/objects occupy 264,000 bytes. Recursive object accounting deduplicates shared identities. These measurements exclude temporary allocations and peak RSS; a different fixed-policy representation could use less memory. They are not minimum-memory theorems.

Conditional crossover, not universal speedup. For a declared cohort-mean deployment model, write

$$C_m(Q,\lambda) = S_m + Q(\bar{t}_m + \lambda \bar{q}_m), \qquad \Delta q = \bar{q}_f - \bar{q}_a > 0.$$

Then the adaptive policy is no more costly than the fixed policy precisely when

$$\lambda \geq \frac{(S_a - S_f)/Q + \bar{t}_a - \bar{t}_f}{\Delta q}. \tag{18}$$

For nonnegative latency, a negative right side means adaptivity already wins at zero latency. We compute this threshold separately within each paired repeat and then summarize it; subtracting unrelated medians need not give the paired median. At $Q = 1000$, the median threshold is 0.822 ms per reply, with range $[0.225, 1.050]$ ms. At the complete-census horizon $Q = 10,974$, it is 0.0653 ms, with range $[0.00865, 0.1162]$ ms. These are sensitivity calculations, not observed external latencies. For smaller Q, the model uses full-cohort online means, not measured prefixes of an

arbitrary nonuniform task sequence. At zero external latency, the frozen fixed comparator is faster in all three paired full-census runs. This negative result is retained rather than replaced by the favorable comparison to fixed joint entropy.

Reproduction and trust boundary. With project dependencies installed and `PYTHONPATH=.:src`, run `scripts/audit_semantic_strong_revision.py` with `--mode cost --repeats 3`, `--mode horizons --max-length 8`, or `--mode checker`. The scripts print their records without overwriting legacy evidence. Queries are issued through an oracle-only traversal function; evaluation target IDs are not policy inputs. An incompatible reply leads to abstention. Correctness certificates establish uniqueness relative to the bounded public language and supplied transcript. They do not authenticate the oracle, prove that an out-of-language target belongs to the language, or imply robustness to noisy replies. These premises are necessary for interpreting "certified correctness."

I Prior Sensitivity and Budget-Optimal Fixed Coverage

This additional audit keeps all five primitives, all 16 inputs, the length bound eight, and all 10,974 behaviors unchanged. It changes only the declared design and evaluation prior. The analysis is post hoc and exhaustive for this finite language; it is not a new external benchmark or an estimate of human intent.

Preserving the syntax prior changes the policy. Let m_f count the legal syntax strings representing behavior f. Independent raw-syntax enumeration gives $\sum_f m_f = 488{,}281$ and $1 \le m_f \le 29{,}409$. Uniform sampling from all strings of lengths zero through eight induces $\pi(f) = m_f/488{,}281$, not a uniform distribution over behaviors or over lengths. Entropy-greedy acquisition with these aggregated masses, without any branch-specific repair, has worst-case depth six. Its histogram over all behaviors is $(1, 458, 6303, 3965, 247)$ for depths two through six. Every one of its 10,974 final transcripts passes the independent raw-syntax checker. The previously established five-query obstruction is independent of prior and still applies, since every class has positive mass. Consequently this unmodified, alias-aware greedy policy also attains the optimal worst-case six on this instance. This is not a general greedy-optimality theorem, and the uniform-class greedy policy still needs seven queries before refinement.

Global mean-optimal fixed baselines for each prior. For positive integer class masses m_f and a queried subset A, define

$$a_m(A) = \sum_f m_f \, \mathbf{1}\{\exists g \neq f : f|_A = g|_A\}.$$

For any fixed order with singleton stopping, its weighted total query count is the sum of $a_m(A)$ over prefix subsets. Thus the recurrence

$$F_m(A) = a_m(A) + \min_{x \in \mathcal{X} \setminus A} F_m(A \cup \{x\}), \qquad F_m(\mathcal{X}) = 0,$$

optimizes the mean over *all* fixed orders on the entire probe interface. States already separating all behaviors have value zero. The audit evaluates all $2^{16} = 65{,}536$ subsets, not just permutations of the nine-input minimax design. Integer objectives avoid a rounding-dependent optimum. Randomizing a fixed order cannot improve its expected objective, which is a mixture of the deterministic objectives.

For uniform behavior, the optimal order is $(5, 8, 2, 15, 14, 7, 3, 11, 12, 9)$, with total query count $52{,}454$, mean 4.7798432659, and worst case ten. For uniform syntax, the corresponding order is $(2, 14, 7, 13, 8, 5, 11, 3, 15, 1, 4)$, with weighted total $2{,}232{,}220$, mean 4.5715889007, and worst case eleven. These means optimize different objectives; the optimal fixed *worst case* remains nine under both full-support priors. The audit independently checks all target certificates for both fixed orders, the original refined adaptive policy, and alias-aware entropy-greedy.

The matched syntax-prior result uses weighted total $2{,}149{,}127$; the mismatched refined tree uses $2{,}201{,}207$. Mean query savings persist for both declared priors, but their magnitude is not prior-invariant. This experiment supports the mass-preservation requirement in Eq. (2); it does not establish optimal expected adaptive cost or robustness to arbitrary priors. No new timing advantage is inferred from these weighted query counts.

Table 7: **Prior sensitivity against a separately optimized global fixed baseline.** All rows use the same complete behavior language and exact certificate condition. A mismatch means that the class-prior refined tree is retained but evaluated under syntax mass. These are finite-population means, not estimates from independently sampled tasks.

Evaluation prior	Adaptive policy	Adaptive mean	Fixed optimum	Reduction
Uniform behavior	Class-prior refined	4.3305	4.7798	9.40%
Uniform syntax	Class-prior refined (mismatch)	4.5081	4.5716	1.39%
Uniform syntax	Alias-aware entropy	4.4014	4.5716	3.72%

The per-budget frontier can cross. For budget k, the largest certifiable mass of any fixed design is $\sum_f m_f - \min_{|A|=k} a_m(A)$. It is obtained in the same exhaustive subset audit. Different budgets may require different, nonnested designs; this is an upper envelope over fixed policies, not the performance of one fixed order. Under uniform behavior, the exact fixed-envelope counts at budgets $2, 3, 4, 5, 6, 8, 9$ are respectively $17, 694, 5256, 8869, 10305, 10952, 10974$. The refined adaptive tree identifies $1, 560, 7044, 10716, 10974, 10974, 10974$ at those budgets. Thus the adaptive tree exceeds every fixed design at budgets four through six, but a budget-specialized fixed design does better at budgets two and three. Its lower mean and optimal worst case do not imply pointwise dominance at every partial budget. This is an observation–certified-coverage frontier, not a claim that all these designs lie on the deployment-runtime frontier.

Reproduction. `scripts/audit_prior_sensitive_frontier_20260916.py` prints the complete prior-specific orders, exact counts, checksums, and independent certificate counts without changing historical results. The compact evidence record is `paper/revision_20260916_independent/prior_frontier_results.json`. `src/tango/exact_fixed_frontier.py` implements the subset audit and mass-aware acquisition. Tests compare its expected optimum with every permutation and its budget envelope with every subset on six small generated instances, and cover singleton rosters, wide integer replies, invalid masses, and duplicate behaviors. Passing these tests is distinct from a human audit of the code and does not authenticate oracle replies or extend the legal language.

J Independent Fixed-Design Audit and the Behavior-Output Cost Boundary

This audit preserves the four-bit bank, horizon eight, all 10,974 behaviors, exact replies, and authentic-oracle assumption. It strengthens the fixed baseline and separately checks whether returning a behavior table really needs source synthesis. Neither addition is a new acquisition algorithm. The fixed frontier is exact for the declared uniform behavior population; deployment measurements are implementation-dependent repeated timings, not complexity lower bounds.

J.1 An independent forward recurrence for every fixed design

Let $a(S)$ be the number of behaviors in nonsingleton cells after projection onto a queried subset S. For any fixed ordering $x_1, \dots, x_k$, the total number of replies consumed with immediate singleton stopping is

$$\sum_{i=0}^{k-1} a(\{x_1, \dots, x_i\}).$$

A forward dynamic program therefore satisfies

$$F(\varnothing) = 0, \qquad F(S) = \min_{x \in S} \big[F(S \setminus \{x\}) + a(S \setminus \{x\})\big]. \tag{19}$$

The optimum under worst-case budget b is $\min\{F(S) : |S| \le b,\ a(S) = 0\}$. The recurrence considers the last query of each possible ordering; induction on $|S|$ proves completeness. A repeated query has a deterministic reply already known, so removing it cannot worsen identification. Every useful fixed policy is consequently covered by an ordering of a subset of the sixteen inputs. Randomization is a mixture of these policies and cannot lower their minimum expected cost.

The implementation evaluates all 65,536 subsets with exact integer objectives. Four-bit outputs in sixteen positions occupy exactly 64 bits; the projection keys are lossless encodings, not potentially colliding hashes. All 524,288 subset-inclusion edges satisfy ambiguity monotonicity. A separately implemented backward recurrence in `src/tango/fixed_design_audit.py` returns the same unrestricted optimum. This is an implementation cross-check, not an independent research-team replication or proof-assistant formalization; both implementations use NumPy and the same independently checked behavior roster.

Table 8: **Exact mean–worst-case frontier for fixed designs.** The means average all 10,974 behaviors uniformly. Larger admissible worst-case budgets beyond ten do not improve the optimal mean.

Budget	Total replies	Mean replies	Attaining fixed order
0–8	—	—	No separating design
9	52,595	4.792692	$(4, 9, 3, 14, 15, 6, 2, 10, 13)$
10–16	52,454	4.779843	$(5, 8, 2, 15, 14, 7, 3, 11, 12, 9)$

The refined adaptive tree consumes 47,523 replies in total, giving $(52454-47523)/52454 = 9.4006\%$ fewer replies than every fixed design in expectation under this prior. This does not establish optimal adaptive mean cost. The prior-sensitive audit in Appendix I evaluates a different weighted objective and also shows why this expected advantage must not be described as dominance at every observation budget.

J.2 A NO-SEARCH CERTIFICATE FOR FULL-DOMAIN BEHAVIOR READOUT

For a finite declared input domain $\mathcal{X}$, query every $x \in \mathcal{X}$ and return the table $g(x) = Y_x$. An independent checker validates input/output ranges, consistent repeated replies, full domain coverage, and equality of the claimed table to the replies. If oracle replies are authentic and deterministic, then $g = f_W$ on the whole task domain. This guarantee does not require enumerating $\mathcal{W}$, computing its quotient, or trusting a proposer state identifier. It is the elementary full-observation baseline, not a new theorem about program synthesis. The code rejects incomplete transcripts and corrupted claims.

This baseline identifies the same semantic object as the partial-observation policies, but does not output a legal word over the original primitive bank. When a legal DSL representative is a required deliverable, extra synthesis must be charged and this readout alone is not a complete solution. Conversely, when the estimand is the behavior table, forcing readout through an exhaustive syntax checker would charge it for a task the specification does not require. The behavior-cost comparison therefore uses a shared behavior output contract and native independent checkers, not an identical-source-output or identical-checker contract. All methods have the same input/reply interface; none authenticates the oracle messages themselves.

For all 10,974 behaviors, the full-domain checker accepts the complete certificate, rejects deletion of its final reply, and rejects a corrupted first entry of the claimed table. Each of the four methods also identifies and certifies every behavior in each of three timed runs. The full-readout checker is independent of the finite-language proposer and exhaustive checker, although these deterministic audits do not establish security against every possible malformed implementation input.

J.3 MATCHED TIMING, UNCERTAINTY, AND OBSERVATION-COST TRADEOFFS

Each repeat randomizes method order and uses a paired random permutation of all behaviors. The partial-observation methods receive their frozen fixed orders or branch-repair recipe for free and pay for graph compilation, policy-tree construction, and independent raw-syntax checker initialization. All methods pay for oracle lookup, acquisition, certificate packaging, and their native checking. Common imports and public metadata handling are excluded; full readout has no additional bank-specific setup. The evaluation roster supplies hidden oracles and is not charged to readout. Discovery of the fixed order, repair recipe, or research proof is excluded symmetrically from these shared-bank deployment measurements, not asserted to be free in an end-to-end discovery workflow.

The zero-latency negative result is strong: full readout is faster in every paired run. Global fixed has the lower median census total than adaptive, but adaptive is faster in one of the three paired runs; no

Table 9: **Repeat-level ranges for the new native-checker experiment.** Each entry is median [minimum, maximum] across three paired repeats, not a confidence interval. Online time includes checking. Initialization and the complete-census total are measured separately.

Policy	Setup (s)	Online (μs)	Census total (s)
Refined adaptive	1.682 [1.679, 2.162]	77.7 [77.4, 84.2]	2.532 [2.531, 3.085]
Global mean-optimal fixed	1.537 [1.468, 1.830]	83.2 [82.8, 116.5]	2.451 [2.377, 3.108]
Minimax-optimal fixed	1.574 [1.466, 1.929]	82.4 [82.4, 123.6]	2.478 [2.370, 3.285]
Full-domain readout	0 additional	11.8 [11.8, 12.0]	0.129 [0.129, 0.132]

uniform timing ranking or statistical significance is inferred. Earlier same-checker measurements in Appendix H are a separate run series, not pooled or silently replaced. The new mean-optimal fixed baseline was not included in that earlier timed cohort.

For a shared-bank cohort of size Q, let S_m be measured setup, $\bar{t}_m$ the full-population mean online time, and $\bar{q}_m$ its mean reply count. Under an external constant per-reply latency λ, use the sensitivity model $C_m(Q, \lambda) = S_m + Q(\bar{t}_m + \lambda \bar{q}_m)$. Against a competitor f with $\bar{q}_f > \bar{q}_a$, the nonnegative crossing for adaptive a is

$$\lambda^*(Q) = \max\left\{0, \frac{(S_a - S_f)/Q + \bar{t}_a - \bar{t}_f}{\bar{q}_f - \bar{q}_a}\right\}. \tag{20}$$

Computing the crossing within each paired repeat and then summarizing gives Table 10. This calculation assumes the reported behavior population and constant reply latency. It neither measures an external service nor guarantees the runtime of an arbitrary sequence of Q targets. In particular, medians of separate quantities must not be substituted for the paired-repeat calculation.

Table 10: **External latency needed to repay adaptive overhead.** Units are milliseconds per reply; entries are the paired-repeat median [range]. Zero means adaptive is already faster in that repeat at zero added latency.

Q	Versus global mean-optimal fixed	Versus full-domain readout
1	474.733 [314.751, 737.823]	144.114 [143.864, 185.246]
1,000	0.463 [0.302, 0.666]	0.150 [0.150, 0.191]
10,974	0.016 [0.000, 0.031]	0.019 [0.019, 0.023]

Figure 5 in the main paper summarizes the same full-census cost records at $Q = 1000$; this appendix provides the underlying crossing formula and complete paired timing table.

J.4 Evidence and reproduction

`scripts/audit_global_fixed_frontier.py--cross-check--repeats3` prints the fixed-design frontier, integer-array checksums, backward-recurrence cross-check, and a same-checker timing audit. `scripts/audit_behavior_output_contract.py--repeats3` prints the separate four-method native-checker timing experiment used in the main table. Run with `PYTHONPATH=.:src` in the project environment. Both scripts are read-only with respect to evidence files. The recorded outputs, retaining every repeat-level result, are `paper/revision_20260916_independent/global_fixed_frontier.json` and `paper/revision_20260916_independent/behavior_output_contract.json`. Historical cost runs remain in their original files. An additional three-repeat execution is stored separately in `paper/revision_20260916_independent/behavior_contract_replication.json`. It again certifies every behavior under all four methods and keeps full readout fastest in each paired run. Its median full-census times are 2.561, 2.492, 2.509, and 0.130 seconds for adaptive, global fixed, minimax fixed, and full readout, respectively. The adaptive–global-fixed crossing at $Q = 1000$ is 0.353 ms [0.254, 0.444], illustrating host/load variation rather than a new exact runtime constant. These repeats are not pooled into the primary tables.

The forward recurrence is in `src/tango/global_fixed_audit.py`; the no-search proposer and checker are in `src/tango/full_domain_certificate.py`. Tests compare every fixed-order permutation and every worst-case budget on small rosters, cross-check the backward recurrence,

and exercise invalid inputs, singleton languages, transcript coverage, conflicting replies, and corrupted claims. Source hashes are included in the evidence records. Human verification of the proofs, code, citations, artifact anonymization, and final submission is separate from these automated results.

K External Scope Test: Adaptive Membership Queries on Knowledge Graphs

We use a frozen UltraQuery executor to construct balanced conditional-collision tasks on FB15k-237 and NELL995. Within each collision group, graph, source, ordered support-source tuple, primitive multiset, candidate roster, checkpoint, and support size are identical; only outcome content varies. Candidate endpoint sets are reachable and pairwise distinct, so the alternatives define distinct behaviors on the declared task probes.

On FB15k-237, visible-only recovery is exactly $.500/.333$ for balanced $2p/3p$ collisions, while two support outcomes recover all candidates. A frozen FB15k-237 checkpoint transferred to NELL995 gives visible-only $.500$ and outcome recovery 1.000 on the zero-shot $2p$ collision set. Permuting outcome content destroys recovery.

For the larger held-out membership test, three seeds yield nine disjoint $K = 24$ groups and 216 balanced tasks. Exact symbolic membership trees have worst-case depth five in every group, attaining the counting lower bound $\lceil \log_2 24 \rceil = 5$. A neural information-gain query policy recovers $.801$ of targets with five bits and $.991$ with eight; random querying at eight bits gives $.079$, and within-collision permuted replies give zero. The $K = 6$ aggregate contains one group that requires four bits despite the three-bit counting bound, showing that probe restrictions can make the counting bound unattainable.

Figure 4 in the main paper displays the complete exact membership tree for one held-out $K = 24$ group. These experiments use the same semantic-identification viewpoint—candidate behaviors are distinguished by their observable consequences—but the large-space certificate theorem in the main paper is specific to the bitvector accumulator language.

L Proof-Carrying Symbolic Identification and External Specifications

This extension changes the representation and the evaluation task, rather than claiming that a larger table census establishes scalability. Distinguishing-input synthesis and oracle-guided component synthesis are established (Jha et al., 2010); independently checking clausal refutations is also established (Wetzler et al., 2014). Our contribution here is their explicit instantiation under the paper's semantic identification contract, with all proof-generation and replay costs measured and failed scalability attempts retained. It is not a new SAT method or a formally verified synthesis system.

L.1 Exact public language and source adaptation

The state is (x, v) with x immutable and $v = x$ initially. Eight total instructions transform v into $v, \neg v, -v, v + 1, v - 1, v \wedge x, v \vee x, v \oplus x$, respectively; arithmetic is modulo 2^b. The output is v. Equivalently, the original transformation formulation operates on (x, v) restricted to diagonal initial states; retaining the known x makes equality of final states equivalent to equality of the returned value. Every word of length zero through L is legal, including repeated instructions. Padding by identity realizes this language with 8^L instruction assignments. This is the count of padded assignments, not the number of distinct behaviors, and neither count is enumerated by inference.

The external oracle specifications are the first eight numbered unary Hacker's Delight tasks in `comp/2017/General_Track` of the public SyGuS repository. The fixed directory tree is `7b9fe3f5112814eda01111e1abce533a1694ea9a`; each original file's immutable Git blob identifier and its extracted reference expression are recorded in `configs/hd_semantic_identification_20260919.json`. They compute, in order,

$$x \wedge (x-1), \quad x \wedge (x+1), \quad x \wedge (-x), \quad x \oplus (x-1),$$
$$x \vee (x-1), \quad x \vee (x+1), \quad (\neg x) \wedge (x+1), \quad (\neg x) \wedge (x-1).$$

We retain the original 32-bit specifications but replace their task-specific expression grammars by the common accumulator language. Task nine requires an arithmetic shift absent from this bank and is not included. The selected list was recorded before the first full comparison, but method development used these tasks: this is exploratory adaptation, not independent preregistration, held-out generalization, or an official SyGuS-track comparison. Each target has a legal representative within the primary horizon four. The identifying routine receives only an execution callable, bit width, horizon, public policy and budgets; the reference expression and task name remain in the evaluator.

L.2 Sound certificate and non-enumerative representation

A candidate w must first be a legal, transcript-consistent word. The checker then reconstructs Eq. (8) from public semantics and Z. A satisfying assignment is an alternative legal program and an input where it disagrees with w. Its absence, established by a checked refutation, implies that every legal behavior consistent with Z equals f_w. Authentic replies and membership of the target in the public language imply that the target is one such consistent behavior. These implications prove Proposition 4. Without the membership assumption, the certificate only identifies the unique in-language explanation; without a consistent representative, UNSAT could merely indicate inconsistent observations. Neither issue is repaired by trusting a solver's status flag.

The implementation allocates two length-L instruction vectors and bit-blasts their executions. It first finds a compatible representative, then fixes one instruction vector to that representative while asking whether the other can disagree at a free b-bit input. Each new observation adds its execution constraints for both vectors. One-hot instruction encoding has constant size per step because the bank has eight members. Bitwise operations and ripple-carry increment/decrement/negation use $O(b)$ gates per step. Tseitin encoding therefore uses $O((|Z|+1)Lb)$ clauses and variables, including both copies and the disagreement guard. This bound concerns constructing the formula, not the difficulty of solving it, refutation size, or worst-case checker time. A full-domain transformation table would instead require $b2^b$ output bits per behavior.

Behavior-preserving normalization. The optimized protocol permits identities only as suffix padding and forbids adjacent NOT–NOT, negation–negation, increment–decrement, decrement–increment, repeated AND-with-input, repeated OR-with-input, and repeated XOR-with-input. The first, second, third, fourth and seventh pairs cancel; repeated AND or OR reduces to one copy. Removing identities and applying these length-decreasing rewrites terminates in a word of no greater length with the same transformation for every (x, v). Appending identity padding gives a representative satisfying every restriction. Thus normalization preserves all original bounded-language behaviors; it is not a favorable replacement target class. It does not preserve syntax probabilities, and no syntax-prior claim is made for this SAT policy. The checker additionally verifies that a fixed reference obeys the same normal-form restrictions. Otherwise fixing an illegal normalized reference would create vacuous UNSAT; an explicit negative test covers this case.

Solver-free refutation replay. Glucose3, exposed by PySAT, emits a DRUP log. For every added clause c, our checker requires unit propagation to derive a contradiction from the current formula together with the negations of all literals in c (the RUP rule). Induction makes every accepted addition a logical consequence. A checked empty clause then proves UNSAT. Deletion commands may be ignored soundly because retaining previously implied clauses only strengthens propagation. Unchecked RAT-only steps, missing empty clauses, invalid literals, malformed inputs and timeouts fail closed. The checker does not invoke a SAT or SMT solver.

A small Python implementation and a separate C++17 replay implementation share this contract, not search state. Both rebuild the input formula through the same deterministic Python circuit compiler. The compiler and replay code are trusted components: they are not proof-assistant verified or independently reviewed by a second research team. Exhaustive primitive tests on widths one through four, small-language miter checks, normalized/raw behavior equality at small widths, randomized SAT/RUP cross-checks, and certificate corruption tests provide implementation evidence, not a formal compiler-correctness proof. An additional Z3 expression encoding compares each returned program to its hidden reference over every 32-bit input. This is evaluator-only cross-validation; it is not available to the identifying policy and is not the deployed certificate.

Table 11: **External 32-bit specifications; same legal-program output.** Eight tasks, three paired timing repeats. Means include encoding, search, fresh proof production, serialization and independent replay. Imports, one-time checker compilation and evaluation-only reference equivalence are excluded. All methods use the same normalized language and native RUP checker.

Policy	Certified	Replies	Total (s)	Check (s)
Incremental witness	24/24	2.375	0.747	0.345
Restarted witness	24/24	2.750	0.883	0.407
Fixed edge probes	24/24	4.375	0.999	0.509
Seeded random probes	24/24	3.458	0.992	0.512
Fixed eight-probe batch	24/24	8.000	1.383	0.779

L.3 Matched policy and proof costs

All five primary methods use the same public normalized language, exact scalar oracle, returned legal-program contract, and native independent checker. Incremental witness acquisition retains the SAT instance as observations arrive. Restarted witness acquisition rebuilds the SAT instance from accumulated constraints after each observation; it still reuses the public encoding and is not a no-compilation-cache baseline. These policies may choose different witnesses, so their runtime difference is not a transcript-controlled causal estimate of learned-clause reuse. Fixed edge probes use zero, all ones, one, all ones minus one, alternating masks and complements, and the top-bit boundaries, followed by powers of two and seeded filler probes. Random probes are uniform without replacement. These are declared useful fixed comparators, not globally optimal fixed designs over 2^{32} available inputs. The batch comparator takes its first eight edge probes before its first identification check; it does not pay for checks at all eight prefixes.

For every method the final refutation is generated by a fresh proof-producing solver initialized with the public transcript formula and the fixed reference. That additional proof search is timed. The identifying SAT state may still be reused between observations. All methods pay for encoding, solving, oracle calls, proof packaging, serialization, checker process startup, formula reconstruction and RUP replay. Common imports, the one-time native-checker build and evaluator-only reference checking are excluded. No artificial reply latency is introduced. The CPU timings are host/load dependent; neither peak RSS nor a remote service's latency is measured. The primary record retains all 120 task-level total times and query counts, grouped by eight targets and three repeats.

Table 11 reports arithmetic means across these paired complete-task runs. Three timing repeats do not create 24 independent specifications. Random-probe sequences vary by repeat, while the other declared policies have deterministic transcripts. Incremental acquisition uses two or three replies on these targets, not a proved worst-case bound over the entire legal language. Its mean internal cost is lower than each tested comparator's aggregate, but individual target comparisons can reverse. No population-level significance or superiority over a separately engineered oracle-guided system is inferred.

L.4 Retained development and scaling failures

The first unnormalized, accumulated-proof comparison also certified all eight tasks with each of four methods. However, incremental witness acquisition used 3.125 replies and 1.455 seconds on average, versus 2.875 and 1.166 for restarting. Thus naive retention of learned solver history was worse. A Python-only replay pilot additionally exposed high checking costs. Normalization and native replay were applied to all methods, and fresh terminal proof construction was charged rather than treated as free. The final and earlier means are separate run series, not a controlled attribution of the improvement to any single optimization.

At horizon six, with the same 50,000-conflict per-call limit, only task hd04 is certified: four replies, 14.231 seconds total, including 8.817 seconds of checking, and an 8,216,514-byte proof log. Seven tasks return UNKNOWN. At horizon eight, all eight return UNKNOWN. Their $8^8 = 16,777,216$ padded assignments are not materialized, but a succinct formula has not solved the combinatorial problem. An additional two-task horizon-eight pilot requested fixed fallback observations after UNKNOWN; both still abstained at sixteen observations. Those extra replies were not misreported

as certificates. Resource-limited failure is not a proof of impossibility, and no general long-program scalability is claimed.

L.5 Reproduction and evidence locations

The experiment used Python 3.10.20, PySAT 1.9.dev15 and Z3 5.1.0. With these packages and a C++17 compiler installed, run:

```
export PYTHONPATH=.:src
g++ -O2 -std=c++17 src/tango/rup_check.cpp -o tmp/rup_check
python scripts/benchmark_symbolic_identification.py \
  --methods incremental rebuild fixed_edges random batch8 \
  --horizon 4 --normalize --proof-mode fresh --repeats 3
```

The runner prints task transcripts, candidate words, proof hashes, statuses and costs. Add `--include-proof` to export complete regenerated certificates; compact archived records contain checksums rather than all raw proof bytes. `paper/revision_20260919_symbolic/primary_results.json` retains the primary numerical records and source hashes; `horizon_limits.json` in that directory retains every horizon-six/eight outcome and the fallback pilot; `development_results.json` retains the earlier negative comparison. `tests/test_symbolic_identification.py` exercises the encoding and both solver-free replay implementations. These automated checks do not replace author verification of source provenance, proofs, implementation and submission.

M A Linear Width Cutoff for a Fixed Accumulator Language

The result in this section is an analytic reduction of the identification and verification domain. It is not extrapolation from successful small-width tests. The public bank is fixed: identity, NOT, negation, increment, decrement, and AND/OR/XOR with an immutable input register x. The accumulator initially equals x; the returned value is its final contents. Arithmetic is modulo 2^B. Words have no width-dependent constants or instructions. Let $a(w)$ count the negation, increment, and decrement occurrences in w.

M.1 Streaming semantics and a decreasing state

Read the input bits from least to most significant. Every arithmetic occurrence has its own carry or borrow bit, initialized to one. Writing v for the bit entering that occurrence and c for its current carry, its output and next carry are

instruction	output bit	next carry/borrow
inc	$v \oplus c$	$v \wedge c$
dec	$v \oplus c$	$(\neg v) \wedge c$
neg	$(\neg v) \oplus c$	$(\neg v) \wedge c$.

Negation is bitwise complement followed by addition of one. The bitwise instructions have no state: at a given bit position they combine the accumulator bit with that position's immutable input bit. Process instructions in word order at each position, using each occurrence's old carry to compute its output and storing its new carry for the next position. Induction over bit positions and then over instructions proves equality with ordinary modular execution. Every carry satisfies $c' \leq c$, so the joint state can change at most $a(w)$ times along any input. Arbitrarily many positions can be self-loops, but cannot create a new carry once one is zero.

Proposition 6 (A shortest witness in a decreasing-state product)**.** *Consider two deterministic, time-homogeneous, letter-to-letter transducers. Suppose the reachable product has a nonnegative integer rank, initially at most h, which strictly decreases on every non-self-loop transition. If some input distinguishes their output streams, an input of length at most $h + 1$ does so.*

Proof. Choose a shortest distinguishing input and stop at its first differing output symbol. No earlier step can be a product self-loop: deleting that input symbol would leave the state before the remaining

suffix unchanged, preserve the later mismatch, and give a shorter witness. Hence every step before the final mismatch strictly decreases rank. There are at most h such steps, followed by the one mismatching output symbol. The final transition need not decrease rank. □

For two accumulator words, use the sum of their carry bits as rank. The product starts with rank $a(u) + a(v)$. Monotonicity makes every changed product state strictly decrease this rank. This bounds witness length by the number of arithmetic occurrences, not by the possibly exponential number of product states. The proposition is a simple finite-state distinguishability argument; we do not claim a new general automata decision procedure.

Theorem 5 (Length-dependent, width-independent equivalence cutoff)**.** *If two words u, v in the stated bank differ at any bit width, they differ on some input of width at most $a(u) + a(v) + 1$. Consequently, for all words of length at most L and every $B \geq k = 2L + 1$,*

$$f_u^{(B)} = f_v^{(B)} \iff f_u^{(k)} = f_v^{(k)} \iff f_u^{(b)} = f_v^{(b)} \text{ for every } b \geq 1.$$

The cutoff is sufficient; no tightness or optimality claim is made.

Proof. The first assertion follows from Proposition 6 and the streaming construction. Every bank instruction commutes with low-bit projection, and hence so does every word:

$$f_w^{(B)}(x) \bmod 2^b = f_w^{(b)}(x \bmod 2^b), \qquad b \leq B. \tag{21}$$

If two words of lengths at most L differed at any width, their short witness would have width at most $a(u) + a(v) + 1 \leq 2L + 1 = k$. Zero-extending that input to width k, and then to B, preserves its differing low output bit by Eq. (21). Thus equality at either k or B excludes any difference. Equality at every width trivially implies the other equalities. □

Stabilization of the behavioral quotient. For the same legal words at every width and $B \geq 2L+1$, restriction to the low $2L + 1$ bits induces a bijection between the deployment-width and cutoff-width behavioral quotients. Surjectivity follows by executing the same word; injectivity is Theorem 3. Thus the quotient stabilizes while the input domain grows. This does not enumerate the classes for free, minimize queries, or preserve the information in each full-width reply.

M.2 From a narrow refutation to a full-width identification certificate

Fix the externally declared deployment width B, bank, and horizon L. Let $k = \min(B, 2L + 1)$. A real oracle query uses a legal B-bit input and returns a full B-bit reply. Symbolic search uses only the projected transcript $\overline{Z} = \{(x \bmod 2^k, y \bmod 2^k) : (x, y) \in Z\}$. The returned certificate consists of Z, a legal representative w, and a solver-free-checkable RUP refutation of the narrow disagreement formula

$$\mathrm{Legal}_L(\theta) \wedge \bigwedge_{(\bar{x},\bar{y})\in\overline{Z}} E_\theta^{(k)}(\bar{x}) = \bar{y} \wedge E_\theta^{(k)}(z) \neq E_w^{(k)}(z),$$

where z is a symbolic k-bit input. The checker binds B, L and the bank to external public arguments, reconstructs the narrow formula, verifies the proof, and scalar-replays w against every *full* reply in Z. It rejects an attempt to lower k, change the horizon or bank, mismatch the projected transcript, or modify a high reply bit. A legal, consistent representative is required to exclude vacuous uniqueness from an empty version space.

To prove soundness, let u be the actual legal target. Authentic replies and Eq. (21) make u satisfy the narrow observations. The refutation therefore forces $f_u^{(k)} = f_w^{(k)}$. If $k = B$ this already is the required behavior; otherwise Theorem 3 lifts equality to B and all widths. The projected version space may contain extra programs that fail the full replies. Proving uniqueness in this *superset* is sound, although it can require more queries. Low-bit projection need not preserve entropy scores, query optimality, or the original version-space cardinality.

For a fixed bank, the narrow circuit has $O((|Z|+1)L\min(B, 2L+1))$ clauses and variables. Neither the input domain nor the behavioral quotient is enumerated. This is a formula-size bound, not a polynomial bound on SAT search, refutation length, checking time, or memory. The implementation conservatively runs the existing narrow checker once inside identification and once in the transfer wrapper; both calls remain charged in the end-to-end table. The separate terminal-proof ablation checks each proposed certificate only once on each side.

What the theorem excludes. General addition of two changing registers can regenerate a carry; shifts, rotations, multiplication, arbitrary constants, signed comparisons, and width-dependent instructions do not satisfy the present proof. Adding any such primitive requires a new analysis. Even in this bank, testing an arbitrary smaller width is unsound: 2^b increments agree with identity at width b but not at width $b + 1$. Our test suite includes this counterexample; it does not establish that the conservative $2L + 1$ bound is tight.

M.3 Paired observations, total cost, and a controlled proof ablation

The primary experiment retains the eight public 32-bit specifications and the common length-at-most-four language from Appendix L. It is an exploratory continuation, not eight independent application domains. Every method receives full 32-bit replies and returns a legal source representative plus independently checked uniqueness. Each repeat shuffles all task–method pairs. Three repeats measure timing variation, not 24 independent specifications. All 120 attempts certify, and every returned word passes the separate evaluation-only 32-bit Z3 equivalence check against its reference.

Table 1 reports all five methods. The mean of the three complete eight-task timing means is reported. Their observed ranges are $[0.797, 0.827]$ seconds for native-width witness search, $[1.034, 1.094]$ for native fixed probes, $[0.303, 0.306]$ for cutoff witness search, $[0.331, 0.338]$ for cutoff fixed probes, and $[0.298, 0.310]$ for cutoff search with solver rebuilding. These are repeat ranges, not confidence intervals. The same public edge-panel recipe is instantiated at each proof width before execution; all queries remain valid inputs in the common 32-bit interface. The panels are not asserted to be optimal fixed designs.

Relative to native witness acquisition, cutoff acquisition is $2.66\times$ faster in aggregate but uses 3.125 rather than 2.375 replies. Relative to fixed probes with the *same cutoff*, it uses 28.6% fewer replies and 9.25% less total time. The latter advantage is not uniform across tasks. Incremental and rebuilt cutoff runs have virtually equal aggregate times, so these measurements do not establish a general solver-history reuse benefit. Communication is charged at $32|Z|$ bits, not $9|Z|$; the latter counts only bits consumed by symbolic reasoning. No artificial execution delay is added.

To separate proof reduction from changes in acquisition, a second audit freezes the same eight legal words and full transcripts produced by cutoff search and rebuilds the terminal proof at both 32 and nine bits. All 48 attempts pass. Mean encoding-plus-proof-production-plus-checking time is 0.6705 seconds at 32 bits and 0.1501 seconds with transfer, a $4.47\times$ ratio. Mean proof bytes fall from 445,564.5 to 79,556.75. This is a terminal-certification ablation, *not* end-to-end identification timing or a held-out benchmark.

M.4 The remaining horizon limit and a large-width control

At horizon six, a separate paired sweep with the same per-call 50,000-conflict cap certifies one of eight tasks with native-width inference and all eight with the thirteen-bit cutoff. Mean all-attempt times are 4.773 and 8.628 seconds: native-width failures return earlier, so these unequal-coverage means must not be called a speed-to-success comparison. Both average 4.5 observed replies. At horizon eight, all eight cutoff runs return UNKNOWN and abstain. A proved width bound does not remove difficult search over long programs.

Finally, lifting the same eight expressions to 256-bit modular arithmetic, while keeping horizon four, preserves the nine-bit proofs and certifies all eight programs. Mean total time is 0.3102 seconds and mean replies 3.125; the RUP proof hashes exactly match those at 32-bit deployment. These are generated width variants, not unchanged original SyGuS tasks or new independent examples. The all-width conclusion follows from the theorem, not from these eight tests.

M.5 Prior work, implementation checks, and reproduction

Width-independent verification already has substantial prior art. Niemetz et al. (2021) translate parametric-width formulas into quantified integer reasoning. Bhat et al. (2025) mechanize automata/model-checking procedures for width-independent bitvector predicates, including Hacker's Delight verification. We do not claim the first all-width method, a more general decision procedure, or a Lean-verified implementation. Our narrower contract gives an explicit $2L + 1$ cutoff *uniform*

over an unknown bounded program language, then uses actual executions to identify a behavior and supplies a replayable narrow refutation. No runtime comparison to those verification systems is made because they are not black-box acquisition protocols.

`src/tango/carry_cutoff.py` implements the transducer, cutoff, and bound checker. `tests/test_carry_cutoff.py` compares all 585 words of lengths zero through three on all sixteen four-bit inputs, checks 300 seeded longer-word pairs using an independent transducer-product search, tests projection and naive-width counterexamples, and rejects altered bounds, banks, high replies, and invalid proofs. These tests are not formal verification of the theorem or the CNF compiler; responsible authors must review both. A separate audit regenerates all eight 32-bit transferred certificates with matching narrow-proof hashes. It rejects seven mutation classes on every task: wrong deployment width, wrong horizon, smaller proof width, changed bank, corrupted high reply bits, an invalid empty refutation, and deletion of the final observation from both transcript views (56 rejections in total). Run `scripts/audit_cutoff_certificate_binding.py` to reproduce these checks.

With `PYTHONPATH=.:src`, run `scripts/benchmark_carry_cutoff.py` with `--horizon 4 --repeats 3 --compact` for the primary study; `--horizon 6 --repeats 1 --methods native cutoff --compact` for the paired horizon sweep; and `--horizon 8 --repeats 1 --methods cutoff --compact` for the retained boundary. The 256-bit control adds `--width 256 --horizon 4 --repeats 1 --methods cutoff --compact`. `scripts/audit_same_transcript_cutoff.py--repeats3` reconstructs the controlled terminal-proof ablation. The runner's `--include-proof` option exports full certificates; archived compact records retain numerical results and proof hashes, not every proof line. Evidence is in `paper/revision_20260919_cutoff/`. The initial projected fixed panel exhausted its distinct probes prematurely; that comparator bug is recorded in `development_deviations.json` and was corrected before the complete primary study was rerun. Historical timings and horizon-six pilot records are retained separately and are not pooled.