WHEN IS PROGRAM STRUCTURE REQUIRED? CERTIFICATE-GROUNDED CALIBRATION OF PROGRAM-INDUCTION DIFFICULTY

Anonymous authors
Paper under double-blind review

ABSTRACT

Program-induction benchmarks require a distinction between the structure of a generator and the structure demanded by its examples. We introduce certificate-grounded calibration for programming-by-example: complete one-rule enumeration and a two-rule witness certify a change in the shortest legal solution from one rule to two. Paired specifications share a reference program, contain six examples each, and exactly match five edit features of the changed example. On 120 new validation families, Qwen3-8B and Olmo-3-7B-Think exhibit displayed-task success decreases of 45.83 and 30.83 percentage points alongside gains of 7.50 and 9.17 points on a common nine-example audit. A supplemental analysis of a shared, wholly unshown witness pair finds gains of 9.17 and 11.67 points. An independent bounded symbolic solver solves all controls with one-rule programs, whose unseen-audit errors follow from the construction; on separation, it solves 95/120 tasks and passes the unshown audit on 87/120. These results distinguish a certified requirement, the cost of finding and delivering a solution, and agreement with a specified unseen target. The witness gains do not establish improved accuracy on broader rule-active inputs; separate readouts locate failures in final-answer delivery.

1 INTRODUCTION

A benchmark's generating program and its required solution structure are different objects. A two-rule reference can produce examples that a single replacement also satisfies. The solver therefore confronts a class of consistent programs, while the benchmark label describes one particular generator. Measuring program-induction difficulty requires separating these objects.

Ambiguity among example-consistent programs is fundamental to programming-by-example (PBE) (Gulwani, 2011; Gulwani et al., 2017). In model evaluation, shortcut behavior and construct mismatch similarly separate a benchmark score from the capability it is taken to represent (Geirhos et al., 2020; McCoy et al., 2019; Jacobs & Wallach, 2021). Benchmark validity requires a justified link between the task and its intended construct (Bowman & Dahl, 2021). We establish that link through a verifiable solution requirement and then measure its behavioral consequences.

We ask whether one can *certify a change in required program length before inference*, while keeping the number of shown examples fixed. For a restricted string-replacement language, the answer is constructive. We enumerate all one-rule programs satisfying five base examples. A control addition preserves at least one; a matched separation addition excludes every one-rule solution while preserving a known two-rule reference. The minimum legal lengths are thus exactly one and two. The additions have equal input length, output length, edit distance, common-prefix length, and nonoverlapping common-suffix length.

A distinguishing example changes both the admissible solutions and the information available to the solver. We therefore score the same selected final program against two targets: its own displayed specification and a common, broader finite audit. After two discovery studies, independent validation finds a large completion cost and a positive common-audit change for two models. Offline supplements localize the latter on examples unseen in both conditions and contextualize it with a

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.

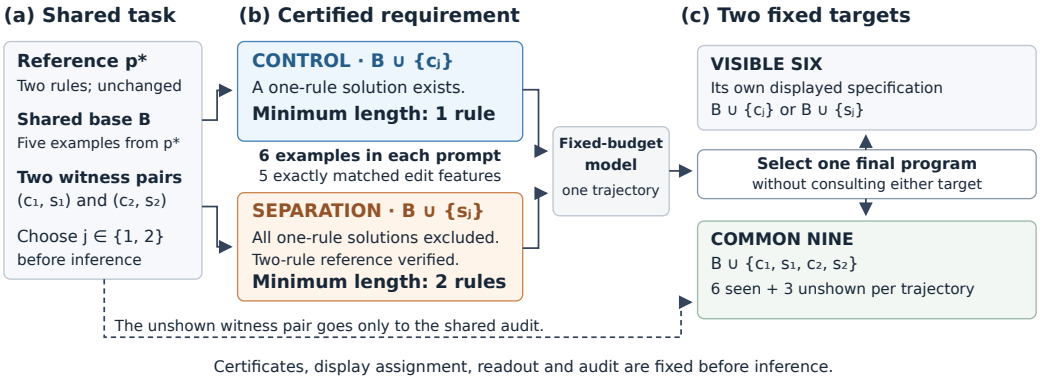


Figure 1: **Certify a requirement, then measure behavior.** A family supplies a shared reference, five base examples, and two witness pairs. One pair is assigned for display before inference. Control retains a one-rule solution; separation excludes all of them. A target-independent selector chooses one final program, which is executed on both the visible six and common nine examples. The three audit-only examples are never shown to the model. The supplemental H_2 target uses the two examples in the entirely unshown witness pair.

symbolic search baseline. A further distinction arises at evaluation: a correct candidate may appear in a trajectory without being delivered as the final answer.

The symbolic baseline exposes inductive preference: its one-rule control solutions violate the separating witnesses by construction, whereas the two-rule reference satisfies them. On separation, bounded symbolic search finds substantially more solutions than the evaluated language-model protocols.

We contribute *certificate-grounded calibration*: a complete short-class certificate, matched specifications with fixed example counts, and independent-family evaluation of completion, audit agreement, and delivery. This links benchmark construction to a verifiable solution requirement and makes the evaluated system's response a separate empirical question. The symbolic and supplemental analyses clarify how solution preferences, output selection, and audit inputs affect that response.

2 Related Work

Synthesis and exclusion. FlashFill resolves ambiguities in string transformations from examples (Gulwani, 2011). Counterexample-guided synthesis in SKETCH (Solar-Lezama et al., 2006; Solar-Lezama, 2008), syntax-guided synthesis (Alur et al., 2013), and query learning (Angluin, 1987) use evidence to restrict candidate programs. SemGuS supports user-defined semantics and synthesis or unrealizability reasoning through constrained Horn clauses (Kim et al., 2021). Building on this tradition, we completely exclude a specific short-program class to construct matched benchmark items before model evaluation. Richer languages motivate corresponding completeness arguments and decision procedures.

Neural induction and executable benchmarks. DeepCoder guides symbolic search with learned program properties (Balog et al., 2017); RobustFill learns string programs from examples (Devlin et al., 2017). ExeDec uses execution subgoals to decompose synthesis (Shi et al., 2024), while Latent Programmer organizes search through discrete latent program codes (Hong et al., 2021). These approaches improve synthesis; our construction certifies task requirements. ReaComp compiles model reasoning into symbolic solvers (Naik et al., 2026b). Our certificate operates at benchmark construction, complementing these synthesis methods. PBEBench controls generated replacement cascades and already analyzes shorter predicted programs and bypassed relations in Appendices F.10–F.11 (Naik et al., 2026a). A generator relation labels one reference, and an output detector labels one prediction; our certificate quantifies over *every* legal one-rule solution. ReplaySCM uses executable mechanisms, counterexamples, and held-out intervention replay (Batzoglou, 2026); F-ICL provides

an exact bounded-program reference and audits runtime and template effects (Ozelim & Zenil, 2026). The distinction here is necessary-length certification under equal-count, surface-matched specifications, followed by new-family validation.

Measurement and finite-budget behavior. Annotation artifacts (Gururangan et al., 2018), HANS (McCoy et al., 2019), CheckList (Ribeiro et al., 2020), and contrast sets (Gardner et al., 2020) motivate tests beyond aggregate accuracy. Measurement modeling makes the link between scores and constructs explicit (Jacobs & Wallach, 2021). Chain-of-thought prompting and broad capability evaluations motivate attention to multi-step behavior (Wei et al., 2022; Srivastava et al., 2023); prefix interventions study harmful continued reasoning (Caldarella et al., 2026). Our prefix analysis measures the development of candidates and final answers within already generated trajectories.

3 Certifying Required Program Length

3.1 Language and complete one-rule enumeration

Let Σ be a finite alphabet. A rule $\mathrm{replace}(a, b)$ replaces all nonoverlapping occurrences of the nonempty string a by b in one pass, using Python string-replacement semantics. Let $\mathcal{P}$ be the set of ordered programs of one to five legal rules over Σ. Experiments use $\Sigma = \{a, \ldots, k, u, v, w, x, y, z\}$ and $1 \le |a| \le 3$, $0 \le |b| \le 3$. The proof below applies to any fixed finite bounds k_a, k_b under the same semantics.

For specification $E = \{(x_i, y_i)\}_{i=1}^{n}$, write $p \models E$ if $p(x_i) = y_i$ for all pairs, and define

$$L_{\min}(E) = \min_{p \in \mathcal{P}:\, p \models E} |p|. \tag{1}$$

All constructed specifications include a nonidentity mapping and a valid two-rule reference, so this minimum exists. Let $\mathrm{Sub}_{1:k}(X)$ collect substrings of lengths one through k from strings in X. We execute every pair in

$$A(E) \times B(E), \quad A(E) = \mathrm{Sub}_{1:k_a}(\{x_i\}), \quad B(E) = \mathrm{Sub}_{1:k_b}(\{y_i\}) \cup \{\epsilon\}, \tag{2}$$

and retain the exact one-rule solution set $\mathcal{S}_1(E)$.

Lemma 1 (One-rule completeness)**.** *For a specification with a nonidentity mapping, this enumeration includes every legal one-rule solution.*

Proof. If $\mathrm{replace}(a, b) \models E$, it acts on a nonidentity example, so a occurs in an input and belongs to $A(E)$. An empty b is explicitly enumerated. Otherwise, the inserted b remains a contiguous substring of that single-pass output and belongs to $B(E)$. The pair is enumerated and passes every example. Conversely, each retained pair is legal and directly checked on E. □

Proposition 1 (Certified one-to-two switch)**.** *If nonidentity specifications E_C, E_S share a valid two-rule reference, $\mathcal{S}_1(E_C) \neq \varnothing$, and $\mathcal{S}_1(E_S) = \varnothing$, then $L_{\min}(E_C) = 1$ and $L_{\min}(E_S) = 2$.*

Proof. A retained one-rule program attains the control minimum. Lemma 1 excludes every one-rule separation solution; the reference attains length two. □

The proof relies on the replacement string surviving in a single-pass output. Intermediate strings in longer programs require a separate completeness argument (Section 6).

3.2 Matched specifications and witness pairs

A family f comprises a two-rule reference p_f^* and five base examples B_f. We use the PBEBench generator's feeding-only reference class and require a nonidentity base with $\mathcal{S}_1(B_f) \neq \varnothing$. A witness is an additional pair produced by p_f^*. A control witness preserves a base one-rule solution; a separation witness eliminates all of them.

For a witness (x, y), we match

$$s(x, y) = (|x|, |y|, d_{\mathrm{edit}}(x, y), \ell_{\mathrm{prefix}}, \ell_{\mathrm{suffix}}), \tag{3}$$

where the common suffix excludes overlap with the prefix. Each accepted family has two strictly matched pairs $(c_1, s_1), (c_2, s_2)$, all nonidentity, with four distinct inputs absent from the base. A predetermined $j \in \{1, 2\}$ yields

$$E_{fC} = B_f \cup \{c_j\}, \qquad E_{fS} = B_f \cup \{s_j\}. \tag{4}$$

Both prompts contain six examples. Every resulting specification is independently re-enumerated, and the reference is checked on all witnesses. Candidate generation, tie breaking, and eligibility are fixed before model evaluation (Appendix A).

The matched intervention preserves the reference, base, example count, and five surface features. Exchanging the sixth example changes the admissible program class together with the identifying information supplied to the solver. The behavioral comparison measures this specification-level intervention.

A concrete certificate. For the reference $\mathrm{replace}(\mathtt{ex}, \mathtt{azw})$ followed by $\mathrm{replace}(\mathtt{wa}, \mathtt{y})$, the five base examples admit exactly one one-rule solution, $\mathtt{exa} \to \mathtt{azy}$. The two witnesses below share signature $(6, 6, 3, 3, 0)$. Control preserves that solution; separation excludes it while retaining the two-rule reference. Appendix A gives all nine mappings.

Witness	Input	Reference output	One-rule output
Control	`iiiexa`	`iiiazy`	`iiiazy`
Separation	`wbawex`	`wbayzw`	`wbawex`

3.3 What a short-solution preference implies

The construction gives a useful prediction about solver behavior, independently of language models.

Corollary 1 (Audit failure of a one-rule control solution)**.** *Any one-rule program satisfying E_{fC} fails each separation witness s_1, s_2. It therefore fails both the common-nine audit and the wholly unshown witness pair defined below.*

Proof. Such a program belongs to $\mathcal{S}_1(B_f)$. Each separation witness was chosen to reject every member of that set. Both audits contain a separation witness. □

A solver that returns the first consistent program in shortest-length order must consequently fail these control audits whenever it returns a one-rule solution. The reference p_f^* satisfies all witnesses, making the role of the shortest-first selection policy explicit. The distinction separates ambiguity of the finite specification from the inductive preference used to resolve it.

4 Study Design and Measurement

4.1 Validation and offline supplements

Two discovery studies used 60 families and 480 outputs each. The second introduced a post hoc common-nine analysis, which motivated a prospectively fixed audit in the independent validation. Validation contains 120 new reference families accepted from 15,004 proposals. A normalized program-identity rule excludes earlier references, proposals, predicted programs, and constructive alternatives. Validation estimates use the new families alone. The base one-rule solution count is one, two, or three in 91, 13, and 16 families, respectively; its minimum, median, and maximum are 1, 1, and 3. Uniqueness here concerns only the one-rule subclass.

Witness assignment and first-executed condition are each balanced 60/60; paired prompts use the same example permutation. Families, prompts, primary audit targets, model revisions, seeds, readouts, and bootstrap indices were fixed before inference. The post hoc supplements reuse these 720 outputs, with analysis plans fixed before supplemental scoring.

4.2 MODELS, INFERENCE, AND FINAL SELECTION

We evaluate Qwen3-8B (Yang et al., 2025), Olmo-3-7B-Think (Team Olmo et al., 2025), and Phi-4-mini-reasoning (Xu et al., 2025) at fixed revisions. All use BF16 weights, temperature 0.6, top-p 0.95, and a cap of 8,192 new tokens. Qwen uses top-k 20 and batch size eight; Olmo and Phi disable top-k and use batch size four. One sampled trajectory is retained per model–family–condition cell. The fixed instruction requests a legal cascade and a final Python list, permitting reasoning first. Inputs contain the displayed examples and the shared synthesis instruction.

The evaluation separates answer delivery from candidate availability. *Native delivery* uses the original code-block extractor. *Canonical final execution*, fixed before validation, maps a finite list of equivalent rule notations without changing their strings or order. For Qwen and Olmo it selects the last complete Python fence after the final closed thinking delimiter, falling back to the last complete list only when no such fence exists. A selected object outside the legal grammar receives zero success, even if an earlier object is correct. Phi uses the same grammar with a frozen endpoint adapter (Appendix C). Program selection precedes target execution. Finally, *candidate availability* checks whether any admissible program expressed anywhere in the trajectory satisfies a target; it provides a separate diagnostic alongside the final score. All truncated, unparseable, and incorrect outputs remain in the denominator.

4.3 TWO PRIMARY TARGETS AND ONE SUPPLEMENTAL UNSHOWN TARGET

Let $\hat{p}_{fc}$ denote the selected final program when one is available. Its primary outcomes are exact success on its displayed six examples and on

$$E_f^{\mathrm{com}} = B_f \cup \{c_1, s_1, c_2, s_2\}. \tag{5}$$

The nine-example audit is identical across conditions and contains six shown plus three unshown examples per trajectory. Common success implies visible success. The supplemental target

$$H_{2,f} = \{c_{3-j}, s_{3-j}\} \tag{6}$$

is also identical across conditions, but both examples are wholly absent from both prompts. The H_2 predicate evaluates this pair independently of displayed-task correctness.

For target T, define $Y_{fc}^T = \mathbf{1}[\hat{p}_{fc} \models T]$, with selection failures scored zero. We estimate the paired difference

$$\widehat{\theta}_T = \frac{1}{120} \sum_f (Y_{fS}^T - Y_{fC}^T). \tag{7}$$

For visible success, T is the condition's own E_{fc}; the other targets are common. Each contrast measures its own success predicate.

We resample family IDs 50,000 times, retaining both conditions and using one archived index array (Efron, 1979; Dror et al., 2018). The six planned model–target contrasts retain percentile quantiles $1/240, 239/240$. The three post hoc H_2 contrasts use $1/120, 119/120$. These Bonferroni-style adjustments provide approximate simultaneous coverage. Other supplements use labeled marginal intervals. The predeclared effect-size criteria are reported in Appendix E; H_2 remains a separate supplemental target.

5 RESULTS

5.1 A SUBSTANTIAL COMPLETION COST ON NEW FAMILIES

Certified separation produces a substantial completion cost (Table 1). Qwen's displayed-task success decreases from 67/120 to 12/120, or -45.83 percentage points; Olmo's decreases from 52/120 to 15/120, or -30.83 points. The adjusted intervals exclude zero. At the family level, Qwen has 55 control-only successes and no separation-only successes; Olmo has 42 and five. Each family contributes one equally weighted binary pair.

For the matched specification intervention, the certified requirement change accompanies a large decrease in displayed-task success. Higher cap rates under separation characterize the same finite-budget response (Section 5.4).

Model	Target	Control	Separation	$S - C$ (pp)	Adjusted interval
Qwen	Visible six	67/120	12/120	-45.83	[-57.50, -34.17]
	Common nine	0/120	9/120	+7.50	[+1.67, +14.17]
Olmo	Visible six	52/120	15/120	-30.83	[-44.17, -17.50]
	Common nine	0/120	11/120	+9.17	[+3.33, +16.67]
Phi[†]	Visible six	5/120	2/120	-2.50	[-8.33, +3.33]
	Common nine	0/120	1/120	+0.83	[+0.00, +3.33]

Table 1: **Primary validation results.** All conditions have 120 families. Differences are separation minus control, with approximate simultaneous intervals for the six planned contrasts. The nine-example audit is common to both conditions. All Phi results use the common implementation documented in Appendix D; its intervals include or touch zero.

Model	Control	Separation	$S - C$ (pp)	Adjusted interval
Qwen3-8B	0/120	11/120	+9.17	[+3.33, +15.83]
Olmo-3-7B-Think	0/120	14/120	+11.67	[+5.00, +19.17]
Phi-4-mini[†]	1/120	1/120	+0.00	[-2.50, +3.33]

Table 2: **Shared, wholly unshown H_2 audit.** Both examples are absent from both prompts. All 120 families remain in each denominator. These supplemental post hoc intervals are approximately adjusted across three model contrasts, separately from the primary six-comparison analysis.

5.2 Agreement improves on common, including wholly unshown, targets

The same outputs show positive common-nine differences: Qwen rises from 0/120 to 9/120, and Olmo from 0/120 to 11/120. Their adjusted intervals, $[1.67, 14.17]$ and $[3.33, 16.67]$ percentage points, support positive transfer on the common audit. Both point estimates are below the predeclared ten-point benchmark, although their adjusted intervals exclude zero (Appendix E). A post hoc paired exact test corroborates the direction after sixfold correction ($p = .02344$ and $.00586$).

The gains extend to a witness pair absent from both prompts. On H_2, Qwen changes from 0/120 to 11/120 and Olmo from 0/120 to 14/120 (Table 2). These outcomes depend solely on examples absent from both prompts. The witnesses were constructed before inference, but this isolated analysis is post hoc on the same families. The H_2 predicate has fewer requirements and its own comparison family. Two H_2-successful separation outputs from each model fail their displayed specification.

Witness-level outcomes make the transfer interpretation concrete. Of Qwen's 67 visible-correct controls, 60 pass the base and both control witnesses but fail both separation witnesses; the corresponding Olmo count is 49/52. All 21 common-nine successes across the three models use alternative rule sequences: 20 have two rules and one has another legal length. They establish *agreement with the finite target* through alternative programs (Appendix F). A post hoc split into 91 single-solution and 29 multiple-solution bases retains both Qwen/Olmo directions in each group. Appendix F.1 reports all models and marginal paired-family intervals for this descriptive grouping.

5.3 An independent symbolic baseline separates selection from solving

A bounded symbolic solver searches the public language using the six displayed examples. It enumerates one-rule programs first, then searches the two-rule space specified in Appendix I; the first visible-consistent solution is committed before hidden-target evaluation. Construction and search share a ten-CPU-second limit per task. Appendix I specifies the candidate order, anytime results, and input isolation.

The baseline solves all 120 controls with one-rule programs and passes neither common-nine nor H_2 on any of them. Corollary 1 predicts these audit outcomes from the selected program class. On separation, it solves 95/120, passes common-nine on 86/120, and passes H_2 on 87/120 (Table 3). Thus the available evidence admits solutions that a simple symbolic search can find much more often than the evaluated language-model protocols.

Condition	Visible six	Common nine	Unshown H_2	Timeouts
Control	120/120	0/120	0/120	0
Separation	95/120	86/120	87/120	25

Table 3: **First-consistent bounded symbolic search.** One run per displayed specification, with a ten-CPU-second cap; all results are reported out of 120, including timeouts. The solver receives no reference, certificate witness, hidden example, or language-model output.

The comparison separates solution availability, inductive preference, and final delivery. The symbolic solver tests candidates on visible examples, while the language models generate a trajectory. Appendix I provides the search protocol and its timing sensitivity.

5.4 Token counts, readout, and budget sensitivity

Completion cost persists at matched token counts. For Qwen and Olmo, the separation-minus-control prompt-token difference has mean -0.075, median zero, and range $[-3, 2]$. Exactly 46/120 families have equal token counts. Within that subset, Qwen changes from 24 to three visible successes (-45.65 points; marginal interval $[-60.87, -30.43]$), and Olmo from 19 to seven (-26.09; $[-43.48, -8.70]$). The completion decrease persists among families with identical prompt-token counts (Appendix G).

Final selection and trajectory development. Two target-blind alternatives—first complete eligible fence/list, and last complete eligible object by endpoint—select the same programs as the primary rule across all 960 Qwen/Olmo comparisons. Final-answer choice is stable across these two rules under the shared grammar. The 2,880 nested prefix records instead show substantial temporal change: at 2,048 tokens none has a canonical visible success; at 4,096, Qwen has 31/2 control/separation successes, increasing to 67/12 at 8,192. Corresponding Olmo counts increase from 6/1 to 52/15. These offline measurements describe the development of fixed trajectories (Appendix J).

Delivery and final-answer availability. Qwen's native and canonical totals coincide at 79/240. Olmo has 4/240 native versus 67/240 canonical successes under the prospectively fixed notation mapping: unchanged text, different delivery contracts. Cap counts rise from 43 to 101 for Qwen and from 63 to 99 for Olmo under separation. Phi has 7/240 visible successes, 21 parseable finals, and 197 capped outputs. These counts quantify the available final-answer support at this budget. Stopping and readout counts are reported in Appendix C.

5.5 Transfer across input distributions

Transfer changes with the input distribution. An additional reference-only audit generates 32 uniform and 32 rule-active inputs per family, excluding the original nine. All 7,680 inputs are scored using the unchanged selected programs, with each family as the statistical unit. Uniform inputs are predominantly identities under the reference (3,325/3,840, or 86.59%). On this layer, Qwen and Olmo have negative unconditional accuracy changes of -36.12 and -21.72 points. On inputs that activate both reference rules, their changes are $+2.16$ and $+1.59$ points, with marginal intervals crossing zero; separation accuracy is 2.76% and 2.01%, respectively. These tests are post hoc and distribution-specific (Appendix H).

Transfer is audit-dependent. Separating witnesses exclude short solutions; uniform inputs emphasize identity behavior and program availability; active inputs test another distribution. Together, these readouts distinguish witness transfer, identity behavior, and accuracy on rule-active inputs.

6 Discussion and Limitations

Certifying the evaluated construct. Complete exclusion supplies a quantifier over every legal short solution. This makes a required-length difference independently verifiable before models are evaluated. Shortest-first symbolic selection then links the certificate to a concrete behavioral prediction, while the neural results measure completion, audit agreement, and delivery separately.

Interpretation of the intervention. The sixth-example exchange changes identifying information together with required length; it measures their combined effect rather than a pure length-mediated effect. The full solution sets need not be nested. The certificate calibrates required program length; behavioral difficulty additionally depends on identifying information, search, and output delivery.

Solver comparison. The neural–symbolic comparison concerns complete evaluated systems: CPU time and generated GPU tokens are different resources. Bounded two-rule search cannot certify nonexistence, and the observed system gap does not localize a neural search mechanism.

The study covers a selected feeding-only reference population in one replacement language. The eligibility criteria select 120 of 15,004 proposals; 119 bases have one nonidentity mapping and one has two. These frequencies describe selection into the eligible generator-conditioned population. A separate constructive search finds nonfeeding minimum-length alternatives for 152/240 separation specifications; 88 remain unresolved. Minimum length and feeding dependence are therefore distinct properties. The certified result concerns this replacement language and length bound; richer languages require corresponding certificates.

Sampling and transfer. Each model–condition cell has one sampled output. Family bootstrap intervals characterize the selected reference population and are approximate; they do not include additional decoding-seed variation. Syntactic identity exclusions do not establish global semantic disjointness. Token-count matching leaves token identity, segmentation, and identifying information variable. The post hoc supplements reuse observed families; the unshown witness pair, solution-count groups, and prefix analyses retain that status. Degenerate empirical intervals do not establish a population null. Finite audits measure agreement on their inputs rather than global functional equivalence, and fixed-trajectory prefixes do not identify a forced-stopping policy. The positive witness-transfer result and the broader-input results consequently describe different evaluation targets.

7 Conclusion

Complete one-rule exclusion and a two-rule witness make a necessary-length change verifiable before model evaluation. Matched new-family comparisons reveal large displayed-task completion costs, alongside gains on a common finite audit and a supplemental wholly unshown witness pair. A bounded symbolic baseline clarifies the role of a short-program preference; token, readout, prefix, and broader-input analyses explain complementary aspects of the measured response. Certificate-grounded calibration connects task construction to an explicit requirement, while keeping solver behavior, delivery, and generalization targets distinct.

AI Use Statement

Generative AI tools assisted hypothesis refinement, experimental design, implementation, proof checking, analysis, literature search, and manuscript preparation. Executable generators produced the synthetic tasks and audit inputs. Numerical claims were checked against stored programs and family-level statistics. The authors are responsible for the claims, references, and code. Figure 1 uses an AI-generated visual reference followed by an editable vector reconstruction with checked semantics.

Reproducibility statement

The paper provides proofs, complete task construction, and inference and analysis protocols. The supplement supplies selected programs, targets, readouts, family indices, numerical reconstructions, and independent CPU checks, including the solution-count groups. The package supports numerical replay of saved outputs and supplies the verbatim synthesis instruction, native templates, pinned revisions, and complete worked-example inputs for model-level replication. The released commands reconstruct program-level and readout-level statistics; the separate research archive retains complete token-level and runtime histories.

Ethics Statement

The study uses generated string-transformation tasks and released models, with no human participants or personal records. The certificates concern the stated program language; behavioral results are interpreted within the specified task and audit distributions.

References

Rajeev Alur, Rastislav Bodík, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design*, pp. 1–17, 2013. doi: 10.1109/FMCAD.2013.6679385.

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987. doi: 10.1016/0890-5401(87)90052-6.

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. DeepCoder: Learning to write programs. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Serafim Batzoglou. ReplaySCM: A benchmark for executable causal mechanism induction from interventions. arXiv:2605.08197v1, 2026. URL https://arxiv.org/abs/2605.08197v1.

Samuel R. Bowman and George Dahl. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4843–4855, 2021. doi: 10.18653/v1/2021.naacl-main.385.

Simone Caldarella, Davide Talon, Rahaf Aljundi, Elisa Ricci, and Massimiliano Mancini. Thinking past the answer: Evaluating harmful overthinking in large reasoning models. arXiv:2606.02835v1, 2026. URL https://arxiv.org/abs/2606.02835v1.

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel rahman Mohamed, and Pushmeet Kohli. RobustFill: Neural program learning under noisy I/O. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 990–998, 2017. URL https://proceedings.mlr.press/v70/devlin17a.html.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1383–1392, 2018. doi: 10.18653/v1/P18-1128.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323, 2020. doi: 10.18653/v1/2020.findings-emnlp.117.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 317–330, 2011. doi: 10.1145/1926385.1926423.

Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Foundations and Trends in Programming Languages*, 4(1–2):1–119, 2017. doi: 10.1561/2500000010.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of NAACL-HLT, Volume 2 (Short Papers)*, pp. 107–112, 2018. doi: 10.18653/v1/N18-2017.

Joey Hong, David Dohan, Rishabh Singh, Charles Sutton, and Manzil Zaheer. Latent programmer: Discrete latent codes for program synthesis. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4308–4318. PMLR, 2021. URL https://proceedings.mlr.press/v139/hong21a.html.

Abigail Z. Jacobs and Hanna Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 375–385, 2021. doi: 10.1145/3442188.3445901.

Jinwoo Kim, Qinheping Hu, Loris D'Antoni, and Thomas Reps. Semantics-guided synthesis. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–32, 2021. doi: 10.1145/3434311.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3428–3448, 2019. doi: 10.18653/v1/P19-1334.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Atharva Naik, Darsh Agrawal, Prakam, Yash Mathur, Manav Kapadnis, Yuwei An, Clayton Marr, Carolyn Rose, and David Mortensen. PBEBench: A multi-step programming by examples reasoning benchmark inspired by historical linguistics. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8877–8918, 2026a. doi: 10.18653/v1/2026.findings-acl.432.

Atharva Naik, Yash Mathur, Prakam, Carolyn Rose, and David Mortensen. ReaComp: Compiling LLM reasoning into symbolic solvers for efficient program synthesis. arXiv:2605.05485v1, 2026b. URL https://arxiv.org/abs/2605.05485v1.

Luan Ozelim and Hector Zenil. Measuring in-context algorithmic reasoning in language models against an exact Bayes-optimal reference. arXiv:2608.01575v2, 2026. URL https://arxiv.org/html/2608.01575v2.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, 2020. doi: 10.18653/v1/2020.acl-main.442.

Kensen Shi, Joey Hong, Yinlin Deng, Pengcheng Yin, Manzil Zaheer, and Charles Sutton. Exedec: Execution decomposition for compositional generalization in neural program synthesis. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13883v2.

Armando Solar-Lezama. *Program Synthesis by Sketching*. PhD thesis, University of California, Berkeley, 2008. URL https://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-176.html.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 404–415, 2006. URL https://people.eecs.berkeley.edu/~sseshia/pubs/b2hd-solar-asplos06.html.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=uyTL5Bvosj.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, et al. Olmo 3. arXiv:2512.13961, 2025. URL https://arxiv.org/abs/2512.13961.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, Shuohang Wang, Weijian Xu, Jianfeng Gao, and Weizhu Chen. Phi-4-Mini-Reasoning: Exploring the limits of small reasoning language models in math. arXiv:2504.21233, 2025. URL https://arxiv.org/abs/2504.21233.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

A CONSTRUCTION, CERTIFICATES, AND SAMPLING

A.1 REPRODUCIBLE ENUMERATION

For n examples, let $u = |A(E)|$ and $v = |B(E)|$. The one-rule solver performs at most uvn replacement checks. Lemma 1 establishes completeness of the example-derived candidate strings. Empty replacements are included; empty find strings are prohibited. Repeated input or output substrings are deduplicated. Each certificate specifies the complete retained one-rule solution set, the two-rule reference, and the example specification.

We generate candidate witness strings from length-two/three strings over the alphabet, base/rule anchors, and deterministic repeated or sampled contexts around those anchors, retaining lengths two through six. Their outputs are obtained only by running the fixed reference. For each five-feature signature and condition, up to four representatives are retained by a fixed hash ordering. A deterministic selection chooses two disjoint control/separation pairs with four distinct witness inputs. This bounded construction defines the eligible sampling population.

Filtering the complete base solution set determines witness eligibility. Independent re-enumeration verifies all 480 six-example certificates and 240 strict matches.

A.2 COMPLETE EXAMPLE

The reference is the ordered program

$$\text{replace}(\texttt{ex}, \texttt{azw}),$$
$$\text{replace}(\texttt{wa}, \texttt{y}).$$

The following nine mappings are the five base examples and both witness pairs:

Role	Input	Output
Base	`yf`	`yf`
Base	`gud`	`gud`
Base	`czua`	`czua`
Base	`dzexa`	`dzazy`
Base	`bwcied`	`bwcied`
Control 1	`iiiexa`	`iiiazy`
Separation 1	`wbawex`	`wbayzw`
Control 2	`hhexa`	`hhazy`
Separation 2	`azwex`	`azyzw`

The base's only one-rule solution is `exa`→`azy`. It remains valid in both control specifications and fails in both separation specifications. Complete re-enumeration finds no other one-rule solution there. The two signatures are $(6, 6, 3, 3, 0)$ and $(5, 5, 3, 2, 0)$. This family was selected for exposition from the design record, without using model correctness.

A.3 ELIGIBILITY AND SAMPLING POPULATION

The validation generator records 15,004 proposals: 11,970 outside the desired reference-relation class; 1,404 without a one-rule base solution; 1,169 excluded by prior identity; 338 without two disjoint strictly matched pairs; three identity cases; and 120 accepted families. The categories sum to the complete proposal count. Among accepted bases, 119 have one nonidentity mapping and one has two. Complete one-rule enumeration gives $|\mathcal{S}_1(B_f)| = 1, 2, 3$ in 91, 13, and 16 families, respectively (minimum/median/maximum $1/1/3$). These counts refer to the one-rule subclass of the full one-to-five-rule language.

The reference key normalizes the program representation, including its recorded symbol mapping. Exclusion checks cover the archived earlier references, proposals, parsed model programs, and constructive alternatives. The exclusion operates on normalized program syntax. The reported family bootstrap treats the accepted families as the independent sampling units in this generator-conditioned design.

The nonfeeding-alternative search is separate from the complete one-rule solver. It finds a constructive minimum-length alternative for 152 separation specifications; the remaining 88 cases are unresolved. Found alternatives establish dispensability of feeding in those cases; the remaining cases retain unresolved status.

B Discovery Evidence and Readout Sensitivity

Discovery 1 used 60 families, two models, two specification conditions, and two instruction versions, yielding 480 outputs. Its primary Qwen completion contrast was -32.50 points. Of these, 22 families met all five strict surface matches. The Olmo candidate analysis exposed a readout confound: the strict notation detector counted 11/120 correct-candidate trajectories under the original instruction and 35/120 under the explicit instruction. Canonicalizing the predefined rule notations changed those counts to 30/120 and 35/120, reducing the apparent instruction effect from 20.00 to 4.17 points. This post hoc analysis measures the sensitivity to the notation readout.

Discovery 2 used 60 new, strictly matched families, two witness pairs, and the fixed explicit instruction. Its 480 outputs gave canonical visible-six effects of -26.67 points for Qwen and -15.83 for Olmo. Both witness-specific point estimates were negative, with heterogeneity intervals that permit unequal witness effects. The common-nine analysis was introduced after observing this study and gave gains of $+21.67$ and $+15.00$ points. Discovery and validation are reported separately.

The validation fixed the common target, canonical parser, witness assignment, and six primary contrasts before new outputs. The validation therefore evaluates the fixed specification contrast. All original native scores and all discovery records remain accessible in the evidence archive.

C Inference Configuration and Final-Answer Rules

C.1 Pinned inference configuration

The exact model repositories and revisions are:

```
Qwen/Qwen3-8B
b968826d9c46dd6066d109eabc6255188de91218
allenai/Olmo-3-7B-Think
d97e442d7cc678210054dbcc9b440894d62c89a4
microsoft/Phi-4-mini-reasoning
0e3b1e2d02ee478a3743abe3f629e9c0cb722e0a
```

The environment uses PyTorch 2.8.0+cu128 and Transformers 4.57.1. Generation uses BF16 weights, SDPA, sampling temperature 0.6, top-p 0.95, one return sequence, and repetition penalty 1.0. Top-k is 20 for Qwen and zero (disabled) for Olmo and Phi. Batch sizes are eight, four, and four. Model-native EOS tokens are respected. The generation budget is 8,192 new tokens for every model.

All model evaluations use an RTX 5090.

Maximum rendered prompt lengths are 272, 339, and 281 tokens for Qwen, Olmo, and Phi. Model revisions, token counts, and random seeds identify the recorded evaluations. The supplement supplies task specifications, selected programs, the verbatim synthesis instruction, complete native input examples, and the three archived chat templates at these revisions. Surface-feature matching and model-native token counts are evaluated separately. The bootstrap seed is `12645822550545217882`; reconstruction uses the preserved $50{,}000 \times 120$ index array to retain the original resampling distribution.

C.2 VERBATIM SYNTHESIS INSTRUCTION AND NATIVE INPUTS

Every validation input uses the following instruction, followed by six input–output pairs. The notation example specifies serialization only.

````
Infer a cascade of string-replacement rules that maps every input to
its paired output.
Apply each rule to every string in order, using Python str.replace
semantics.
Use only characters from abcdefghijkuvwxyz.
The first argument must contain 1 to 3 characters; the second may
contain 0 to 3 characters.
Use at most 5 rules. Any functionally correct cascade is acceptable.
You may reason before answering. Put your final answer in one fenced
block labelled python.
Inside that block write a Python list whose elements are strings. Each
string must be a rule written as replace('a','b'), with two quoted
arguments satisfying the limits above.
For example, the serialization of one rule is:
```python
["replace('a','b')"]
```
This example shows notation only, not a proposed solution. Keep the
order of your rules. Arrow notation, bare character pairs, and a list
of output strings are not the requested rule-list format.
Input-output pairs:
````

For the family in Appendix A, the recorded display order is `gud`, `czua`, `bwcied`, `dzexa`, `yf`, followed by `iiiexa`→`iiiazy` in control or `wbawex`→`wbayzw` in separation. The two conditions contain 265 tokens for Qwen, 332 for Olmo, and 272 for Phi. These counts include native framing. The other witness pair is reserved for evaluation.

The native templates add model-specific framing to a single user message: Qwen uses its user/assistant delimiters with thinking enabled; Olmo inserts its default system message and opening reasoning tag; Phi inserts its default system message. The supplement's `prompt_archive/` provides all six complete rendered strings and token arrays, the three exact templates, and pinned model/tokenizer metadata. Qwen and Phi templates are exact exports of the original tokenizer-configuration fields; Olmo's template is the archived standalone file.

C.3 SELECTION PRECEDES EXECUTION

The canonical selector is deterministic and target-independent. For Qwen and Olmo, it requires the final segment after a closed thinking delimiter. It selects the last complete Python fence, falling back to the last complete list only if no complete Python fence exists. A selected object outside the grammar receives zero final success, including trajectories with an earlier correct candidate. Rule notation mappings are finite, explicit, and frozen; they preserve find/replacement strings and rule order.

Phi's frozen adapter accepts either the same closed-thinking route or an eligible terminal block/list at actual EOS when no explicit unfinished thinking segment is present. A fixed terminal-token set defines eligible suffixes; an unfinished reasoning segment remains ineligible. Evaluation protocol validation includes 19 parsing cases for Qwen and Olmo and 33 model-boundary cases.

Native and canonical scoring use different endpoint eligibility rules. The re-evaluated Phi cohort accordingly has nine native successes and seven canonical successes. Candidate availability performs a distinct target-aware scan of admissible programs already expressed in the trajectory and is reported alongside final success.

For each output, the archive stores the selected span, legal rule sequence, eight executable target scores, all three readout layers, generation length, and EOS/CAP status. Prefix observations at 2,048, 4,096, and 6,144 generated tokens are nested offline measurements of the same trajectory. The principal estimates always retain all 120 outputs per condition.

Model	Condition	Native	Final	Candidate	Parsed	CAP	EOS
Qwen	Control	67	67	67	75	43	77
	Separation	12	12	13	19	101	19
Olmo	Control	3	52	57	57	63	57
	Separation	1	15	17	21	99	21
Phi†	Control	6	5	7	13	97	23
	Separation	3	2	3	8	100	20

Table 4: Stopping and readout counts in the validation cohort, with 120 outputs per row. Final is canonical correctness; candidate means a correct program appears somewhere in the text; parsed does not require correctness. Native and canonical endpoint eligibility differ and need not be nested. †Corrected Phi.

D Attention validation and uniform re-evaluation

D.1 Cohort definition

All reported Phi results use one corrected causal-attention implementation. Correction of the causal-attention position and mask specification was followed by uniform re-evaluation of the entire 240-output cohort. Re-evaluation retained the original tasks, prompts, token IDs, weights, sampling parameters, seeds, batch size, token budget, and readout; every result uses this common implementation. The 480 Qwen/Olmo outputs are unchanged. The cohort retains the original task sample.

D.2 Numerical checks

The corrected implementation was fixed before the full Phi cohort was re-evaluated. Independent causal-plus-padding masks and FP32 attention computations provide the numerical references; the tested full-prefix/reset logits and cache entries match the independent-mask reference bitwise. Full-versus-incremental BF16 comparisons give maximum absolute logit difference 0.3125, maximum mean logit difference 0.05849, and maximum absolute cache difference 0.25, with identical tested argmax predictions. The supplement's implementation-detail document records the context lengths, case counts, and comparison scope.

D.3 Stopping and readout outcomes

The corrected cohort contains five/two visible successes, zero/one common-audit successes, and 13/eight parseable finals under control/separation. EOS occurs on 23/20 trajectories and the token cap on 97/100. Among all 240 outputs, 195 reach the cap without a final boundary, one has a capped unparseable final, one a capped incorrect final, 23 end at EOS without a parseable final, 13 end with an incorrect final, and seven provide a successful final. These categories retain every evaluation and separate stopping from answer delivery.

E Prospective Interpretation and Statistical Supplements

E.1 Effect magnitude and statistical interpretation

Write $\widehat{\Delta} = \widehat{\theta}_{\mathrm{vis}}$ and $\widehat{\Gamma} = \widehat{\theta}_{\mathrm{com}}$. Before inference, the analysis plan set a ten-percentage-point magnitude benchmark. It combined $\widehat{\Delta} \leq -.10$ with a six-comparison upper interval endpoint below zero for completion cost, and $\widehat{\Gamma} \geq +.10$ with a lower endpoint above zero for common-target benefit. A joint magnitude interpretation required both conditions in at least two models. Models with fewer than 12 visible successes over 240 outputs were designated support-limited, with all contrasts retained.

Qwen and Olmo show common-nine changes of 7.50 and 9.17 percentage points, with adjusted intervals $[1.67, 14.17]$ and $[3.33, 16.67]$. Both intervals exclude zero and contain effects on either side of the predeclared ten-point benchmark. The intervals support positive local transfer, while both

point estimates remain below the predeclared magnitude criterion. The distinct H_2 target is evaluated separately.

Marginal 95% intervals for visible/common differences are $[-55.00, -37.50]$/$[3.33, 12.50]$ for Qwen, $[-40.83, -20.83]$/$[4.17, 15.00]$ for Olmo, and $[-6.67, 1.67]$/$[0, 2.50]$ for corrected Phi. The main primary table retains the original six-comparison intervals.

E.2 Exact paired-binary sensitivity

We performed a supplemental exact conditional check after observing the validation outcomes. Let g and ℓ count separation-only and control-only successes. Under independent paired sampling and paired marginal symmetry, $g \mid (g + \ell)$ is $\mathrm{Binomial}(g + \ell, 1/2)$. We use a two-sided exact tail test and $\min(1, 6p)$ for all six original contrasts (McNemar, 1947; Dror et al., 2018). The exact check complements the planned percentile bootstrap.

Model	Target	Gain	Loss	$p_{\mathrm{Bonf},6}$
Qwen	Visible six	0	55	3.33e-16
Qwen	Common nine	9	0	0.02344
Olmo	Visible six	5	42	1.47e-07
Olmo	Common nine	11	0	0.00586
Phi†	Visible six	2	5	1.00000
Phi†	Common nine	1	0	1.00000

Table 5: Post hoc exact paired-binary sensitivity on the original six contrasts. Gain/loss count separation-only/control-only successes. No target is omitted.

For common-nine, Qwen has nine gains and no losses, giving $p = 2/2^9 = .00390625$ and corrected $p = .0234375$. Olmo has eleven gains and no losses, giving $p = .0009765625$ and corrected $p = .005859375$. Phi's one gain leaves its directional contrast unresolved. This test addresses paired success on the specified targets.

E.3 Supplemental uncertainty and chronology

All supplemental analyses are post hoc relative to the neural outputs. The analysis definitions, reference-only new-input generation, solver order, and alternate selectors were frozen before their supplemental evaluation, after the original study. H_2 uses the archived 50,000 family resamples with quantiles $1/120, 119/120$ for three approximate simultaneous intervals. Its marginal intervals are $[4.17, 15.00]$, $[6.67, 17.50]$, and $[-2.50, 2.50]$ points for Qwen, Olmo, and Phi. The supplemental three-comparison family and the primary six retain their separate adjustments.

The remaining supplemental tables and curves are descriptive or use labeled marginal intervals. Equal-token and fixed token-difference strata use recorded subset family IDs and saved 50,000-resample arrays. Small strata ($n < 12$) are descriptive diagnostics with limited support. All supplemental analyses use the original generated outputs.

F Unshown Witnesses, Outcome Localization, and Program Identity

The fixed nine-role order is $(B_1, \ldots, B_5, c_1, s_1, c_2, s_2)$. We execute the same selected final program on all nine mappings and retain every bit, assigning all zeros to a missing program. For displayed index j, the supplemental target is exactly (c_{3-j}, s_{3-j}). Both conditions use these identical two examples; neither prompt contains either example. We also record base-five, both-control, and both-separation success.

Both H_2 examples were withheld from both prompts; their isolated conjunction was selected as a reported outcome after the original model evaluation.

For Qwen, 60 of 67 visible-correct controls have bit vector $(1, 1, 1, 1, 1, 1, 0, 1, 0)$: the only failures are both separation witnesses. Six have $(1, 1, 1, 1, 1, 1, 0, 0, 0)$ and one has $(1, 1, 1, 1, 1, 0, 0, 1, 0)$.

Olmo's corresponding counts are 49, two, and one out of 52. All five visible-correct Phi controls fail only the two separation witnesses. These conditional counts locate the errors within the full 120-family evaluation.

An H_2 success is weaker than common-nine success. Qwen has two additional H_2-successful separation outputs beyond its nine common successes, and both fail the visible specification. Olmo has three additional H_2 successes: two fail visible six and one passes visible six but not all nine. Phi's one control H_2 success fails visible six. The pair and the complete specification therefore test distinct success predicates.

Model	Audit	Condition	Success	Reference	Other 2	Other length
Qwen	Nine	Control	0	0	0	0
Qwen	Nine	Separation	9	0	9	0
Qwen	H_2	Control	0	0	0	0
Qwen	H_2	Separation	11	0	9	2
Olmo	Nine	Control	0	0	0	0
Olmo	Nine	Separation	11	0	10	1
Olmo	H_2	Control	0	0	0	0
Olmo	H_2	Separation	14	0	10	4
Phi†	Nine	Control	0	0	0	0
Phi†	Nine	Separation	1	0	1	0
Phi†	H_2	Control	1	0	1	0
Phi†	H_2	Separation	1	0	1	0

Table 6: Ordered-rule identity among successful audit outputs. "Reference" is exact equality after legal syntax normalization; "Other 2" is a different two-rule sequence, not proof of semantic inequivalence. Counts include all valid models and both conditions.

Rule-sequence equality compares normalized ordered (a, b) pairs. Every common-nine-successful output uses a sequence different from the reference; finite audit agreement measures consistency on the evaluated inputs.

Corollary 1 follows directly from the construction. Every one-rule control solution is in $\mathcal{S}_1(B_f)$; each s_i excludes all such programs. We independently re-enumerated all 480 six-example specifications and checked this implication in all 120 families. The same references pass every witness, demonstrating that the prediction depends on the returned program class.

F.1 Behavior by base solution count

We describe the saved outputs in two groups determined by the base certificates: $|\mathcal{S}_1(B_f)| = 1$ (91 families) and $|\mathcal{S}_1(B_f)| > 1$ (29). Table 7 reports visible-six, common-nine, and H_2 outcomes for every model. Both Qwen and Olmo have a negative visible-six difference and a positive common-nine difference in each group. The completion decrease and positive common-audit change are larger in magnitude in the multiple-solution group. The certificate-defined groups support a descriptive association between multiplicity and outcomes.

The analysis reuses the selected programs without changing the selector or scoring targets. A group bootstrap resamples families while keeping their two conditions paired. All measurements use the original generated trajectories. H_2 retains its supplemental status. In the multiple-solution Phi group, every common-nine and H_2 paired difference is zero, yielding degenerate empirical intervals $[0, 0]$; these are the intervals of the observed bootstrap distribution. All full-sample and group statistics, family memberships, and resampling indices are provided.

Model	$\lvert\mathcal{S}_1(B_f)\rvert$	Target	n	C	S	S − C (pp)	Marginal 95% interval
Qwen	1	Visible six	91	43	5	-41.76	[-51.65, -31.87]
Qwen	1	Common nine	91	0	5	+5.49	[+1.10, +10.99]
Qwen	1	H_2	91	0	7	+7.69	[+3.30, +13.19]
Qwen	> 1	Visible six	29	24	7	-58.62	[-75.86, -41.38]
Qwen	> 1	Common nine	29	0	4	+13.79	[+3.45, +27.59]
Qwen	> 1	H_2	29	0	4	+13.79	[+3.45, +27.59]
Olmo	1	Visible six	91	31	6	-27.47	[-37.36, -17.58]
Olmo	1	Common nine	91	0	5	+5.49	[+1.10, +10.99]
Olmo	1	H_2	91	0	7	+7.69	[+2.20, +13.19]
Olmo	> 1	Visible six	29	21	9	-41.38	[-65.52, -17.24]
Olmo	> 1	Common nine	29	0	6	+20.69	[+6.90, +37.93]
Olmo	> 1	H_2	29	0	7	+24.14	[+10.34, +41.38]
Phi†	1	Visible six	91	2	2	+0.00	[-4.40, +4.40]
Phi†	1	Common nine	91	0	1	+1.10	[+0.00, +3.30]
Phi†	1	H_2	91	1	1	+0.00	[-3.30, +3.30]
Phi†	> 1	Visible six	29	3	0	-10.34	[-24.14, +0.00]
Phi†	> 1	Common nine	29	0	0	+0.00	[+0.00, +0.00]
Phi†	> 1	H_2	29	0	0	+0.00	[+0.00, +0.00]

Table 7: **Paired outcomes in two certificate-defined groups.** C and S are success counts among n families. Intervals use 50,000 paired-family resamples within each group and are marginal, descriptive 95% intervals. This post hoc grouping leaves the original full-sample contrasts and their adjustments unchanged. Phi uses the common validation implementation.

G Prompt Tokenization Diagnostics

Counts use the archived model-native prompt token IDs, including template overhead. A paired delta is separation length minus control length. Summary statistics appear in Table 8; Figures 2–3 show each model separately.

Model	Mean	Median	Range	Zero difference
Qwen	-0.0750	0.0	[-3, 2]	46/120
Olmo	-0.0750	0.0	[-3, 2]	46/120
Phi†	-0.0083	0.0	[-3, 3]	55/120

Table 8: Actual prompt-token differences on 120 paired families per model. Qwen and Olmo each have 46 equal-count families (38.33%); Phi has 55 (45.83%).

The zero-delta subset retains large negative visible effects. Its common-nine changes are $+6.52$ points ($[0, 15.22]$ marginal) for Qwen and $+13.04$ ($[4.35, 23.91]$) for Olmo; its H_2 changes are $+10.87$ ($[2.17, 19.57]$) and $+17.39$ ($[6.52, 28.26]$). Phi has one visible success in each zero-delta condition. These fixed-subset sensitivities retain the main analysis and quantify the contrast at equal prompt length.

Model	Token difference bin	n	Visible $S - C$ (pp)	Marginal interval
Qwen	`delta<=-2`	5	-80.00	[-100.00, -40.00]
Qwen	`delta=-1`	37	-35.14	[-51.35, -21.62]
Qwen	`delta=0`	46	-45.65	[-60.87, -30.43]
Qwen	`delta=1`	24	-54.17	[-75.00, -33.33]
Qwen	`delta>=2`	8	-50.00	[-87.50, -12.50]
Olmo	`delta<=-2`	5	-40.00	[-80.00, +0.00]
Olmo	`delta=-1`	37	-29.73	[-45.95, -13.51]
Olmo	`delta=0`	46	-26.09	[-43.48, -8.70]
Olmo	`delta=1`	24	-29.17	[-50.00, -8.33]
Olmo	`delta>=2`	8	-62.50	[-87.50, -25.00]
Phi†	`delta<=-2`	6	+0.00	[+0.00, +0.00]
Phi†	`delta=-1`	28	-10.71	[-25.00, +0.00]
Phi†	`delta=0`	55	+0.00	[-5.45, +5.45]
Phi†	`delta=1`	23	+0.00	[-13.04, +13.04]
Phi†	`delta>=2`	8	+0.00	[+0.00, +0.00]

Table 9: Visible-six paired effects in all five predefined token-difference strata. Intervals are marginal, post hoc, and unadjusted across strata. Strata smaller than 12 families are descriptive low-support results; degenerate intervals do not prove a population zero.

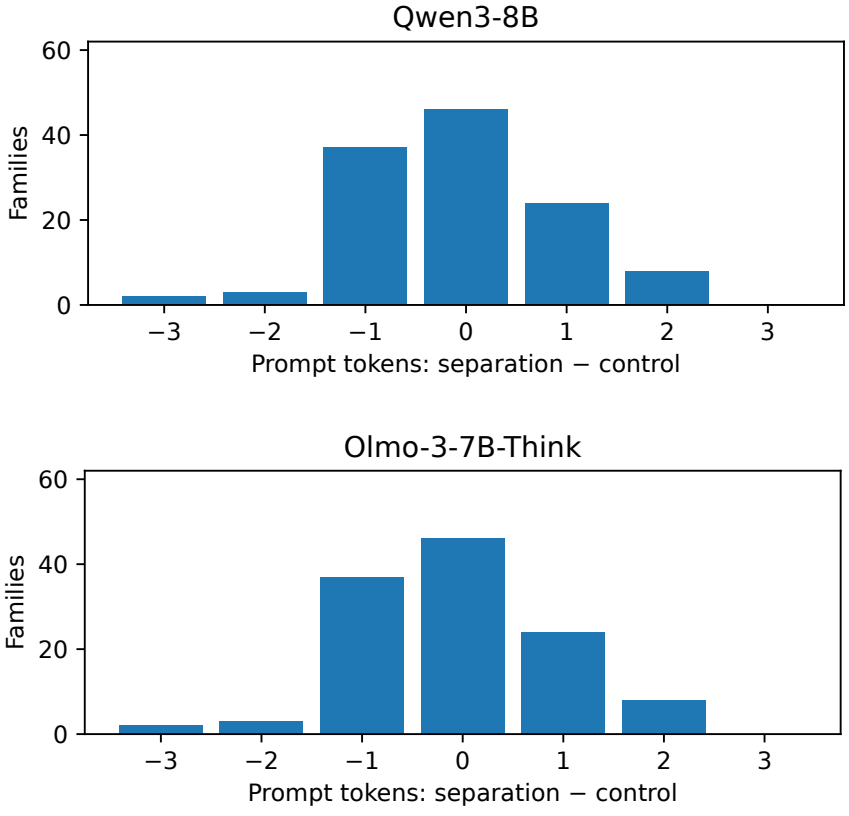


Figure 2: Token-difference distributions for Qwen and Olmo. Histograms show separation-minus-control prompt-token counts.

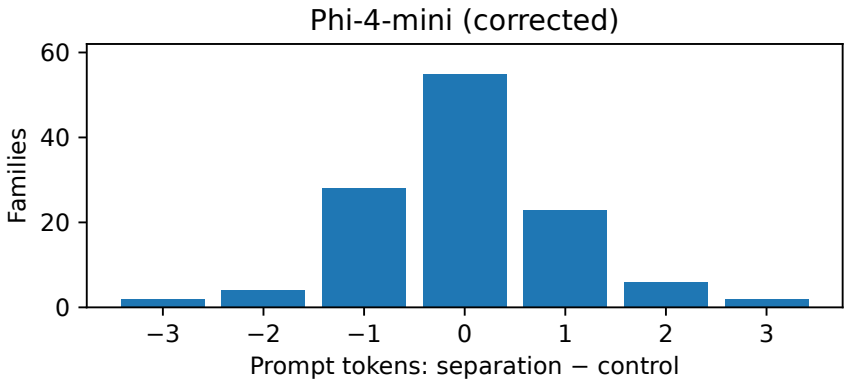


Figure 3: Token-difference distribution for Phi prompts.

H Additional Reference-Only Input Audits

H.1 Generation and metrics

For each family, we generate 32 uniform and 32 active inputs, excluding all original nine inputs and duplicates within the two layers. Uniform inputs use the same 17-character alphabet with lengths two through twelve. Active proposals combine fixed reference-rule anchors with sampled contexts and are accepted only when both reference replacements change their respective intermediate strings. Reference execution alone defines the activation criterion, independently of model programs and their correctness. Seeds, proposal counts, rejections, and the accepted inputs were fixed before supplemental scoring. Each layer has a 10,000-proposal ceiling per family; all requested inputs were obtained.

For program $\hat{p}_{fc}$ and input stratum A_f of size 32, we report

$$Z_{fc} = \frac{1}{32} \sum_{x \in A_f} \mathbf{1}[\hat{p}_{fc}(x) = p^*_f(x)],$$

We set $Z_{fc} = 0$ when no final program is selected. The main quantity is the mean of $Z_{fS} - Z_{fC}$ over 120 families; all-pass rates are retained in the machine-readable supplement. The independent sample consists of 120 families; the 3,840 inputs per layer and 7,680 overall are repeated tests of the selected programs. Both conditions use the same input sets. Input construction depends on the reference program alone.

Model	Stratum	C (%)	S (%)	Difference (pp)	Marginal interval
Qwen	Uniform	48.10	11.98	-36.12	[-43.93, -28.36]
Qwen	Active	0.60	2.76	+2.16	[-0.29, +5.34]
Olmo	Uniform	34.30	12.58	-21.72	[-29.53, -13.70]
Olmo	Active	0.42	2.01	+1.59	[-0.31, +4.22]
Phi†	Uniform	6.46	3.23	-3.23	[-7.84, +1.22]
Phi†	Active	0.60	0.00	-0.60	[-1.80, +0.00]

Table 10: Additional-input mean accuracy, including all failures to deliver a program. All six model–stratum combinations are reported. Intervals are marginal family bootstrap intervals, not pooled across strata.

H.2 What the broader audits support

Of the 3,840 uniform inputs, 3,325 (86.59%) are unchanged by the reference. None of the active inputs is a reference identity. Uniform accuracy is therefore substantially influenced by identity

behavior as well as program availability. The positive Qwen/Olmo active-layer point estimates have intervals spanning zero. Identity behavior, answer availability, and rule-active agreement contribute distinct information to the audit.

For context, executing the frozen symbolic outputs gives uniform mean accuracies of 87.19%/70.26% and active means of 0.42%/40.42% under control/separation. These are descriptive evaluations of the same selected outputs, with timeouts scored zero, under the original search budget. Together with Table 10, they show why performance depends on which inputs the audit emphasizes. The witnesses and broader input distributions evaluate different aspects of transfer.

I Independent Bounded Symbolic Search

I.1 Solver inputs and search space

Each solver task contains an opaque ID, the legal language specification, and the ordered six visible pairs. A separate evaluator holds the reference, certificates, unshown witnesses, audit inputs, and model outputs. The solver receives only these task inputs. Selected programs are committed before the external evaluator tests hidden targets. The public string-replacement semantics define the search language.

Candidate rules are ordered by $(|a| + |b|, |a|, a, b)$ after deterministic deduplication. One-rule enumeration is complete as in Lemma 1. For the first rule of a two-rule program, find strings come from visible input substrings of length at most three; replacements come from visible input/output substrings, public alphabet singletons, and the empty string. Each candidate produces an intermediate input vector. An unchanged vector is skipped. The second rule is enumerated completely from that intermediate vector to the visible outputs, using the one-rule lemma. The first consistent program is returned and never replaced using audit performance.

The first-rule replacement pool omits possible latent intermediate strings. Two-rule search is therefore incomplete even though the second-rule subproblem is complete for each chosen first rule. An exhausted pool or time limit leaves the unrestricted two-rule existence question unresolved.

Algorithm 1 First-consistent bounded solving of a displayed specification

Require: Visible specification E; public language; CPU limit τ
Start timer before candidate construction
for one-rule candidate r in the complete, fixed order **do**
    Record elapsed-time/candidate checkpoints; stop if limit reached
    if $r \models E$ **then**
        return $[r]$
    end if
end for
for first rule r_1 in the fixed restricted pool **do**
    Compute intermediate inputs $z_i = r_1(x_i)$; skip an unchanged vector
    for second rule r_2 in the complete one-rule search from z_i to y_i **do**
        Record checkpoints; stop if limit reached
        if $[r_1, r_2] \models E$ **then**
            return $[r_1, r_2]$
        end if
    end for
end for
return no selected program (not an unrealizability certificate)

I.2 Budgets and recorded execution

Each of the 240 tasks runs once with a ten-process-CPU-second cap and a 60-second wall-clock safeguard. The same search records checkpoints at 0.1, one, and ten CPU seconds and at 1,000, 10,000, and 100,000 full candidate-program checks. A selected program is frozen after discovery, and is retained at later checkpoints. Candidate construction and search are both included in CPU time;

process startup is separately reported. The numerical record distinguishes candidate construction, search, and process startup.

CPU limit (s)	Condition	Visible six	Common nine	Unshown H_2
0.1	Control	120/120	0/120	0/120
0.1	Separation	1/120	1/120	1/120
1	Control	120/120	0/120	0/120
1	Separation	32/120	28/120	29/120
10	Control	120/120	0/120	0/120
10	Separation	95/120	86/120	87/120

Table 11: Same-run solver checkpoints by CPU budget. All counts use the full 120 tasks per condition. The ten-second row is the primary committed run.

Candidate limit	Condition	Visible six	Common nine	Unshown H_2
1,000	Control	40/120	0/120	0/120
1,000	Separation	0/120	0/120	0/120
10,000	Control	120/120	0/120	0/120
10,000	Separation	0/120	0/120	0/120
100,000	Control	120/120	0/120	0/120
100,000	Separation	10/120	10/120	10/120

Table 12: Same-run candidate-check checkpoints. One candidate check tests a full program against the visible specification, with early rejection allowed; per-example execution counts are separately recorded. Candidate counts and CPU seconds are distinct budget views.

The primary run solves 215 tasks, with 25 reaching the CPU-time limit. It uses 10.47 CPU-minutes in total and approximately 1.924×10^8 candidate checks, with peak working-set memory of 23.7 MiB. All 120 control selections contain one rule; all 95 separation selections contain two.

A repeated execution agrees on 238/240 final selections. The two differences occur at the time limit; the 215 tasks solved in both executions have identical programs and candidate counts. Tables 11–12 use the original run throughout. Fixed candidate-order checkpoints complement the machine-dependent time-budget measurements.

J Offline Prefixes and Alternative Final Selection

J.1 Nested trajectory measurements

The 720 valid trajectories provide four nested prefixes at 2,048, 4,096, 6,144, and 8,192 generated tokens, totaling 2,880 records. A trajectory ending before a cut retains its actual EOS; otherwise the prefix is marked capped. Text is decoded with the archived tokenizer and original boundary contract. We retain parseability, visible/common scores, and correct-candidate availability at every cut. The fixed full-trajectory denominator is retained, including outputs without an eligible final segment.

At 2,048 tokens, Qwen already expresses ten control and one separation correct candidates, while Olmo expresses two control candidates, but neither has a successful canonical final. This separates observable candidate availability from final delivery. The analysis retains the original 8,192-token generation cap and measures candidate expression and delivery within it.

J.2 Two fixed alternative selectors

For Qwen and Olmo only, alternative A selects the first complete Python fence in the eligible final segment, or the first complete list if no fence exists. Alternative B selects the last complete object by character endpoint across fences and standalone lists, deduplicating lists contained within a fence. Both retain the frozen final-segment eligibility and legal-rule parser. Object selection precedes parsing and execution: a chosen object outside the grammar receives zero final success.

Model	Tokens	C parsed	C final	C cand.	S parsed	S final	S cand.
Qwen	2,048	0	0	10	0	0	1
Qwen	4,096	34	31	43	3	2	7
Qwen	6,144	61	53	61	12	10	10
Qwen	8,192	75	67	67	19	12	13
Olmo	2,048	0	0	2	0	0	0
Olmo	4,096	9	6	24	1	1	4
Olmo	6,144	30	27	48	5	2	11
Olmo	8,192	57	52	57	21	15	17
Phi†	2,048	0	0	0	0	0	0
Phi†	4,096	2	1	2	0	0	0
Phi†	6,144	7	4	6	2	1	2
Phi†	8,192	13	5	7	8	2	3

Table 13: Prefix development of parseable finals, displayed-task successes, and correct-candidate availability. Each count uses all 120 trajectories per condition. Prefixes are nested observations of the same runs.

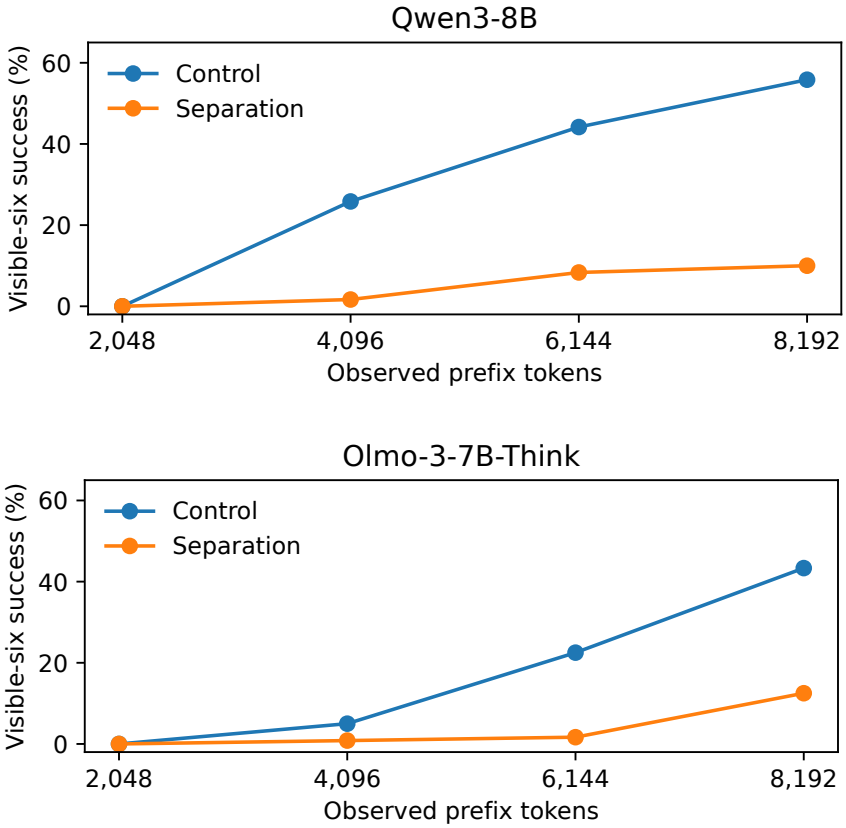


Figure 4: Observed final-success development along the existing Qwen and Olmo trajectories. Each curve follows nested prefixes of the same 8,192-token trajectory.

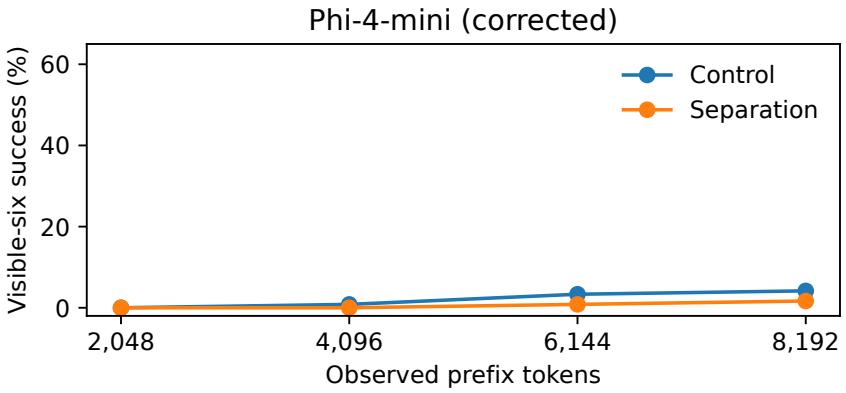


Figure 5: Corresponding Phi validation-cohort prefix diagnostic. The same 240 outputs are used at every cut.

Across 480 outputs times two alternatives, all 960 selected rule sequences and their visible-six, common-nine, and H_2 scores agree with the primary selection. Phi's endpoint rule is unchanged. Agreement holds for the two specified alternatives under the common grammar.

K Data and Code Availability

All reported model analyses use the same 720 valid-runtime outputs: 480 unchanged Qwen/Olmo trajectories and 240 corrected Phi trajectories. The earlier Phi implementation and both discovery studies remain separate from the reported validation cohort. Supplemental audit inputs are generated by seeded procedures; symbolic-search outcomes follow the documented time budget. These analyses reuse fixed model trajectories while introducing explicitly specified supplemental evaluation designs.

The supplemental reproducibility archive contains selected programs, targets, model-native token counts, prefix readouts, alternative-selector outputs, and symbolic-search records, with a file manifest. Independent CPU checks cover three objects: complete short-program certificates, saved model programs and their audit outcomes, and the bounded symbolic solver. The supplement documents the full reconstruction inventory and provides the corresponding executable checks. These checks validate analysis of the saved observations; model generation and timed symbolic search require their respective execution environments.

The accompanying code includes reconstruction data, the isolated solver implementation, and scripts for rebuilding the tables and figures. Numerical tables are reconstructed directly from the recorded data. The accompanying provenance records distinguish original observations, supplemental analyses, and runtime validation. Full raw-token and implementation histories are retained in a separate reproducibility archive.