FREEZE, REFIT, COMPARE: A PROCEDURE-LEVEL CONTAMINATION AUDIT FOR SYMBOLIC ARC SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic solvers for the Abstraction and Reasoning Corpus (ARC) build their answers from libraries of hand-written operations, so every solution can be read. Readable code can still hide evaluation contamination: iterating on the public evaluation tasks can leave traces of them in operation bodies, validators and special cases, invisible to a check of method names for evaluation identifiers. We propose an audit of the whole executable procedure: freeze the solver, refit its trainable part on a development split of the training set, and compare success on reserved training tasks with success on the public evaluation set. Because the evaluation set is deliberately harder, an evaluation excess is a warning sign. On our own solver, the frozen pipeline's evaluation rate was 3.3 to 6.3 times its reserved-training rate on each of four splits, and commit history points to operations written during evaluation iteration as the likely source. A recount of our own provenance check found two counting errors, including evaluation tasks counted as their own held-out support because they reappear in ARC-AGI-1. With support recounted, manifests selected without evaluation outcomes and nested controls reverse the gap, as does an independently written solver; under our rule these outcomes are inconclusive, not clean passes, and for the frozen manifests the reversal is consistent with the measured difficulty shift. Under an independent-task model at reserved-training rates of 5 to 9%, the audit flags an evaluation excess with 80% probability only when the excess reaches about 9 to 11 percentage points. Inspectable is not the same as clean: contamination can hide in code that nobody calls trained, and an audit costing one training split can expose it.

1 INTRODUCTION

Motivation. The Abstraction and Reasoning Corpus (ARC) (Chollet, 2019; Chollet et al., 2025) scores a system by whether it produces the exact output grid for a test input after seeing a few demonstration pairs. Symbolic ARC solvers answer with an executable rule assembled from a library of operations, which makes each solve inspectable. Inspectability does not settle provenance. When a team iterates on the public evaluation tasks, information about those tasks can enter persistent executable code (an operation body, a validator, a special case) that sits outside the components conventionally described as "trained". The final artifact can look generic, pass a check for evaluation identifiers in its method names, and still owe part of its public-evaluation score to that contact. The same blindness affects provenance checks themselves: in our own case, several "held-out" support tasks turned out to be evaluation tasks that also appear in the earlier benchmark version.

Significance. A public-evaluation score can lose its out-of-sample interpretation even when the final artifact appears generic and no identifier check fails. The adaptive-data-analysis literature formalizes this mechanism, score inflation from repeated adaptive contact with a fixed evaluation set (Dwork et al., 2015; Blum & Hardt, 2015), and offers prospective protections. Going forward, the remedy is clean validation discipline and a private-set result. That remedy does not answer the retrospective question faced by a reader of an existing public-evaluation result, or by a team that, like ours, developed parts of its solver against public-evaluation scores: given an artifact whose development history is already on record, what does its behaviour say about the information it absorbed?

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.

The question may matter more when solver code is written with coding assistants that iterate against benchmark feedback: such loops can produce persistent code quickly, and each iteration on the public evaluation set is a chance for its tasks to enter that code.

Contributions. (1) A *procedure-level audit* whose object is the entire frozen executable procedure π, not the surface form of its method names: the trainable component is refit on an 800-task development partition of the ARC-AGI-2 training set and π is run, unchanged, on the remaining 200 training tasks and on the 120 public evaluation tasks, with the comparison interpreted by a three-way rule whose power at these pool sizes we state (Section 4, Figure 1). (2) A forensic case study on our own solver, LGE: the audit flags a large evaluation excess on every proposer-training split (Section 5.1); a recount of LGE's own support-based provenance check by task identity and content exposes two counting errors that had inflated our earlier diagnostics, one of them the counting of evaluation tasks that reappear in the ARC-AGI-1 evaluation set as their own held-out support (Section 5.2); manifests selected without evaluation outcomes reverse the direction of the gap (Section 5.3), as do nested-selection controls and an independently authored solver, ARGA (Xu et al., 2023) (Sections 5.4–5.5), and the reversed gap is consistent with a measured difficulty shift (Section 5.6). Figure 2 places the main bands reported below. (3) An operational checklist (Appendix C), instantiations of the audit for other solver classes (Table 2), and a harness with one external adapter.

What this paper is not. The audit is an external diagnostic layer applied to a frozen target procedure. It does not improve the solver, it does not replace validation-set discipline or private evaluation, and a freeze after historical evaluation contact cannot certify that earlier code was uncontaminated. It does not target test-time synthesis: adapting to the current task's demonstrations within a fixed budget, by program synthesis or otherwise, is what ARC is designed to reward. Its target is persistent code or weights, carried across tasks and runs, that encode information from earlier contact with the evaluation tasks. A flagged outcome is a pattern consistent with contamination, subject to power and distributional assumptions; it is not proof of historical contact, and an outcome that is not flagged is not a certificate. LGE is not a state-of-the-art ARC system: under the corrected protocol it solves no public evaluation task, and frontier language models solve most of them (Appendix H). We have not run LGE on the private or semi-private evaluation sets.

2 Background and audit object

ARC. A task consists of two to seven demonstration pairs (x_i, y_i) of coloured grids and one or more test inputs $x^\star$; success requires the exact output grid for every test input (*task-exact*, TE). For example, the demonstrations may show every enclosed region of the input being filled with the colour of its border; the solver must apply the same transformation to a test grid with different regions and colours. ARC-AGI-2 has 1,000 public training tasks and 120 public evaluation tasks; the public evaluation set is deliberately harder than training (Chollet et al., 2025), and a private evaluation set is held by the organizers. A rule that reproduces every demonstration pair is *demo-fit* (DF); demo-fit rules need not be test-exact. Table 1 collects the notation and the bands used below.

Symbolic solvers. ARGA (Xu et al., 2023) abstracts a grid into a graph of per-colour connected components, acquires constraints from the demonstrations, and searches a vocabulary of filters and transformations, its domain-specific language (DSL), for a program that reproduces every pair; GPAR (Lei et al., 2024) casts the same problem as generalized planning. LGE follows the same pattern with a larger, learned library: an encoder maps a grid to objects and regions; a Naive-Bayes proposer trained on traces from the training tasks ranks operation families; candidate executors (generic `_try_*` functions, 387 in the audited inventory) are instantiated against the demonstrations; survivors of an exact-demo-fit gate and a label-free shape gate are reranked by empirical-Bayes reliability priors (Appendix I, Figure 3).

The audited object. We call the entire executable induction pipeline a *procedure* π: the executor inventory, the proposer and its training code, the scoring rule, and the seed. A *manifest* is a fixed list of executors. A procedure is *frozen* at a commit hash. The contamination channel this paper audits is the authoring of executor bodies during development iteration on public evaluation tasks; of the 54 executors selected in LGE's family-supported evaluation, 44 were committed in April 2026 during ARC-AGI-2 iteration (10 have no recoverable date). The configuration behind 54/120 was

Table 1: Notation and the bands that appear in this paper.

D, T_{held}, E	800-task development partition; 200-task reserved training subset (seeded; the manifest audits and ARGA use a separate draw, Table 5); 120 public evaluation tasks
π	a frozen executable procedure (inventory, proposer training, scoring, seed)
p_H, p_E, Δ	task-exact rate on T_{held} and on E; $\Delta = p_E - p_H$ (positive = evaluation excess)
legacy validators	task-specific validators from earlier development; 67–68/120 with them enabled, reported only as a diagnostic
family-supported	selected by LGE's original protocol: exact demo fit, label-free shape check, no evaluation identifier in the executor name (54/120)
held-out-supported	the selected executor exact-fits ≥ 1 distinct support-pool task (8/120; 4/120 after identity and content exclusion)
substrate-general	the selected executor exact-fits ≥ 2 distinct support-pool tasks (1/120)
oracle	any demo-fit rule in the manifest is test-exact (an upper bound over the enumerated candidates only)
consistency picker	the top-ranked demo-fit rule under a label-free consistency score (deployable)

also chosen on evaluation outcomes: on 2026-04-24, successive runs scored on the evaluation set raised the proposer's top-k from 20 to 30 and tried several reranking variants, taking selected solves from 35/120 to 54/120, the number of evaluation tasks with a test-exact rule among the retained candidates.

3 Related work

ARC methods. Approaches include direct language-model transduction and test-time training (Akyürek et al., 2025; Li et al., 2025), language-model program generation with verification (Butt et al., 2024; Pourcel et al., 2025), products of experts over augmented views (Franzen et al., 2025; 2024), per-task compression (Liao & Gu, 2025), object-centric representations (Xu et al., 2024), symbolic graph and constraint search (Xu et al., 2023; Lei et al., 2024; Rocha et al., 2025), and library learning (Ellis et al., 2021; Grand et al., 2024; Bowers et al., 2023). Our per-task prototypes (Appendix F) reimplement ARGA's ingredients at small scale: two connectivity abstractions, four filter families, up to 24 transformation searches and a constraint-acquisition loop. The external replication instead runs ARGA's published code with five compatibility patches (Appendix D).

Evaluation audits and contamination. Three lines are relevant. Corpus-overlap and probabilistic-model audits address trained models: Carlini et al. (2019) plant canaries to measure unintended memorization; Min-K% Prob (Shi et al., 2024) detects pretraining data from token probabilities; Oren et al. (2024) give a likelihood-based black-box test that exploits canonical-order preference under exchangeability. Fresh-test and leaderboard studies examine the consequences of repeated evaluation: Recht et al. (2019) construct fresh CIFAR-10 and ImageNet replicas and find that accuracy drops are better explained by difficulty than by adaptivity, and Roelofs et al. (2019) find little evidence of substantial overfitting across Kaggle public-versus-private splits. The adaptive-data-analysis line (Dwork et al., 2015; Blum & Hardt, 2015) formalizes score inflation from repeated adaptive contact with a fixed evaluation set and offers prospective protections such as the reusable holdout and the Ladder. Kapoor & Narayanan (2023) provide a taxonomy of eight types of leakage, a survey of 294 papers across 17 fields, and model info sheets. Our difference is scoped: the audited object is a persistent authored procedure rather than a trained model, and the instrument is a retrospective, executable, outcome-based comparison rather than corpus membership, likelihoods, fresh task collection, prospective holdout mechanisms or self-reporting.

4 The audit

Protocol. Fix a seed; shuffle the 1,000 training tasks; take T_{held} as the first 200 and D as the remaining 800. Fit every statistically fitted component of π on D only; hand-written code stays as authored. Freeze π at a commit. Run the identical procedure with the identical per-task time budget on T_{held} and on E, recording a per-task trace. Report p_H, p_E, and Δ with a Newcombe hybrid-score 95% confidence interval (CI; Newcombe, 1998). The preregistered test is one-sided Fisher

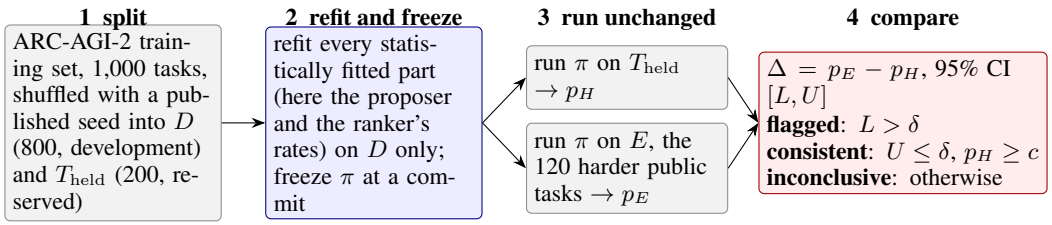

Figure 1: The audit. Every statistically fitted component is refit on D alone, while hand-written code, possibly written with the whole training set in view, stays as authored; the procedure is then frozen and run unchanged, with the same per-task budget, on the reserved training tasks and on the public evaluation tasks. Without evaluation contact the harder evaluation set should score lower, so an excess whose interval clears the tolerance δ is flagged; an excess that survives the refit points to the frozen part of π.

exact for the flagged direction, $p_E > p_H$; when the observed difference runs the other way we report the reverse-direction exact p descriptively. Allowed information for the frozen procedure: the demonstration pairs and the test input grid of the current task, and whatever was fit on D. Forbidden: test outputs, evaluation outcomes of any kind, and task identifiers used for dispatch.

Decision rule. Let δ be a tolerance for evaluation excess stated before the audit and c a capacity floor for p_H. The outcome is *flagged* if the lower confidence bound on Δ exceeds δ; *consistent* if the upper bound on Δ is at most δ and $p_H \geq c$; and *inconclusive* otherwise. A nonsignificant test is inconclusive, not a pass. We use $\delta = 0$ throughout, because the public evaluation set is harder than training, so any positive excess is unexpected for a procedure without evaluation contact; a difficulty-adjusted δ is discussed in Section 5.6. We did not preregister a capacity floor c for the retrospective analyses below, so no manifest in this paper is scored as consistent. We have not calibrated δ or c on an independent population of contaminated and uncontaminated systems; the rule is an interpretation convention, not a validated detector.

Power. At these pool sizes the rule is coarse. Under a model of independent tasks in which both pools share a success rate between 5% and 9% (the range of the procedure audits, the corrected manifests and ARGA; the nested cells run at 3.0–4.5%), the rule flags with probability about 0.026. Its flag probability reaches 0.5 only at evaluation excesses of 5.9–7.4 points and 0.8 only at 9.1–11.1 points (exact enumeration; Appendix B). An evaluation rate less than about 6–7 points above the reserved-subset rate, including one lifted by contamination just enough to cancel the difficulty shift, is more likely to be scored inconclusive than flagged, which is a further reason why an inconclusive outcome is not a pass.

Freeze chronology. The original seed-42 procedure audit ran on 2026-04-26; the 59-method manifest (v4) was frozen on 2026-04-27 and the 105-method manifest (v5) on 2026-04-29. Each manifest was frozen before its own oracle evaluation run, after earlier public-evaluation contact by the same development effort; no manifest predates all evaluation contact.

Other solver classes. The protocol needs only a freezable procedure; any component fit on training data must be declared and refit on D, and what the audit can localize depends on which parts that refit covers (Table 2). An excess that survives the refit points to the frozen part. Wherever π starts from a pretrained model, the refit cannot remove pretraining exposure: an excess could not be separated from other frozen sources, and exposure covering both training and evaluation tasks would raise p_H as well as p_E and need not produce an excess. We have applied the audit to two symbolic solvers only; the other rows are instantiations, not results.

Table 2: Instantiating the audit. Only the first row is evaluated in this paper.

solver class	refit on D	frozen part an excess would point to
symbolic DSL solver	any learned proposer or ranker (LGE: both; ARGA: none)	executor and DSL code, validators, scoring
program search with a persistent library	library growth, prompt exemplars, any fine-tuning	base model, prompts, hand-written tools
test-time-training transducer	offline fine-tuning from a fixed base checkpoint, including on synthetic tasks	base checkpoint, synthetic-task generators, test-time-training recipe, augmentations, voting
zero-shot language model	nothing	everything; the audit reduces to a frozen comparison

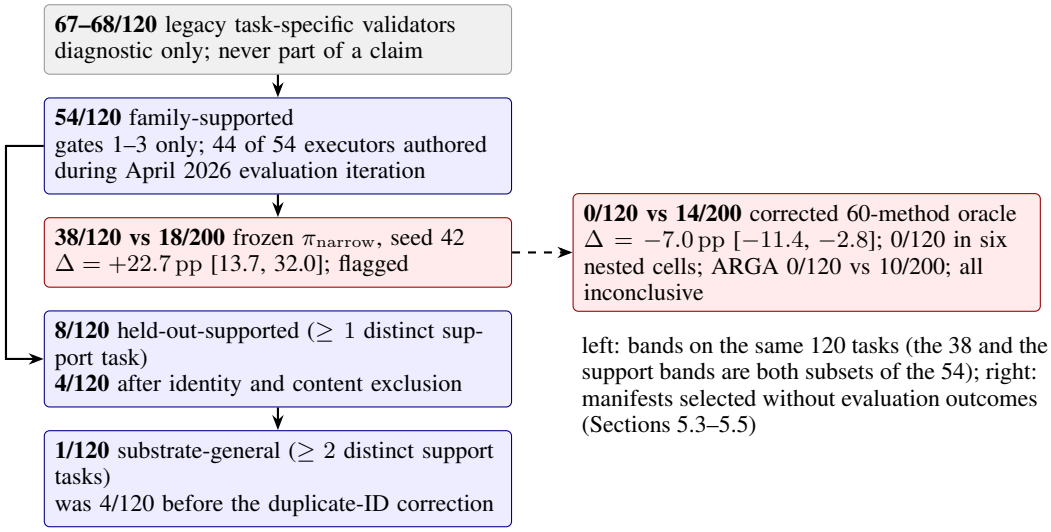

Figure 2: The result ladder. The frozen-procedure numerator (38) and the support bands (8, 4, 1) are each subsets of the 54 family-supported solves; the support bands are nested in one another but two of the eight held-out-supported tasks lie outside the 38. The support bands are diagnostics computed after the family-supported run, not gates of the pipeline that produced 54/120. The right-hand reserved counts come from a different 200-task draw than the left (Table 5).

5 Results

5.1 Procedure audit of the existing pipeline

π_{narrow} is LGE's proposer-and-gate pipeline (every operation family with a record in D, 19–21 per split, since top-$k = 30$ exceeds their number; exact-demo-fit gate, label-free shape gate, no-evaluation-identifier name filter) with the Naive-Bayes proposer refit on D. Candidates are ranked by a score that combines the proposer's family probability with per-family and per-method rates recomputed from D's trace records; because those records contain only exact solves, every rate is 1 for a family or executor with a record in D and a fixed 0.05 otherwise, so the score mainly favours executors observed in D (Appendix I). The empirical-Bayes reranker behind the 54/120 run is not part of π_{narrow}, but π_{narrow}'s top-k and ranking formula were fixed during the same development. Table 3 reports the original split and three further splits with the same procedure, plus one extended procedure. Every split is flagged: the evaluation rate exceeds the reserved-subset rate by 18.7–26.7 percentage points, with the lower confidence bound above zero in every row. The evaluation count varies with the refit proposer (38/120 in four rows, 32/120 in one), so the evaluation rate is not a constant of the inventory. A reliability filter (dropping families with a recomputed rate below 0.05) removed no family, since every rate is 1 or 0.05, and left every outcome unchanged; two batches of object-relational synthesizers (2 and 10 templates) changed the reserved-subset count by at most one

Table 3: Procedure-level audit of π_{narrow} across proposer-training splits. $\Delta = p_E - p_H$ in percentage points with a Newcombe hybrid-score 95% CI; p is one-sided Fisher exact for $p_E > p_H$. The seed-7 row uses the extended procedure (ten added synthesizers) and is reported separately.

split	procedure	T_{held} (200)	E (120)	Δ [95% CI]	p
seed 42	π_{narrow}	18 (9.0%)	38 (31.7%)	+22.7 [13.7, 32.0]	3.8×10^{-7}
seed 1	π_{narrow}	10 (5.0%)	38 (31.7%)	+26.7 [18.1, 35.7]	2.0×10^{-10}
seed 2	π_{narrow}	16 (8.0%)	32 (26.7%)	+18.7 [10.2, 27.7]	8.6×10^{-6}
seed 3	π_{narrow}	14 (7.0%)	38 (31.7%)	+24.7 [15.8, 33.9]	1.3×10^{-8}
seed 7	$\pi_{\text{extended v2}}$	11 (5.5%)	38 (31.7%)	+26.2 [17.5, 35.3]	6.2×10^{-10}

task at seed 42. These tested expansions did not explain the gap, which is not the same as showing that no capacity explanation exists.

5.2 Auditing the support audit

LGE's original provenance check asked, for each executor selected on an evaluation task, whether the same executor exact-fits other tasks in three support pools meant to exclude evaluation tasks (1,000 ARC-AGI-2 training, 400 ARC-AGI-1 training, 400 ARC-AGI-1 evaluation). An executor with support on at least two such tasks was called substrate-general and one with support on at least one was called held-out-supported; an earlier version of this support check reported 4/120 and 8/120 on that basis. Two errors invalidated those counts.

Duplicate identifiers. The three pools contain 1,800 records but only 1,033 distinct task identifiers (391 identifiers shared between ARC-AGI-2 training and ARC-AGI-1 training, 376 between ARC-AGI-2 training and ARC-AGI-1 evaluation). A success on one task could satisfy the two-record threshold by appearing in two pools. Under the corrected criterion of at least two *distinct* successful support tasks, 60 of the 105 methods in the v5 manifest qualify and the substrate-general band falls from 4/120 to 1/120 (sole survivor `_try_diff_shape_recolor` on task `1818057f`, supported by two distinct training tasks; its docstring mentions a task identifier, so the survivor is not an identifier-free trace-derived executor). The held-out-supported band stays at 8/120 under this correction.

Evaluation identities in the support pools. Six ARC-AGI-2 public-evaluation task identifiers (`0934a4d8`, `136b0064`, `16b78196`, `981571dc`, `aa4ec2a5`, `da515329`) also occur in the ARC-AGI-1 evaluation pool used as support, with the same test pair and the same demonstrations up to order. For three of the eight held-out-supported executors the only distinct support task is the evaluation task the executor itself solved, present under the same identifier in the ARC-AGI-1 evaluation pool: `_try_8block_extract` (`0934a4d8`), `_try_substitution_pattern` (`136b0064`) and `_try_encapsulate_shapes` (`aa4ec2a5`). Excluding evaluation identities leaves 5/120 held-out-supported. A pair-level content check finds one more case: evaluation task `db695cfb` shares all five demonstration pairs with ARC-AGI-1 evaluation task `55783887`, which has another identifier and another test pair. `55783887` is the sole support of `_try_diagonal_cross`, the executor selected for `db695cfb`, whose docstring names `db695cfb`; excluding evaluation tasks by identity and content leaves 4/120 held-out-supported. None of the six identifiers supports any of the corrected 60 methods of Section 5.3, but `55783887` is one of the two supports of `_try_diagonal_ray_intersection`, the one method that manifest adds to v4. Without it the manifest is v4's 59 methods (Table 5): one training-pool success (`1f876c06`) and one evaluation demo fit (`db695cfb`; 29 become 28) are lost, and no test-exact count on E or on the reserved subset of Table 5 changes. No count in Table 6 changes (two of its manifests lose this method), but the corrected-60 step of the nested ladder falls from 35 to 34, since `1f876c06` lies in the seed-103 reserved subset. The lesson is that label independence must be enforced on task identity and content, not on dataset-directory names.

Table 4 gives the corrected cross-tabulation of authorship class and support for the 54 selected executors. The direction that survives is one trace-derived executor with two distinct supports against no post-hoc executor with more than one; excluding evaluation tasks by identity and content moves all four post-hoc borderline executors to singleton as well, leaving no post-hoc executor with any held-out support. Across all 387 executors the distinct-identifier semantics move 62 executors from

Table 4: Provenance audit of the 54 family-supported evaluation solves under distinct-identifier semantics (general ≥ 2 distinct supports; borderline $= 1$; singleton $= 0$). Excluding evaluation tasks by identity moves three post-hoc borderline executors to singleton (0/1/49), and content exclusion one more (0/0/50).

authorship class	general	borderline	singleton	total
trace-derived (method-strict)	1	3	0	4
post-hoc (family-supported)	0	4	46	50
total	1	7	46	54

Table 5: Frozen-manifest restrictions. Counts are recorded successes under the implemented bounded search (30–60 s per task). The 200-task subset participated in full-pool manifest selection and is not a held-out selection test (see Section 5.4). p: reverse-direction one-sided Fisher exact for $p_H > p_E$, reported descriptively. Superseded 105-method rows are kept for the record: both of their evaluation successes came from single-support executors authored in April 2026. Reserved subset: a separate 200-task draw (seed 20260426) that shares 39 tasks with the seed-42 subset of Table 3; ARGA uses the same draw.

manifest	selection	E (120)	T_{held} (200)	training (1,000)	p
corrected 60 methods	oracle	0 (0.0%)	14 (7.0%)	69 (6.9%)	1.2×10^{-3}
corrected 60 methods	consistency top-1	0 (0.0%)	11 (5.5%)	53 (5.3%)	5.1×10^{-3}
v4, 59 methods (2026-04-27)	oracle	0	14	68	—
v5, 105 methods (superseded)	oracle	2	19	87	—
v5, 105 methods (superseded)	consistency top-1	1	15	66	—
ARGA, patched, 60 s search	published search	0 (0.0%)	10 (5.0%)	—	8.3×10^{-3}

"general" to "borderline" (70 general, 66 borderline, 251 singleton). These are descriptive support diagnostics, not estimates of success on an independent held-out distribution.

What the bound does and does not say. A Hoeffding-plus-union-bound argument (Appendix A) gives, for 387 executors, $N = 1{,}033$ distinct support identifiers and $\delta = 0.05$, a slack of 0.068; the two-distinct-task threshold corresponds to $2/N = 0.0019$, and a threshold above the slack would require at least 71 successes. Neither a two-success nor a ten-success admission threshold is therefore certified in the probably-approximately-correct (PAC) sense. The bound also presumes a fixed executor class and an i.i.d. audit sample, neither of which holds for the historical, adaptively inspected support pools.

5.3 Manifests selected without evaluation outcomes

The opposite-direction control restricts π to a manifest of executors chosen by a rule that uses no evaluation outcome: generic name, no task identifier in code or docstring, and distinct-task support in the support pools. The corrected 60-method manifest (the 59 methods of the April v4 manifest plus one) is a retrospective sensitivity analysis, not a preregistered manifest. For each task we run every primitive and every (geometric operation, primitive) two-step chain and record whether any demo-fit rule is also test-exact (the *oracle*, an upper bound over the enumerated candidates only) and whether a deployable label-free consistency picker (Appendix E) selects a test-exact rule (Table 5).

The evaluation outcome is exhaustively certified: a full rerun of all 120 evaluation tasks under the 60-method manifest records zero test-exact candidates, 29 demo-fit tasks, zero timeouts and at most 6.04 s per task. Reserved-subset and training negatives were not exhaustively rerun after the manifest correction; any success missed there would only strengthen the $p_H > p_E$ direction. Under the decision rule the corrected manifests are not flagged ($\Delta = -7.0\,\text{pp}$, 95% CI $[-11.4, -2.8]$ for the oracle) but they are *inconclusive*, not consistent: no capacity floor was preregistered and the reserved-subset rate is only 5.5–7%; demo-fit candidates exist for 29 evaluation tasks without any being test-exact; and under the 105-method manifest (hence a fortiori under its 60-method subset)

Table 6: Genuinely nested selection (support counted only outside T_{held}). Tasks whose first test-exact primitive dropped out were re-run under the nested manifest at 8 s per (method, task) call; other outcomes follow from the subset property. p: reverse-direction one-sided Fisher exact for $p_H > p_E$.

threshold	seed	manifest size	T_{held} (200)	E (120)	p
$\geq$ 2 distinct	101 / 102 / 103	49 / 52 / 47	7 / 9 / 9	0 / 0 / 0	0.036 / 0.014 / 0.014
$\geq$ 3 distinct	101 / 102 / 103	31 / 30 / 17	6 / 9 / 7	0 / 0 / 0	0.058 / 0.014 / 0.036

three-step composition adds no test-exact rule on any pool. An oracle over enumerated candidates is not the maximum over every program expressible in the DSL.

5.4 Nested selection controls

The original manifests were selected with support counted over all 1,000 training tasks, including the 200 later scored as T_{held}. For three fresh splits (seeds 101–103) we recomputed selection using distinct-identifier support outside T_{held} only, at thresholds of two and three distinct supports (Table 6). Manifests shrink to 17–52 methods, the reserved-subset count falls to 6–9/200, and the evaluation count is 0/120 in every cell. Reverse-direction exact p-values range from 0.014 to 0.058; all six cells are inconclusive under the rule. The six cells share the evaluation tasks and overlap in training tasks, so they are related sensitivity analyses, not six independent replications. Nesting covers selection only: the executor bodies were authored with visibility of the full training pool, so per-task leakage through executor code is not removed by this control. Within the same reserved subset the ladder 105 methods $\rightarrow$ corrected 60 $\rightarrow$ nested selection gives 44 $\rightarrow$ 35 $\rightarrow$ 25 successes over the three seeds combined; the first drop is the semantics correction and the second is the pure nesting effect.

5.5 External case study: ARGA

ARGA (Xu et al., 2023) is an independently authored symbolic solver that predates ARC-AGI-2. Its public code assumes equal input and output shapes, which is often false on ARC-AGI-2; we applied five patches that remove shape-mismatch crashes and hash large token strings (Appendix D). With a 60 s search limit, patched ARGA records 10/200 on T_{held} and 0/120 on E (reverse-direction exact $p = 0.0083$; inconclusive under the rule). ARGA uses a different DSL and budget from LGE, so it is an external case study in the same audit framework rather than a matched causal control, and its low rate limits the power of the comparison. Because the patches change which candidates are reached within the time limit, we do not claim that they preserve every solution the unpatched code would have found, and an adapter defect can only lower ARGA's counts (Appendix D).

5.6 How much harder is the evaluation set?

Label-free structural proxies differ sharply between the 1,000 training and 120 evaluation tasks: mean input area 182 versus 373 cells (a factor of 2.05; Mann–Whitney $p = 3 \times 10^{-24}$), mean output area 1.98 times larger, 5.8 versus 7.2 distinct colours, 1.08 versus 1.39 test inputs per task, and 3.2 versus 3.0 demonstration pairs; the shape-change frequency (32.0% versus 32.5%) and the output/input size ratio are matched. Within the training pool these proxies predict solvability: a ridge-logistic model fit on the v4 manifest's per-task oracle outcomes is calibrated on the reserved subset (predicted 14.5 versus observed 14 successes) and predicts an evaluation rate of 1.1% (1.3 expected successes) from the measured shift alone; the observed 0/120 has a Poisson-binomial tail probability of 0.26 under this model versus 2×10^{-4} under a no-shift null, and 0.045 under a conservative tercile-binned variant. The proxies are structural (size, colours, counts), not semantic, so this analysis bounds only the measurable part of the shift. Its use is twofold. For the frozen manifests it shows that the reversed gap is statistically consistent with the measured difficulty shift alone, which is one reason their outcome is inconclusive rather than a pass. For π_{narrow} it shows how far the flagged excess lies from any difficulty-based expectation: a procedure without evaluation contact and with the training-pool difficulty response would be expected to lose most of its rate on evaluation, whereas π_{narrow} triples it.

Split sensitivity. Because the frozen-manifest audits refit nothing per split, each task's outcome is a fixed property and the reserved-subset count over any 200-of-1,000 split is exactly hypergeometric (v4: mean 13.6, 95% interval 8–20). The $p_H > p_E$ direction holds in more than 99.999% of splits and reverse-direction Fisher significance at 0.05 in 99.1%; the split actually used sits at the 62nd percentile. This analysis is descriptive and conditional on the fixed outcome vector; selection dependence is addressed by the nested controls, and π_{narrow}, which refits per split, by the splits of Table 3.

5.7 Interpretation

In the rule's own vocabulary the tally is: flagged, π_{narrow} on all five rows of Table 3; inconclusive, the two corrected-manifest rows, the six nested cells, and ARGA; consistent, none (Figure 2). The pattern is consistent with evaluation-contact inflation localized in the executor-body authoring step and outside the proposer, and it is corroborated by provenance: 44 of the 54 selected executor bodies were committed during April 2026 evaluation iteration. The pattern does not, by itself, identify the mechanism. Distribution shift, selection dependence between the reserved subset and the manifests (removed only for selection by the nested controls), and differing solver capacities remain live explanations for parts of each gap, and a nonsignificant reverse direction can coexist with contamination when a procedure has very low power. Eight engines authored independently or restricted to manifests selected without evaluation outcomes never score higher on evaluation than on the reserved subset, while the eight contaminated configurations (the rows of Table 3 and three further seed-42 variants) exceed it by 18.7–26.7 points; the engines are correlated and labelled by the history the rule is meant to detect, so this separation shows exploratory utility, not a calibrated criterion.

6 Limitations

The audit applies to any freezable procedure; localization is strongest for modular symbolic systems, and wherever π starts from a pretrained model the refit cannot remove pretraining exposure (Table 2). At ARC-AGI-2's pool sizes it flags an excess with 80% probability only at about 9–11 points, under a model of independent tasks without distribution shift (Appendix B, which also gives one shift-adjusted figure). The released harness records freeze claims but cannot verify prior human contact; its current adapters receive test outputs under a documented honesty contract, and a label-isolated task object with targets visible only to scoring is required before the harness can be said to enforce the information boundary. Task independence within ARC is approximate: a pair-level check finds seven evaluation tasks that share demonstration or test pairs with support-pool tasks, all treated in Section 5.2, and one pair of support-pool tasks that are identical under different identifiers (`070dd51e`, `40853293`); near-duplicates (the same rule on different grids) were not audited. Public evaluation is the only split available to us; a private-set rerun is the definitive remedy for any capability claim. LGE is one system and ARGA one external case; a calibrated detector would need controlled exposure and no-exposure comparisons with estimates of false-positive rate and power across differing task distributions.

7 Conclusion

Checking method names for evaluation identifiers is too weak a provenance test for symbolic ARC systems, and so, we found, was our own support-based check. A procedure-level audit that freezes the executable pipeline, refits its trainable part on a development partition, and compares reserved-training and public-evaluation outcomes under an explicit three-way rule flagged our existing pipeline on every split and returned inconclusive outcomes for the manifests selected without evaluation outcomes, whose reversed gap is consistent with a measured difficulty shift; a recount of our support diagnostics by task identity and content exposed two counting errors along the way. The audit is a diagnostic, not a certificate, and at ARC-AGI-2's pool sizes it flags an excess with 80% probability only at about 9 to 11 points. It makes suspected contamination a reproducible statistical question and, combined with commit provenance, points to the executor-authoring step as the locus of the excess. It is cheap enough to run before public-evaluation gains from a symbolic solver are reported, and that is where we recommend it be run.

AI USE STATEMENT

Generative AI tools were used in this work as follows. (i) Language-model coding assistants were used to write and refactor parts of the solver, the audit harness, the correction and verification scripts, and the analysis code; LGE's perception layer optionally uses a language model as a symbolizer on three text tasks, as stated in Appendix G. (ii) Language-model assistants were used to search and check the literature, to review the manuscript's claims against the saved artifacts (the recounts reported in Section 5.2 were first produced by such a review and then re-executed), to check the proof and recompute the numerical instantiation of the bound in Appendix A, to help design the audit protocol and its statistical conventions, and to draft and edit text. Generative AI tools were not used to generate or alter experimental outcomes. The authors reviewed the AI-assisted code and text, re-ran the correction scripts whose outputs appear in the tables, and take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The paper reports errors in our own earlier diagnostics and corrects them; all corrected numbers are traceable to released artifacts. The work uses public benchmark data and public third-party code (ARGA, under its published license) and involves no human-subject data. The audit is intended as a research-integrity instrument; it should not be used to accuse specific systems of misconduct, since a flagged outcome has explanations other than evaluation contact.

REPRODUCIBILITY STATEMENT

The supplementary archive contains the procedure-audit harness, the LGE package and the audited procedures needed to rerun every ARC-AGI-2 audit (including the reranker script, its training report and the cached candidates behind 54/120), the ARC task files used, the ARGA adapter and patch diff, the corrected 60-method manifest, the split seeds, and the per-task JSON artifacts behind every table: the seed-42 procedure audit and the seed-1/2/3 reruns at a single commit, the corrected oracle and consistency-picker reruns, the full-evaluation certificate, the nested-selection controls, the difficulty-shift and split-sensitivity analyses, the support-pool recount that produced Section 5.2, the pair-level content-overlap check, and the power enumeration of Appendix B. Appendix C gives the operational protocol; Appendix A the bound and its corrected constants; Appendix D the ARGA patches; Appendix G the supporting benchmark results with their artifacts. The settings of the audited procedures (top-k, ranking formulas, smoothing constants) were fixed during the April 2026 development that also iterated on the public evaluation set (Section 2); none was changed after the audits were frozen, and no parameter of the correction analyses was tuned on the evaluation split. All scripts use archive-relative paths; a script recomputes the reported ARC-AGI-2 numbers from the per-task artifacts, and a smoke test reruns slices of the audits from source and compares them with those artifacts.

REFERENCES

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 942–963, 2025.

ARC Prize Foundation. ARC-AGI-2 public evaluation frontier model attempt traces. Hugging Face dataset, revision e321bbeb of March 19, 2026, accessed April 2026, 2026a. URL `https://huggingface.co/datasets/arcprize/arc_agi_v2_public_eval`.

ARC Prize Foundation. GPT-6 Astra results on ARC-AGI-1, ARC-AGI-2 and ARC-AGI-3. ARC Prize results table, accessed September 5, 2026b. URL `https://arcprize.org/results/openai-gpt-6-astra`.

Qiming Bao, Alex Yuxuan Peng, Tim Hartill, Neset Tan, Zhenyun Deng, Michael Witbrock, and Jiamou Liu. Multi-step deductive reasoning over natural language: An empirical study on out-

of-distribution generalisation. In *16th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy), IJCLR 2022*, 2022.

Avrim Blum and Moritz Hardt. The Ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1006–1014, 2015.

Matthew Bowers, Theo X. Olausson, Lionel Wong, Gabriel Grand, Joshua B. Tenenbaum, Kevin Ellis, and Armando Solar-Lezama. Top-down synthesis for library learning. *Proceedings of the ACM on Programming Languages (POPL)*, 7, 2023.

Natasha Butt, Blazej Manczak, Auke Wiggers, Corrado Rainone, David Zhang, Michaël Defferrard, and Taco Cohen. CodeIt: Self-improving language models with prioritized hindsight replay. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.04858.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pp. 267–284, 2019. arXiv:1802.08232.

François Chollet. On the measure of intelligence, 2019. arXiv:1911.01547.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems, 2025. URL https://arxiv.org/abs/2505.11831. arXiv:2505.11831, revised January 15, 2026.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)*, 2021.

Daniel Franzen, Jan Disselhoff, and David Hartmann. The LLM ARChitect: Solving ARC-AGI is a matter of perspective. Technical report, ARC Prize 2024 (first place; paper award runner-up), 2024. URL https://github.com/da-fr/arc-prize-2024/blob/main/the_architects.pdf.

Daniel Franzen, Jan Disselhoff, and David Hartmann. Product of experts with LLMs: Boosting performance on ARC is a matter of perspective. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2505.07859.

Gabriel Grand, Lionel Wong, Matthew Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning interpretable libraries by compressing and documenting code. In *International Conference on Learning Representations (ICLR)*, 2024.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9):100804, 2023. doi: 10.1016/j.patter.2023.100804.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. BIG-Bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, 2025. doi: 10.18653/v1/2025.acl-long.1285.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, 2018.

Chao Lei, Nir Lipovetzky, and Krista A. Ehinger. Generalized planning for the abstraction and reasoning corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. arXiv:2401.07426 (extended version).

Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M. Dunn, Hao Tang, Wei-Long Zheng, Yewen Pu, and Kevin Ellis. Combining induction and transduction for abstract reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2411.02272.

Isaac Liao and Albert Gu. ARC-AGI without pretraining. arXiv:2512.06104; ARC Prize 2025 Paper Award, third place, 2025. URL https://arxiv.org/abs/2512.06104.

Robert G. Newcombe. Interval estimation for the difference between independent proportions: comparison of eleven methods. *Statistics in Medicine*, 17(8):873–890, 1998. doi: 10.1002/(SICI)1097-0258(19980430)17:8<873::AID-SIM779>3.0.CO;2-I.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori B. Hashimoto. Proving test set contamination in black-box language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. Self-improving language models for evolutionary program synthesis: A case study on ARC-AGI. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2507.14172.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5389–5400, 2019.

Filipe Marinho Rocha, Inês Dutra, Vítor Santos Costa, and Luís Paulo Reis. Program synthesis using inductive logic programming for the abstraction and reasoning corpus. In *Hybrid Models for Coupling Deductive and Inductive Reasoning (HYDRA 2024)*, Communications in Computer and Information Science, pp. 52–68. Springer, 2025. doi: 10.1007/978-3-031-89366-7_4.

Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. A meta-analysis of overfitting in machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations (ICLR)*, 2023.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Yudong Xu, Elias B. Khalil, and Scott Sanner. Graphs, constraints, and search for the abstraction and reasoning corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias B. Khalil. LLMs and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *Transactions on Machine Learning Research*, 2024. arXiv:2305.18354.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-Discover: Large language models self-compose reasoning structures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

A HELD-OUT AUDIT BOUND

Let $\mathcal{E}$ be a finite class of executors fixed independently of an i.i.d. support sample H of size N, $\hat{p}_h$ the empirical exact-solve rate of h on H and p_h its population rate.

Theorem 1. *For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the draw of H, every $h \in \mathcal{E}$ satisfies $p_h \geq \hat{p}_h - \sqrt{(\ln|\mathcal{E}| + \ln(2/\delta))/(2N)}$.*

Proof. Hoeffding's inequality (Hoeffding, 1963) gives $\Pr[|\hat{p}_h - p_h| \geq \epsilon] \leq 2e^{-2N\epsilon^2}$ for each h; a union bound over $\mathcal{E}$ and solving $2|\mathcal{E}|e^{-2N\epsilon^2} = \delta$ for ϵ yields the claim. □

Numerical instantiation. For $|\mathcal{E}| = 387$, $N = 1{,}033$ distinct support identifiers and $\delta = 0.05$, the slack is $\sqrt{(\ln 387 + \ln 40)/2066} \approx 0.0683$, whereas $2/N \approx 0.0019$; an admission threshold above the slack would require at least 71 successes out of 1,033. Counting the 1,800 pool records as if distinct gives a slack of 0.052 and a requirement of 94; neither a two-success nor a ten-success threshold is certified under either count. Grouping tasks that share a demonstration or test pair, the pools hold 1,032 distinct tasks (Section 6); the slack is then 0.0684 and the requirement still 71 successes. The historical support pools were assembled and inspected adaptively and contain duplicated tasks, so deduplication alone does not make them an i.i.d. sample; the bound is a conditional illustration, not a guarantee for the retrospective audit.

B POWER OF THE DECISION RULE

We model the reserved subset and the evaluation set as independent binomial samples of $n_H = 200$ and $n_E = 120$ tasks with rates p_H and p_E, and compute by exact enumeration over all 121×201 outcome pairs the probability that the rule flags, i.e. that the lower bound of the Newcombe hybrid-score 95% interval on Δ exceeds $\delta = 0$ (Table 7). The same interval code reproduces the five intervals of Table 3 and the corrected-oracle interval of Section 5.3. For comparison, the one-sided Fisher exact test at level 0.05 flags with probability 0.028–0.034 at equal rates and reaches 80% power at excesses of 8.7–10.4 points. The model assumes independent tasks and no distribution shift. Under the measured shift a procedure without evaluation contact is expected to score lower on E, which lowers the false-flag rate and raises the excess needed over that expectation: for a procedure with the v4 manifest's training-pool difficulty response ($p_H \approx 7\%$, expected evaluation rate 1.1%; Section 5.6), an 80% flag probability requires an evaluation rate of about 17%, some 16 points above its difficulty-adjusted expectation.

Table 7: Power of the decision rule ($\delta = 0$, $n_H = 200$, $n_E = 120$): the flag probability when $p_E = p_H$, and the evaluation excess $p_E - p_H$ at which the flag probability reaches 50% and 80%.

p_H	flag rate at $p_E = p_H$	excess for 50% (pp)	excess for 80% (pp)
5%	0.026	5.9	9.1
7%	0.026	6.7	10.2
9%	0.026	7.4	11.1

C OPERATIONAL AUDIT CHECKLIST

1. Declare the target claim: a public-development diagnostic or an unseen-generalization claim.
2. Canonicalize task identifiers and content across every pool; remove overlaps, including benchmark-version reuse of identifiers, before counting support.
3. Commit split seeds before execution; sample T_{held} from the training set with a published seed.
4. Fit every statistically fitted component on D only; freeze code, model, DSL, manifest, prompts, tools, ranker, budgets and thresholds at a commit hash dated before the audit run, and record any earlier evaluation contact.

5. Hide test outputs from the inducer; expose only demonstrations and test inputs through a task object whose targets are visible to scoring alone.

6. Allow only preregistered label-free test-input heuristics (Appendix E).

7. Run identical budgets on T_{held} and E; log per-task traces, errors, timeouts, hashes and environment.

8. Preregister the flagged-direction test, report Δ with a difference interval and the reverse-direction exact test descriptively; apply the flagged / consistent / inconclusive rule with a stated tolerance and a preregistered capacity floor; state a difficulty-adjusted expectation if one is used.

9. Before the run, compute the rule's power at the pool sizes and the expected p_H (Appendix B), and state the smallest excess the audit flags with 80% probability.

10. After any evaluation contact, obtain new unseen or private data for the capability claim.

D ARGA PATCHES FOR THE EXTERNAL CASE STUDY

ARGA's published code (`https://github.com/khalil-research/ARGA-AAAI23`) assumes equal input and output shapes in scoring and constraint acquisition; on ARC-AGI-2 this raises structural exceptions that prevent any audit. Five patches to `task.py` (upstream revision `9787bf0`) were applied: (1) `Task.calculate_score` scores a candidate whose reconstructed grid has a different shape than the expected output as a full-mismatch penalty instead of raising; (2) the unconditional plotting calls in `Task.solve` are skipped when images are not requested; (3) `Task.constraints_acquisition_global` skips demonstration pairs with mismatched shapes and sets `no_movements=False` for them; (4) the integer-string key with which the search discards duplicate candidates is replaced by a SHA-256 hash, avoiding Python's integer-string conversion limit; (5) the final test-output check in `Task.solve` counts every cell as an error when the shapes differ instead of raising. The audit artifact's label "4-patch" counts patches 1–4; the audited file also contains patch 5. Patches 2 and 5 run after the search and change neither scores nor the program returned; without patch 2, two of the ten reserved-subset successes raise at the plotting call, and patch 5 affects only whether a task is recorded as induced. Patch 4 changes neither scores nor priorities, but the upstream key merged all-zero token strings of different lengths, so the patched search keeps some candidates that the upstream search discards. Patches 1 and 3 change control flow on shape-changing tasks and can change which candidates are reached within the budget. ARGA's own search limit was 60 s, inside a 75 s task timeout enforced on a best-effort basis (three tasks, all failures, ran 79–203 s). ARGA's output has its input's shape, so on shape-changing tasks the adapter returns ARGA's best non-zero-score program, which is scored as incorrect. The adapter applies a returned program to a test input without ARGA's table of static objects, so programs that insert a static object fail at test time; this can only lower ARGA's counts, and it changed no outcome among the 27 reserved-subset tasks whose search ended before the time limit (the other tasks were not checked). We report the patched behaviour as ARGA's behaviour under these patches. The patches were checked by inspection of the affected code paths; no patched-versus-unpatched comparison on ARC-AGI-1 tasks is released, so we make no preservation claim. The patched file, the diff and the upstream revision are in the supplementary archive.

E CONSISTENCY SCORE FOR DEPLOYABLE SELECTION

For each demo-fit transform τ and test input $x^\star$ the score sums: $+1$ if the output-minus-input shape delta matches some demonstration's delta, else -0.5; $+1$ if the output palette is contained in the union of demonstration and test-input palettes, else -0.5 per extraneous colour; $+0.5$ if the demonstration outputs share a mode colour and the output has it, else -0.5; $+0.3$ if the non-zero connected-component count lies within the demonstrations' range plus two, else -0.2. The picker selects the highest-scoring demo-fit transform. The weights were fixed by hand before any evaluation run and were not tuned on evaluation.

F PER-TASK RULE-INDUCTION PROTOTYPES

To test the capacity explanation at a higher effort level than the in-body synthesizer batches, we built a symbolic trace extractor (per-object features, three-pass matching, per-object delta classes) with three typed synthesizers (global colour map, translate-by-predicate, recolor-by-predicate). Its predicted gain of six to ten recoveries was recorded before the run; it recovered none of the 133 reserved-subset tasks that π_{narrow} failed at the prototype's commit. A constraint-acquisition engine in the ARGA/GPAR style (two connectivity abstractions, four filter families, nine to 24 transformation searches, optional two-step composition and refinement) lifted the reserved subset monotonically from 7/200 to 10/200 across five versions while evaluation stayed at 0/120. An abductive prototype fitted a path-flow task (`332f06d7`) it was built on and none of four blind tasks committed before it was written. These prototypes show that the tested expansions of the rule-induction machinery move the reserved-subset rate by a few tasks without moving evaluation; they do not exhaust the space of capacity explanations.

G SUPPORTING EVIDENCE OUTSIDE ARC

These results are reported for context on what the LGE substrate does where rule induction is well posed; none of them bears on the audit. **SCAN** (Lake & Baroni, 2018): a 16-entry rule table learned from the training split (four primitive verbs, two turn directions, two *opposite* modifiers, four *around* modifiers, two repetition counts and two connectors) reaches 100% on all 15,808 test examples of the simple (4,182), add-primitive-jump (7,706) and length (3,920) splits. **BBH** (Suzgun et al., 2023): on 14 structured tasks (3,396 examples; 250 per task except 146 for *penguins*) the micro-average is 84.25% (2,861/3,396) and the macro-average 84.2%; the 11 tasks solved with deterministic parsers average 86.1%, and three tasks use a language-model parser (Table 8). **BBEH** (Kazemi et al., 2025): Table 9 reports all 12 tasks with released per-example artifacts (200 examples each); the substrate is competitive on a few structured tasks and near zero on most, and Kazemi et al. (2025) report GPT-4o at 8.0% and o3-mini at 55.0% on *dyck languages* for reference. **PARARULE-Plus** (Bao et al., 2022): 88.22% (9,526/10,798) by forward chaining with the frozen library. **PrOntoQA** (Saparov & He, 2023): proof success 100% on 7,900 out-of-distribution examples (mean gold-fact step recall 0.959). **bAbI**: 900/900 across nine reasoning types (a saturated sanity check). **GSM8K**: 47% (95% CI 37–57), reported as a negative result where reasoning is entangled with comprehension.

Table 8: BBH structured tasks. $n = 250$ except *penguins* ($n = 146$). †: language-model parser. GPT-4 chain-of-thought numbers from Zhou et al. (2024).

task	LGE (%)	GPT-4 CoT (%)
word_sorting	100.0	90
boolean_expressions	100.0	85
dyck_languages	100.0	69
web_of_lies	100.0	80
multistep_arithmetic	100.0	92
tracking_shuffled_objects	85.6	80
colored_objects†	85.6	80
temporal_sequences	82.8	99
penguins† ($n = 146$)	82.9	100
object_counting	81.6	100
navigate	80.8	90
logical_deduction	65.2	70
date_understanding†	63.6	80
geometric_shapes	50.8	56
macro-average (micro)	84.2 (84.25)	83.6

H FRONTIER CONTEXT ON ARC-AGI-2 PUBLIC EVALUATION

Table 10 lists task-exact counts on the same 120 public evaluation tasks computed from the public attempt traces released by ARC Prize (ARC Prize Foundation, 2026a) in April 2026 (pass@2, miss-

Table 9: BBEH: all 12 tasks with released per-example artifacts (200 examples each; task-exact %). †: language-model symbolization variant (the deterministic variant scores 9.5 on *shuffled objects* and 0.0 on *object properties*).

task	LGE (%)	task	LGE (%)
dyck_languages	64.0	spatial_reasoning	6.0
shuffled_objects†	57.5	boolean_expressions	4.0
web_of_lies	20.0	time_arithmetic	4.0
word_sorting	20.0	multistep_arithmetic	2.0
temporal_sequence	13.5	object_counting	2.0
object_properties†	1.0	zebra_puzzles	0.0

ing coverage counted as incorrect). The sole substrate-general LGE solve (`1818057f`) is solved by all seven frontier panels, and every task in the held-out-supported band after identity and content exclusion (`1818057f`, `5dbc8537`, `89565ca0`, `e3721c99`) is solved by at least three of them, so the audited substrate offers no complementary coverage. The three tasks among LGE's 54 family-supported solves that no panel solves (`13e47133`, `269e22fb`, `9bbf930d`) were solved by executors with no distinct held-out support at all. These traces predate the September 2026 report of GPT-6 Astra at 95.0% on ARC-AGI-2 at maximum reasoning effort (ARC Prize Foundation, 2026b); frontier scores on the repeatedly inspected public set and on the organizers' verified semi-private set have different exposure and are not comparable to LGE's numbers as a matched baseline.

Table 10: Task-exact counts on the 120 public evaluation tasks (April 2026 traces).

system	covered	task-exact / 120
GPT-5.4 pro (xhigh)	107	96 (80.0%)
GPT-5.4 (high)	120	86 (71.7%)
Claude Opus 4.6 Thinking (max)	117	83 (69.2%)
GPT-5.2 Thinking (xhigh)	107	59 (49.2%)
Gemini 3 Deep Think Preview	117	48 (40.0%)
Claude Opus 4.5 Thinking (64k)	120	43 (35.8%)
Gemini 3 Pro Preview	116	28 (23.3%)
LGE family-supported	120	54 (45.0%)
LGE held-out-supported, identity and content excluded	120	4 (3.3%)
LGE substrate-general	120	1 (0.8%)
LGE corrected 60-method manifest (oracle)	120	0 (0.0%)

I PIPELINE AND RERANKER

For each method m and family p the trace miner records n_m, n_p (training tasks fit) and e_m, e_p (held-out training outputs solved exactly). The run that produced 54/120 ranks candidates with an empirical-Bayes reranker fit on the traces of all 1,000 training tasks: $\hat{\rho}_p = (e_p + 5\rho_0)/(n_p + 5)$ with $\rho_0 = 0.32$ the global exact rate, $\hat{\rho}_m = (e_m + 3\hat{\rho}_p)/(n_m + 3)$, and lexical token priors smoothed toward ρ_0 with strength 5, whose geometric mean over a method's name tokens is $\hat{\rho}_t$; an unseen executor name backs off to the geometric mean of its family and token priors. A candidate scores $\hat{\rho}_m\sqrt{\hat{\rho}_p\hat{\rho}_t}\,(0.25+0.75q)$, where q is the proposer's family probability, times a support factor $n_m/(n_m+1)$ for executors seen in the traces, and zero if it fails the label-free shape check. The audited procedure π_{narrow} (Section 5.1) instead scores $(\rho_p+0.01)^2(\rho_m+0.01)(0.05+0.95q)$, where ρ_p and ρ_m are the exact fractions among D's trace records for the family and the executor. Those records contain only exact solves, so $\rho_p = \rho_m = 1$ for every family and executor with a record in D, and 0.05 is used otherwise; the score therefore mainly favours executors observed in D and, among them, families with high proposer probability. The proposer is a categorical Naive-Bayes model with $\alpha = 0.5$ smoothing and top-$k = 30$, which exceeds the 19–21 families with records in D. No smoothing constant was retuned per evaluation task. Figure 3 shows the pipeline and the point at which the diagnostics of Section 5 are applied.

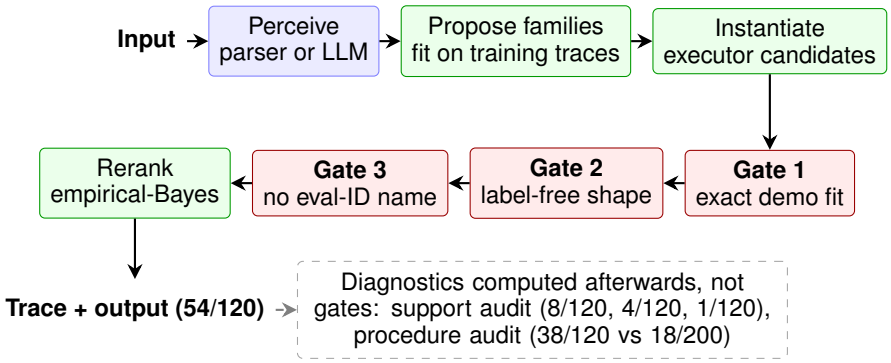

Figure 3: The LGE pipeline behind the 54/120 run, with proposer and reranker fit on the traces of all 1,000 training tasks; π_{narrow} refits the proposer on D and replaces the empirical-Bayes reranker with the score given above. The family-supported number (54/120) is produced by gates 1–3; the support bands and the procedure audit are diagnostics applied to that output and to the frozen procedure, not gates that the pipeline enforces.