HOLDING THE ANSWER FIXED: WHAT HIDDEN-STATE PROBES AND INTERVENTIONS ESTABLISH ABOUT CORRECTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hidden-state probes are often interpreted as measuring whether a model's answer is correct. But correctness is relational: the same answer can be right for one task and wrong for another, while standard evaluations usually allow both the task and the answer to vary. We study what probe success establishes when answer identity is held fixed. We construct paired task variants in which the answer is identical but its checker outcome flips. Across six 7B–32B code and tool-use models, newly fitted probes almost perfectly distinguish these fixed-answer task conditions. Yet this strong decodability does not imply that an existing correctness score is reusable: probes learned from freely generated answers transfer inconsistently to the same candidate states and can even reverse their ordering under the tested shift. We uncover a second separation within a single frozen score. Two probes correctly order 93% and 95% of fixed-answer pairs, while incorrect answers still outrank correct answers from other problems in about one third of cross-problem comparisons; generation-matched mazes exhibit the same gap. Input controls show that the surviving signal can arise from different task-dependent sources. Finally, late whole-state patches reproduce donor outputs whether those outputs are correct or incorrect. Together, these results show that decodability, score reuse, fixed-answer responsiveness, cross-problem ranking, and output restoration are distinct properties. Correctness probes should therefore be evaluated against the comparison required by their intended use, rather than by pooled discrimination alone.

1 INTRODUCTION

Hidden-state probes are increasingly used to ask whether a model "knows" that its answer is correct. Trained to distinguish correct from incorrect answers from internal representations, their held-out accuracy or AUC measures how readily correctness-related labels can be decoded (Kadavath et al., 2022; Marks & Tegmark, 2024; Vilas et al., 2026). The prospect extends beyond interpretability: a reliable score could support verification, answer selection, or selective prediction without an external checker for every output.

Correctness, however, relates an answer to a task. The same program may satisfy one specification and fail another after the required output convention changes; the same route may become nonoptimal when a shortcut opens. Standard evaluations usually allow both task and answer to vary. A high AUC can then draw on answer identity, task information, or their interaction. Which of these supports the score matters when the task changes or the score is reused.

We study this ambiguity by holding the answer fixed and changing whether it is correct. Within each problem family, two task variants have the same model output—or supplied candidate—but opposite external-checker outcomes. Every deterministic answer-only score must tie within the pair, so correct ordering requires information beyond answer identity. The changed task may still supply easy input or compatibility cues. We use input controls to identify these sources, then examine score reuse, ranking, and interventions (Figure 1).

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.

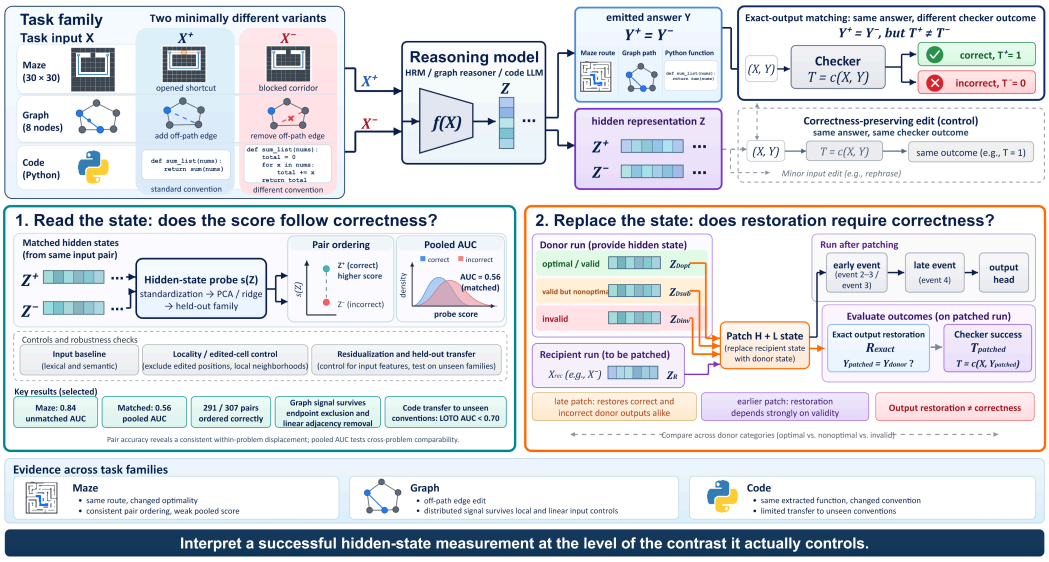


Figure 1: **Separating answer identity from correctness.** Task variants change the checker outcome of a fixed answer. Readouts compare paired ordering with pooled discrimination and test input explanations; interventions compare output restoration across donor categories and computation steps. The generation-matched branch is illustrated for mazes, graphs, and code; the code inset summarizes the original convention study (Appendix E). Candidate validation extends the comparison by replaying a supplied answer under both task conditions (Section 2.2).

The experiments reveal both sides of this distinction. Correctness-preserving route changes move an ordinary maze probe about thirteen times more than fixed-route correctness changes. Yet fixed-answer distinctions remain highly decodable: newly fitted probes reach 0.998–1.000 AUC across six 7B–32B code and tool-use models.

Does an existing score track that decodable distinction? We train probes on freely generated answers and freeze them on supplied candidates. On identical candidate states, newly fitted probes are nearly perfect, while frozen scores range from reliable transfer to reversed ordering. Generated and supplied answers differ jointly in provenance, format, and error composition. This composite shift tests score reuse in a new population.

A second separation holds within a single frozen score and test population. Code 7B and tool 32B correctly order 93% and 95% of fixed-answer pairs at pooled AUCs of only 0.67 and 0.68; generation-matched mazes order 291/307 at 0.56. Within-problem responsiveness asks whether the score follows revised requirements. Cross-problem ranking also depends on differences in score level between problems. We formalize this distinction: even perfect pair ordering permits pooled AUC approaching $1/2$. Candidate selection and screening provide two practical uses with different ranking requirements.

Input controls locate different sources of the surviving response. Maze occupancy, code prompt–program compatibility, and tool length/likelihood explain strong readouts in the constructed populations. Graph discrimination survives both excluding edited endpoints and removing linearly predictable adjacency information, while nonlinear input structure remains available. Interventions pose a parallel question: late whole-state patches in maze and Sudoku reasoners reproduce correct and incorrect donor outputs alike. Measuring output equality and checker success separately distinguishes support for an answer from success on its task.

These results distinguish the hidden-state claims required by different uses. Our contributions are:

1. **Fixing the answer identifies task-dependent response.** Ordinary scores can follow answer changes while fixed-answer distinctions remain highly decodable (Sections 2–3).

2. **Decodability and score reuse separate on identical states.** Frozen generated-answer probes transfer inconsistently where newly fitted probes succeed (Section 3).
3. **Within-problem responsiveness and cross-problem ranking impose different demands.** A single score can reliably order pairs yet rank across problems weakly; we characterize this gap formally (Section 4).
4. **Output restoration and correctness are distinct outcomes.** Late patches reproduce both correct and incorrect outputs. Input controls and donor comparisons tie each measurement to its task and outcome (Sections 5–6).

2 Testing Correctness at a Fixed Answer

2.1 An accurate probe can respond mainly to answer changes

The released Hard Maze Hierarchical Reasoning Model (HRM) (Wang et al., 2025) predicts routes through 30×30 grids; the checker accepts any shortest route. Opening a corridor can make a persisting route nonoptimal. Alternatively, an edit can change the route while preserving optimality. These edits vary the answer and its correctness separately.

A linear probe on pooled states reaches 0.84 AUC on ordinary runs, where routes vary. Freezing it reveals that large correctness-preserving route changes move its score about 13 times as much as fixed-route correctness changes. With the route fixed, correctness-changing and correctness-preserving edits produce similarly small responses. The ordinary detector is most sensitive to route-associated variation.

The probe uses 1,088 runs in 100 maze families, each held out in full. The matched comparison retains its representation and fitting procedure (Appendix C.4).

2.2 Matching answers across a change in correctness

Let X denote a task, Y an answer, and Z the associated hidden state. An external checker assigns $T = c(X, Y) \in \{0, 1\}$. We construct task variants within a problem family and compare pairs satisfying

$$Y_i^+ = Y_i^-, \qquad c(X_i^+, Y_i^+) = 1, \qquad c(X_i^-, Y_i^-) = 0. \tag{1}$$

The superscripts denote checker outcomes. Any answer-only score $g(Y)$ is identical for the two members. A probe that orders them correctly must therefore use information beyond which answer is present.

For mazes and graphs, independent runs on both task variants are retained when their complete path marks or path-token sequences match; the original code study matches extracted programs. This selects the output-persistence population. Opening a shortcut changes the answer in 630 of 783 cached maze runs, while 153 retain it with a changed checker outcome. Matching examines these retained cases; Appendix B.1 reports both edit directions and retention counts.

For supplied answers, a language model processes the same candidate under two task conditions, one it satisfies and one it fails. Replay fixes its identity even when independent generations rarely repeat. These candidate-checking states support the score-reuse test in Section 3.

2.3 Measuring the response

Training excludes the evaluation problem family. We call the score's response to a task change within a family, with its answer fixed, *fixed-answer responsiveness*. Strict *pair accuracy* measures its direction: the fraction of pairs in which the correct member scores higher. Ties are not wins; frozen code 32B has one tie. Pooled AUC also includes cross-problem comparisons and gives ties half credit. Section 4 examines why these measurements can disagree.

The same held-out states support two measurements. Fitting on matched training examples measures decodability for a readout adapted to that population. Keeping an earlier probe frozen measures score reuse. Their comparison holds the evaluation states fixed.

2.4 What the fixed-answer comparison identifies

Proposition 1 (answer-only null). On m exact-answer pairs, every deterministic answer-only score $g(Y)$ has zero strict pair accuracy and pooled AUC $1/2$ under the usual half-credit tie convention:

$$g(Y_i^+) - g(Y_i^-) = 0, \qquad \widehat{\mathrm{AUC}}(g) = \tfrac{1}{2}. \tag{2}$$

The positive and negative answer multisets are identical, so their score distributions coincide (proof in Appendix J). A task-dependent $h(X, Y)$ can still exploit compatibility cues; the controls test these remaining explanations. Generation matching conditions on output persistence, while replay fixes the candidate externally. Appendix K gives the executable protocol and concrete examples.

3 A New Probe Can Succeed Where an Existing Score Fails

3.1 From generated answers to supplied candidates

Can a generated-answer score be reused for supplied candidates? Model generations and reference candidates differ jointly in provenance, error types, and code presentation. Under this composite shift, the frozen G-trained probe and a probe fitted on separate V-training families score identical candidate test states.

The code study uses Qwen2.5-Coder-Instruct at 7B, 14B, and 32B (Hui et al., 2024); the tool study uses Qwen3 at 8B, 14B, and 32B with thinking disabled (Qwen Team, 2025). All use BF16. We construct 917 code problems from HumanEval and MBPP (Chen et al., 2021; Austin et al., 2021) by changing the required final output, for example by reversing a returned sequence. The 715 synthetic tool problems vary operation and argument requirements. In each candidate pair, the same reference program or tool call passes one version of the task and fails the other.

We fit linear logistic probes to the middle decoder block at the last answer token, using training-only standardization. Test problems are held out by source family: 192 code and 138 tool families, each contributing one correct and one incorrect candidate example. All six models have complete generation and candidate-validation runs. Readout sites and probe settings are fixed across sizes. Appendix H gives the construction and extraction details. Table 1 labels generated answers as G and supplied-candidate validation as V.

3.2 Score reuse across models and sizes

Frozen scores behave differently across both tasks and sizes (Table 1B). Code 14B and tool 14B reverse the desired ordering, with AUCs of 0.454 and 0.201; code 32B reaches 0.547. Tool 32B retains reliable paired responses at a pooled AUC of 0.681, a distinction examined below. Refitting on V improves AUC in every model, reaching 0.998–1.000 on the identical test states (Figure 2). Thus, strong decodability on a population does not imply that an existing correctness score can be reused on that population.

The shift includes changes in failure type: 548 of 583 tool 14B G failures are disallowed-key or parsing errors, while V has neither. Appendix H.4 describes these changes; Appendix L.2 shows the full paired-score distributions.

3.3 What makes the candidate distinction easy to read?

Input and likelihood controls reveal strong compatibility cues behind the answer-end readouts (Table 1A).

In code, a prompt–program compatibility rule reaches 1.000 accuracy and AUC: it checks whether the prompt requests a changed output contract and whether the reference program contains its recognizable transformation wrapper. Prompt-only and length/likelihood controls are near chance, missing this informative interaction. The post-hoc rule identifies an explicit solution to the construction; the state probe measures readability under the same conditions.

In tool use, length and likelihood reach 0.989–0.998 AUC, close to the state probe. Both constructions therefore make compatibility accessible even when score reuse fails. Removing candidate

Table 1: **Candidate readout and frozen score reuse on the same test states. A:** six complete V runs, with probes fitted separately to each feature. Length/LP uses input length, candidate length, and mean token log-probability; prompt uses TF–IDF. **B:** all six G-trained probes, frozen on the same V test sets as in A. Pair columns count strict wins; † marks one additional tie. Code uses Qwen2.5-Coder; tool use uses Qwen3. The post-hoc code prompt–candidate compatibility rule reaches 1.000 AUC at every size.

A. Fixed candidates: states and controls

		Pooled AUC				
Task	Size	Prompt	Length/LP	Prompt state	Answer state	Correct pairs
Code	7B	.486	.509	.688	.998	192/192
	14B	.486	.549	.643	1.000	192/192
	32B	.486	.512	.680	.999	192/192
Tool	8B	.544	.989	.431	1.000	138/138
	14B	.544	.998	.474	1.000	138/138
	32B	.544	.991	.494	1.000	138/138

B. Generated answers → fixed candidates: frozen score transfer

Task	Size	G AUC	V AUC	V AUC, refitted	Correct V pairs
Code	7B	.785	.672	.998	179/192
	14B	.826	.454	1.000	71/192
	32B	.735	.547	.999	129/192†
Tool	8B	1.000	.854	1.000	138/138
	14B	1.000	.201	1.000	19/138
	32B	1.000	.681	1.000	131/138

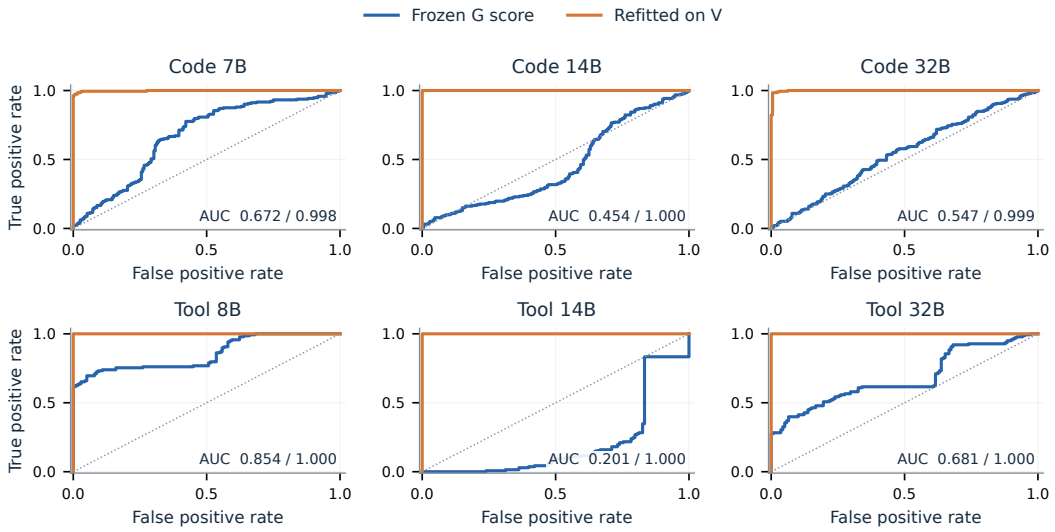


Figure 2: **Readable candidate distinctions need not survive score reuse.** Frozen and V-refitted ROC curves on identical candidate test states across all six models. Each panel reports frozen/refitted AUC, using 192 code or 138 tool test families. The probe and its preprocessing stay fixed during transfer; refitting uses separate V-training families. Paired-score distributions and group-bootstrap intervals appear in Appendix H.

objects seen in training preserves fitted-readout performance; unseen transformation rules remain a separate setting (Appendix H.6).

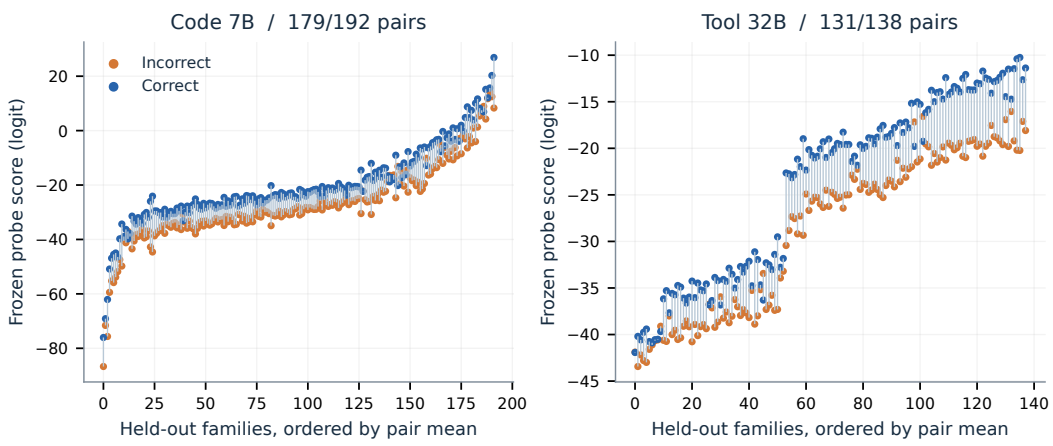


Figure 3: **One frozen score, two ranking questions.** Every held-out family is shown, ordered by its mean score. Segments join the incorrect (orange) and correct (blue) members of an identical-candidate pair. Code 7B orders 179/192 pairs correctly at AUC 0.672; tool 32B orders 131/138 at 0.681. Between-family variation overlaps the paired response. The displayed logits preserve the probability ordering in these two panels.

4 Fixed-Answer Responses and Cross-Problem Ranking

We next separate fixed-answer responsiveness within a problem from ranking comparability across problems. Consider correct/incorrect scores of $(101, 100)$ for one pair and $(1, 0)$ for another. Both pairs are ordered correctly, but the first problem's incorrect answer scores above the second problem's correct answer. Pair accuracy measures fixed-answer responsiveness; pooled AUC also depends on differences in score level between problems. AUC evaluates ranking, not probability calibration.

Let $q(a, b) = \mathbf{1}[a > b] + \frac{1}{2}\mathbf{1}[a = b]$ assign half credit to ties. Write the midpoint m_i and signed response d_i as

$$m_i = \tfrac{1}{2}(s_i^+ + s_i^-), \quad d_i = s_i^+ - s_i^-, \quad s_i^+ - s_j^- = (m_i - m_j) + \tfrac{1}{2}(d_i + d_j). \tag{3}$$

Proposition 2 (ranking decomposition). With one positive and one negative per family and $m > 1$,

$$\widehat{\mathrm{AUC}} = \frac{Q}{m} + \frac{m-1}{m}\widehat{\mathrm{AUC}}_{\mathrm{cross}}, \qquad Q = \frac{1}{m}\sum_i q(s_i^+, s_i^-). \tag{4}$$

Here $\widehat{\mathrm{AUC}}_{\mathrm{cross}}$ averages $q(s_i^+, s_j^-)$ over $i \neq j$; Q equals strict pair accuracy when there are no ties. Splitting the m^2 comparisons into diagonal and off-diagonal terms proves the identity. Even perfect strict pair accuracy permits AUC $(m+1)/(2m) \to 1/2$ (tight construction in Appendix J). This is score geometry, not a decomposition into neural mechanisms.

4.1 The gap appears within one frozen LLM score

The frozen code 7B probe orders 179/192 pairs correctly at pooled AUC 0.672; tool 32B orders 131/138 at 0.681. Figure 3 shows their paired responses against the variation in score level across families. Tool 8B also exhibits the gap, ordering all 138 pairs correctly at AUC 0.854. The effect thus occurs in both domains and at different model sizes.

For the same fixed probe and test set, incorrect members outrank correct members from other families in 32.9% (code 7B) and 32.1% (tool 32B) of cross-family comparisons. These constitute over 99% of pooled AUC terms; pair accuracy uses only each family's own members (Appendix H.5).

Using the score. The two 32B probes also evaluate four stochastic answers to each original task in disjoint, unfiltered cohorts. Best-of-N selection compares candidates within a task; screening ranks

Table 2: **The paired–global gap also appears under generation matching.** Maze scores use event-4 pooled states and family-held-out class-mean probes. The unmatched and matched probes are fitted separately. Terciles divide matched pairs by $\|\Delta Z\|$, with probes refitted within each group. Brackets give 95% family-bootstrap AUC intervals conditional on held-out scores.

Evaluation population	Families	Pooled AUC	Correct pairs
Unmatched runs	100	.84	—
Matched, all	307	.56 [.55, .58]	291/307
Smallest state change	103	.52	103/103
Intermediate	102	.53	101/102
Largest state change	102	.66 [.62, .71]	93/102

the predetermined first answer across tasks. These two uses put the ranking distinction into practice while keeping the fitted probe unchanged.

On 148 tool tasks, retaining the top 75% of first answers gives error rates of 7.2% for the answer-state probe and 24.3% for likelihood; input-text and prompt-state controls reach 5.4% and 2.7%. Screening therefore benefits from predictive structure already present in the input. Selecting among four candidates changes 109 correct tasks to 113, a gain of 2.7 percentage points with interval [0.0, 6.1]; code changes 111/155 to 109/155. The same score supports useful screening while offering little within-task selection gain. Appendix I.3 reports both cohorts, all controls, and uncertainty.

4.2 Generation-matched mazes reveal the same geometry

In 307 independent maze families, the model emits the same route before and after an edit changes whether it is shortest. A separately fitted probe, using the ordinary probe's representation and preprocessing, orders 291/307 pairs correctly at only 0.56 pooled AUC (Table 2). Weak cross-problem ranking again conceals consistent fixed-answer responsiveness.

Separately fitted probes within each state-change tercile also show the gap. In the smallest-change third, all 103 pairs are correctly ordered at 0.52 pooled AUC. Larger state changes do not correspond to better paired ordering in these fits; the subgroup counts are not a partition of the overall probe's predictions.

Ridge readouts and the original panel's family-centering analysis give complementary diagnostics (Appendix C.2). Centering uses both members at evaluation time to expose their relative displacement; it is not a correction available to a score that receives only one answer.

The gap in the single frozen LLM probe also rules out combining differently fitted cross-validation scores as its explanation.

5 Tracing the Signal Beyond Answer Identity

What explains the signal after matching? Task inputs still change, as the code wrapper rule and tool-likelihood baseline illustrate. In mazes and graphs, we know which cell or edge changed and compare correctness-changing with correctness-preserving edits. These controls distinguish registration of an edit from selective response to its task consequence.

5.1 Maze readouts distinguish pooled sensitivity from local input encoding

The matched maze probe responds more strongly to correctness-changing edits than to comparable preserving edits. We pair each of 103 preserving edits with a changing edit matched on wall direction and geometry. Their median absolute score changes are 0.024 and 0.048, respectively: a $1.97\times$ response to correctness changes (paired Wilcoxon $p = 1.5 \times 10^{-9}$). Unlike the ordinary detector in Section 2.1, this matched-fitted score is selectively sensitive to the checker-changing edit.

The strongest local readout has a simpler explanation. A probe at the edited cell reaches AUC 1.00, as does raw occupancy: in the matched pool, the cell is a wall in the optimal member and open in the nonoptimal member. Its state also changes by a similar amount under correctness-changing and

Table 3: **Earlier and late restoration across donor categories.** Joint $H+L$ patches in the released Hard Maze and Sudoku-Extreme models reproduce complete outputs. Earlier sites are event 3 for mazes and events 2–3 for Sudoku; the late site is event 4. Sudoku uses different samples at the two sites; these rates do not identify a universal timing boundary. Confidence intervals and displacement controls appear in Appendix F.

Task	Donor category	Earlier restoration	Late restoration
Maze	Optimal	250/259 (.97)	259/259 (1.00)
	Valid, nonoptimal	83/83 (1.00)	83/83 (1.00)
	Invalid	143/258 (.55)	258/258 (1.00)
Sudoku	Valid	24/24 (1.00)	64/64 (1.00)
	Invalid	13/24 (.54)	64/64 (1.00)

preserving edits. The perfect local score thus reproduces an available input cue, while the weaker pooled score carries a more selective paired response. Figure 4 and Appendix C give the local measurements and full controls.

5.2 Graph signals extend beyond the edited locations

For graph reasoning, we train recurrent message-passing, graph-attention, and Transformer models to predict shortest paths in 8-node graphs. An off-path edge edit changes whether an emitted path is shortest while preserving its full token sequence. Both the matched state probe and the edited-edge baseline reach AUC 1.00. This again makes the location and source of the signal the next question.

Excluding both edited endpoints from the pooled state retains AUCs of 0.85–1.00. We then fit a linear prediction of hidden states from the full adjacency input on training families, subtract that prediction, and probe the remaining states. AUC remains 0.70–0.76. Moreover, correctness-changing edits move the trained score more than preserving edits even though they move the state by a comparable norm. The per-architecture results appear in Appendix D.

The graph distinction is distributed beyond the edited locations and survives removal of linear adjacency information. The local maze score, by contrast, reproduces the edited cell's occupancy. The same matched comparison thus locates a direct input cue in one representation and a distributed task-dependent response in another.

Answer-only controls test what matching removes; prompt-only and edited-location baselines test task cues; compatibility rules test interactions those marginal baselines miss. Correctness-preserving edits test edit sensitivity, while endpoint exclusion and linear residualization restrict specific spatial and linear explanations. The surviving graph signal may still contain nonlinear input structure.

6 Output Restoration Across Donor Correctness and Time

Does restoring an answer also restore task success? We compare output equality and checker success across donor categories and computation steps. We use whole-state activation patching, replacing the recipient's joint $H+L$ state with a donor state and resuming computation (Heimersheim & Nanda, 2024). We record exact restoration separately from checker success:

$$R_{\text{exact}} = \Pr[\hat{Y}_{\text{patched}} = \hat{Y}_{\text{donor}}], \qquad T_{\text{patched}} = c(X_{\text{eval}}, \hat{Y}_{\text{patched}}). \tag{5}$$

To relate restoration to success, evaluate both answers on the same X_{eval}, as in the donor-task maze evaluation. If $A_{\text{patch}} = \Pr[T_{\text{patched}} = 1]$ and $A_{\text{donor}} = \Pr[c(X_{\text{eval}}, \hat{Y}_{\text{donor}}) = 1]$, then

$$|A_{\text{patch}} - A_{\text{donor}}| \leq 1 - R_{\text{exact}}. \tag{6}$$

Checker outcomes can disagree only when outputs differ. In particular, perfect restoration reproduces the donor's success rate, including its failures (Appendix J).

Late maze patches restore all 600 outputs, including 258 invalid routes; only 259 are optimal. One event earlier, valid donors—including nonoptimal routes—restore almost perfectly, versus 55% for invalid donors. Sudoku likewise shows an earlier validity association and complete late restoration

(Table 3). These category comparisons do not isolate validity as an independent cause. The earlier difference persists after matching maze donors by state-displacement norm. Equal-norm noise rarely changes outputs; Sudoku's displacement norms are comparable across categories (Appendix F). The association concerns validity: valid but longer routes are incorrect under the probe's optimality target, yet restore reliably at the earlier site.

Graph interventions show the converse separation. A donor with a different optimal path rarely restores the designated counterfactual target, yet produces an optimal patched output in 89% of cases. Equality uses the original target sequence; checker success accepts alternative optima (Appendix F.6). These metrics identify different successful interventions. The separate 10×10 step-800 timing study uses a different checkpoint and population (Appendix F). Whole-state replacement tests output support, rather than the fitted probe direction. Appendix I.2 reports the four-model LLM prefill study.

7 Related Work

Decodability and behavioral use. Probes predict self-knowledge, truth, and answer quality (Kadavath et al., 2022; Marks & Tegmark, 2024; Azaria & Mitchell, 2023; Burns et al., 2023; Orgad et al., 2025), with established interpretive limits (Belinkov, 2022): irrelevant properties remain decodable (Ravichander et al., 2021), while amnesic interventions test use (Elazar et al., 2021). Control tasks and information-theoretic probes address capacity and learning cost (Hewitt & Liang, 2019; Pimentel et al., 2020; Voita & Titov, 2020). We fix answer identity and use frozen transfer to distinguish decodability from score reuse.

Contrastive evaluation and uncertainty. Contrast sets test targeted input changes (Gardner et al., 2020); heuristic challenge sets expose simple benchmark solutions (McCoy et al., 2019). We require identical answers with opposite checker outcomes and test input explanations. Semantic uncertainty groups equivalent generations (Kuhn et al., 2023; Farquhar et al., 2024), and entropy probes estimate it from hidden states (Kossen et al., 2024). Exact-output matching instead asks how a fixed answer is represented under changed requirements.

Verification and selection. Outcome verifiers rank solutions (Cobbe et al., 2021); process supervision evaluates steps (Uesato et al., 2022; Lightman et al., 2024). Latent trajectories and activation changes support selection (Vilas et al., 2026; Liang et al., 2026), while sampling can amplify verifier errors (Rohatgi et al., 2026). We connect score reuse with selection and filtering. Calibration (Guo et al., 2017), risk–coverage evaluation (Geifman & El-Yaniv, 2017), and risk-controlled selection (Zecchin et al., 2025) require further evaluation beyond AUC (Appendix M).

Interventions and faithfulness. Causal tracing and interchange interventions link representations to behavior (Vig et al., 2020; Meng et al., 2022; Conmy et al., 2023; Geiger et al., 2021; 2025), with interpretations shaped by intervention and outcome definitions (Heimersheim & Nanda, 2024; Zhang & Nanda, 2024; Chan et al., 2022). Faithfulness studies separate a rationale's causal role from answer accuracy (Turpin et al., 2023; Lanham et al., 2023; Han et al., 2026). Our hierarchical-reasoner comparisons (Wang et al., 2025) instead separate output equality from checker success across donor categories and time.

8 Conclusion

Hidden-state "correctness" measurements support several distinct claims. Fixing the answer reveals task-dependent distinctions that can be highly decodable, while scores learned on another population may fail to transfer. Even within a single frozen score, reliable fixed-answer responsiveness can coexist with weak cross-problem ranking. Input controls locate different sources of the surviving signal; whole-state interventions separate output restoration from checker success.

Decodability, score reuse, fixed-answer responsiveness, cross-problem ranking, and output restoration each require their own comparison. The fixed-answer protocol distinguishes these claims and tests the one required by the intended use. Appendix M discusses scope and deployment limitations.

Reproducibility statement

The supplementary material specifies matching rules, checkers, probes, model settings, intervention protocols, and a claim-to-record evidence index. Appendix G summarizes how the evidence is organized; Appendix H documents the language-model data, completed runs, CPU analysis, and plotted predictions.

LLM usage statement

LLMs assisted with manuscript organization, prose revision, and analysis code. Reported experimental values are checked against saved artifacts; the authors are responsible for the scientific claims and their interpretation.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of EMNLP*, pp. 967–976, 2023.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7/.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *ICLR*, 2023.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *NeurIPS*, 2023.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021. doi: 10.1162/tacl_a_00359. URL https://aclanthology.org/2021.tacl-1.10/.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024. doi: 10.1038/s41586-024-07421-0. URL https://www.nature.com/articles/s41586-024-07421-0.

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 EMNLP*, 2020. URL https://aclanthology.org/2020.findings-emnlp.117/.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *NeurIPS*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *NeurIPS*, 2021.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Yunseok Han, Yejoon Lee, and Jaeyoung Do. RFEval: Benchmarking reasoning faithfulness under counterfactual reasoning intervention in large reasoning models. In *ICLR*, 2026. URL https://openreview.net/forum?id=2Gc8aj0afg.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *EMNLP*, 2019.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in LLMs. *arXiv preprint arXiv:2406.15927*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *ICLR*, 2023. URL https://arxiv.org/abs/2302.09664.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Zhenwen Liang, Ruosen Li, Yujun Zhou, Linfeng Song, Dian Yu, Xinya Du, Haitao Mi, and Dong Yu. Your reasoning model is secretly a reward model: Optimization-free verification from experience. In *ACL*, 2026. Anthology 2026.acl-long.788.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024. URL https://arxiv.org/abs/2305.20050.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *COLM*, 2024.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*, pp. 3428–3448, 2019. doi: 10.18653/v1/P19-1334. URL https://aclanthology.org/P19-1334/.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *NeurIPS*, 2022.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *ICLR*, 2025.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. In *ACL*, pp. 4609–4622, 2020. doi: 10.18653/v1/2020.acl-main.420. URL https://aclanthology.org/2020.acl-main.420/.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? In *EACL*, pp. 3363–3377, 2021. URL https://aclanthology.org/2021.eacl-main.295/.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J. Foster. Taming imperfect process verifiers: A sampling perspective on backtracking. In *International Conference on Learning Representations*, 2026.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *NeurIPS*, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. URL https://arxiv.org/abs/2211.14275.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *NeurIPS*, 2020.

Martina G. Vilas, Safoora Yousefi, Besmira Nushi, Eric Horvitz, and Vidhisha Balachandran. Tracing the traces: Latent temporal signals for efficient and accurate reasoning. In *ICLR*, 2026. arXiv:2510.10494.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. In *EMNLP*, pp. 183–196, 2020. doi: 10.18653/v1/2020.emnlp-main.14. URL https://aclanthology.org/2020.emnlp-main.14/.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Matteo Zecchin, Sangwoo Park, and Osvaldo Simeone. Adaptive learn-then-test: Statistically valid and efficient hyperparameter selection. In *ICML*, pp. 74018–74036, 2025. URL https://proceedings.mlr.press/v267/zecchin25a.html.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *ICLR*, 2024.

A Notation and Evaluation Guide

The main experiments use one distinction throughout: an answer is an object produced by the model, whereas correctness is a relation between that answer and a specified task. Table 4 collects the notation.

Table 4: Notation.

Symbol / term	Meaning
X	Task input, including the maze, graph, or program specification.
Y	Answer used for matching: emitted path marks, graph path tokens, or exact extracted program source.
Z	Hidden representation at a stated site and event.
$T = c(X, Y)$	External checker outcome; maze and graph optimality, or code test success.
H, L	HRM's slow and fast hidden modules.
Event k	Micro-step k within the final outer cycle.
Family	A base problem together with its related input variants.
Input baseline	A score using the edited input feature, evaluated on the same pairs as the hidden-state probe.
LOTO	Leave-one-template-out: probe fitted on all other convention templates.
R_{exact}	Complete patched-output equality with the designated donor output.

How to interpret the report. An unmatched score quantifies prediction on its stated population. A matched pair comparison removes the answer as a discriminator within that pair. A pooled matched score additionally depends on ordering across families and on the held-out probe fits. Input baselines, correctness-preserving edits, and leave-one-condition-out transfer establish what the remaining readout responds to and how far it travels. For patches, exact fidelity specifies the restored object, while the declared checker input specifies the task under which it is evaluated. Each measurement is interpreted for its stated population and prediction target.

B Maze Construction and Semantic Evaluation

The official counterfactual census contains 600 maze candidates. The released HRM produces 342 valid routes, of which 259 are shortest, and 258 invalid outputs. None matches the fixed canonical BFS label exactly. On the separate mechanism split, 382/384 examples admit multiple shortest paths, and exact canonical comparison rejects all 127 tested alternative optima. These results motivate evaluating validity and optimality independently of the canonical training label.

Checker implementation. The evaluator decodes the predicted grid, checks its length against the question grid, and collects the marked cells together with the start and goal. It checks for wall collisions and connectivity from start to goal on the marked graph. Extra marked cells outside the recovered route and missing interior marks invalidate the answer, excluding branches, cycles, and truncated markings. Optimality compares the valid route length with the BFS shortest length; canonical exactness compares with the fixed-neighbor BFS label. Marks form a set of grid cells.

Matched sampling. The original cache covers base mazes 500–599, with 1,088 traces and a median of nine traces per family; 91 families contain both optimal and nonoptimal traces and 29 supply a retained fixed-path correctness pair. Harvests on mazes 0–249, 250–499, and 600–999 add 278 families, for 307 in the primary pool. Each family contains one member of each checker class and retains identical endpoints and emitted path marks; 305 have a unique single-cell edit. The typical one-cell edit changes about 0.001 of the grid input. The designed correctness-preserving pool contains 103 one-cell pairs.

B.1 Behavioral selection and both edit directions

The matching rule conditions on an unchanged answer and a changed checker outcome. Table 5 reports the cached behavioral census before choosing independent evaluation families. Two edit directions produce matched pairs: opening a shortcut so that a persisting route becomes nonoptimal, and blocking a competing corridor so that a persisting route becomes optimal. The retained cases

describe output persistence after a task change, which is the regime in which recognizing that an existing answer has become wrong is separable from generating a replacement.

Table 5: Behavioral response before matched-family selection, and matched statistics at the edit level. Counts refer to cached edit runs, not the 307 independent evaluation families. Matched AUC and ordering use the frozen primary readout applied to the persisting, correctness-changing edits of each direction.

Edit direction	All edits	Output changes	Persists, T flips	Matched AUC	Ordered
Open a shortcut	783	630	153	0.57	145/153
Block a competing cell	5,543	3,359	2,138	0.57	2,070/2,138

Both directions give the same matched picture (AUC 0.57, ordering 95–97%), so the maze results do not depend on which way the task condition is changed. Output persistence is not hidden-state invariance: the retained pairs support 291/307 correct orderings, their path-token logit margins differ (Appendix C), and the graph construction retains the same path while supporting perfect matched discrimination. Matched scores characterize the retained output-persistence population.

C Probe Procedures and Supplementary Diagnostics

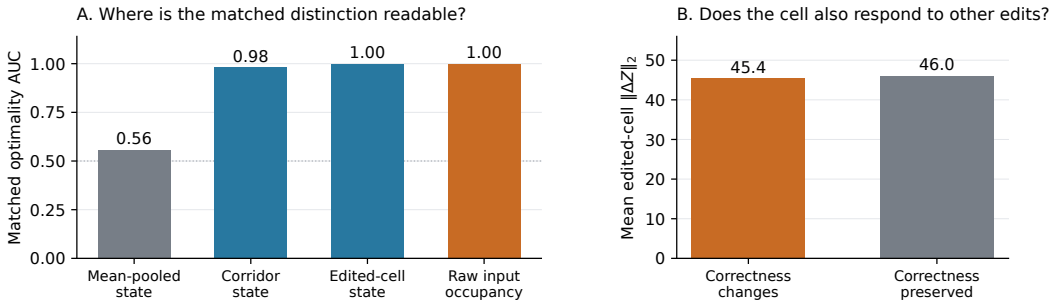


Figure 4: **Input controls explain the strongest local probe score.** Left: matched optimality AUC for the mean-pooled state, alternative-corridor mean, edited-cell state, and raw occupancy. Right: the edited-cell state changes by a similar mean L_2 magnitude under correctness-changing and correctness-preserving edits. Local hidden-state discrimination mirrors the edited input feature. Local probe results use the original 104-pair panel; the input baseline is also checked on the expanded pool.

C.1 Readouts, Preprocessing, and Uncertainty

The primary comparison applies training-fold standardization and PCA-16 before fitting the class-mean direction in both the unmatched and 307-family matched populations. PCA-16 ridge uses the same preprocessing and a regularized linear readout. The primary pooled maze readouts in this subsection use the released Hard Maze HRM checkpoint and event-4 representation site. Family-bootstrap intervals resample families while retaining their held-out scores; exact binomial intervals describe pair-order counts. The primary matched mass-mean AUC is 0.565 [0.550, 0.583] with pair accuracy 291/307 [0.917, 0.970].

State-change terciles. Each tercile uses a separately refitted family-held-out probe, including training-fold preprocessing; its counts therefore need not sum to the wins of the overall 307-family probe. The within-pair state change $\|\Delta Z\|$ over the 307 pairs has quantiles $p_{10} = 0.090$, $p_{25} = 0.128$, $p_{50} = 0.184$, $p_{75} = 0.313$, $p_{90} = 0.815$; tercile cut points are 0.144 and 0.250. Pooled AUC by tercile is 0.519, 0.530, and 0.660 (largest-tercile family bootstrap [0.621, 0.709]); strict pair ordering is 103/103, 101/102, and 93/102. For the smallest tercile we report the pooled AUC point estimate and the exact binomial interval for its 103/103 ordered pairs.

C.2 Pair ordering, family centering, and the source of the offset

For the primary pool, PCA-16 mass-mean scores order 291 pairs and PCA-16 ridge 280. On the earlier 104-family panel, the counts are 100, 95, and 94 for mass-mean, PCA-16 ridge, and uncompressed ridge; centering each two-member family yields pooled AUCs of 1.00, 0.98, and 1.00. Centering computes $z_1-(z_1+z_2)/2$ and $z_2-(z_1+z_2)/2$ and uses the counterpart state at evaluation time.

Family centering is a paired diagnostic: it requires both states at evaluation time, whereas an independently scored answer supplies only one. The ridge and centering comparisons above are retained as supplementary checks on score geometry.

Across events 1–5, the full-grid matched readouts correctly order 94–100 of the 104 panel families. A separate classifier distinguishes output-changing pairs from correctness-changing pairs at AUC 0.98 [0.93, 1.00]; its label is the type of comparison, not the identity of an emitted path.

C.3 Task features, confidence, and behavioral responses

Task-feature probes use path length, endpoint Manhattan distance, turns, mark count, validity, and extra/missing-mark counts. The length/slack/turn subset gives AUC 0.71, and the full set 0.85. A probe using the emitted mark set alone gives 0.48, and a path-length-only scalar 0.53. Path-token logit margins vary within matched pairs (Wilcoxon $p = 2.7 \times 10^{-5}$). Residualizing the unmatched state on this confidence scalar gives mass-mean AUC 0.497, compared with 0.859 before residualization, with cosine 0.691 between the two fitted directions.

C.4 Frozen-detector census

Table 6: **Answer changes move the ordinary probe more than fixed-answer correctness changes.** Median absolute changes in the frozen probe's standardized score, relative to a fixed-route optimality flip. Route-change census rows use single-cell edits; the filtered pool retains one large route change per family (route Jaccard overlap ≤ 0.80). Training excludes the evaluation family.

Answer	Optimality	Pairs	Relative $\lvert\Delta s\rvert$
Different maze family (reference)	—	400	21×
Route change, filtered pool	preserved	158	13×
Route change, $\geq$ 21 cells	preserved	532	3.0×
Route change, 6–20 cells	preserved	1,531	1.2×
Route change, 2–5 cells	preserved	1,030	0.7×
Fixed route	changes	307	1×
Fixed route	preserved	103	0.8×

Table 6 expresses medians relative to the fixed-route optimality flip. In absolute standardized units of the frozen detector, the fixed-route flip median is 0.016 on the full census and 0.014–0.018 on the family-level pools; the per-family filtered output-flip pool (Jaccard overlap ≤ 0.80 between routes) has median 0.20 on 158 pairs and 0.17 on the corresponding 443-pair harvest for the 307 pool (ratios 13 and 11); the census strata have medians 0.011 (2–5 cells, n=1,030), 0.018 (6–20 cells, n=1,531), and 0.047 ($\geq$ 21 cells, n=532); random different-family pairs (n=400) have median 0.33. Pairs in which both the route and optimality change (297 in the family-level pool) have median 2.23.

C.5 Spatial readouts and input controls

Edited-cell mass-mean AUC rounds to 1.00 at every event, with 103/104 correctly ordered panel pairs. At event 4, the one-hop neighborhood gives approximately 0.92 and the two-hop neighborhood 0.79; the competing-corridor mean gives 0.98. The input-occupancy baseline associates wall/empty with optimal/nonoptimal in all 305 unique-edit primary pairs. On the correctness-preserving pool all 103 occupancies change while no within-pair checker label does; pooled occupancy AUC is 0.50. The mean edited-cell state change is 45.4 for correctness-changing and 46.0

for correctness-preserving pairs; original-versus-edited classification at this token gives 0.81 and 0.62, and on mean-pooled late states 0.50 [0.48, 0.51] on the preserving pairs.

Symmetric pooled-state control. The main comparison trains a standardized PCA-16 mass-mean probe only on correctness-changing exact-output pairs. Five folds are assigned by base-maze identifier; each fold's fitted score is normalized by its training-score standard deviation and applied without refitting to held-out changing and preserving pairs. Of the 307 primary pairs, 305 have one identifiable edited cell; 290/305 receive a positive optimal-minus-nonoptimal margin. Each of the 103 preserving edits is paired one-to-one with a changing edit of the same wall direction, minimizing standardized differences in distance to the emitted path, distance to the nearest endpoint, row, and column. Median absolute score changes are 0.048 and 0.024 (ratio 1.97; paired one-sided Wilcoxon $p = 1.5 \times 10^{-9}$). Without geometry matching, over all 305 changing and 103 preserving edits, the ratio is 2.11 and an original-versus-edited classifier on the preserving pool scores 0.51.

C.6 Capacity and permutation diagnostics

The original 29-family cache supports a 24-configuration panel: six probe pipelines (PCA-4/16/32 ridge, uncompressed ridge, RBF SVM, ExtraTrees), two pooling rules, and events 4 and 5. No configuration was selected using its reported score. Pooled AUCs range from 0.527 to 0.630, while uncompressed full-grid ridge orders 27/29 pairs at both events, corroborating the distinction between pooled and within-family discrimination (Table 7). Swapping labels within the 104 panel pairs while holding scores fixed gives an upper-tail $p = 10^{-4}$ over 9,999 permutations; a PCA-16 ridge diagnostic with full refitting on the 29 families gives $p = 0.005$ over 199 permutations.

Table 7: Full probe-capacity panel on the original 29 maze families. Path and full denote path-token and full-grid pooling.

Event / pooling	Probe	Pooled AUC	Conditional 95% CI	Pair accuracy
4 / path	ridge 4	0.528	[0.495, 0.577]	0.655
4 / path	ridge 16	0.536	[0.514, 0.589]	0.793
4 / path	ridge 32	0.568	[0.548, 0.616]	0.897
4 / path	ridge full	0.596	[0.565, 0.659]	0.931
4 / path	rbf svm	0.534	[0.526, 0.576]	0.931
4 / path	extra trees	0.555	[0.518, 0.608]	0.793
4 / full	ridge 4	0.541	[0.520, 0.585]	0.793
4 / full	ridge 16	0.591	[0.558, 0.652]	0.862
4 / full	ridge 32	0.598	[0.553, 0.674]	0.897
4 / full	ridge full	0.608	[0.566, 0.688]	0.931
4 / full	rbf svm	0.558	[0.530, 0.609]	0.828
4 / full	extra trees	0.617	[0.565, 0.693]	0.828
5 / path	ridge 4	0.527	[0.495, 0.577]	0.655
5 / path	ridge 16	0.533	[0.510, 0.579]	0.759
5 / path	ridge 32	0.567	[0.546, 0.616]	0.897
5 / path	ridge full	0.591	[0.560, 0.650]	0.862
5 / path	rbf svm	0.529	[0.521, 0.567]	0.897
5 / path	extra trees	0.550	[0.511, 0.606]	0.690
5 / full	ridge 4	0.545	[0.527, 0.586]	0.828
5 / full	ridge 16	0.593	[0.560, 0.652]	0.862
5 / full	ridge 32	0.602	[0.560, 0.674]	0.897
5 / full	ridge full	0.600	[0.564, 0.674]	0.931
5 / full	rbf svm	0.555	[0.529, 0.605]	0.828
5 / full	extra trees	0.630	[0.585, 0.706]	0.897

D Graph Reasoners and Matched-Pair Construction

Each input is an 8-node undirected graph with node 0 as start and node 7 as goal. The 18-dimensional node features concatenate an adjacency row, a one-hot node identifier, and start/goal indicators; a two-layer encoder maps them to width 128. The GRU model performs six rounds of degree-normalized message passing; the GAT model replaces the GRU update with attention over neighbors; the Transformer model applies self-attention over all nodes without an adjacency mask. The readout concatenates the start, goal, and mean node states and predicts up to nine tokens including EOS. Training uses AdamW (learning rate 8×10^{-4}, weight decay 0.01, batch size 64, gradient clipping 1.0) for 2,000 steps on 6,000 sampled graphs. Development seed 20260911 fixes the final message-passing site; confirmatory GRU results use seeds 20260901–20260910 with 120 pairs

each; the five GRU control seeds use 120 pairs each, and the GAT and Transformer controls use 80–97 pairs under the same training protocol.

Table 8: Graph input controls. GRU entries average five independently trained seeds; GAT and Transformer use one seed each with the identical protocol. Residualization and probes are fitted outside the evaluated family. The preserving-edit column reports the median cross-fitted score change and the original-versus-edited AUC on correctness-preserving pairs.

Diagnostic	GRU	GAT	Transformer
Matched AUC (full readout)	1.00	1.00	1.00
Edited-edge input bit AUC	1.00	1.00	1.00
Mean state excluding edited endpoints AUC	0.967	0.996	0.847
Adjacency-residualized state AUC	0.739 [0.699, 0.759]	0.763	0.695
Median $\|\Delta s\|$: changing / preserving edits	1.72 / 0.155	1.87 / 0.16	0.79 / 0.33
Original-vs-edited AUC on preserving pairs	—	0.489	0.507
Mean $\|\|\Delta Z\|\|$: changing / preserving	7.79 / 6.05	—	—

The input bit confirms that the construction permits a trivial separator; the remaining rows test whether the hidden-state result reduces to it. Correctness-preserving edits move the state by a comparable norm but move the truth-trained score 11 (GRU), 12 (GAT), and 2.4 (Transformer) times less; excluding edited endpoints retains strong discrimination; and cross-fitted linear adjacency residualization leaves AUC 0.70–0.76. These controls support a distributed task-dependent signal aligned with the checker label that survives removal of the edited endpoints and the linearly predictable adjacency component. Nonlinear input information remains available in the residual states. Output-changing control pairs are available for the GRU (141/166 edits change the decoded path) and GAT (197/200) models; for the Transformer encoder, off-path edits change the decoded route into an invalid path in all 67 candidates, so its output-changing control is not defined and only its matched and input-control rows are reported.

E Code-Generation Protocol and Full Scores

E.1 Specifications, generation, and matching

All language models use greedy decoding. Each convention template (integer averaging, negative floor division, remainder sign, half-up rounding, indexing base, interval overlap, empty-string splitting, inclusive bounds, even-length median, case-sensitive sorting, first-versus-last deduplication, and 39 further conventions in the expanded set) has several numeric problem families with two prompt variants each; the shared description and worked example jointly specify the desired function. The checker evaluates the exact extracted `def f` program on hidden tests corresponding to the variant's convention. The 50-template construction uses eight numeric families per template (400 families per model); the 14B construction uses 41 templates with six families each (247 families). A matched pair requires identical extracted programs within a family and different checker outcomes. Pool sizes: Qwen2.5-1.5B 253, Qwen2.5-3B 228, Qwen2.5-7B 100 (216 families), Qwen2.5-14B-AWQ 100, SmolLM2-1.7B 99 (11 templates). The 50-template pools replace the initial 11-template Qwen pools as the primary report; the 11-template panel is retained for the supplementary prompt-set comparison below.

Input baseline and output persistence. A classifier on the $(\text{template}, \text{variant})$ indicator separates matched pairs at AUC 0.97–1.00 for every model. The fraction of families in which the model follows the convention change with a different, correct program is 0.06 (1.5B), 0.10 (3B), 0.14 (7B), and 0.23 (14B); the fraction emitting identical code in both variants is 0.77 (1.5B), 0.71 (3B), and 0.47 (14B). These rates describe how the matched pools are selected from each model's generated answers.

Checker depth. Hidden tests were deepened from two to five–seven per family after harvesting. The deepened checker retains 247/253 (1.5B), 223/228 (3B), 58/60 (7B, 11 templates), and 97/100 (14B) pairs and leaves all reported transfer AUCs below 0.70 (LOTO 0.51, 0.52, 0.43, and 0.68 respectively).

Natural-task reference. A separate pool of ordinary specifications without convention flips (200 traces initially, 160 in the expansion; Qwen2.5-1.5B accuracy 0.865 and 0.869) gives natural-task AUC 0.98–0.99 for all models, providing a predictive reference on ordinary prompts. Table 9 reports the convention pool and matched pool side by side so that prompt-distribution differences remain explicit.

Table 9: Code readouts at the final output token. Readout: PCA-16 ridge for the 1.5B, 3B, and SmolLM2 rows, as in the primary report; mass-mean for the 7B and 14B rows, with ridge in parentheses where it differs materially. Convention pool: all paired convention prompts before output matching. LOTO: leave-one-template-out matched AUC. Unique: number of distinct extracted programs among the matched pairs; matched AUC restricted to one pair per unique program is 0.51 (1.5B), 0.52 (3B), 0.35 (7B/11), 0.55 (7B/50), and 0.65 (14B; template-cluster bootstrap interval [0.57, 0.75]).

Model / templ.	Pairs	Pool	Matched	LOTO	Ordered	Unique
Qwen2.5-1.5B / 50	253	0.64	0.52	0.51	129/253	40
Qwen2.5-3B / 50	228	0.67	0.52	0.52	129/228	35
SmolLM2-1.7B / 11	99	0.60	0.57	0.49–0.56	52/99	—
Qwen2.5-7B / 11	60	0.75	0.68 (0.88)	0.38 (0.42)	46/60	9
Qwen2.5-7B / 50	100	—	0.62	0.56	69/100 [0.59, 0.78]	28
Qwen2.5-14B-AWQ / 41	100	0.70	0.70 (0.66)	0.67 (0.65)	81/100 [0.72, 0.88]	24

E.2 Sites, transfer, and controls

Table 10: Prompt-end versus output-end readouts. Family-held-out AUC, with LOTO in parentheses where computed; strict pair ordering where recorded.

Model / templates	Prompt end	Output end	Prompt ordered	Output ordered
Qwen2.5-1.5B / 50	0.53	0.52 (0.51)	169/253	129/253
Qwen2.5-3B / 50	0.52	0.52 (0.52)	131/228	129/228
Qwen2.5-3B / 11	0.80	0.54	78/95	56/95
SmolLM2-1.7B / 11	0.53	0.57	56/99	52/99
Qwen2.5-7B / 11	0.69 (0.43)	0.69 (0.37)	—	46/60
Qwen2.5-14B-AWQ / 41	0.65 (0.56)	0.69 (0.67)	—	81/100

Template transfer. Leave-one-template-out evaluation fits the probe on all templates but one and scores the held-out template's matched pairs. All evaluated language models have low AUC under this test (< 0.70). Tables 9 and 10 retain the individual scores and readout choices. The small-panel results illustrate why the transfer test matters: a within-template ridge AUC of 0.88 falls to 0.42 on an unseen template, and a prompt-end AUC of 0.80 on 11 templates falls to 0.52 after expansion to 50 templates. These observations motivate reporting template-held-out scores alongside family-held-out scores.

Quantization. On the 37 matched families shared by the 4-bit and AWQ versions of Qwen2.5-7B under the same 11 templates and seed, matched AUC is 0.671 and 0.688.

Correctness-preserving convention rewrites. For 40 families per model, the convention description is rewritten without changing the worked example or which behavior is correct; the checker outcome is retained in 94–95% and the emitted program in 85% of cases. Median standardized score change of the matched-fitted readout is 6.24 under correctness-changing edits and 3.64 under wording-only rewrites for 14B (ratio 1.71), and 1.72 versus 1.67 for the 7B 11-template readout.

Interpreting a persistent program. Matching conditions on the generator emitting the same extracted function under specifications with different test outcomes. Pair ordering measures whether the hidden-state score responds in the correct direction within a specification family; template-held-out AUC measures whether that score remains comparable across conventions. The observed paired differences and uniformly low transfer AUCs support the main text's distinction between fixed-answer responsiveness and a broadly reliable correctness score.

Arithmetic reference. On a 14B arithmetic behavior set, Qwen2.5 follows a changed numeric condition in 96% of families, so exact-output correctness flips occur in only 1%; the fixed-answer regime for arithmetic is rare at this scale, which is why the code convention task is the setting for the 14B probe. On a multi-operand arithmetic sample, Qwen2.5-1.5B's unmatched final-output ridge AUC is 0.99, and twelve pairs with the same emitted integer but unrelated prompts yield AUC 1.00, illustrating that identical answers across unrelated tasks still leave substantial input differences; the maze and program constructions specify the task change within a family.

F Patching Design, Timing, and Specificity

F.1 Measurement and relation to the probing experiments

Activation patching replaces a recipient's hidden state with a state from a donor run and resumes computation (Heimersheim & Nanda, 2024). We record exact output restoration and correctness separately:

$$R_{\text{exact}} = \Pr[\hat{Y}_{\text{patched}} = \hat{Y}_{\text{donor}}], \qquad T_{\text{patched}} = c(X_{\text{eval}}, \hat{Y}_{\text{patched}}). \tag{7}$$

Here $\hat{Y}$ is the complete decoded output, including the full grid for structured tasks. The intervention replaces the joint $H{+}L$ state. Its outcome describes the ability of that state to reproduce a donor answer. As defined in Section 6, this output-restoration outcome is distinct from the fitted probe score measured under exact-output matching.

These measurements follow the broader use of causal tracing and interchange interventions to test support for a specified behavior (Vig et al., 2020; Meng et al., 2022; Conmy et al., 2023; Geiger et al., 2021; 2025). The choice of contrast and outcome determines their interpretation (Zhang & Nanda, 2024; Chan et al., 2022). Here, separating restoration from checker success makes it possible to compare donors with different task outcomes.

F.2 Summary across tasks and events

Table 11: **Exact output restoration by donor validity and intervention time.** Exact fidelity of the joint $H{+}L$ intervention by donor category and event. Maze: full 600-donor census on the donor's counterfactual maze; the norm-matched row selects optimal donors whose event-3 displacement (Δ-norm 300–500, mean 379) matches the invalid donors (mean 376). Sudoku: 24 donors per category at events 2–3 and 64 at event 4; Δ-norms 320 (valid) and 335 (invalid). Equal-norm random perturbations change the recipient output in at most one case per category (maze 0/20, 1/20, 1/8; Sudoku 0/24, 0/24). Brackets are exact 95% intervals.

Task	Donor category	Donors	Event 3 (Sudoku: 2–3)	Event 4
Maze	Optimal	259	0.97 [0.94, 0.98]	1.00
Maze	Optimal, norm-matched to invalid	24	1.00 [0.86, 1.00]	1.00
Maze	Valid, nonoptimal	83	1.00 [0.96, 1.00]	1.00
Maze	Invalid	258	**0.55** [0.49, 0.62]	1.00
Sudoku	Valid	24 / 64	1.00 [0.86, 1.00]	1.00
Sudoku	Invalid	24 / 64	**0.54** [0.33, 0.74]	1.00

Table 11 extends the official-maze comparison in Table 3 with uncertainty, the displacement-matched subset, and Sudoku results. At event 4, all 600 maze donors and all 128 Sudoku donors restore their complete outputs, including incorrect outputs. At event 3 in the maze, restoration is 250/259 for optimal donors, 83/83 for valid but nonoptimal donors, and 143/258 for invalid donors. In the released Sudoku-Extreme model, events 2–3 give 24/24 valid and 13/24 invalid donor restorations. These are category associations in the stated populations; displacement and noise controls address specific perturbation explanations.

The early comparison separates valid from invalid donors. In the maze, valid but nonoptimal donors belong to the negative class of the main probe's optimality target, yet restore as reliably as optimal donors. This category comparison describes an association between donor validity and output

restoration in the tested checkpoint. The intervention replaces the joint state; its measured effect is the reproduction of the donor output.

F.3 Donors and checker context

In the directed maze experiment, the recipient follows the base-maze run and receives a joint $H{+}L$ state from its counterfactual donor at the stated event. Exact restoration compares the full patched output with the donor's output. The semantic evaluation uses the donor's counterfactual maze, making the restored output's donor-task validity and optimality explicit. Unrelated-donor exactness is scored against the designated counterfactual target. The full census contains 259 optimal, 83 valid-nonoptimal, and 258 invalid donors. At event 4, directed and reverse exact fidelity are 1.00 in every category and unrelated target restoration is 0. A stratified screen in the same official model of 25 donors per category gives complete-output equality and complement preservation of 1.00 with zero collateral changes; its 75 candidates span 29 distinct base mazes (maze-level one-sided lower bound 0.90).

F.4 Event-resolved restoration and controls

Table 12: Event-3 restoration on the full maze census, with displacement-matched and perturbation controls (fixed seed). Δ-norm is the L_2 distance between donor and recipient event-3 states. The noise column counts recipients whose output changes when the donor state is replaced by a random perturbation of the recipient state with the same norm.

Donor category	Donors	Mean Δ-norm	R_{exact} (donor patch)	Noise
Optimal	259	138	250/259 = 0.965 [0.935, 0.984]	0/20
Optimal, Δ-norm 300–500	24	379	24/24 = 1.00 [0.86, 1.00]	—
Valid, nonoptimal	83	—	83/83 = 1.00 [0.96, 1.00]	1/8
Invalid	258	376	143/258 = 0.554 [0.491, 0.616]	1/20

Table 12 uses the released 30×30 Hard Maze checkpoint identified by revision and checksum in the supplementary provenance table. The 600 donors are counterfactual edits from its test-maze census. All patches replace the joint $H{+}L$ state, and exactness compares the complete patched output with the counterfactual donor output. The norm-matched row limits an explanation based solely on displacement magnitude; it does not identify validity as an independently manipulated cause.

Separate 10×10 timing study. The 40 both-correct block pairs use a separately trained step-800 model on independently generated 10×10 mazes; its revision and checksum are recorded in the supplementary provenance table. Selection requires both clean and counterfactual predictions to equal their target answers. Exact restoration is 0/40 at events 1–3 and 40/40 at events 4–5, with zero unrelated-target restoration. Both studies use the same event-index convention in the final outer step (outer index 15 of 16), joint-state replacement, and full-output equality; the checkpoint, task scale, and sampling conditions differ. They therefore do not estimate the same event-3 restoration rate. The existing records do not isolate which of these differences explains the change in timing, and this smaller-model study is not corroboration of the official model's event-3 validity association.

Official-model module and spatial support (60 donors, event 4): L alone 43/60 [0.59, 0.83]; H alone 0/60 [0.00, 0.06]; path tokens alone 0/60; non-path tokens alone 26/60 [0.31, 0.57].

Separate locally trained model: output compatibility. Experiments with the 10×10 step-800 checkpoint test compatibility of H and L: same-donor $H{+}L$ gives 1.00 on 40 pairs, conflicting modules from different families give zero, output-equivalent temporal donors ($n = 80$) can be crossed in either direction with fidelity 1.00, and distinct-history recovered donors succeed on all 38 eligible pairs. Patching at events 4–5, immediately before the logits, or at the logits gives fidelity 1.00 on a 40-pair control with zero hard-negative restoration.

Sudoku. Fresh puzzles and a fresh seed provide donors for the released Sudoku-Extreme HRM. Donor validity is checked on the donor puzzle; saved patched-validity rates use the recipient puzzle. In the late sample this recipient-task rate is zero in both donor categories despite complete restoration. The donor-category labels therefore cannot be substituted for donor success on the recipient

task in Equation 6. At event 4 the intervention restores 64/64 valid and 64/64 invalid grids (one-sided lower bound 0.954 per category). At events 2 and 3 (24 donors per category), valid donors restore 24/24 and invalid donors 13/24 (0.54 [0.33, 0.74]); mean Δ-norms are 320 and 335; equal-norm noise changes the output in 0/24 cases in either category. Patched outputs never match the donor grid under noise (0/24).

F.5 Additional localization and donor controls

A learned direction in activation-change space restores exact outputs at rate 0.80 on held-out families, while a norm-matched random orthogonal direction restores none. A spatial locator predicts continuous RMS localization scores on a fit/test family split; for the smallest spatial budget covering 90% of the oracle score, learned values are 0.842 and 0.628 against oracle 0.841 and 0.646, with random references near 17. Hard-negative donors are matched within edit type using equally weighted gaps in changed-cell count, route length, path overlap, edit geometry, and output overlap; the unrelated-donor matcher uses robust standardized Euclidean distance over route lengths, changed-cell count, edit distances, and activation-change RMS.

F.6 Graph output-restoration replication

The causal replication uses the first three confirmatory GRU seeds. The 900 candidate single-edge comparisons yield 842 eligible directed interventions. Directed patches restore the designated counterfactual target in 706/842 cases (.838), with 722/842 optimal patched outputs (.857); unrelated donors restore that target in 57/842 (.068). Different-input, same-output donors give 680/840 target restorations (.810) and 704/840 optimal outputs (.838). Different-token optimal donors give 36/565 target restorations (.064) and 503/565 optimal outputs (.890). In the last comparison, exactness is measured against the original designated counterfactual target, not the replacement donor's different path. The supplementary evidence index provides seed-level numerators and denominators; the reported ratios pool the three seeds without averaging ratios with different denominators.

G Evidence Organization and Reproduction

The anonymous supplementary package contains a claim-to-record index, a SHA-256 manifest, processed result records, deterministic checkers, and the CPU analysis and figure-generation code. Table 13 summarizes the package by evidence family. The accompanying evidence index gives the exact record names and maps every empirical table and figure to its inputs and verification command; keeping those implementation details in the package lets this appendix emphasize the scientific comparisons.

Table 13: Organization of the supplied evidence. Exact filenames, checksums, and commands appear in the supplementary evidence index.

Evidence family	Material supplied	Independent checks
Maze readouts and matched controls	Pairing census, held-out scores, edit-direction strata, state-change analyses, and local/input controls.	Reconstruct pooled AUC, strict paired wins, selection counts, and state-change summaries.
Maze and Sudoku interventions	Donor metadata, patch sites, restoration and checker outcomes, displacement-matched and noise controls.	Reconstruct rates, targets, denominators, and confidence intervals.
Graph replication	Seed-level readouts, architecture and spatial controls, and directed, unrelated, same-output, and different-token donors.	Preserve seed denominators before aggregation and check each donor comparison.
Generated-code studies	Matched programs, checker outcomes, prompt-template splits, site comparisons, checker-depth, quantization, and rewrite controls.	Reproduce held-out-template summaries and pair-level results.
Fixed-candidate language-model studies	Frozen and refitted predictions, group/split identities, input and compatibility controls, ranking counts, and uncertainty calculations for all six models.	Verify candidate identity, AUC, paired wins, transfer intervals, and ranking identities.
Utility and exploratory analyses	Candidate selection, risk–coverage, retrieval, and prefill-intervention records.	Rebuild the reported tables, curves, and exact or paired tests.

Verification. The package README separates CPU-only verification from optional model inference. CPU entry points check split integrity, candidate identity, numerical summaries, figure inputs, and all supplied file hashes from the frozen records. Public or locally trained checkpoints that are

too large for upload are identified by revision and checksum; they are not required to verify the reported manuscript values.

H Language-Model Candidate Validation and Frozen Transfer

H.1 Populations, contracts, and checkers

The new code experiments use 917 source groups: 132 retained HumanEval problems and 785 retained MBPP problems. These are contract-transformed subsets, not scores on the original benchmarks. A deterministic construction changes the final returned value through Boolean negation, numeric negation, or outer-sequence reversal, with transformed reference behavior and executable tests. The base and transformed contracts define the checker outcome of the same reference candidate. The code groups split into 555 training, 170 development, and 192 test groups. Generation has four prompt variants per group (3,668 traces); validation has two members per group (1,834 traces). No test candidate program hash appears among training candidates in V.

The tool experiments use 715 synthetic groups covering ledger, inventory, workflow, and composite operations. Contracts specify allowed operations, arguments, and state-dependent constraints. The checker parses the tool-call object and evaluates it under the corresponding contract; matching compares canonical object hashes. The source groups split into 436 training, 141 development, and 138 test groups. Generation has four variants per group (2,860 traces); validation has two (1,430 traces). Each V group has one correct and one incorrect member by construction. Group identities are disjoint across splits, although recurring call objects can occur in different tool problems. The unseen-output analysis below removes this overlap.

G and V should not be interpreted as interchangeable samples. G uses greedy independent generation. V replays a supplied candidate to obtain its likelihood and hidden states under each prompt. No generated answer is substituted when a fixed candidate fails its checker. Neither V label balance nor its perfect pairing rate estimates the frequency with which the model would independently generate both members. All six G test sets contain no eligible exact-output, correctness-flipping pair.

H.2 Models and representation extraction

The six models are Qwen2.5-Coder-7B/14B/32B-Instruct and Qwen3-8B/14B/32B. Inference uses BF16, greedy decoding, a prompt limit of 12,288 tokens, and a generation limit of 768 tokens. Qwen3 uses `enable_thinking=false`. The run seed is 20260917. We retain prompt-end and answer-end representations from early, middle, and late decoder blocks; the main analysis fixes the middle block, selected by the saved half-depth rule. The answer site is the last non-EOS candidate token, and the prompt site is the last prompt token. Decoder-block residual states are captured before the final normalization; the final-layer state returned by a model API is not substituted for this hook. The same extraction and model environment are used for each completed G/V comparison. Resolved model metadata and full configurations are preserved with the run records.

All new readouts use training-only standardization followed by logistic regression ($C = 1$, maximum 3,000 iterations, seed 17), without PCA. The probe settings, extraction site, and test split are held fixed across model sizes. Prompt controls use word unigrams and bigrams, sublinear TF–IDF, a 20,000-feature cap, and logistic regression. The length/likelihood control uses input-token count, output-token count, and mean candidate-token log-probability, with the same standardized logistic readout. These controls are fitted on training groups, not on the held-out labels.

The original code convention study in Appendix E uses different models, readout sites, prompts, and in some cases quantization. Its values are not a matched model-scaling comparison with the BF16 study. The new near-ceiling V scores are also insufficient to resolve a scaling trend.

H.3 Frozen transfer and uncertainty

The G probe's standardization and fitted parameters remain unchanged when scoring V. We do not reverse its sign, select a new threshold, or use V labels to adjust the score. The V-trained comparison uses a separate fit on V training groups and the identical held-out V examples. Because

every V training example belongs to a matched pair, "ordinary" and "matched" fitting in the analysis software use the same training rows in V; these are one result, not independent replications.

Table 14: Frozen generated-answer probes on fixed candidates. Intervals resample held-out source groups 1,000 times, retaining the already fitted probe. They quantify test-population uncertainty, not variation from probe refitting or model training. Strict counts exclude ties; the pair credit Q assigns them half credit. Code 32B has one tie.

Model	V AUC [95% interval]	Strict V pairs	Pair credit Q [95% interval]
Code 7B	.672 [.646, .703]	179/192	.932 [.896, .964]
Code 14B	.454 [.423, .483]	71/192	.370 [.302, .438]
Code 32B	.547 [.530, .567]	129/192	.674 [.604, .740]
Tool 8B	.854 [.826, .886]	138/138	1.000 [1.000, 1.000]
Tool 14B	.201 [.155, .240]	19/138	.138 [.080, .196]
Tool 32B	.681 [.660, .711]	131/138	.949 [.913, .978]

For the V-fitted answer-state probes, family-bootstrap AUC intervals are [.994, 1.000] for code 7B, [1.000, 1.000] for code 14B, and [.996, 1.000] for code 32B. All tool intervals are [1.000, 1.000]. A degenerate bootstrap interval occurs when all resampled groups remain perfectly separated; it does not imply zero population uncertainty. The strict counts are 192/192 for code and 138/138 for tool use. Under an independent-group binomial model, corresponding two-sided 95% Clopper–Pearson lower bounds are .981 and .974. These conditional intervals do not account for shared templates or other dependence across source problems.

H.4 Candidate provenance, format, and error-type shifts

The transfer comparison includes a provenance and format shift. In code 7B, 2,448 of 3,668 G traces end with a code fence, whereas no V candidate has that ending. This is one concrete reason to interpret G-to-V as a test of score reuse under the stated validation shift, rather than a causal estimate of removing answer variation alone.

The checker records also show an error-type shift. Across all train, development, and test rows of tool 14B G, 2,277 of 2,860 traces succeed. Of the 583 failures, 496 are disallowed-key errors, 52 are parsing failures, four violate a transfer precondition, and 31 fail the goal check. The 1,430 tool 14B V rows contain no disallowed-key or parsing errors; their 715 failures are goal-check failures. These are whole-run descriptive counts, not test-set error-stratified probe performance. They identify a concrete alternative explanation for G-to-V behavior without establishing which cues the probe uses. The supplementary checker audit contains the underlying row-level categories; generated code has a less specific `no_completion_marker` failure category, so the records do not provide an equally fine code error taxonomy.

Figure 2 in the main text shows the complete frozen-versus-refitted curves. No error-type-matched or format-aligned state extraction is available in these comparisons, so their individual contributions to transfer remain unresolved.

H.5 Within-family and cross-family contributions to AUC

This calculation enumerates comparisons among already saved test predictions. It changes neither the examples nor the scores and involves no fitting or model inference. Each of the m candidate families contributes one positive score s_i^+ and one negative score s_i^-. With $q(a,b) = \mathbb{1}[a > b] + \frac{1}{2}\mathbb{1}[a = b]$, pooled AUC averages $q(s_i^+, s_j^-)$ over all m^2 positive–negative comparisons. Separating diagonal and off-diagonal terms gives

$$\mathrm{AUC} = \frac{1}{m} Q + \frac{m-1}{m} \mathrm{AUC}_{\mathrm{cross}}, \qquad \mathrm{AUC}_{\mathrm{cross}} = \frac{1}{m(m-1)} \sum_{i \neq j} q(s_i^+, s_j^-). \tag{8}$$

Here $Q = m^{-1} \sum_i q(s_i^+, s_i^-)$ is the tie-adjusted pair credit from Equation 4. Code 32B has one probability-saturation tie: both scores round to 1 although their logits are 42.244 and 38.360. Its 129 strict wins give $Q = 129.5/192 = 0.6745$. The unchanged decision function gives 130 strict wins and AUC 0.54652, compared with probability-based AUC 0.54651. The tables retain the

original probability protocol. All other within-family comparisons and all cross-family comparisons are tie-free.

Table 15: Exact accounting of the saved frozen scores. A reversal means an incorrect member from one family scores above a correct member from another. The table decomposes the existing evaluation and is not an additional experimental replication.

Model	Strict own wins	Cross-family reversals	Cross-family AUC	Pooled AUC
Code 7B	179/192	12,069/36,672 (32.9%)	.671	.672
Code 14B	71/192	20,008/36,672 (54.6%)	.454	.454
Code 32B	129/192	16,655/36,672 (45.4%)	.546	.547
Tool 8B	138/138	2,774/18,906 (14.7%)	.853	.854
Tool 14B	19/138	15,098/18,906 (79.9%)	.201	.201
Tool 32B	131/138	6,066/18,906 (32.1%)	.679	.681

Within-family terms carry weights of $1/192 = 0.52\%$ for code and $1/138 = 0.72\%$ for tool use; over 99% of pooled comparisons are across families. Shared observations make these comparisons dependent, so their count is not an independent-trial sample size. The supplementary package provides the saved scores, pair indices, exact counts, and an executable check of this identity for every reported AUC.

H.6 Compatibility and unseen-output controls

The code compatibility rule uses two observed features: whether the prompt contains the changed-final-output-contract instruction, and whether the candidate contains both the `_contract_transform` function and the `_base_namespace` wrapper. It predicts success when these indicators agree. The rule uses no source identifier, saved checker label, or test execution. It was identified after inspecting the construction and reaches accuracy and AUC 1.000 on training, development, and test pairs for all three code sizes. This is a post-hoc demonstration of a simple solution to this dataset, not evidence that the language model implements that rule internally.

Table 16: Candidate-validation results after removing test answer objects seen in probe training. Test groups remain disjoint from training throughout. The tool filter retains 93 families and 88 distinct canonical objects; each family still has both members. All displayed pair counts are strict wins with zero ties.

Model	Full test families	Unseen-output families	AUC	Correct pairs
Code 7B	192	192	.998	192/192
Code 14B	192	192	1.000	192/192
Code 32B	192	192	.999	192/192
Tool 8B	138	93	1.000	93/93
Tool 14B	138	93	1.000	93/93
Tool 32B	138	93	1.000	93/93

The full tool V test set has 131 distinct canonical objects, 43 of which overlap training objects. The filter in Table 16 removes every affected family and evaluates existing predictions without refitting. It tests answer-object overlap, not unseen contract templates. For the tool length/likelihood baseline, strict pair accuracy is 98.55%, 99.28%, and 99.28% at 8B, 14B, and 32B, respectively. This corroborates the near-perfect pooled scores from that control in Table 1A.

H.7 Available runs and evidence provenance

All six code/tool G jobs and six V jobs are complete: each code G/V run has 3,668/1,834 traces and each tool G/V run 2,860/1,430. Four prefill-patch studies and two 32B selection studies are also complete. The final snapshot contains 27 finished jobs including six pilots and three RAG runs. The analysis counts only top-level completed records, excluding preserved historical shards from sample totals.

The records retain group and split identities, exact candidate hashes, checker outcomes, token counts, likelihoods, model configuration, hidden states, and patch metadata. All 35,707 files in the final snapshot match their manifest; the 25,686 previously received traces and their states are unchanged. CPU analysis refits probes from saved states, checks split integrity, and recomputes tool/RAG verdicts; code uses the saved sandbox verdicts. The manuscript consistently uses local CPU fits. The supplementary package includes the completed-run summaries, predictions, ranking counts, unseen-output controls, analysis provenance, and figure builders; no model inference is needed to regenerate the reported analyses.

I Exploratory Language-Model Comparisons

I.1 Retrieval-context ordering

An additional retrieval-augmented answering set contains 400 traces per Qwen3 size (8B/14B/32B), formed from 200 question/context-order pairs and 199 source groups. The variants change context order rather than reversing the checker outcome of a fixed candidate. We therefore report them as an ordinary answering comparison, not as another exact-output correctness test. Success requires both the audited answer and the evidence identifiers.

Table 17: Exploratory retrieval answering on 60 held-out traces per model. Correctness requires both answer and citation checks. AUC is undefined for 14B because its test labels contain no successes. Sparse positives limit the precision and scope of the two defined AUCs.

Model	Successes	Answer passes	Citation passes	Answer AUC	Prompt AUC
Qwen3 8B	6/60	23/60	13/60	.883	.302
Qwen3 14B	0/60	19/60	11/60	—	—
Qwen3 32B	5/60	29/60	10/60	.942	.749

Across all splits, strict successes are 26/400, 29/400, and 60/400. Prompt TF–IDF test AUC is .448 at 8B and .796 at 32B; length/likelihood gives .395 and .411. The answer-state discrimination in Table 17 concerns this strict combined target and should not be conflated with answer correctness alone or generalized from the small positive test classes.

I.2 Prefill state replacement

The prefill study includes code 7B/32B and tool 8B/32B, with 144 patch jobs per model over 24 recipient groups, two donor categories (same label or unrelated), and three layers. Each job includes self, noise, and donor conditions. No eligible same-answer donor cohort is available in any model. The intervention replaces the last-prompt-token residual state during prefill, whereas the main probe reads the answer-end state.

Table 18: Prefill state replacement. Success counts are over intervention records, which reuse recipient groups and are not independent task trials. Self-patching reproduces baseline successes. Exact donor-output restoration is zero in all four models.

Model	Baseline success	Self patch	Donor patch	Noise	Exact donor restoration
Code 7B	78/144	78/144	77/144	60/144	0/144
Code 32B	72/144	72/144	72/144	73/144	0/144
Tool 8B	96/144	96/144	96/144	29/144	0/144
Tool 32B	132/144	132/144	132/144	96/144	0/144

Donor patches leave all 144 tool outputs unchanged at both scales. Code 7B retains the raw output in 122/144 records; code 32B retains the self-patched token sequence in 86/144. For code 32B, the late unrelated-donor condition repairs one answer and breaks one; other donor conditions preserve checker labels. Noise uses the full donor-state norm, rather than donor–recipient displacement. These results characterize the tested prefill site; answer-end probe-direction interventions and same-answer donor contrasts are separate causal questions.

I.3 CANDIDATE SELECTION AND SELECTIVE FILTERING

Cohorts and frozen scores. We evaluate code and tool 32B on 155 and 148 task groups disjoint from all formal G/V groups. Code contains 20 HumanEval and 135 MBPP tasks with their original contracts; tool use contains 148 original synthetic ExecutableStatePlans-v2 tasks. Neither cohort is filtered by generated outcomes. Each task has four samples at temperature 0.6, top-p 0.95, and a 768-token answer budget. The G-trained middle-block answer-end probe and its preprocessing stay frozen. Selection compares candidates within a task; selective filtering ranks the predetermined first candidate across tasks. The latter uses test-set score quantiles to report coverage.

Table 19: **Selection from the same four-candidate pool.** Entries are successful tasks; every rule sees the same generated candidates. Length/LP is the original three-feature G-trained logistic control. Oracle uses checker labels and is an upper bound.

Task	Tasks	First	Answer probe	Mean LP	Length/LP	Oracle
Code 32B	155	111	109	113	114	117
Tool 32B	148	109	113	112	114	116

Within-task selection. The answer probe changes code success from 71.6% to 70.3%, repairing two tasks and breaking four. Tool success changes from 73.6% to 76.4%, repairing five and breaking one. The paired gains are -1.29 percentage points (95% group-bootstrap interval $[-4.52, 1.94]$) and $+2.70$ points ($[0.00, 6.08]$); two-sided exact paired tests give $p = 0.688$ and 0.219. Only 11/155 code and 18/148 tool candidate pools contain both correct and incorrect answers, limiting available selection gains. All four tool samples have the same extracted object in 117/148 tasks. The count comparisons retain failed and truncated candidates rather than replacing them.

Table 20: **Ranking the first answer across tasks.** At 75% coverage, code retains 116/155 tasks and tool retains 111/148. Parentheses give error counts. Controls use G training groups and the original fitting settings; no utility labels are used for fitting.

	Code 32B		Tool 32B	
Score	AUC	Risk at 75%	AUC	Risk at 75%
Answer-end probe	.650	24.1% (28/116)	.975	7.2% (8/111)
Mean log-probability	.573	26.7% (31/116)	.726	24.3% (27/111)
Length/LP	.661	21.6% (25/116)	.829	14.4% (16/111)
Input TF–IDF	.609	24.1% (28/116)	.954	5.4% (6/111)
Prompt-end probe	.547	25.0% (29/116)	.985	2.7% (3/111)

Cross-task screening. Tool answer-state AUC is .975 with a 95% task-bootstrap interval of [.950, .991], versus .726 [.645, .804] for mean likelihood. At 75% coverage, the risk difference is -17.12 points with interval $[-25.23, -9.91]$, resampling tasks and reranking at each of 3,000 draws. At 50% coverage, the probe retains 74 answers with no observed errors; the descriptive two-sided binomial interval has upper endpoint 4.86%. Input TF–IDF and prompt-end states also predict these outcomes well (Table 20), locating substantial predictive structure before generation. Code answer-state AUC is .650 [.553, .740], and its filtering differences from likelihood have intervals spanning zero.

Analysis provenance. Best-of-four outcomes and first-answer curves are recomputed from saved candidate states; the added input controls use the existing TF–IDF and standardized logistic settings on G/train. The intervals condition on the fitted model and probe. First-answer AUC and control comparisons are post-hoc analyses; the reported coverage grid is inherited from the saved utility evaluation. Input-only scores tie across a task's candidates, so their natural use here is cross-task screening. We interpret prompt-state scores for first-answer filtering, avoiding candidate selection driven by small numerical differences in full-sequence replay. The 10,000-draw paired bootstrap and exact tests use independent task groups, not individual candidates. CPU refitting reproduces all selection counts; at the 38-task code coverage point it yields six errors versus five in the received fit. The supplementary audit retains both versions and their numerical difference; all paper values use the local fit.

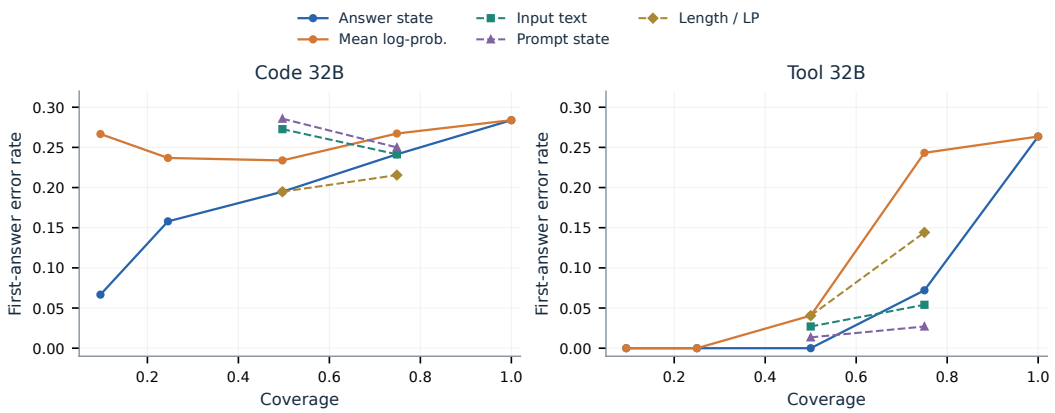


Figure 5: **Selective filtering and its input controls.** All methods rank the same first answers. Solid curves show the original coverage grid; dashed controls are evaluated at 50% and 75% coverage. The answer-state probe improves tool screening over raw likelihood, while input-only and prompt-state controls are also strong. These are empirical risk–coverage comparisons, without a calibrated deployment threshold or risk guarantee.

J Finite-Sample Identities and Proofs

The following statements formalize the measurement distinctions used in the main text. They are elementary finite-sample identities and bounds, not a generalization theorem for correctness detection. Assume a deterministic checker and m families, each containing one positive and one negative example with the same answer. Scores are evaluated after fitting; no independence assumption is needed for the algebra. Statistical uncertainty and deployment interpretation require additional assumptions discussed in Appendix M.

J.1 Answer-only discrimination under exact matching

Proof of Proposition 1. Write $u_i = g(Y_i^+) = g(Y_i^-)$ and use $q(a,b) = \mathbf{1}[a > b] + \frac{1}{2}\mathbf{1}[a = b]$. Every pair ties, so strict pair accuracy is zero and tie-adjusted pair accuracy is $1/2$. Moreover,

$$\widehat{\mathrm{AUC}}(g) = \frac{1}{m^2}\sum_{i,j} q(u_i, u_j) \tag{9}$$

$$= \frac{1}{m^2}\left(\frac{m}{2} + \sum_{i<j}\{q(u_i,u_j) + q(u_j,u_i)\}\right) = \frac{1}{2}, \tag{10}$$

because $q(a,b) + q(b,a) = 1$. This conclusion allows repeated answers across families. It is stronger than saying an answer-only classifier fails within a pair: on this balanced construction its pooled AUC is also exactly chance. The result does not apply to unmatched populations, unequal class weights within families, or scores that additionally use the task. In particular, $h(X,Y)$ may detect a prompt–candidate interaction without performing general verification.

J.2 Paired versus cross-family ranking

Proof of Proposition 2. Define

$$Q = \frac{1}{m}\sum_i q(s_i^+, s_i^-), \qquad \widehat{\mathrm{AUC}}_{\mathrm{cross}} = \frac{1}{m(m-1)}\sum_{i\neq j} q(s_i^+, s_j^-). \tag{11}$$

The m diagonal terms contribute Q/m to the pooled average over m^2 comparisons; the other $m(m-1)$ terms contribute $(m-1)\widehat{\mathrm{AUC}}_{\mathrm{cross}}/m$. This proves Equation 4. The coefficients count comparisons, not independent observations. Each family participates in many terms, so treating the $m(m-1)$ comparisons as independent Bernoulli trials would understate uncertainty.

Tight bound with every pair strictly ordered. If $s_i^+ > s_i^-$ for every i, then

$$\frac{m+1}{2m} \leq \widehat{\mathrm{AUC}} \leq 1. \tag{12}$$

For any distinct i, j, at least one of $s_i^+ > s_j^-$ and $s_j^+ > s_i^-$ must hold: otherwise adding their negations contradicts $(s_i^+ - s_i^-) + (s_j^+ - s_j^-) > 0$. Thus the two cross terms together contribute at least one, while every diagonal contributes one. The total is at least $m + \binom{m}{2}$, yielding the lower bound. To attain it, choose $s_i^- = 3i$ and $s_i^+ = 3i + 1$: within each family the positive wins, but between two families only the higher-offset positive beats the other negative. Hence $\widehat{\mathrm{AUC}} = (m+1)/(2m)$, despite perfect pair accuracy. Figure 6 shows this construction. Conversely, putting every positive above every negative attains the upper bound.

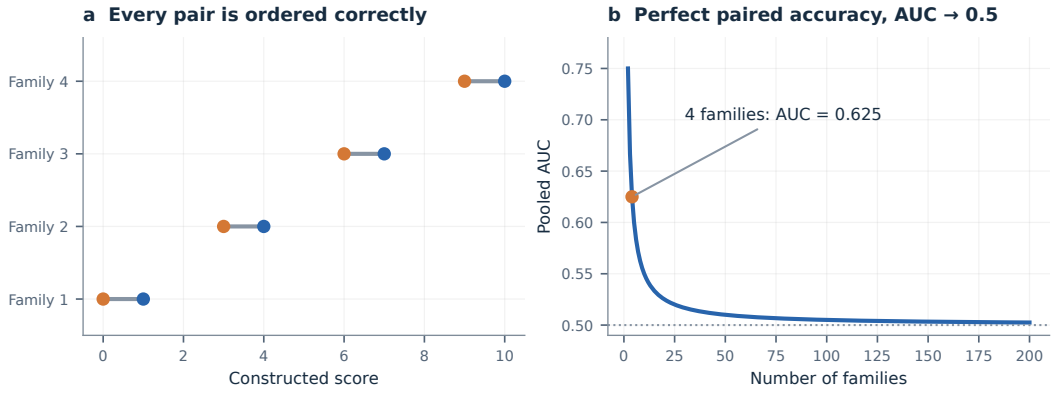


Figure 6: **A constructive separation between two ranking requirements.** Left: four synthetic score pairs, with negative scores in orange and positive scores in blue. Every pair is ordered correctly; their pooled AUC is $10/16 = 0.625$. Right: extending the construction makes AUC approach chance as the family count grows. This is an algebraic illustration, not model data or an additional experiment.

The midpoint identity in Equation 3 explains the role of between-family offsets. A strictly increasing transformation preserves every q comparison and therefore cannot repair a ranking discrepancy. Subtracting a family-specific offset can change cross-family rankings but requires a reference that an independently scored answer may not provide. Neither operation establishes candidate-selection utility for a fixed task.

J.3 Restoration and checker success share a coupling bound

Let $E = \{\hat{Y}_{\mathrm{patched}} = \hat{Y}_{\mathrm{donor}}\}$, $C = c(X_{\mathrm{eval}}, \hat{Y}_{\mathrm{patched}})$, and $D = c(X_{\mathrm{eval}}, \hat{Y}_{\mathrm{donor}})$. Both checker evaluations must use the same X_{eval} for each intervention. Since E implies $C = D$,

$$|\mathbb{E}C - \mathbb{E}D| \leq \mathbb{E}|C - D| \leq \Pr(E^c) = 1 - R_{\mathrm{exact}}. \tag{13}$$

This proves Equation 6. For $0 < R_{\mathrm{exact}} < 1$, the exact decomposition is

$$A_{\mathrm{patch}} = R_{\mathrm{exact}} \Pr(D = 1 \mid E) + (1 - R_{\mathrm{exact}}) \Pr(C = 1 \mid E^c). \tag{14}$$

No independence of donor correctness and restoration is assumed. At $R_{\mathrm{exact}} = 1$, patched success equals donor success. The official maze census illustrates this endpoint: all 600 outputs restore, but only $259/600$ are optimal on the donor task. If equality is defined against a different designated target, or the checker uses the recipient task, the stated coupling must be reformulated. Alternative-optimal graph donors are precisely why the equality target and checker context are reported separately.

K Evaluation Protocol, Prompt Templates, and Worked Cases

K.1 A reproducible sequence of comparisons

The comparison is designed to separate answer identity, task dependence, and score reuse. The following protocol states what is fitted and what remains fixed; it does not introduce a new training algorithm.

1. **Define the object and checker.** Specify the exact answer representation Y, the evaluation task X, and $c(X, Y)$. Use full maze path marks, graph token sequences, extracted code, or canonical tool objects as appropriate.
2. **Construct pairs and retain their provenance.** For generation matching, run each variant independently and retain equal answers with opposite labels, reporting the selection census. For V, replay one reference candidate under both contracts and verify the label flip.
3. **Split before fitting.** Exclude test source families from both G and V training. Fit preprocessing only on training families. Candidate-object filtering is a separate diagnostic and does not imply unseen transformation rules.
4. **Compare scores on the same states.** Fit the G probe once and freeze its preprocessing, coefficients, and orientation. Separately fit a V probe on V training families. Evaluate both on identical V test states without sign reversal or test-based threshold selection.
5. **Report distinct outcomes.** Compute pooled AUC and strict paired wins; report tie handling. Evaluate input and compatibility controls. For interventions, state the donor population, checkpoint, site, equality target, and checker input, then report restoration and checker success separately.

The LLM model families, BF16 settings, extraction positions, generation budgets, seeds, and logistic-regression settings are specified in Appendix H. Maze and graph training and state sites are specified in Appendices B–D. The supplementary evidence index identifies the saved inputs and deterministic generation command for each empirical panel in Appendix L.

Recorded extraction environment. The six V run manifests record Python 3.11.15, PyTorch 2.7.0+cu128, Transformers 4.56.2, CUDA 12.8, BF16, and SDPA attention on NVIDIA RTX PRO 5000 72GB Blackwell GPUs. The 32B models use two visible GPUs; the smaller models use one. The zero-based middle-block indices are 14/24/32 for Code 7B/14B/32B and 18/20/32 for Tool 8B/14B/32B. These are extraction settings, not a hardware-normalized efficiency comparison; exact model revisions and device maps appear in the supplementary provenance tables.

K.2 Code prompt template and a held-out checker example

Each code prompt uses the following structure. Bracketed fields are template placeholders; the contract text is from the saved construction. The changed contract applies to the final returned value, not to arguments or side effects.

> **Shared instruction:** Implement the requested task. Return one complete Python code block, including imports.
> **Original contract:** FINAL OUTPUT CONTRACT: Return the original result exactly as described in ORIGINAL SPEC; do not apply any output transformation.
> **Changed contract:** FINAL OUTPUT CONTRACT: The final returned value must follow this changed output contract: negate booleans; negate numeric scalars; reverse the order of strings, lists or tuples (only the outer sequence). Compute the original task result first, then apply exactly this transformation. Input arguments and side effects remain as in the original task. Examples in the ORIGINAL SPEC below describe the original result before this transformation.
> **Task slot:** ORIGINAL SPEC: [source problem and required signature].

For the held-out source `mbpp-62`, the specification asks for the smallest number in a list, with signature `smallest_num(xs)`. Both V records replay exactly this candidate:

```
def smallest_num(xs):
  return min(xs)
```

Its outputs on the three saved test inputs are $1, 1, 45$. The original contract accepts them; the changed numeric contract requires $-1, -1, -45$. The saved checker implements the change by applying the involutive transformation to the candidate output before comparing with the original expected value. Thus the same candidate passes the original tests and fails the changed tests. This example was selected for brevity from the held-out dataset, not for a particular probe score, and is a reference candidate rather than a sampled model answer.



Figure 7: **Two executable fixed-candidate cases.** Condensed displays of real held-out records: MBPP-62 and inventory-0017. The candidate is identical within each pair while the requested result changes. The displayed outcomes were rechecked with the saved deterministic tests and tool checker. These illustrate construction and checker semantics, not case-level evidence of the probe's mechanism.

K.3 Tool prompt template and a held-out state transition

The tool template specifies all transition rules and a finite action budget. The common instruction is: "Plan actions reaching every goal. Calls execute sequentially; stop after at most 6 actions. Return ONLY a JSON list of {`tool`:`name`, `args`:{...}} objects." It then supplies the rules, initial state, tool argument types, and final-state goals. No model judge supplies the label.

The held-out `tool-inventory-0017` pair uses the rule: reserve a positive integer quantity by subtracting from stock and adding to reserved, provided stock is sufficient. The initial stock is `itemA`=17 and `itemB`=16, with no reserved items. Both prompts use the same `reserve(item: str, quantity: int)` tool and both replay:

```
[{"tool": "reserve", "args": {"item": "itemA",
"quantity": 5}},
 {"tool": "reserve", "args": {"item": "itemB",
"quantity": 9}}]
```

The calls leave stock $(12, 7)$ and reserved quantities $(5, 9)$. The original goals require reserved quantities $(5, 9)$ and pass. The edited goals require $(6, 9)$ and fail, while parsing and transition preconditions still succeed. The supplementary worked-case record preserves the full rows, byte-identical candidate strings, split labels, source hashes, and checker results. Figure 7 condenses the two cases without replacing their original records.

K.4 How examples relate to the claim

The code case exposes a recognizable contract transformation, and the tool case exposes a simple final-state mismatch. They explain why high V readout performance can coexist with strong non-hidden-state controls. They do not establish that the probe executes the checker or uses the displayed rule. The generation-matched maze and graph populations answer a different selection question because their equal outputs arise from independent model runs. The full prompt templates make this difference inspectable instead of treating every fixed-answer comparison as the same experimental setting.

L Additional Visual Diagnostics from Saved Results

These panels display the predictions and counts underlying the tables. The supplementary package provides the plotted values and SHA-256 provenance for both the six-model transfer views and the earlier control, case, and structured-task panels. Multiple views of the same observations serve explanation rather than independent replication.

L.1 Readout controls across all six candidate models

Figure 8 makes the feature comparison in Table 1A visible across models. The answer-end probe is near ceiling, while prompt-only and prompt-end scores are substantially weaker. Tool likelihood is prompt-conditioned and almost matches the answer-state readout. It therefore provides a competitive simpler score, not an answer-only explanation. Code has a different control: the prompt–program compatibility rule achieves AUC 1.000 even though the separate prompt and length/likelihood baselines do not.

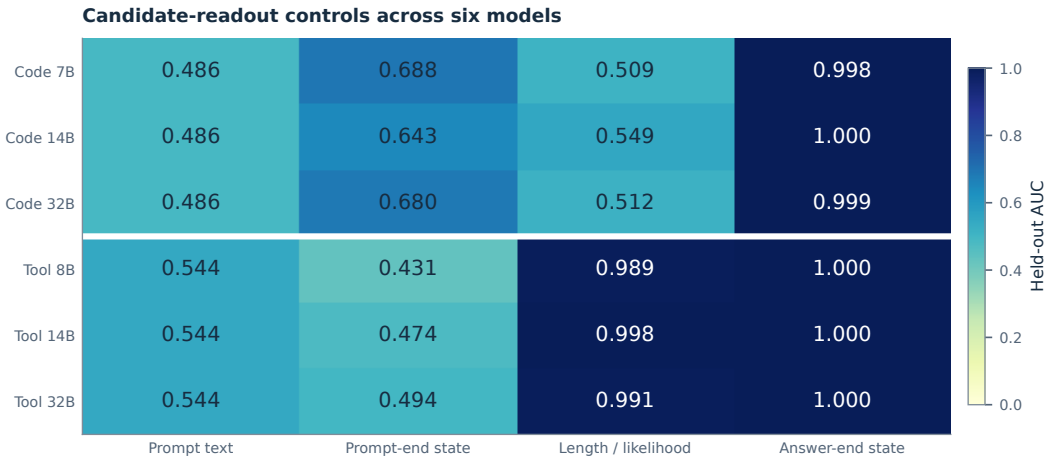


Figure 8: **Six-model control matrix.** Held-out V AUCs, with a separately fitted probe for each feature and model. All values reproduce Table 1A. Prompt text is TF–IDF; length/likelihood includes input length, candidate length, and mean conditional token log-probability. Near-ceiling answer-state performance measures readability of the constructed labels.

L.2 Paired-score distributions across six models

Figure 9 complements the ROC curves in Figure 2. Negative differences reverse the desired within-family ranking. Code 14B and tool 14B place most pairs in this region; code 7B, tool 8B, and tool 32B mostly respond in the desired direction. Code 32B has 129 strict wins and one probability tie. These distributions retain the original raw probe probabilities, with no sign adjustment.

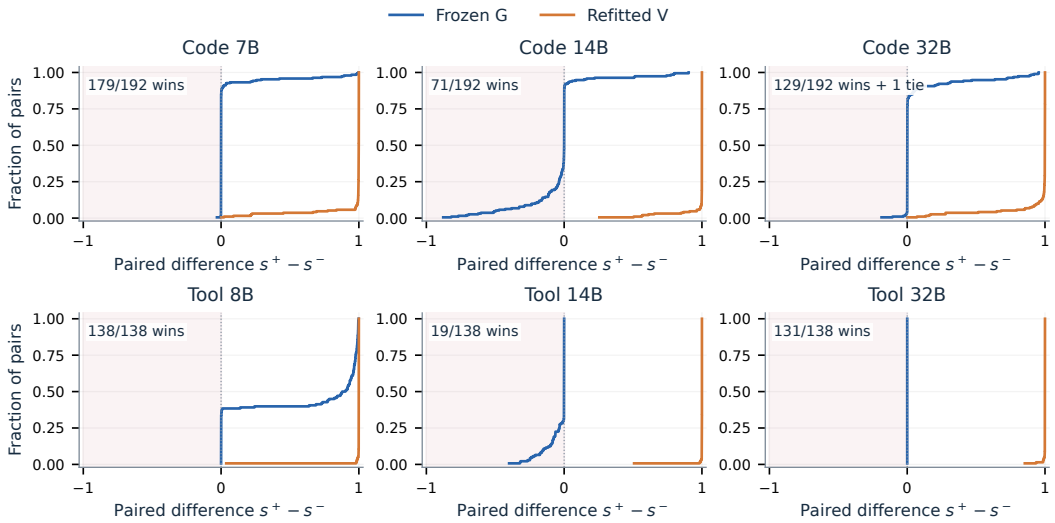


Figure 9: **Paired responses of frozen and refitted scores.** Every test pair contributes one difference. Shading marks reversed ordering; counts use strict wins. Refitting uses separate V-training families and orders every held-out pair correctly.

L.3 Every cross-family comparison behind pooled AUC

Figure 10 displays all m^2 comparisons for all six frozen scores. Both axes order families by their mean frozen logit, keeping the same family on the diagonal. A blue cell means the row family's positive member scores above the column family's negative member; a rose cell means the reverse and white marks a tie. The diagonal concerns fixed-answer responses, while the off-diagonal region dominates pooled AUC. These images explain the existing ranking accounting in Appendix H.5; the cells are dependent because they share observations.

L.4 What changes between generated and supplied tool answers

Figure 11 displays the error census from Appendix H.4. It is conditional on failure and pools all data splits. The distinct error mixtures offer an alternative explanation for score-transfer behavior; the comparison does not isolate error taxonomy from candidate provenance or formatting. Code-end formatting shifts are described separately in that appendix.

L.5 Uncertainty and sample sizes in restoration comparisons

Figure 12 presents the category counts of Table 3 with two-sided 95% Clopper–Pearson intervals. The maze panel uses the same official-model census at both sites. Sudoku's earlier and late samples differ, so its plotted contrast is not a paired estimate of an event effect. The intervals are descriptive binomial intervals at the donor level; maze donors can share base problems, and they should not be read as cluster-adjusted uncertainty or evidence for a population-independent timing boundary. The separate local 10×10 checkpoint is deliberately not pooled into either panel.

L.6 Reading the visual diagnostics together

The controls compare predictive features; the ordering matrices unpack pooled AUC; the error census describes population shift. Restoration intervals expose sample sizes. These complementary views clarify existing evidence rather than supplying independent replications. Empirical scope and downstream requirements follow.

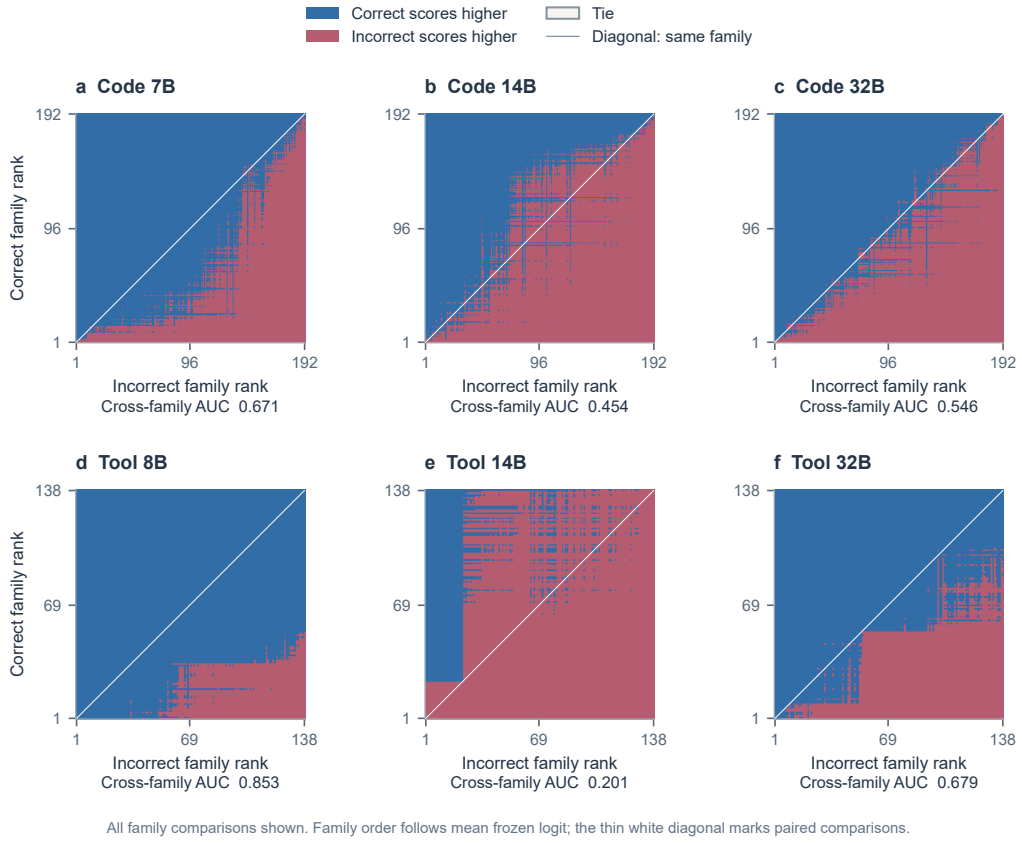


Figure 10: **Paired responses occupy only the diagonal.** Complete frozen-score ordering matrices for all three code models (192^2 cells each) and all three tool models (138^2 cells each). Cross-family AUC excludes the diagonal. The thin white diagonal locates within-family comparisons; exact counts are in Table 15. No scores are refitted or reoriented for this display.

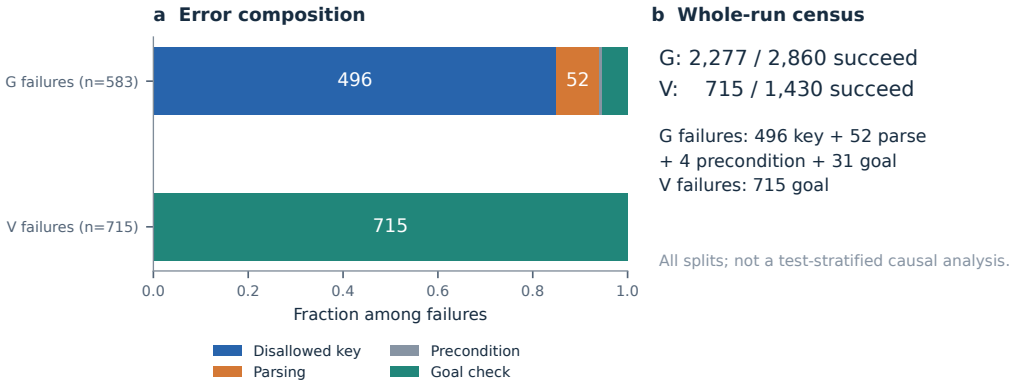


Figure 11: **Tool 14B transfer also changes the failure mixture.** The bars normalize within 583 G failures and 715 V failures, using whole-run checker records. G has 496 disallowed-key, 52 parsing, four precondition, and 31 goal-check failures; all V failures concern the goal. The adjacent census gives the success denominators. This is a descriptive composition plot, not test-set error-stratified AUC.

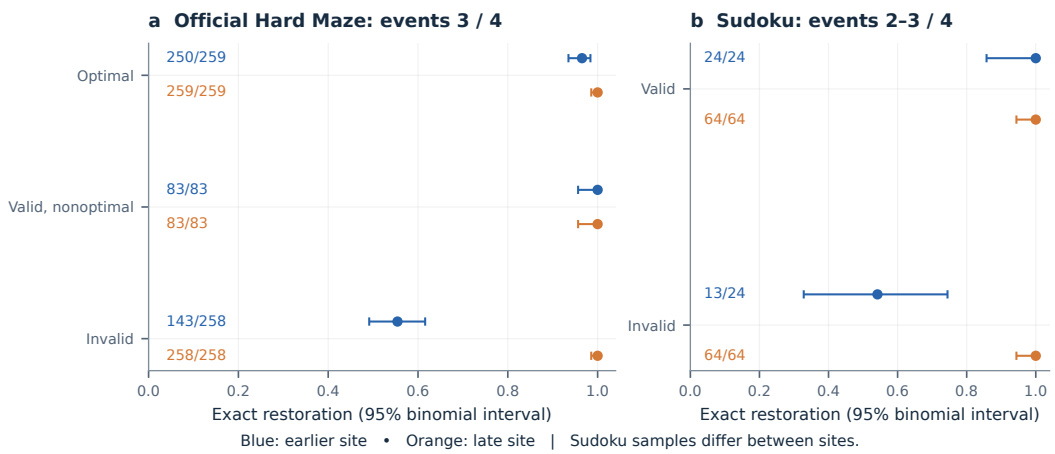


Figure 12: **Restoration by category with explicit denominators.** Blue marks the earlier site and orange event 4. Whiskers are two-sided 95% binomial intervals computed from the displayed saved counts, without a new intervention run. The figures summarize output equality; complete restoration can include checker failure. Timing provenance and displacement controls remain in Appendix F.

M Discussion and Limitations

The protocol separates five properties that ordinary correctness-probe evaluation can conflate: decodability, score reuse, fixed-answer responsiveness, cross-problem ranking, and output restoration. Pair accuracy measures the within-problem response; pooled AUC also evaluates ranking across problems. For interventions, checker success identifies the task consequence of output equality. These comparisons can be applied to existing scores and interventions.

Fixed-answer evaluation holds the candidate constant while requirements change; best-of-N holds the task constant while candidates change. Task-specific offsets can disrupt cross-problem ranking while cancelling within one task. The two 32B utility cohorts illustrate why selection and screening warrant separate tests: tool filtering improves over likelihood, but input controls explain much of the available predictability, and within-task selection gains are small. Empirical risk at a chosen coverage describes the evaluated cohort; deployment requires a threshold selected independently and validation under the intended task distribution. Balanced candidate pairs alone do not supply deployment error rates.

The empirical scope follows the construction. Generation matching selects output-persistence cases; replay fixes candidates under synthetic contract changes with strong compatibility cues. The six post-hoc transfers combine provenance, format, and error-type shifts. Natural specification changes and unseen transformation rules remain open: in the original code convention study, every model stays below 0.70 cross-template AUC (Appendix E). Graph residualization restricts linear input explanations; nonlinear structure may remain. Whole-state patches establish output support at specific sites and populations. Testing a fitted probe direction's causal role requires intervening on that direction.