FALSIFICATION-TIME COMPUTE: SCALING INFERENCE BY SEARCHING FOR COUNTEREXAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

When candidate answers can be executed, additional inference-time compute can seek new evidence instead of another answer. Falsification-Time Compute (FTC) actively constructs an executable world—a witness—on which the live candidates behave differently. Execution reveals the disagreement; a separate source of semantic authority decides whether it supports rejection. On 100 Spider tasks with frozen model-generated candidate pools, FTC improves selection from 66% to 73% under true Test Suite Accuracy (TSA), against an 83% pool oracle. The gain comprises seven recoveries, zero regressions, and 41.2% closure of the measured Public-to-Oracle gap. Controlled and natural-pool studies show that candidate conditioning can strengthen weak witness proposals, while strong shared search can also find useful distinctions. Open witness search retains positive TSA value without explicit failure-family routing. Finding a witness is not enough, however: candidate-blind learned interpretation introduces an additional SQL regression and loses both available search recoveries in a small code diagnostic. FTC thus separates two inference-time problems—discovering executable evidence and establishing enough semantic authority to act on it.

1 INTRODUCTION

What should additional inference-time compute do when candidate answers can be executed? Most test-time scaling methods generate more candidates, explore more reasoning paths, judge answers, or revise them (Wang et al., 2023; Madaan et al., 2023; Snell et al., 2025). Executable domains permit another action: change the evidence.

Consider two SQL queries that both return `3` on the public database. One counts rows with `COUNT(*)`; the other counts distinct continent values. Add a second row with a repeated continent and their behavior separates immediately: one returns `2`, the other `1`. The tiny database did not create a new hypothesis. It exposed a difference between hypotheses already in the pool. We call such a concrete, replayable world an *executable witness*.

This example also exposes the central difficulty. Execution tells us that the queries disagree, but not which output the question intended. If the task asks for rows, `2` is appropriate; if it asks for distinct continents, `1` may be. Finding a disagreement and knowing what it means are different computational roles.

Falsification-Time Compute (FTC) uses inference-time computation to actively choose or construct a witness because it can distinguish the current live candidates. The resulting loop is simple: maintain candidate hypotheses, acquire a witness, execute the candidates, interpret the resulting behavior groups, and update the candidate set. Candidate generation expands the hypothesis set. FTC changes the evidence available about a fixed set. A third role—semantic authority—determines whether the observed disagreement warrants rejection.

FTC therefore has two central questions. First, can search find a world that separates plausible answers in a useful way? Candidate access, task-level structure, engineered SQL knowledge, and learned proposals are alternative sources of search power. Second, once candidates disagree, what can authorize an elimination? A trusted specification can certify failure. When none is available, a learned model may interpret the task, witness, and anonymous output classes, but its judgment remains fallible even though the executions are exact.

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.

Our experiments follow these questions. On 100 Spider tasks with frozen candidate pools, family-routed FTC raises true TSA from 66% to 73%, against an 83% pool oracle. All seven recoveries survive the multi-database evaluation, with zero regressions. Controlled studies then show why search policy matters: candidate conditioning raises weak generic test generation from .182 to .977, while a strong task-level family bank reaches 1.000. On natural pools, actively selected shared tests can also match family-routed search.

We next compare two ways to construct witnesses. On an 80-task SQL pool, family-routed search reaches 58/80 under TSA. Open search, which proposes small databases without explicit failure categories, improves Public from 52/80 to 55/80 with no damage. Learned candidate-blind interpretation preserves those three recoveries but adds a separate regression, yielding 54/80. In code, the separation is sharper: open input search moves 23/29 to the 25/29 pool ceiling, while learned interpretation falls to 22/29. Search and authority are empirically distinct bottlenecks.

Our contributions are:

- We formulate FTC as active executable evidence acquisition: a use of inference-time compute distinct from generating or judging candidate answers.
- We establish natural-pool selection value under true TSA and identify how candidate access, shared structure, engineered families, and open proposals affect witness search.
- We separate witness discovery from semantic authority in SQL and code, showing that useful executable evidence can be found and then used incorrectly.

2 Related Work

Inference-time scaling. Chain-of-thought, self-consistency, Tree-of-Thoughts, and iterative refinement spend test-time compute on reasoning, sampling, search, or revision (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023; Yao et al., 2023; Madaan et al., 2023; Shinn et al., 2023). Compute-allocation work asks how to distribute this budget (Snell et al., 2025). FTC adds an action available when answers are executable: acquire a world that changes what can be observed about the current pool.

Verifiers, tests, and execution feedback. Outcome and process verifiers score candidate solutions (Cobbe et al., 2021; Lightman et al., 2024; Uesato et al., 2022); reviewer models, CodeT, and self-debugging use judgments or executions in code generation (Zhang et al., 2023; Chen et al., 2023; 2024; Jiang et al., 2024). Generated-test and repair systems use tests and counterexample-guided feedback (Chen et al., 2025; Orvalho et al., 2025; Liang et al., 2025); execution-guided SQL decoding filters invalid behavior (Wang et al., 2018). Fixed tests and execution consensus evaluate candidates under evidence already chosen. FTC instead spends compute selecting a witness for the live hypotheses.

Counterexamples and semantic testing. CEGIS and property-based testing exploit counterexamples under executable specifications (Solar-Lezama, 2008; Gulwani, 2010; Claessen & Hughes, 2000). Distilled SQL suites evaluate queries across databases (Zhong et al., 2020); BIRD, LiveSQL-Bench, and REFUTE motivate broader executable evaluation and failure search (Li et al., 2024; Zhang et al., 2025; Gu et al., 2025). FTC brings these ideas to fixed-pool inference: it studies both how compute finds a discriminative witness and what authority permits the system to act when no trusted specification accompanies that witness.

3 Method

3.1 FTC in One Round

For a task x, let $Y = \{y_1, \ldots, y_N\}$ be the live candidate pool and $\mathcal{C}(x)$ the space of executable worlds. A witness $c \in \mathcal{C}(x)$ is a small database for SQL or a concrete input for code. FTC actively chooses c because executing the current candidates on it is expected to reveal a useful distinction.

Figure 1 illustrates one round. Three queries are plausible on the public database. Search duplicates one continent value in a tiny table. Execution then produces two behavior groups: candidates A and

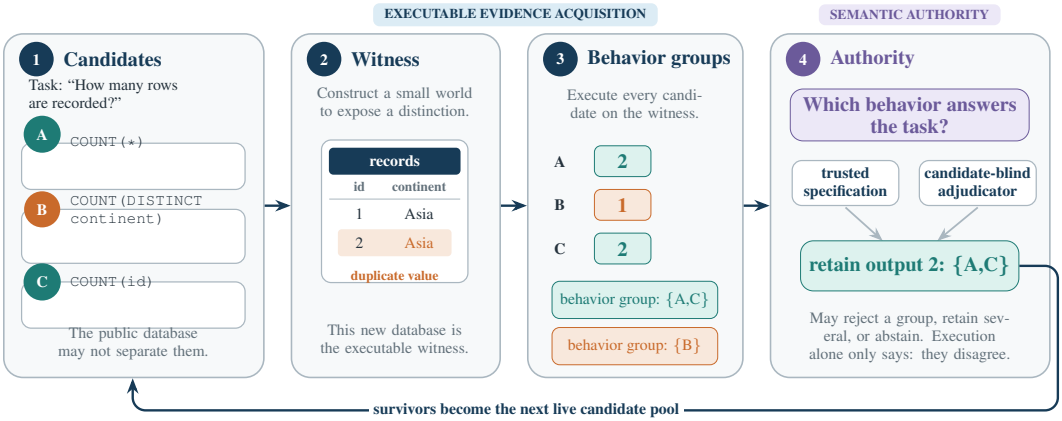

Figure 1: One illustrative FTC round. The candidate pool stays fixed while search changes the evidence by constructing a duplicate-containing database. Execution exposes two behavior groups. Only the authority layer can turn that disagreement into rejection; it may also retain several groups or abstain.

C return `2`, while B returns `1`. This partition is evidence, but not yet a verdict. A trusted specification or learned interpreter must decide whether any group can be rejected; survivors become the pool for the next round.

This definition sets FTC's boundary. Sampling another candidate expands Y; judging plausibility scores members of Y; executing a fixed test suite reuses pre-selected evidence. FTC's defining action is to choose or construct a new executable world for the current pool. An admissibility check enforces declared constraints, although schema validity may not capture every meaningful-world constraint.

3.2 Finding a Useful Witness

Let $E(y, c)$ be the observable result of executing candidate y on witness c, and let $[o]$ denote the equivalence class of output o under the declared comparison. The candidates induce a set of output classes $\mathcal{O}(c)$, with

$$z_i(c) = [E(y_i, c)] \in \mathcal{O}(c), \qquad i = 1, \ldots, N. \tag{1}$$

Order-sensitive tasks require an order-aware comparison. Our open-witness experiments select among admissible proposals using the split utility

$$U(c) = 1 - \frac{\max_{z \in \mathcal{O}(c)} |\{y_i \in Y : z_i(c) = z\}|}{|Y|}. \tag{2}$$

Zero utility means no discrimination; higher utility means a smaller largest behavior group. The score rewards a split, not correctness.

Search policies. All search variants implement the same action. *Candidate-conditioned* search uses one candidate to aim at a likely weakness. *Shared* search chooses a test for the pool without inspecting a particular candidate. *Family-routed* search uses engineered SQL knowledge—joins, NULLs, counts, aggregation, and existence—to construct worlds. *Open* search asks a learned proposer to build a discriminative world directly. These are policy choices inside FTC, not separate methods.

3.3 From Disagreement to Rejection

Write $R(x, y) = 1$ when candidate y is correct under the intended task specification. Execution identifies behavior groups; rejection requires an authority that connects behavior to intent.

Certified failure evidence. Let $V(x, y, c)$ be a trusted failure verifier, with $V = 1$ indicating established failure and $V = 0$ indicating that failure has not been established. Its required one-sided contract is

$$V(x, y, c) = 1 \implies R(x, y) = 0. \tag{3}$$

A sound executable specification or formal checker can supply this authority. A verified failure supports rejection; finding no failure does not certify correctness. Reference-assisted comparison is valid only when the reference and witness reflect the intended specification.

Candidate-blind learned semantic adjudication. Without a trusted specification, a candidate-blind adjudicator receives the task, schema or function specification, witness, and anonymous output classes. It does not see candidate programs, identities, candidate-to-output mappings, reference answers, or correctness labels. The controller privately maps retained classes back to candidates. The adjudicator may retain several classes or abstain by retaining all of them; its authority remains learned rather than certified.

Conservative union. For m blinded adjudications, let $Z_j(c) \subseteq \mathcal{O}(c)$ be the classes retained by adjudication j. Retain their union:

$$Z_{\cup}(c) = \bigcup_{j=1}^{m} Z_j(c). \tag{4}$$

A class is eliminated only if every adjudication excludes it. This rule is conservative in a deterministic sense—one retaining judgment preserves a class—but it provides no probabilistic guarantee. A semantic mistake shared by every adjudication survives repetition.

3.4 Where Inference Compute Goes

Let

$$B = B_{\text{gen}} + B_{\text{evidence}} + B_{\text{authority}}, \tag{5}$$

where the terms pay for new candidates, witness search and execution, and interpretation of the resulting disagreement. A complete policy also specifies admissibility, output comparison, and a fallback when evidence leaves multiple survivors. Our main experiments freeze the candidate pool, so they isolate the value of B_{evidence} and $B_{\text{authority}}$ rather than claim an optimal allocation against additional sampling. Post-freeze reference interpretation measures search power; it is a diagnostic, not a deployable reference-free selector.

4 Experiments

The experiments answer four questions. RQ1 tests whether new executable evidence improves selection. RQ2 asks what gives witness search power. RQ3 examines what additional compute changes inside the pipeline. RQ4 asks whether useful evidence is necessarily used correctly. Public denotes selection using public execution evidence; the pool oracle is the fraction of tasks containing at least one candidate scored correct. Recoveries correct Public errors; damage loses Public-correct selections.

4.1 RQ1: Can Executable Evidence Improve Selection?

We begin with the central empirical question: if the candidate answers are held fixed, can spending compute on new executable worlds improve which answer is selected? Our primary pool contains 100 Spider tasks whose model-generated candidates pass weak public execution tests. We score with true Spider Test Suite Accuracy (TSA), which requires official execution equivalence across every available distilled database for each represented schema.

With two-row witnesses, family-routed FTC raises accuracy from 66/100 to 73/100 against an 83/100 pool oracle (Table 1). All seven discordant tasks favor FTC: seven recoveries and zero regressions. The paired gain is +7 percentage points, with a 95% bootstrap interval of +2 to +12 points and exact two-sided McNemar $p = 0.0156$. FTC closes $7/(83 - 66) = 41.2\%$ of the measured gap.

Table 1: Primary fixed-pool result under true Spider TSA using two-row witnesses. The candidate pool is unchanged; FTC acquires additional executable evidence.

Setting	Public	FTC	Oracle	Recoveries	Damage	Gap closed
Public-overfit full100	.66	.73	.83	7	0	41.2%

TSA lowers Public and FTC by ten points relative to single-database scoring, yet preserves every recovery. Seventeen tasks contain no TSA-correct candidate, and ten contain one that FTC still misses. Thus executable evidence has measurable selection value, but search neither creates a missing answer nor guarantees that every available answer is found. Secondary frozen-pool TSA results and evaluator audits appear in Appendix A.

4.2 RQ2: What Gives Witness Search Power?

New evidence helps only when search finds a world that exposes a relevant difference. We compare what information the proposer uses and how explicitly the search is structured.

A controlled SQL bank contains 176 augmented variants from 44 base tasks whose public instances leave candidates indistinguishable. Generic task-conditioned tests select correctly at .182. Conditioning search on a candidate raises accuracy to .977, exceeding an LLM judge at .932 (Table 2). Yet a strong task-only family bank reaches 1.000. Candidate access can focus weak proposals, but sufficiently targeted shared knowledge can cover the same distinctions.

Table 2: Controlled fixed-pool comparison. The 176 stress variants derive from 44 base tasks and use controlled-bank labels, not Spider TSA. Oracle regret is the gap to the pool oracle.

Selector / proposal policy	Accuracy	Oracle regret
Public tests	.000	1.000
LLM judge	.932	.068
Generic task-conditioned tests	.182	.818
Candidate-conditioned falsifier	.977	.023
Hand-built task-only family bank	1.000	.000

Natural pools show the same pattern. Actively selected shared tests match family-routed FTC on three of four official-single settings and slightly exceed it on the fourth; execution consensus wins once, ties once, and trails twice (Table 19). No conditioning strategy uniformly dominates.

We then compare explicit SQL families with open witness construction on the 80-task non-pilot pool under TSA. Family-routed search reaches 58/80. Open search uses one Qwen2.5-7B-Instruct proposal per task, followed by admissibility checks and deterministic top-$U(c)$ selection. With reference SQL consulted only after witness freeze, open search raises Public from 52/80 to 55/80, recovering three errors with no damage (Table 3).

Table 3: Search-policy comparison on the same frozen non-pilot80 pool under true TSA. Reference interpretation occurs only after witness selection.

Policy	Non-pilot80 TSA
Public	52/80
Family-routed FTC	58/80
Open search + post-freeze reference adjudication (Stage A)	55/80
Pool oracle	67/80

Open search therefore has positive selection value without enumerating SQL failure categories, although the engineered policy remains stronger. The common object is the witness; conditioning and routing determine how effectively search finds one.

4.3 RQ3: What Does Additional Compute Change?

Our fixed-pool design deliberately spends no additional calls on candidate generation. It measures what compute can do after the hypotheses already exist. The controlled allocation sweep makes this

distinction visible: as the fraction assigned to falsification rises from 0/20/40/60/80%, accuracy follows .000/.182/.932/.977/1.000. These operating points mix policies and are not a universal frontier, but they show that compute can buy evidence coverage rather than more samples.

Coverage is the immediate constraint. In the local SQL run, 54/80 proposals are parseable, only 16/80 produce a usable nontrivial witness, and 26/80 hit the frozen 1400-token cap. A higher-coverage provisional proposer finds witnesses on 41/80 tasks, suggesting search headroom without furnishing a controlled model comparison. More evidence-side compute helps only if it reaches distinctions that the current proposal process misses.

Compute accounting also separates accuracy from efficiency. On the controlled end-to-end bank, FTC reaches 1.000 with 4.00 model calls and 6.09 executions per task, while judge reranking reaches .932 with 3.00 model calls and 4.00 executions. The judge has better correct-per-call ratios; FTC has higher accuracy. Token-, latency-, and dollar-normalized comparisons remain unavailable (Appendix E). Thus these experiments establish a distinct use of compute, not its universal optimality relative to additional sampling.

4.4 RQ4: Does Useful Evidence Guarantee Correct Use?

No. Holding the selected SQL witnesses fixed, post-freeze reference interpretation reaches 55/80 under TSA. Candidate-blind union-of-3 preserves the same three recoveries but introduces a separate regression, yielding 54/80 (Table 4). Search succeeds on those three tasks; learned authority adds an independent error.

The same separation appears more sharply in code. On MBPP/66, the input `[1,-2,0]` splits an implementation that counts $x \geq 0$ from implementations that count $x > 0$: their outputs are `2` and `1`. Open input search finds this witness, but learned union retains both behaviors, so the baseline tie-break remains wrong. Across the frozen MBPP+29 pool, reference interpretation moves 23/29 to the 25/29 oracle. Learned interpretation loses both recoveries and introduces one baseline-correct regression, falling to 22/29.

Table 4: Search and learned authority on fixed candidate pools. SQL uses true TSA; code uses the frozen MBPP+29 protocol. Reference interpretation occurs only after witness freeze.

Domain / contract	Public	Search + reference	Learned authority	Oracle
SQL / true TSA	52/80	55/80	54/80	67/80
Code / MBPP+29	23/29	25/29	22/29	25/29

Fixed-witness SQL diagnostics reveal why repetition is insufficient. For a Caribbean surface-area question, one country joins to two city rows. The intended/reference output is `100`; the adjudicator repeatedly favors the duplicated join total `200`, across both tested prompts and all three matched-prompt permutations. Union can soften an idiosyncratic exclusion, but not a semantic mistake shared by every decision.

The SQL and code losses differ in size and mechanism, so they do not establish a universal failure pattern. They establish the decomposition: finding an informative executable world does not guarantee that learned authority will use it correctly. Full prompt controls and frozen-response audits appear in Appendices C and D.

5 Limitations

The experiments hold candidate pools fixed, so they measure the value of evidence acquisition rather than the optimal trade-off against more candidate samples. The primary result covers 100 Spider tasks, and the controlled bank contains augmented variants from 44 base tasks. TSA evaluates many distilled databases but does not prove equivalence on every possible world; shared-search and fixed-witness diagnostics that retain official-single scoring are labeled separately.

Witness validity and authority remain substantive assumptions. Schema checks may miss latent constraints, references can diverge from natural-language intent, and learned adjudicators can share errors. Open SQL search remains below family-routed search, and its 16/80 usable-witness coverage

confounds proposer capability with the frozen token budget. The code pool has only two recoverable baseline errors. These results support a mechanism decomposition, not broad domain-level performance or a normalized compute-optimal frontier.

6 Conclusion

What should additional inference-time compute buy? Sometimes, not another answer, but a world that distinguishes the answers already available. FTC turns that possibility into an explicit loop: acquire a witness, observe behavior, interpret the disagreement, and update the pool. Natural-pool gains show that this evidence can improve selection; controlled, SQL, and code experiments show that search power and semantic authority can fail independently. The resulting allocation problem spans three roles: compute for candidate generation, compute for executable evidence discovery, and compute for deciding what that evidence means.

AI use statement

We used LLMs to assist with language polishing and manuscript editing. We also used LLM-based coding agents to write and revise research code. All AI-assisted code and outputs were manually reviewed and tested, and separate independent LLM-agent runs provided an additional audit of the code, experimental artifacts, and manuscript consistency. The systems evaluated in this work are themselves LLM-based, including witness proposers and semantic adjudicators. The authors reviewed all AI-assisted work and take responsibility for the final text, code, analyses, figures, claims, and artifacts.

Reproducibility statement

Appendices document evaluation contracts, frozen selections, prompts, blinding, model settings, resource accounting, and protocol audits. We will release the original code and experimental artifacts as anonymous supplementary material with the submission.

References

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations*, 2023.

Xiancai Chen, Zhengwei Tao, Kechi Zhang, Changzhi Zhou, Xinyu Zhang, Wanli Gu, Yuanpeng He, Mengdi Zhang, Xunliang Cai, Haiyan Zhao, and Zhi Jin. Revisit self-debugging with self-generated tests for code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations*, 2024.

Koen Claessen and John Hughes. QuickCheck: A lightweight tool for random testing of Haskell programs. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*, pp. 268–279, 2000.

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.

Alex Gu, Jacob Steinhardt, et al. Can language models falsify? evaluating algorithmic reasoning with counterexample generation. *arXiv preprint arXiv:2502.19414*, 2025.

Sumit Gulwani. Dimensions in program synthesis. In *International Conference on Formal Methods in Computer-Aided Design*, pp. 13–24, 2010.

Nan Jiang, Xiaopeng Li, Shiqi Wang, Qiang Zhou, Soneya Binta Hossain, Baishakhi Ray, Varun Kumar, Xiaofei Ma, and Anoop Deoras. LeDex: Training LLMs to better self-debug and explain code. In *Advances in Neural Information Processing Systems*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, et al. BIRD: A big bench for large-scale database grounded text-to-SQL evaluation. In *Advances in Neural Information Processing Systems*, 2024.

Yaxin Liang, Rui Ying, Takuya Taniguchi, Chengguang Gan, and Zhenzhong Cui. RECODE: Leveraging reliable self-generated tests and fine-grained execution feedback to enhance LLM-based code generation. In *Advanced Intelligent Computing Technology and Applications*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Pedro Orvalho, Mikolas Janota, and Vasco Manquinho. LLM-CEGIS-repair: Counterexample-guided program repair with large language models. In *arXiv preprint arXiv:2502.07786*, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.

Armando Solar-Lezama. *Program Synthesis by Sketching*. PhD thesis, University of California, Berkeley, 2008.

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.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*, 2018.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Tianyi Zhang, Tao Yu, Tatsunori B. Hashimoto, Mike Lewis, Wen-tau Yih, Daniel Fried, and Sida I. Wang. Coder reviewer reranking for code generation. In *International Conference on Machine Learning*, 2023.

Yunjia Zhang et al. LiveSQLBench: A contamination-free and continuously evolving benchmark for text-to-SQL. In *International Conference on Learning Representations*, 2025.

Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic evaluation for text-to-sql with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 396–411, 2020.

A Evaluation Protocols and Statistical Evidence

This appendix records the benchmark contracts supporting the selection results. True TSA is used for the primary natural comparison, the additional frozen Public/FTC selections, and the fully specified local non-pilot80 composition. Official-single scores remain explicit for other mechanism diagnostics. The historical evaluator audit is kept separate from these results.

A.1 True Test Suite Accuracy Protocol

The TSA evaluation uses the official Spider test-suite implementation at commit `e97acc546ecbee8fa27fa8dbf025ef61493a876c`, with `plug_value=False`, `keep_distinct=True`, and a 60-second timeout. Ordering sensitivity is determined by whether `ORDER BY` appears in the gold query. A candidate must satisfy official execution equivalence across every available distilled test-suite database for its task's schema.

The official archive is 1,269,456,098 bytes, with SHA-256

`9ec24ea8debc6bd04abfe137b5f1a739b5a8836f32c0464e4dfc94eb7f41da96`

Its extraction contains 28 schema directories: eight classical datasets and the 20 Spider schemas represented in the evaluated frozen natural pools. These 20 schemas contain 695 distilled SQLite databases, approximately 25–60 per schema; the archive includes canonical `<db_id>.sqlite` and distilled `<db_id>vNNNN.sqlite` instances. Evaluation uses all official distilled databases

present for these represented schemas. This inventory describes the evaluated archive, not verified coverage of every Spider schema ever distributed.

Single-database sanity gate. Before enabling the distilled suite, the same official evaluator code path was run only on each task's original Spider database. Across all 562 frozen candidates in public-overfit full100, it reproduced Public 76/100, FTC 83/100, and pool oracle 87/100 exactly, with zero task-vector mismatches and zero unexplained evaluator mismatches. The subsequent TSA run evaluated 100/100 tasks, relabeled all 562 candidates, and completed with zero gold execution errors. Candidate pools and selected answers remained fixed.

Table 5: The primary public-overfit full100 pool under two evaluation contracts. TSA is the primary result; official-single is retained as an evaluator-strength comparison. The paired difference is +7 percentage points under both contracts.

Contract	Public	FTC	Pool oracle	Gap closed
Official-single	76/100	83/100	87/100	63.6%
True TSA	66/100	73/100	83/100	41.2%

Candidate-label changes. Table 6 shows 72 official-single-correct candidates becoming TSA-incorrect and zero upgrades. In this audited pool, TSA is strictly more demanding. Public and FTC selected-candidate correctness each change on 10 tasks; pool-oracle status changes on four tasks, and paired Public/FTC outcomes change on 11 tasks. The aggregate paired result remains seven recoveries and zero regressions.

Table 6: Official-single versus true TSA labels for all 562 candidates in the primary full100 pool. The 72 changes are downgrades; no candidate upgrades in this pool.

Official-single label	TSA correct	TSA incorrect
Correct	336	72
Incorrect	0	154

For SPIDERDEV/0002, the gold query counts all rows in `CONTINENTS`, while a candidate effectively computes `COUNT(DISTINCT Continent)`. They agree on the original database. The distilled instance `car_1v5080.sqlite` contains duplicate continent values, separating `COUNT(*)` from the candidate. The example makes the role of executable coverage concrete: hypotheses that agree in one world can differ in another.

A.2 Frozen-Pool Selection and Paired Statistics

The full100 and non-pilot80 analyses below retain exact frozen task manifests and selected candidate indices. Only their evaluation labels change across contracts. Additional official-single search-policy comparisons are collected in Appendix B.3.

Paired TSA uncertainty for public-overfit full100. The analysis uses the exact frozen full100 row=2 task manifest and compares existing family-routed FTC with Public under TSA. Only evaluation labels change; no models are rerun and no candidate pools, selectors, prompts, or task manifests change. Public is correct on 66/100 tasks, FTC on 73/100, and the pool oracle on 83/100. The paired counts are 66 both correct, 7 FTC-only correct, 0 Public-only correct, and 27 both wrong; the last group comprises 17 pool misses and 10 residual selection errors. For $\Delta = \mathrm{Acc}_{\mathrm{FTC}} - \mathrm{Acc}_{\mathrm{Public}} = +0.07$, the exact two-sided McNemar test on the seven discordant pairs gives $p = 0.015625$. We resample tasks with replacement, retaining each Public/FTC correctness pair together, for 100,000 paired bootstrap replicates with fixed seed 20260909. The percentile 95% confidence interval is $[0.02, 0.12]$. The interval measures task-level sampling variation under TSA labels, not semantic equivalence beyond the evaluated databases.

Table 7: Paired task outcomes under true TSA for the primary public-overfit full100 comparison. All seven discordant pairs favor FTC.

	FTC correct	FTC wrong
Public correct	66	0
Public wrong	7	27

Fully specified non-pilot80 under TSA. A purely offline relabel scores the exact frozen selections from the fully specified local 80-task experiment. The official-single results were first independently reproduced: Public 60/80, family-routed FTC 66/80, Oracle 69/80, and Stages A and B each 63/80. The selection-identity gate matched all 80 frozen selections exactly for each stage. The same validated TSA evaluator and candidate-level TSA labels used in the primary full100 audit were then applied. No models were rerun; candidates, witnesses, prompts, selectors, adjudications, task manifests, and selected candidate indices remained unchanged. Only the correctness labels of the frozen selected outputs changed.

Table 8: The fully specified non-pilot80 experiment under two evaluation contracts. True TSA is the preferred contract; official-single values document the reproduced historical results. All pools and selected candidate indices are frozen.

System	Official-single	True TSA
Public	60/80	52/80
Family-routed FTC	66/80	58/80
Pool oracle	69/80	67/80
Stage A	63/80	55/80
Stage B	63/80	54/80

Under TSA, family-routed FTC retains six recoveries with zero damage. Stage A has three recoveries, zero damage, and $3/(67-52) = 20.0\%$ gap closure. Stage B has three recoveries, one regression, and $2/(67-52) = 13.3\%$ gap closure. Both stages preserve the same three recoveries as under official-single: 0027, 0075, and 0094. The additional Stage-B regression is 0022, which is TSA-correct under Public and Stage A but incorrect under Stage B. Thus Stage B preserves all three search recoveries; it does not lose one of them. Stronger provisional and prompt-control selections remain official-single diagnostics and are not relabeled by this audit.

Table 9: Additional frozen Public and family-routed FTC selections under true TSA. Recovery and damage are task-level changes relative to Public.

Pool	Public	FTC	Recovery	Damage
Official100 llama3:8b	.68	.69	2	1
Official100 qwen2.5:7b	.87	.87	0	0
Official100 mixed, row=2	.87	.87	0	0
Public-overfit full100 qwen2.5:7b, row=2	.85	.85	0	0
Public-overfit full100 qwen2.5-coder:7b, row=2	.84	.86	2	0
Public-overfit pilot20 llama, row=2	.65	.75	2	0
Public-overfit pilot20 llama, sameanswers	.65	.70	1	0

A.3 Historical Evaluator and Label Audits

Repository-to-official-single audit. The read-only audit relabeled existing natural-pool candidates; it did not regenerate their answers. The repository oracle executed candidate/reference queries on one Spider database and applied unconditional `sorted(rows, key=repr)`. The official-single evaluator uses the official execution-equivalence semantics on that database. Across the audited candidate records, the confusion matrix is:

Table 10: Candidate-level oracle audit across all audited natural pools. These are candidate records, not independent task counts or TSA outcomes.

Repository label	Official-single correct	Official-single incorrect
Correct	13,085	63
Incorrect	598	4,495

All 63 positive-to-negative flips are ORDER BY artifacts concealed by unconditional row sorting. The 598 reverse flips arise because official result equivalence accepts some denotational permutations the repository canonicalizer rejected. These official-single labels remain the scoring contract for shared-search and the stronger provisional and fixed-witness adjudication diagnostics. The primary natural-pool comparison and the fully specified local non-pilot80 composition are further evaluated with true TSA.

Historical official-single uncertainty (descriptive). Under the previous official-single contract, the full100 paired counts were 76 both correct, 7 FTC-only correct, 0 Public-only correct, and 17 both wrong. Its paired-bootstrap interval was also $[.02, .12]$, and 99.911% of resamples had a positive difference; this fraction is a resampling diagnostic, not a posterior probability. In the official-single subset of 87 tasks with a labeled-correct pool candidate, Public reached $76/87 = .8736$ and FTC $83/87 = .9540$, a paired difference of $7/87 = .0805$. The secondary paired-bootstrap 95% interval was $[.0345, .1379]$, with exact two-sided McNemar $p = 0.015625$. The 13 excluded pool misses were both wrong. These historical conditional results are not TSA subset estimates; the primary TSA analysis uses all 100 tasks.

Why labeled-correct candidates can disagree on witnesses. Residual audits found queries that count distinct teacher names instead of teacher rows, collapse duplicate cartoons with DISTINCT when all cartoons are requested, duplicate states through missing DISTINCT, or count only entities participating in an extra join. Such queries can agree with a reference on one database without being semantically equivalent. These historical witness audits explain why single-database labels may miss distinctions; the candidate-level TSA changes above are reported separately rather than assigning new labels to unaudited residual cases.

Four explanations must remain separate: heuristic-property misspecification, weak single-database correctness labels, latent world constraints absent from the schema, and candidate-blind adjudication errors. A raw reference disagreement is not sufficient to assign one of these explanations. In particular, reference/NL disagreement and witness admissibility need independent inspection. The historical hits have not been exhaustively partitioned among these causes.

B Search Policies and Supporting Diagnostics

This appendix groups implementation details and supporting search experiments. The operator library specifies one FTC search policy; controlled ablations and additional natural-pool comparisons characterize its behavior without defining the framework by that implementation.

B.1 Family-Routed Operators and Diagnostic Definitions

The family-routed candidate-conditioned implementation parses joins, predicates, aggregates, and NULL-sensitive constructs and routes to a finite collection of perturbation operators. It can use public examples and prior accepted hits. Shared subset search that acquires discriminative witnesses and open pool-conditioned proposals are alternative FTC policies; merely executing fixed tests is a comparator. The tables document this engineered policy. Open witness search does not route proposals through this taxonomy.

Table 11: SQL falsifier operator families. Operators are candidate-conditioned and produce small schema-consistent database instances.

Operator	Candidate signal	Generated artifact
Join-cardinality perturbation	`JOIN`, foreign keys, counts	Duplicate or missing matching rows
NULL-sensitive perturbation	Predicates, aggregates, outer joins	Rows containing NULL boundary values
Aggregation boundary	`COUNT`, `SUM`, `AVG`, `GROUP BY`	Empty, singleton, duplicate groups
Predicate boundary	Comparisons and equality filters	Rows at or near predicate thresholds
Row-preserving metamorphic	Order-insensitive queries	Shuffled or permuted database instances

Table 12: Code falsifier operator families. Operators generate executable inputs that are replayed against candidate programs.

Operator	Applies to	Examples
Boundary values	Numeric, string, list inputs	Empty, singleton, min/max, off-by-one
Type/shape mutation	Lists, dictionaries, strings	Nested, repeated, duplicate, unsorted inputs
Public-test perturbation	MBPP/HumanEval examples	Nearby inputs preserving the same intent
Adversarial hidden-case prompt	General functions	LLM-proposed failing input against candidate code
Replay constraints	Repaired candidates	Previously accepted test evidence

Keep evidence types separate in reporting. For certified evidence, a verified-failure rate is $\Pr[V(x, y, c) = 1]$ under the stated sampling protocol and verifier contract. For heuristic properties,

report acceptance rates and their label-relative audits instead. For anchored executable evidence, report retained classes, candidate eliminations, reference-class retention, public-error recoveries, and damaged public-correct selections separately. Execution discrimination is measured by output partitions, not by assuming each different output is wrong. Fixed-pool regret is measured relative to the declared evaluation oracle; it is not semantic regret when the oracle is only official-single. Repair utility and replay pass rates likewise require specifying both the evidence interface and evaluation labels.

B.2 Controlled Bank and Historical Property Audits

The following controlled-bank measurements are retained from the original manuscript, not recomputed under the natural-pool oracle audit. The 176-bank end-to-end configuration includes repair; its 1.000 score must not be conflated with .977 fixed-pool selection. "Accepted hits" below means acceptance by that implementation's property checker, not independently certified semantic failure. A verifier-relative hit becomes certified evidence only when its soundness contract and witness assumptions hold.

Bank construction and configurations. The augmented bank contains 176 controlled variants from 44 base tasks, including three row-shuffled variants per base task, eight schema families, and twelve failure modes. These variants are not 176 independent natural examples. The original SQL implementation uses family-specific executable properties without calling `reference_sql` during falsification or repair. Its end-to-end configuration reaches 1.000 accuracy, compared with .932 for judge reranking and .057 for random database testing; fixed-pool selection is the separate .977 configuration reported in the main text.

Table 13: Deployable end-to-end SQL evaluation on 176 augmented variants from 44 base tasks. These are controlled stress cases, not independent natural benchmark examples. The FTC loop does not use `reference_sql`; the main comparison is against scalar judging and untargeted database testing.

Method	Acc.	Accepted hits	Repairs	Model Calls	Exec Calls
Single-shot	0.000	0.000	0.000	1.00	4.00
Self-consistency	0.000	0.000	0.000	3.00	7.00
Judge rerank	0.932	0.000	0.000	3.00	4.00
Execution feedback	0.000	0.000	0.000	1.00	5.00
Random DB test	0.057	0.057	0.057	1.06	6.00
FTC	**1.000**	**1.000**	**1.000**	4.00	6.09

Scope of the engineering ablation. Removing join/cardinality or NULL checks harms the original family-routed implementation; generic metamorphic checks alone fail on this controlled bank. This is evidence about that implementation, not a requirement that every FTC instance enumerate semantic failure families. The zero-family natural-pool diagnostics examine a different search policy and cannot be used to erase this controlled engineering cost. Domain knowledge and semantic authority remain necessary even when failure-family routing is absent.

Table 14: Deployable SQL verifier ablation on the 44-task full bank. Performance degrades sharply when task-relevant semantic checks are removed.

Verifier	Acc.	Accepted hits
Reference-assisted	1.000	1.000
Deployable full	**1.000**	**1.000**
No join/cardinality	0.614	0.614
No NULL checks	0.750	0.750
Metamorphic-only	0.000	0.000

Table 15: Design cost of deployable SQL verifier components. Domain expertise and reusability are qualitative diagnostics for portability, not measured costs.

Verifier component	Domain expertise	Reusable?	Failure modes covered
Schema consistency	Low	High	Invalid databases, broken keys
Join/cardinality checks	Medium–high	Medium	Join omission, duplication, many-to-one errors
NULL-sensitive checks	Medium	High	NULL predicates, aggregates, missing filters
Aggregation/count checks	Medium	Medium	Count, group, duplicate, empty-group errors
Row-preserving metamorphic checks	Low	High	Order sensitivity, unstable execution artifacts

Table 16: Cross-slice SQL robustness checks. Slices vary task composition and failure-mode coverage under the same oracle-assisted falsification interface.

Slice	Tasks	Failure Modes	Judge	FTC	Falsifier
full44	44	12	0.932	**1.000**	0.977
balanced32	32	9	0.906	**1.000**	1.000
failureaware32	32	11	0.938	**1.000**	0.969
cappedstress24	24	11	0.917	**1.000**	0.958

Historical mutation-derived bridge. The old 32-example real-database bridge used reference-derived executable mutations rather than naturally sampled answers. It is retained only as a historical selection-isolation diagnostic. Its reported zero hit rate on labeled-correct candidates does not establish a soundness theorem, nor is this table an official-single relabeling of natural pools. The full44 rate .067 was a label-relative audit finding, not a measured universal false-positive probability.

Table 17: Historical label-relative property-hit diagnostics, not a soundness certificate. Bridge candidates are reference-derived; these rates have not been reinterpreted as revised natural-pool measurements.

Setting	Public	Falsifier	Labeled-correct hit	Labeled-incorrect hit
Spider strong public	1.000	1.000	0.000	0.859
Spider weak public	0.281	**0.875**	0.000	0.890
Full44 controlled	0.182	0.977	0.067	0.986

The original controlled coverage summary spans 18 named strategies in six families. The 16 hits against labeled-correct candidates are localized to `tiny_database_no_active_task` in the absence/existence family. This localization identifies where to audit; it does not determine whether each event reflects a misspecified property, a weak label, or an inadmissible world.

Table 18: Historical controlled-bank property hits by family. Correct/incorrect columns use the original evaluation labels; a hit against a labeled-correct candidate is not by itself a proven semantic false positive.

Family	Accepted property hits	Wrong-candidate hits	Correct-candidate hits
null_semantics	64	64	0
boundary_threshold	64	64	0
absence_existence	56	40	16
aggregation_extrema	44	44	0
count_cardinality	40	40	0
join_cardinality	32	32	0

B.3 Additional Natural-Pool Search Diagnostics

These tables preserve explicitly labeled official-single configurations and cross-contract failure decompositions. We retain official100, independent-first100, and public-overfit pool names rather than pooling their results. Official100 is the selector-diagnostic slice historically called "official-diverse100." The recorded K identifier is distinct from the explicit falsification-attempt count in the public-overfit sweep. Primary TSA statistics and the named non-pilot80 contract comparison appear in Appendix A.

Table 19: Natural-pool search-policy comparisons under official-single scoring, not TSA. Shared tests are task-conditioned; Exec-MBR aggregates execution consensus. Neither policy uniformly dominates family-routed FTC.

Setting	Public	Shared	Exec-MBR	FTC	Oracle
Official100 llama	.77	.78	.79	.78	.85
Official100 mixed, row=2	.92	.93	.83	.92	.95
Public-overfit pilot20, row=2	.75	.85	.85	.85	.95
Public-overfit full100, row=2	.76	.83	.78	.83	.87

Table 20: Official100 fixed-pool results under official-single scoring, retained for comparison across recorded configurations. These oracle values are not TSA oracles.

Pool / configuration	Public	Falsifier	Oracle
qwen2.5:7b, $K = 8$	.92	.92	.93
qwen2.5:7b, $K = 16$	.92	.92	.93
qwen2.5:7b, $K = 16$, row=2	.92	.92	.93
llama3:8b, $K = 8$	.77	.78	.85
mixed, $K = 8$	.92	.92	.95
mixed, row=2	.92	.92	.95

Table 21: Independent-first100 single-generator results under official-single scoring; TSA outcomes for these configurations are not reported.

Pool / configuration	Public	Falsifier	Oracle
llama3:8b, $K = 8$	.81	.82	.86
qwen2.5:7b, $K = 8$	.91	.93	.95

Independent mixed configurations are omitted because exact corrected row-level results could not be established.

Table 22: Fixed-pool public-overfit sweeps with llama3:8b under official-single scoring. Attempts and row limits change the falsifier, not the candidate pool. These sweeps are separate from the relabeled TSA selections in Table 1.

Pool	Attempts	Rows	Public	Falsifier	Oracle
pilot20	4	1	.75	.75	.95
pilot20	8	1	.75	.80	.95
pilot20	16	1	.75	.80	.95
pilot20	8	2	.75	.85	.95
pilot20	16	2	.75	.85	.95
full100	4	1	.76	.78	.87
full100	8	1	.76	.80	.87
full100	16	1	.76	.80	.87
full100	8	2	.76	.83	.87
full100	16	2	.76	.83	.87

Under official-single scoring, the full100 two-row gain is seven percentage points out of an eleven-point oracle gap: $7/11 = 63.6\%$ gap closure. The primary TSA result instead closes $7/17 = 41.2\%$. Under official-single scoring on pilot20, the existing falsifier closes $2/4 = 50.0\%$ of the public-to-oracle gap. The reference-assisted zero-family pilot diagnostic closes $4/4 = 100\%$ of that measured official-single gap, an upper-bound result on this frozen pool.

Table 23: Public-overfit full100 cross-generator replication under official-single labels, row=2. Oracle also equals pool-correct coverage.

Generator	Public	Falsifier	Oracle
llama3:8b	.76	.83	.87
qwen2.5:7b	.89	.89	.91
qwen2.5-coder:7b	.88	.90	.93

Table 24: Failure decomposition by evaluation contract. Counts refer to tasks; the four categories sum to each pool size. Only the primary full100 decomposition is reported under TSA; the other rows retain official-single labels.

Setting	Contract	Pool miss	Public sufficient	Recovered	Oracle residual
Full100, row=2	TSA	17	66	7	10
Full100, row=2	Official-single	13	76	7	4
Pilot20, row=2	Official-single	1	15	2	2
Official100 mixed, row=2	Official-single	5	92	0	3

Full100 and pilot20 in this table denote the public-overfit pools. Each decomposition uses its stated contract; official-single residual assignments are not carried forward as TSA labels.

C SQL WITNESS AND ADJUDICATION PROTOCOLS

The fully specified named-model non-pilot80 composition uses true TSA, with official-single values retained for historical comparison (Appendix A.2). The stronger provisional, pilot, and 41-witness prompt-control diagnostics below retain official-single scoring; their exact selected outputs have not been TSA-relabeled here.

There are two distinct zero-family witness sets on non-pilot80: 16 selected Qwen witnesses from the fully specified named-model replication and 41 selected witnesses from the stronger provisional session-agent experiment. The latter support both the stronger-anchor diagnostics and the separate 123-decision Ministral replay. Only the named Qwen-to-Ministral composition specifies both proposer and anchor; its 48 anchor decisions must not be conflated with that earlier replay (Appendix F).

C.1 FULLY SPECIFIED LOCAL CONFIGURATION

Frozen proposal protocol. The experiment `experiments/pcftc_named_proposer_v1` reuses the same 80 non-pilot public-overfit tasks, fixed candidate pools, public selector, schema/task inputs, and verbatim zero-family proposer prompt as the original prospective protocol. Earlier outcomes were already known; this is a reproducibility/control replication, not a new prospective or unseen evaluation. It retains the generic schema-derived admissibility kernel, split utility in Equation 2, and deterministic top-U witness selection, with no semantic failure-family routing. Exactly one Qwen2.5-7B-Instruct call is made per task. Proposals, selected witnesses, manifests, and hashes are frozen before reference/gold/correctness-label access; the per-task proposer prompt SHA-256 is recorded.

Stage A: search-only diagnostic. There are 54/80 parseable proposals. The remaining 26/80 completions hit the pre-specified 1400-token cap and truncate mid-output, yielding no usable witness. Usable admissible nontrivial witnesses occur on 16/80 tasks, including 4/9 recoverable public-error tasks under the original official-single labels; the corresponding stronger provisional proposer counts are 41/80 and 8/9 under that same contract. These historical recoverable-error denominators are not TSA coverage estimates. Across the 16 selected nontrivial witnesses, top-U ranges from .125 to .75, with mean .324. Among parsed tasks, the mean number of admissible witnesses is approximately 5.95 and the mean number of distinct output partitions is approximately 4.27. Many generated worlds are minor variations of a common sparse-table structure. These search diagnostics characterize coverage and structural homogeneity, not semantic authority.

Only after proposal and witness freezing is reference SQL unlocked to identify the reference output class. Under true TSA, Stage A reaches 55/80 against Public 52/80, family-routed FTC 58/80, and Oracle 67/80. It recovers 0027, 0075, and 0094 with zero damage, closing $3/15 = 20.0\%$ of the measured gap. All three official-single recoveries therefore survive TSA. This is a search-power diagnostic using post-freeze reference interpretation, not a deployable reference-free selector or a semantic correctness certificate. The two proposer operating points identify search capability and budget as coverage limitations without isolating their effects; no larger-token-budget intervention was tested.

Stage B: fully local Qwen-to-Ministral composition. The 16 selected Qwen witnesses receive candidate-blind semantic adjudication from Ministral-3-14B-Instruct-2512, GGUF Q4_K_M, llama.cpp, greedy decoding, using the exact frozen matched anchor prompt. Each selected witness receives three fresh blinded class-label/order permutations, for $16 \times 3 = 48$ calls, with responses frozen before reference scoring. Under true TSA, union-of-3 reaches 54/80: it preserves the three recoveries (0027, 0075, and 0094) but introduces one additional regression on 0022, closing $2/15 = 13.3\%$ of the gap. It eliminates 47 candidates across the 16 anchored tasks, with zero retain-all unions and two explicit abstentions. Two raw reference-class exclusions occur, on 0022 and 0087; these are not two established semantic errors.

Under official-single, Stages A and B both score 63/80, each closing $3/9 = 33.3\%$ of the historical gap with three recoveries and zero damage. True TSA separates them: Stage A scores 55/80 and Stage B 54/80. The post-freeze audit identifies 0087 as the recurring tie-versus-`LIMIT 1` reference/convention case. Case 0022 was already identified as a genuine adjudicator semantic miscount, but the original single database did not penalize the selected candidate. Under TSA, Public

and Stage A are correct on 0022 while Stage B is incorrect, making this a task-level regression rather than a harmless finding under that benchmark label. TSA exposes this particular adjudication error; it does not validate every manual semantic judgment or prove equivalence on all possible databases. Limited witness discovery remains the larger bottleneck, while semantic authority introduces an additional one-task loss. The recovery of 0094 here does not contradict its persistent error on the different session-agent witness analyzed below. Those 41-witness diagnostics retain their official-single interpretation.

C.2 Provisional Fixed-Witness Diagnostics

Pilot mechanism and semantic-complexity diagnostic. The remaining diagnostics use provenance-limited session-agent witnesses. On the frozen pilot20 pool, corrected official-single public/existing-FTC/oracle scores are 15/20, 17/20, and 19/20. Zero-family search with reference-only post-selection checking reaches 19/20. Reference access here adjudicates selected witnesses and is not part of deployable proposal generation.

On the frozen top-4 witnesses, 19 tasks have a labeled-correct candidate. Fifteen already have correct public selection and require no semantic class decision for that selection objective. The four remaining tasks—0011, 0013, 0016, and 0017—can each be recovered with one local decision among at most three output classes. No task requires two to four decisions; all 19 pool-correct tasks are recoverable in this retrospective diagnostic. A stricter objective, removing every officially-incorrect survivor, requires one decision on eight tasks and zero on the rest. The class-label encoding length does not measure the semantic knowledge needed to interpret the question.

Fresh-context pilot replication. Eight frozen candidate-blind cards were each tested under three class-label/order permutations, giving 24 fresh isolated contexts with one frozen prompt. Anchors had no repository, reference, or candidate access. Reference-class retention was 24/24, wrong reference-class exclusions 0/24, and nontrivial decisions 21/24. Each permutation recovered three of the four public-error tasks and reached 18/20. The unrecovered task reflected retain-all behavior, not a wrong exclusion. These results show partial recovery with no observed reference-class exclusion, not consistent recovery of every hard task or a guarantee about unseen contexts.

Provisional non-pilot80 protocol and blinding boundary. The protocol was frozen before evaluating the 80 non-pilot tasks. One open pool-conditioned witness proposal per task used no semantic failure-family routing. The protocol selected the highest-utility admissible nontrivial witness among available artifacts. Generic schema admissibility is not a proof of every latent data constraint. The anchor saw only the question, schema, tiny world, and anonymous output classes. Candidate programs and their output mapping remained hidden; reference access was unlocked after the response freeze. Forty-one tasks yielded usable witnesses and 39 fell back to public.

Table 25: Provisional session-agent trust diagnostics under official-single labels. Counts of damage and recovery are observed outcomes, not probability bounds.

Diagnostic	Single strong anchor	Union of three
Accuracy	66/80 (.825)	65/80 (.8125)
Public-error tasks recovered	6	5
Public-correct tasks damaged	0	0
Public-to-oracle gap closed	66.7%	55.6%
Candidates eliminated	111	90

Union-of-3 uses three fresh blinded replicates per selected witness and retains any class retained by any replicate; 5/41 unions retain all classes. The 90/111 elimination ratio is 81.1%, not a safety probability. Union converted one observed genuine single-anchor semantic mistake into non-elimination. Two raw reference-class mismatches remaining after union were manually attributed to a witness/reference literal mismatch and a reference SQL/NL mismatch. This post-hoc audit does not prove the other eliminations semantically sound. We do not combine pilot and prospective scores into a new full100 open-witness claim: protocols and adjudication conditions differ.

C.3 Prompt Controls and Semantic Case Audits

Matched-prompt local adjudication protocol. The local model remains `Ministral-3-14B-Instruct-2512`, GGUF Q4_K_M, running with llama.cpp/llama-server on CUDA, greedy decoding, temperature 0, top_k=1, top_p=1, and a fixed seed. Each blinded card receives a stateless request. The first run used the alternate semantic-anchor prompt and reached 62/80 single and 61/80 union, with one union-level damaged public-correct selection, 10/41 retain-all unions, and 11.1% union gap closure. This remains a prompt-sensitivity comparator, not the sole local result.

The matched-prompt replay uses the stronger anchor system's exact frozen prompt verbatim on every card. It changes no task-specific instructions or failure-case rules. The same 80 non-pilot tasks, candidate pool, 41 selected candidate-blind cards, frozen witnesses, $U(c)$ selection, candidate-blind input fields, reference-unlock discipline, and $R_{\text{official-single}}$ labels are retained. Three fresh class-label/order permutations per card yield $41 \times 3 = 123$ stateless decisions; all responses are frozen before reference access. These are permutations of the same 41 cards, not independent task samples. A programmatic integrity audit confirmed agreement with sealed card manifests, absence of candidate SQL, gold SQL, and forbidden labels from anchor inputs, unchanged responses after freeze, and freeze preceding scoring. This checks protocol integrity, not semantic correctness or upstream proposer reproducibility.

Table 26: Matched-prompt Ministral replay under official-single scoring on the 41 frozen session-agent witnesses from the original prospective pool, not the TSA-scored 16-witness named-model composition. M1 is the pre-specified single-anchor replicate, not the best replicate selected after scoring. Reference-class exclusions and damage measure different events.

Diagnostic	Single M1	Union-of-3
Accuracy	63/80	64/80
Public-error tasks recovered	5	4
Public-correct tasks damaged	2	0
Public-to-oracle gap closed	33.3%	44.4%
Retain-all decisions/unions	1/41	9/41

Table 27: Candidate-blind adjudication on the same 41 fixed SQL witnesses under official-single scoring. The rows compare authority choices, not end-to-end search systems.

Anchor bundle	Single	Union-of-3	Union damage	Union gap closed
Provisional	66/80	65/80	0	55.6%
Ministral, alternate prompt	62/80	61/80	1	11.1%
Ministral, matched prompt	63/80	64/80	0	44.4%

M1 makes nine reference-class exclusions; these must not be equated with nine genuine semantic errors. The three single-anchor accuracies are 63/80, 63/80, and 64/80. Union eliminates 87 candidates and reaches $60 + 4 - 0 = 64$ correct selections, while M1 reaches $60 + 5 - 2 = 63$. Relative to the alternate prompt, union improves from 61/80 to 64/80, gap closure from $1/9 = 11.1\%$ to $4/9 = 44.4\%$, observed damage from one to zero, and retain-all unions from 10/41 to 9/41. The result is mixed: prompt matching changes the power–robustness trade-off, but does not remove all systematic semantic errors.

Post-freeze case diagnostics, not method exceptions. Case 0034 asks for names of nations founded after 1950. The alternate-prompt union excludes the reference class through a join-restricted reading. Under the matched prompt, one replicate retains both classes, allowing union to retain the correct/reference class; the task is no longer damaged. Other replicates still exhibit the restrictive reading. Thus prompt choice changes whether this error tendency becomes a hard elimination, rather than uniformly removing the tendency.

Case 0094 asks for the total surface area of Caribbean countries. Its witness has one Caribbean country with two city rows. The intended/reference result counts the country once, yielding 100; Ministral selects the join-fan-out total 200 under both prompts and all three matched-prompt permutations. This is the clearest observed persistent correlated semantic error. It does not establish an effect of quantization, a universal model-size limit, or a shared bias of all possible anchors.

Case 0089 is ambiguity-sensitive: on wording involving "each time," two replicates favor a more specific per-event/per-ship interpretation and one abstains. We treat this as a borderline semantic case, not an independently established genuine semantic error or an inferred union-level failure. Cases 0023 and 0061 belong to different audit categories. In 0023, the reference literal `JetBlue Airways` differs from the witness's `Jetblue Airways`, producing an empty reference class: a witness/reference literal mismatch. In 0061, the question asks for the youngest dog but the reference query omits the dog filter, so a younger cat changes its output: a reference SQL/NL mismatch. These two cases persist across systems and are not counted as evidence of anchor-model semantic failure.

D Code Witness and Adjudication Protocols

The MBPP+29 diagnostic holds the candidate pool fixed: Public reaches 23/29 and the family-routed falsifier and pool oracle both reach 25/29. We report the open-search protocol and frozen-response audit first, then preserve input-contract development and repair experiments as separate supporting records.

D.1 Open-Input Search and Blinded Adjudication

The code replication asks whether open input search can recover the two available errors without the original code failure-family routing, and whether learned adjudication preserves those recoveries. The corrected run (v2) retains the frozen MBPP+29 tasks, candidate pools, public selector and tie-break, split utility $U(c)$, deterministic witness selection, and evaluation protocol. It uses Qwen2.5-7B-Instruct with the same local configuration documented in Appendix F: greedy decoding, seed 20260909, a 1400-token generation cap, and exactly one proposer call per task. Only the defective proposer I/O contract is corrected: inputs use named function parameters in a structured Python-literal dictionary parsed with `ast.literal_eval`, without executing proposer-generated code. This is a protocol-corrected replication after a known interface defect, rather than an unseen holdout.

Stage A: witness-search power. The proposer produces parseable outputs on 28/29 tasks and admissible inputs on 26/29; all 26 have a usable nontrivial witness. After proposals and selected witnesses are frozen, reference adjudication yields 25/29, recovering 2/2 recoverable baseline errors with zero baseline-correct damage. It reaches the pool ceiling and matches the original falsifier (Table 4). Reference adjudication isolates search power; it is not a reference-free selection method.

Stage B: candidate-blind semantic adjudication. The same Ministral-3-14B-Instruct-2512 adjudicator and union-of-3 policy are applied to fresh cards for all 26 witnesses, yielding $26 \times 3 = 78$ fresh blinded adjudications. No v1 responses are reused. Each presentation contains the natural-language task, function signature/specification, concrete input, and anonymous output classes. Candidate programs, identities, output-to-candidate mappings, the reference implementation, hidden tests, and correctness labels are withheld. Responses are frozen before reference scoring.

Stage B reaches 22/29, preserving neither Stage-A recovery and introducing one baseline-correct regression. The union excludes the reference class on nine witnesses; these are class-level exclusions, not nine task-level regressions. Its survivor set agrees with reference-assisted Stage A on 16/26 witnesses. The three permutations are repeated adjudications of each witness, not independent tasks. Thus this small pool supplies second-domain evidence for open witness search, while the tested learned adjudicator fails to preserve the resulting selection gain.

D.2 Frozen-Response Semantic Audit

The analyses below are post-hoc descriptions of the 78 frozen Stage-B responses. No models are rerun, and neither the v2 protocol result nor its interpretation as an unsuccessful learned composition is replaced by an alternative aggregation.

Aggregation sensitivity. Table 28 applies alternative aggregation rules to the same responses. None of the examined rules reaches the Stage-A ceiling of 25/29. This suggests that aggregation choice alone does not explain the loss: the recorded judgments themselves lack sufficient semantic

reliability to preserve the ceiling. M1's 24/29 is the best replicate identified post hoc, not a selected experimental method.

Table 28: Post-hoc aggregation of the same 78 frozen code adjudications. Union-of-3 remains the protocol result; all other rows are descriptive reanalyses, not alternative protocol-level method comparisons.

Aggregation	Accuracy
Union-of-3 (protocol)	22/29
Intersection-of-3	23/29
Majority	22/29
Replicate M1 alone	24/29
Replicate M2 alone	22/29
Replicate M3 alone	22/29

Output shape. The frozen responses retain non-empty collections relatively often and scalar values less often (Table 29). All six empty-collection presentations are reference-correct, representing two tasks across three permutations, yet all are rejected. On this pool, the tested model/prompt/interface tends to reject empty or exceptional-looking behavior. This descriptive pattern neither establishes a general property of language models nor identifies its cause.

Table 29: Post-hoc output-shape retention across code adjudicator presentations, not independent tasks. Counts are given where available; qualitative entries do not imply unreported denominators. All six empty-collection presentations are reference-correct.

Output shape	Observed retention
Non-empty collections	Relatively frequent
Scalar values	Less frequent
`None`	0/36 presentations retained
Empty collections	0/6 presentations retained
Exceptions, timeouts, errors	0 retained

How the two recoveries are lost. For MBPP/2, the selected witness produces candidate outputs `None`, `()`, and `()`, with reference behavior `()`. Stage A separates the mutant from the correct candidates. All three learned adjudications instead reject both available classes, including the correct empty tuple. With no survivors, the selector falls back to the original baseline and loses the recovery. For MBPP/66, the input `[1,-2,0]` separates an implementation counting $x \geq 0$ from those counting $x > 0$: candidate outputs are 2, 1, and 1, and the reference class is 1. The learned union retains both classes, eliminates no candidate, and returns control to the frozen baseline tie-break, which selects incorrectly. The first failure is over-exclusion; the second shows why restraint cannot by itself compensate for insufficient fallback evidence.

The task-level regression. For MBPP/9, `find_Rotations('abab')` produces output classes 1 and 2, with reference behavior 2. The baseline selects correctly, but the learned union retains the wrong class and excludes the reference class, causing the single baseline-correct regression. The witness exposes an informative distinction; the learned interpretation reverses its selection value.

D.3 Protocol Audit: Input-Contract Correction

The initial frozen run (v1) is retained as a protocol-development record, not as evidence about search quality. Its proposer contract simultaneously required strict JSON and Python literals and left positional argument nesting ambiguous. The interface mechanically rejected valid separating inputs, including inputs already generated for a recoverable task. The named-parameter contract in v2 corrects this defect while preserving the remaining experimental settings; the runs are separate records.

D.4 Historical MBPP+29 Selection and Repair

The frozen MBPP+29 baselines reproduce exactly: self-consistency/public selection is 23/29 (.793), the original family-routed selection falsifier reaches 25/29 (.862), and the pool oracle is 25/29. Only

two baseline errors are recoverable by selection. The historical weak end-to-end repair configurations remain at .103. Their logs report no invalid counterexamples and .897 repair-stuck runs, locating a repair bottleneck in those configurations. The .931 stronger-repair value is an oracle-style upper bound, not a deployable measured repair model. These historical configurations are separate from the open-search and learned-adjudication configuration in Section 4.4.

Table 30: Historical code selection and repair on MBPP+29. Selection and weak end-to-end repair are separate configurations; the stronger-repair row is an oracle-style upper bound. The open-search replication is reported separately in Table 4.

MBPP+29 variant	Accuracy	Key interpretation
self-consistency	0.793	strong generation baseline
selection-only falsifier	**0.862**	falsification signal helps selection
FTC no repair	0.103	CE alone does not fix answer
current repair	0.103	repair stagnates
current repair + replay	0.103	replay adds cost but no accuracy
stronger repair upper bound	0.931	evidence is useful if repair is strong

E Resource Accounting

The measurements below describe compute consumption for specified configurations. They do not establish a token-, latency-, or dollar-normalized comparison between spending additional compute on candidates and spending it on evidence.

E.1 Historical Selector and Generation Logs

Inherited judge, MBR+FTC, shuffled-hit, count-aware, adaptive, and localized-selector variants are not compared under $R_{\text{official-single}}$ because their selected outputs were unavailable for relabeling. Corrected comparisons are reported in Section 4.1 and Appendix A.2; the measurements below describe resource use only.

Costs unaffected by relabeling. The logged call, token, latency, and generation measurements below are historical resource measurements for the named configurations. Relabeling correctness does not by itself alter resource consumption, but these costs alone do not establish a cost-adjusted accuracy ranking. Accuracy-versus-cost plots using the old labels are omitted.

Table 31: Cost proxies for stronger judge selectors. Input tokens and latency are selector-side diagnostics, not full dollar-normalized Pareto accounting.

Setting	Selector	Invoke rate	Model calls/task	Input tokens/task	Latency/task
independent mixed	ensemble	0.920	10.02	7,366	23.78s
independent mixed	pairwise	0.920	16.59	13,775	39.35s
public-overfit full100	ensemble	0.980	16.80	11,709	39.62s
public-overfit full100	pairwise	0.980	47.61	34,294	111.83s

Table 32: Natural candidate-pool generation-token accounting. This is not a full selector-side token, latency, or dollar accounting.

Setting	Sampled/Task	Kept/Task	Kept Frac.	Total Gen. Tokens	Avg Tokens/Task
independent first100 mixed, row=2	16.0	4.57	0.286	579,161	5,791.61
official100 mixed, row=2	16.0	4.72	0.295	578,943	5,789.43
public-overfit pilot20, row=2	32.0	5.10	0.159	401,984	20,099.20

Historical relative attack-work ratios for localized tie-only search are .731 on independent mixed, .748 on official100 mixed, and .990 on pilot20. These ratios measure work reduction, not preservation of accuracy under the revised oracle.

E.2 Controlled Allocation and Call Counts

The controlled allocation sweep gives accuracies .000/.182/.932/.977/1.000 at falsification fractions 0/20/40/60/80%. This is policy- and task-dependent and does not establish an optimal universal allocation. One solver, judge, falsifier, or repairer invocation counts as one model call; executions are counted separately. The ratios below favor the judge per model/execute call even though the

end-to-end FTC configuration obtains the highest controlled-bank accuracy. Prompts, batching, and reusable artifacts make these coarse proxies rather than dollar-normalized deployment comparisons.

Table 33: Coarse compute view of the deployable end-to-end SQL configuration on 176 augmented variants from 44 base tasks. Efficiency ratios are proxy diagnostics, not token-, latency-, or dollar-normalized claims.

Method	Acc.	Model Calls	Exec Calls	Correct / Model	Correct / Exec
Single-shot	0.000	1.00	4.00	0.000	0.000
Self-consistency	0.000	3.00	7.00	0.000	0.000
Judge rerank	0.932	3.00	4.00	**0.311**	**0.233**
Execution feedback	0.000	1.00	5.00	0.000	0.000
Random DB test	0.057	1.06	6.00	0.054	0.010
FTC	**1.000**	4.00	6.09	0.250	0.164

F Model Provenance

Named zero-family proposer. The named-model replication uses Qwen2.5-7B-Instruct, loaded directly from the local Hugging Face snapshot `Qwen2.5-7B-Instruct` with `local_files_only`. The backend is bitsandbytes NF4 4-bit with double quantization, transformers 5.16.1, and torch 2.11.0+cu128 on an RTX 5070 with 12 GB memory. Decoding is greedy with `do_sample=False`, fixed seed `20260909`, and `max_new_tokens=1400`; exactly one proposer call is made per task. The prior prospective protocol's frozen zero-family prompt is used verbatim, with a per-task SHA-256 recorded. All proposals, selected witnesses, selection manifests, and hashes are frozen before reference access. The experiment `experiments/pcftc_named_proposer_v1` is a frozen-protocol replication on the already studied 80-task pool, not a new prospective holdout.

Named local semantic anchor. Stage B uses `Ministral-3-14B-Instruct-2512`, GGUF Q4_K_M, with llama.cpp and greedy decoding. The exact frozen matched anchor prompt is applied to three fresh blinded class-label/order permutations per selected Qwen witness; union-of-3 aggregates the responses. All $16 \times 3 = 48$ responses are frozen before reference scoring. Together with the specified Qwen proposer, this is a fully local zero-family composition, not merely downstream replay of unidentified proposals.

The separate 41-card Ministral replay uses llama.cpp/llama-server on CUDA, greedy decoding, temperature 0, top_k=1, top_p=1, a fixed seed, and one stateless request per card permutation. Its 123 decisions use the stronger system's frozen prompt uniformly and are frozen before reference access. The integrity audit checks sealed card manifests, blinded inputs, frozen-response immutability, and freeze-before-scoring order. This earlier replay does not specify or rerun its session-agent proposer and is not a fully controlled cross-model comparison.

Code replication. The protocol-corrected code replication in Appendix D uses the same local Qwen proposer and Ministral adjudicator configurations specified above. Its corrected proposer contract supplies named function arguments; the code adjudicator receives task specifications, inputs, and anonymous output classes. These code inputs are distinct from the SQL prompts. The 26 selected code witnesses receive 78 fresh blinded adjudications, separate from both the 48-decision SQL composition and the 123-decision SQL replay. The code v1/v2 labels refer to the initial and corrected input contracts, not to the SQL experiment directory above.

Stronger provisional components. The stronger zero-family proposer and stronger semantic anchor use a session-agent execution route whose stable model/backend identity remains unavailable. Their higher-coverage results remain provisional mechanism diagnostics, not headline model comparisons. The execution route does not identify a provider or model, and frozen outputs alone cannot establish reproducibility of these inference steps. The named replication supplies independent local evidence for the search mechanism without resolving this separate provenance limitation.

Table 34: Model identities and provenance boundaries. The named replication specifies both local proposer and anchor; stronger session-agent results remain provisional.

Component	Provenance status
Named zero-family proposer	Qwen2.5-7B-Instruct, local HF snapshot, NF4 4-bit with double quantization; specified transformers/torch backend, greedy decoding, fixed seed, frozen prompt, 1400-token cap, one call/task. Fully specified local replication.
Named local semantic anchor	Ministral-3-14B-Instruct-2512, GGUF Q4_K_M, llama.cpp, greedy decoding; matched frozen prompt and union-of-3 over fresh blinded permutations. Stage B: 48 calls on 16 Qwen witnesses; separate replay: 123 calls on 41 session-agent witnesses.
Code proposer and adjudicator	Same specified local Qwen and Ministral configurations; corrected named-argument input contract and code-specific blinded cards. 78 fresh adjudications on 26 code witnesses; protocol union-of-3.
Stronger zero-family proposer	Session-agent execution route; stable model/backend identity unresolved; provisional.
Stronger semantic anchor	Session-agent execution route; stable model/backend identity unresolved; provisional.
Historical components	Documented natural-pool generators: llama3:8b, qwen2.5:7b, qwen2.5-coder:7b. Full historical solver/judge/falsifier/repairer model and backend identities are not established here.