ZERO-TRAINING FRACTURE ATTRIBUTION VIA DIRECTIONAL PROVENANCE DIFFERENCING

Anonymous authors
Paper under double-blind review

ABSTRACT

An actionable agent diagnosis must identify which write to change, the evidence for changing it, and the downstream work to replay. We introduce a zero-attribution-training framework that connects these objects through versioned artifact provenance. A bounded, directional local carrier compares fixed defect anchors against a write and its pooled parents, separating wrong-content introduction from required-content loss; confidence-aware pooling preserves anchor budgets at merges. Output ancestry determines relevance, while optional field contracts distinguish legitimate projection from deletion. On 226 executable failures with unbalanced optimal transport (UOT) as the carrier, restricting scoring to output ancestors raises exact-write hit1 from 11.5% to $\mathbf{50.9}\%$, and replacing unsigned with directional local change adds a further $+\mathbf{21.2}$ points (paired 95% workflow-cluster interval $[16.3, 26.9]$). With supplied contracts, an exact lexical/typed carrier reaches 100% from 76.1%, and typed UOT gains likewise. Executed replay separates localization, repair, and regeneration cost; fixed missing-information experiments measure non-target selection, abstention, and replay-closure errors. Observation-equivalence ceilings separate what dependency structure alone can identify from what frozen defect anchors add. The result is an inspectable interface from defect evidence to testable interventions, with explicit observation requirements and no trained attribution predictor.

1 INTRODUCTION

A failed agent workflow presents two questions: *what is wrong with the result*, and *which operation should change*? A critic may answer the first without the second. Later code can faithfully implement a faulty design, so changing the final patch leaves the source intact; replaying everything discards useful work. The actionable target is the write to inspect, regenerate, or revert, together with its dependent artifacts.

Trajectory attribution asks which agent or step is associated with a failure; fracture attribution asks which persisted artifact transition changed the evidence needed to produce the observed output. An agent step is a temporal event, whereas a write creates a versioned state that can be inspected, regenerated, or reverted. A faulty decision may appear in one step, be copied by later steps, and remain visible only in a descendant artifact. Conversely, a step may be on the failure trajectory while merely projecting or relaying information without introducing the defect. Our intervention unit is therefore the write whose payload changes across dependency boundaries, together with the descendants that must be replayed after that write is modified.

Figure 1 makes the intervention unit concrete: a write, its input versions, and the content whose support changed. Two structural problems follow. First, chronological adjacency is not dependency: an intervening test-plan write need not replace a design or its implementation, and workflows fork and merge. Second, a content match is not a transition diagnosis: correct content upstream does not locate its loss, and wrong content downstream may have been copied rather than introduced. Attribution therefore needs both dependencies and directional content change.

Learned trajectory tracers offer one route to this problem. AgentTracer (Zhang et al., 2025a) trains an agent-and-step predictor from injected faults using GRPO (Shao et al., 2024). New pipelines can change their roles, tools, and artifact schemas before such experience is available. We study a com-

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.

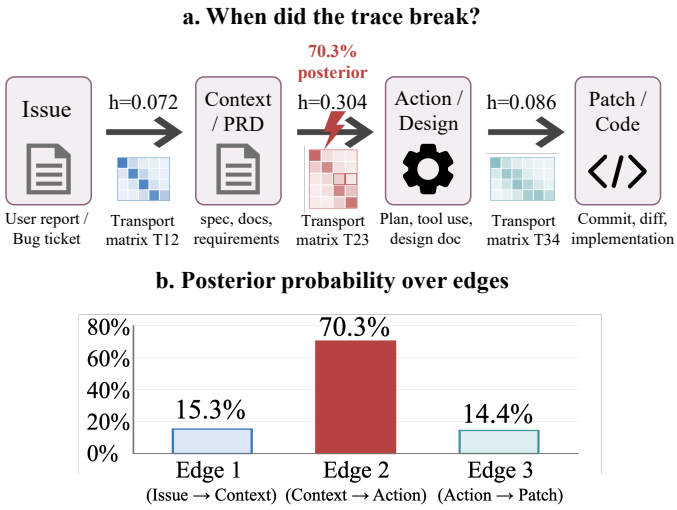


Figure 1: Motivating diagnostic interface on a path through an artifact workflow. The highlighted context-to-action transition identifies a write to inspect instead of naming only an agent. In the general provenance graph, one write can consume several parents.

plementary operating point: *zero attribution training*, with an explicit computation over observed writes and optional pretrained models for semantic evidence.

Our method follows a dependency-driven interface. We first materialize tool writes, bind reads to their observed resource versions, and preserve narrative as low-confidence dependency evidence. We then compare fixed defect anchors against each write and its pooled parents using a local carrier, and retain separate increases for wrong content and decreases for required content. Finally, a ranked write and its descendants are passed to replay so that localization and intervention outcome can be evaluated separately. The carrier can be UOT, cosine, or exact lexical/typed matching; UOT provides the primary bounded coupling used in the main directional comparison.

A no-reference variant obtains anchors from a compact judge that sees only the task and final artifact. Reference-based word differences provide a separate, deterministic diagnostic proxy.

Our contributions are:

1. **An inspectable intervention interface.** Versioned writes, output ancestry, and optional field obligations connect a defect to an editable operation and its replay descendants. Narrative supplies evidence without becoming an intervention target.
2. **A bounded local fracture diagnostic.** Fixed anchor identities, a separate supply cap, and confidence-aware parent pooling support distinct introduction and loss scores. We establish boundedness, raw-score locality, and conditional exact-carrier localization, with explicit abstention.
3. **Evidence that separates the interface's mechanisms.** Direct slicing and local-difference controls isolate scope, direction, and responsibility. Executed interventions measure repair separately from localization; controlled observation corruption and observation-equivalence ceilings expose the information required by the interface.

2 Related Work

Failure attribution in agent trajectories. Multi-agent frameworks such as AutoGen and MetaGPT make intermediate roles and messages explicit (Wu et al., 2023; Hong et al., 2024), creating a natural substrate for diagnosis but not fixing its unit of attribution. AgentTracer (Zhang et al., 2025a) learns agent- and step-level labels from counterfactual replay and thousands of injected faults. Automated failure-attribution work uses LLM judges to identify a responsible agent and time (Zhang

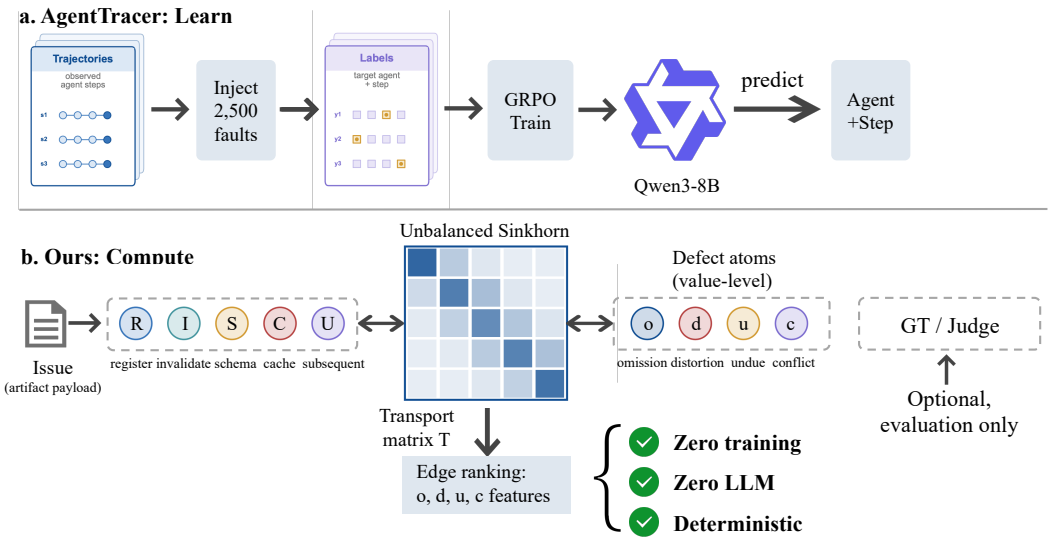


Figure 2: Learned attribution versus per-trace computation. The top depicts training an agent-and-step predictor from labeled injections; the bottom depicts the motivating explicit-value transport pipeline, whose "zero LLM" label applies to that regime.

et al., 2025b); TRAIL organizes evidence for issue localization in long traces (Deshpande et al., 2025); AgentTrace represents deployed systems as causal graphs (Wang, 2026); and LLM-based root-cause analysis maps operational evidence to likely causes (Roy et al., 2024). Our target is not the participant or dialogue event associated with failure, but the artifact transition whose payload can be inspected and regenerated.

Slicing and provenance. Program slicing identifies statements relevant to a computation (Weiser, 1984); data provenance traces an output to its contributing inputs (Buneman et al., 2001). Our backward slice operates on observed artifact writes. It supplies output scope, while directional support and field obligations distinguish a defect-producing transition from a relevant but legitimate one. Direct slice and slice-plus-difference controls test this increment.

Feedback, process supervision, and credit assignment. Self-Refine, CRITIC, and Reflexion produce feedback or verbal experience for subsequent attempts (Madaan et al., 2023; Gou et al., 2024; Shinn et al., 2024); ReAct interleaves reasoning and actions (Yao et al., 2023); verifiers and textual gradients supervise reasoning or translate evaluations into updates (Cobbe et al., 2021; Yuksekgonul et al., 2024). Feature attribution distributes a prediction's credit over inputs (Lundberg & Lee, 2017; Sundararajan et al., 2017); temporal-difference learning and return decomposition propagate delayed outcomes (Sutton, 1988; Arjona-Medina et al., 2019); multi-agent credit reassignment allocates shared rewards across language agents (He et al., 2025). Fracture attribution instead routes feedback to a persisted write, using fixed requirement/defect content rather than a scalar return or verbal critique.

Optimal transport. OT couples non-identical supports (Frogner et al., 2015; Peyré & Cuturi, 2019); UOT relaxes mass conservation to avoid forced matches (Chizat et al., 2018). Sentence embeddings supply soft costs (Reimers & Gurevych, 2019), while exact lexical and typed predicates provide controls. An optional language-model judge (Zheng et al., 2023) diagnoses the defect and supplies anchors for the no-reference regime.

3 METHOD

Figure 3 gives the conceptual interface: defect evidence is anchored at the output, read-time provenance defines the dependency context, directional local comparison identifies the fracture, and replay tests the resulting intervention.

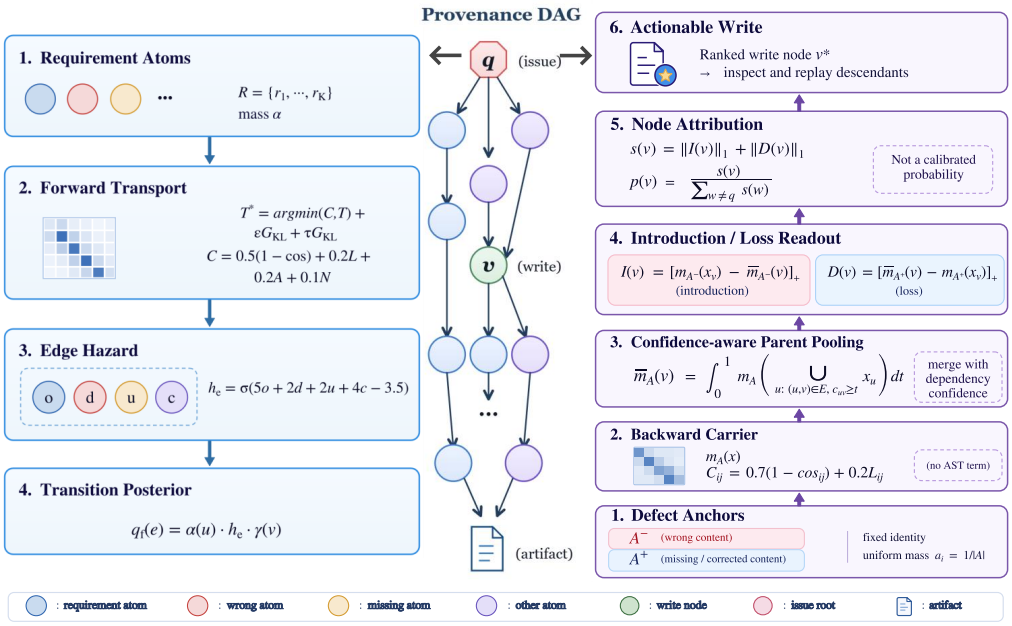


Figure 3: Conceptual overview of the bidirectional attribution schematic over the artifact-write provenance DAG. **Forward chain** (left): requirement atoms $R = \{r_1, \ldots, r_K\}$ (mass α) are propagated from the issue through the unbalanced-transport carrier, and an edge hazard/posterior ranks transitions. **Backward chain** (right): defect anchors d (from a reference or the no-reference judge) are transported in reverse to score each write's introduction of novel wrong content and loss of required content (the two-axis drop/intro readout), ranking actionable writes.

3.1 Fracture Attribution Interface

The intervention unit is a write together with its input versions and replay descendants. A write may change the artifact state that later operations consume, so attribution is defined over transitions between versioned states rather than isolated dialogue events. The provenance context determines which upstream states are compared, while the local fracture score determines how evidence changes across the transition.

The interface separates three operations. **Construct provenance:** materialize tool writes and bind reads to their observed resource versions. **Measure a local fracture:** compare fixed defect anchors against a write and its pooled parents. **Verify an intervention:** pass a ranked write and its descendants to a replay system and test the resulting artifact. This decomposition lets scope, local change, and intervention outcome be evaluated independently.

3.2 Artifact Writes and Read-Time Provenance

Let $G = (V, E)$ contain an issue root q and one node per substantive tool write. A write node v records its agent, command index, resource read set R_v, write set W_v, observed payload, and reconstructed output state x_v. Nodes are ordered by integer step and within-step command order. The *write node*, rather than each incoming dependency edge, is one candidate intervention: a merge with three parents is still one operation to regenerate. Dependency edges identify the inputs needed to reproduce that operation.

Temporal order alone is insufficient to recover the dependency structure of an artifact workflow. A write depends on the versions actually observed when it executed, not necessarily on the most recent versions at the end of the step. We therefore bind every resource read to the latest preceding producer at read time and use these versioned producers as the write's parents. Read–modify–write operations also consume the previous version of their destination; a complete overwrite does not

imply this dependency. Multiple reads induce a merge, and repeated consumption of a produced resource induces a fork.

Explicit read dependencies have confidence 1. References to known artifact paths in the writing agent's own narrative provide supplementary confidence-0.5 evidence. Narrative and read-only events remain metadata. An unresolved external read attaches to q; absent explicit read evidence, a structural issue-root edge anchors the write.

We reconstruct states by applying supported full writes, appends, line insertions, and validated replacements to resource versions. Ambiguous edits or unknown base files retain $x_v = \perp$, distinct from a known empty file. Their observed payload can support introduction evidence, but not whole-file omission. The selected output is the last code-resource write, falling back to the last nonempty payload write. Evaluation retains all writes through that command, including independent branches, and discards later events. Appendix B specifies parsing and state rules.

3.3 Defect Anchors

Two fixed sets describe the observed failure: A^+ contains required or corrected content missing from the final artifact; A^- contains erroneous content present in it. Their identities remain fixed across the graph. A reference-based diagnostic takes word-set differences between the reference and the reconstructed artifact.

In the no-reference regime, a defect judge receives the task and complete final artifact. It returns separate JSON strings `wrong` and `missing`. A nonempty `wrong` field is an exact candidate quote; `missing` describes corrected behavior. Empty fields are allowed, including an empty pair when no defect is detected.

For real traces, the implementation extracts at most 80 alphabetic content-word atoms per field using the same deterministic tokenizer in every matching ablation. Structured values and executable fault labels are evaluated separately. Every nonempty axis has uniform mass $a_i = 1/|A|$; an absent axis contributes zero.

3.4 Directional Local Fracture

A single magnitude of local change cannot distinguish two opposite transitions: a write may introduce support for content that should be absent, or remove support for content that should be present. We therefore compare fixed defect anchors across a write and its dependency context, retaining positive increases for wrong content and positive decreases for required content.

For anchor texts $A = \{\alpha_i\}_{i=1}^{|A|}$, let b_x be the capacity of the distinct tokens in carrier text x. Token relevance is its maximum cosine similarity to an anchor; we retain at most 40 tokens with relevance at least 0.4 and normalize their relevance to obtain b_x. A locally cached MiniLM encoder supplies similarities. The primary carrier cost is

$$C_{ij} = 0.7(1 - \cos_{ij}) + 0.2L_{ij}, \tag{1}$$

where L is the fixed character-overlap penalty specified in Appendix B. We solve product-reference unbalanced transport

$$\begin{aligned} T^\star = \arg\min_{T \geq 0} \langle C, T \rangle + \varepsilon\, \mathrm{GKL}(T \| a \otimes b_x) \\ + \tau\, \mathrm{GKL}(T\mathbf{1} \| a) + \tau\, \mathrm{GKL}(T^\top \mathbf{1} \| b_x), \end{aligned} \tag{2}$$

with $\varepsilon = 0.05$ and $\tau = 0.5$. The entropy reference is part of the objective: the scaling kernel is $(a \otimes b_x) \odot \exp(-C/\varepsilon)$.

A separate row cap produces the diagnostic plan

$$\widehat{T}_{ij} = T^\star_{ij} \min\left(1, \frac{a_i}{\sum_j T^\star_{ij}}\right), \qquad m_A(x) = \widehat{T}\mathbf{1}, \qquad 0 \leq m_A(x) \leq a. \tag{3}$$

Zero-mass rows and columns are excluded exactly, and empty token support gives $m_A(x) = 0$. Row marginals preserve anchor identity.

The carrier determines how local support is measured. UOT provides a bounded coupling across multiple anchors and non-identical token support; cosine provides independent similarity; exact lexical inclusion tests whether an anchor's tokens occur in the carrier. We compare these representations under the same provenance scope and directional readout.

3.5 Local Introduction and Loss on a DAG

A write may read several upstream artifacts. Allocating a full anchor budget to each parent would multiply evidence at merges. Instead, let c_{uv} denote dependency confidence and pool parent text before computing carrier mass:

$$\overline{m}_A(v) = \int_0^1 m_A\left(\bigcup_{u:(u,v)\in E,\ c_{uv}\geq t} x_u\right) dt. \tag{4}$$

The union denotes concatenation followed by token deduplication. The integral is evaluated exactly over the finitely many confidence levels; an empty parent set contributes zero. It integrates nested supports rather than independently summing parent plans, so $0 \leq \overline{m}_A(v) \leq a$. The root supplies issue text; an unknown parent contributes its observed payload as partial evidence.

With $[z]_+ = \max(z, 0)$ componentwise, the two directional changes are

$$I(v) = [m_{A^-}(x_v) - \overline{m}_{A^-}(v)]_+, \tag{5}$$
$$D(v) = [\overline{m}_{A^+}(v) - m_{A^+}(x_v)]_+, \tag{6}$$
$$s(v) = \|I(v)\|_1 + \|D(v)\|_1, \tag{7}$$
$$p(v) = s(v) \Big/ \sum_{w\neq q} s(w). \tag{8}$$

I measures new support for observed wrong content; D measures lost support for required content. Increased support for required content and decreased support for wrong content do not contribute to the fracture score. Set $D(v) = 0$ whenever the reconstructed output state is unknown. If all scores are zero, the method abstains; otherwise p is a normalized attribution weight.

Both axes reuse the same carrier operator without recursive path attenuation. Adding a write that changes neither an existing node nor its parents leaves that node's raw score unchanged; new positive-score candidates can still change normalized weights and rankings. Appendix A proves boundedness, locality, and a conditional exact-carrier localization result.

3.6 Output Applicability and Field Responsibility

A read dependency does not oblige every child to reproduce every requirement. For selected artifact y, let $\mathcal{C}(y)$ contain y and its non-root ancestors through explicit or narrative dependencies. The *artifact* variant uses

$$s_y(v) = \mathbf{1}\{v \in \mathcal{C}(y)\}s(v). \tag{9}$$

It retains every candidate and the original tie rule but changes score eligibility. A separate *observed-loss* variant pools only known non-root explicit/narrative parents for D, leaving I unchanged. We evaluate both switches and their combination (Appendix F).

When a structured write has an externally supplied set of required fields K_v, its loss coordinate becomes

$$D_i^K(v) = \mathbf{1}\{k_i \in K_v\}D_i^{\mathrm{O}}(v) \tag{10}$$

for anchor key k_i. Introduction and anchor budgets are unchanged. An unspecified obligation leaves loss ungated; an explicitly empty one permits no field loss. Multiplication by a binary gate preserves the carrier bounds.

Table 1: Evidence map. Each question is tested against one ground-truth source.

Question	Ground truth	Metric	Controls
Scope, direction	Injected faults	Hit1	Slice, unsigned change
Responsibility	Injected faults; probe	Hit1; FP/Del	Contract off
Repair and cost	Executed replay	Pass, W	Slice-first, uniform
Information needs	Ceilings; corruption	Ceiling; N, A, C	Intact
Natural traces	Blinded review; labels	Agreement	First write, write-ID ranker

3.7 Intervention and Replay

A ranked node identifies the payload to inspect and descendants to replay. Atom-level I, D expose its evidence; re-execution and a verifier establish whether changing that write restores the resulting artifact. Localization and repair therefore remain separate properties of the interface.

4 Experiments

4.1 Experimental Setup

Our evaluation asks whether provenance identifies the relevant scope, whether directional change identifies the responsible transition, whether field responsibility resolves projection versus deletion, whether localization translates into repair, and which observations the interface requires.

Executable cohorts. Sixty self-authored workflows propagate a four-field policy through PRDs, forked configurations, a merge, and Python programs; three program families implement bonus, tariff, and allocation rules, checked by an independent arithmetic oracle. Random topological orderings retain unrelated and post-artifact writes, and each workflow receives three to five faults at distinct eligible writes: threshold changes, condition flips, or offset deletion. The development cohort has 246 attempts (222 clean-pass/injected-fail pairs); a separate seed gives 244 validation attempts and 226 valid pairs on 60 new workflows, Appendix E gives the generator and fault protocol.

Common observations. Predictors receive the failed graph and fixed reference/final-artifact anchors. Raw carriers use token differences; typed carriers use safely parsed key/value declarations, and field obligations are exported from the generator without expected values or fault labels. An all-zero score abstains, and ordinary top-3 includes zero-score tails.

Fair contrasts. The frozen offline protocol compares provenance backward slicing, slicing plus unsigned local changes, and slicing plus directional introduction/loss. Loss uses the same observed-parent rule on both sides. Intervals are pointwise descriptive 95% paired bootstrap intervals from $2{,}000$ source-workflow resamples.

4.2 Does Provenance Identify the Relevant Scope?

A provenance slice identifies writes that can reach the selected artifact; it does not choose which transition introduced the defect. We compare first-in-slice, nearest-to-output, and uniform-selection policies with local-change scores on that same slice. The unsigned control uses absolute anchor-support changes; the directional control retains wrong-content increases and correct-content decreases.

The intact slice contains every injected write, while first and nearest selection provide no exact-write hits. Provenance parents score 14.0% against the chain baseline's 5.0% on development hit1, a $+9.0$-point difference ($[4.1, 14.3]$). Artifact eligibility raises the original UOT hit1 from 11.5% to 50.9% on validation ($+39.4$ points, interval $[31.2, 48.1]$) and raises every other scoped carrier as well: raw lexical 41.2% to 64.2%, typed lexical/exact 58.8% to 76.1%, cosine 10.6% to 23.9%, and typed UOT 0.4% to 17.7% (Appendix G). Provenance therefore supplies the relevance scope in which local attribution operates.

Table 2: Direct framework controls on $222/226$ development/validation failures. Rows are grouped by carrier; within a block, All precedes artifact scope (A), unsigned (U) or directional (D) change, and supplied field obligations (K). Scope, direction, and contracts read top to bottom. All rows except Raw UOT All share output-ancestor scope. W is mean replayed writes. Bold marks within-carrier direction or contract gains.

	Hit1 (%)		Validation		
Policy / carrier	Dev.	Valid.	Top-3 (%)	Pass (%)	W
Slice first	0.0	0.0	17.7	100.0	14.51
Slice nearest	0.0	0.0	17.7	0.0	1.00
Slice uniform	8.0	7.5	22.6	50.7	7.25
Raw UOT All	14.0	11.5	20.4	11.5	1.73
Raw UOT A + U	27.0	29.6	89.4	60.2	7.29
Raw UOT A + D	**54.5**	**50.9**	95.1	72.6	7.34
Raw lexical A + U/D	66.2	64.2	95.1	82.3	7.58
Typed UOT A + U/D	16.7	17.7	42.9	52.2	7.83
Typed UOT A + D + K	41.0	41.6	41.6	41.6	3.09
Typed lexical/exact A + U/D	75.7	76.1	100.0	88.1	7.58
Typed lexical/exact A + D + K	**100.0**	**100.0**	100.0	100.0	7.16

4.3 Does Direction Identify the Fracture?

Holding artifact scope and the carrier fixed, replacing unsigned changes with directional introduction/loss raises UOT hit1 from 29.6% to 50.9%: $+21.2$ points, interval $[16.3, 26.9]$. The corresponding development gain is $+27.5$ points ($[20.2, 34.6]$). The directional coordinates separate wrong-content introduction from required-content loss, whereas unsigned change combines correction and defect evidence into one magnitude.

The directional gain is specific to raw UOT in the aggregate table: raw lexical, typed lexical/exact, and typed UOT retain the same aggregate metrics under unsigned and directional scoring. Lexical/exact carriers already encode much of the relevant distinction directly, while UOT exposes the distinction through changes in coupled anchor support.

4.4 Does Field Responsibility Resolve Projection versus Deletion?

A branch can belong to the output slice without being responsible for every field. With declared obligations, typed lexical/exact hit1 rises from 76.1% to 100% on validation, a $+23.9$-point paired gain ($[19.2, 28.6]$), and from 75.7% to 100% on development. The contract therefore adds responsibility information beyond slicing and typed local differences. Typed UOT with the same gate obtains 41.6% hit1 and 58.4% abstention; its unchanged-key representation does not identify the value changes.

A six-workflow mechanism probe corroborates this at finer grain: declared responsibility removes all 18 measured legitimate-projection losses while preserving all six genuine deletion signals. Full counts and repair outcomes are in Appendix F.4.

4.5 Does Localization Translate into Repair?

Localization and repair evaluate different properties of an attribution method. A broad slice can contain the correct write even when its ranking does not localize the intervention, while a localized write may still fail to restore the required output. We therefore execute the selected intervention and regenerate its descendants, measuring repair success separately from exact-write identification and regenerated work.

The archived studies execute every candidate intervention in both randomized cohorts: $5{,}376$ executions in development including noise checks, and $4{,}373$ in validation. Regeneration uses the forward generator without repeating the injection; non-descendants remain byte-identical. Reusing

these outcomes, directional UOT repairs 72.6% at 7.34 regenerated writes on validation, while exact typed attribution with declared contracts repairs 100% at 7.16 writes. Slice-first also repairs 100% but regenerates 14.51 and never identifies the injected write. Localization and repair therefore provide distinct operating points for the same intervention interface.

4.6 What Observations Are Required?

Identification depends on which observations are available. The 466 valid controlled failures comprise 222 development, 226 validation, and 18 field-responsibility cases. Grouping these by observation equivalence, a predictor restricted to dependency structure (node roles, order, and edge kinds, without payloads or anchors) faces a 28.3% exact-write hit1 ceiling; adding the frozen defect anchors raises this to 64.8%. Each ceiling is a finite-data combinatorial identity over the archived cohorts, bounding predictors restricted to the stated observation (Appendix I).

A fixed plan corrupts 10% or 30% of dependencies, states, or contracts on the validation and six-workflow cohorts. We delete observed edges, add acyclic spurious edges, remove state and payload evidence, omit required-field sets, or toggle one declared field. Candidate identities and defect anchors remain fixed.

Under five representative corruption conditions (intact, missing/spurious edges, unknown states; full eleven-condition, four-carrier table in Appendix H), missing edges exclude the true fault from the observed slice in $134/226$ cases at 10% and $193/226$ at 30%. Spurious edges leave exact-contract localization at 100% while corrupting replay closures in 4.0% and 8.0% of cases. State loss changes both selections and abstentions; missing/wrong obligations shift typed-exact hit1 by at most 6.6 points at either rate and leave raw ungated UOT unchanged. A separate stability certificate bounds when a positive top score is robust to edge-confidence perturbation; details are in Appendix I.

4.7 How Does the Evidence Carrier Affect Attribution?

The attribution operation compares fixed defect anchors across a write and its dependency context; the carrier determines how that local support is represented. Lexical carriers reach 64.2% hit1 against raw UOT's 50.9% on validation, and typed lexical/exact with contracts reaches 100.0% against typed UOT's 41.6%. Exact carriers therefore provide stronger matching when the evidence admits exact lexical or typed overlap, while UOT supplies a soft coupled representation for non-identical support. Carrier choice is thus a representation decision within the same provenance-scoped directional interface.

4.8 Natural Traces

We additionally parse 127 released TracerTraj coding records (Zhang et al., 2025a) and review a fixed observation-only sample of 12 across three write-count strata. Two isolated agent-assisted reviews agree on all 12 defect/attribution statuses and identify a specific target for all five supported cases. Nearest-write alignment yields 110 aligned targets over the 127 released records; first-write gives 61.8% aligned hit1. In the no-reference regime, UOT and cosine each obtain $46.4/66.4\%$ aligned hit1/top-3, against $48.2/52.7\%$ for the trace-level LLM ranker control. Complete review and alignment protocols are in Appendix J and Appendix C.

5 Conclusion

Fracture attribution turns defect evidence into an inspectable write and replay plan without attribution training. Versioned provenance defines output relevance, bounded directional comparison quantifies support changes, and declared obligations distinguish projection from deletion. On 226 executable failures, output scope and direction raise UOT's exact-write hit1 from 11.5% to 50.9%, while declared obligations with an exact typed carrier reach 100% hit1 and repair at 7.16 regenerated writes rather than slice-first's 14.51. Executed replay then measures whether a localized write produces the desired artifact, closing the interface from diagnosis to intervention.

REPRODUCIBILITY STATEMENT

Code, prompts, parsers, frozen evaluation plans, review packets, and per-case outcomes are provided as supplementary artifacts. The main paper specifies the attribution algorithm and evaluation protocol, while the appendix contains implementation details, proofs, ablations, and observation-corruption protocols.

AI USE STATEMENT

Generative AI tools assisted manuscript and code editing, defect judging, ranking, and the two isolated evidence reviews with subsequent adjudication. The review pilot is agent-assisted; the later human confirmation is author-reported. Artifacts distinguish model judgments, local scores, and executed outcomes. An AI writing assistant was also used to review methodological framing and existing ablation results and to draft revised wording for the title, abstract, Related Work, Method, and Results; all adopted interpretations and wording were reviewed by the authors.

ETHICS STATEMENT

The natural-trace evaluation uses released research data. Trace payloads are treated as untrusted data and are not executed. Only bounded, self-authored program templates execute in the controlled experiment. Diagnostic rankings do not authorize deployment of an unverified repair, and API credentials are excluded from code, logs, and manuscript artifacts.

REFERENCES

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *International Conference on Database Theory (ICDT)*, pp. 316–330, 2001.

Lénaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. arXiv:2110.14168.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. TRAIL: Trace reasoning and agentic issue localization, 2025. arXiv:2505.08638.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yang Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhitao He, Zijun Liu, Peng Li, Yi R Fung, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. Advancing language multi-agent learning with credit re-assignment for interactive environment generalization, 2025. arXiv:2502.14496.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. Exploring llm-based agents for root cause analysis. In *Companion Proceedings of the ACM International Conference on the Foundations of Software Engineering (FSE)*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Zhaohui Geoffrey Wang. AgentTrace: Causal graph tracing for root cause analysis in deployed multi-agent systems, 2026. ICLR 2026 Workshop on Agents in the Wild; arXiv:2603.14688.

Mark Weiser. Program slicing. *IEEE Transactions on Software Engineering*, SE-10(4):352–357, 1984.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework, 2023. arXiv:2308.08155.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text, 2024. arXiv:2406.07496.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems?, 2025a. arXiv:2509.03312; numbers cited: step attribution 45.35→72.95 (w/ G, Qwen3-8B + GRPO), 4.8%–14.2% repair-rate gains from attributed feedback (Fig. 3).

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems, 2025b. arXiv:2505.00212.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Appendix

A Transport Objective and Formal Properties

Role: theory and derivations. This section supports the primary results and makes no accuracy claim.

A.1 Product-reference generalized Sinkhorn

For nonnegative arrays of identical shape, define

$$\mathrm{GKL}(x\|y) = \sum_i \left[x_i \log(x_i/y_i) - x_i + y_i\right]. \tag{11}$$

We use $0\log(0/y) = 0$ and positive mass against a zero reference has infinite divergence. Thus the support of a feasible transport plan is restricted to positive entries of $a \otimes b$. On this support define $K_{ij} = a_i b_j \exp(-C_{ij}/\varepsilon)$ and $\rho = \tau/(\tau+\varepsilon)$. The scaling updates are

$$u \leftarrow (a/(Kv))^{\rho}, \qquad v \leftarrow \left(b/(K^{\top}u)\right)^{\rho}, \qquad T = \mathrm{diag}(u)K\,\mathrm{diag}(v). \tag{12}$$

All divisions and powers are elementwise. The implementation operates in log space, stopping when the maximum change of both log scalings is below 10^{-8} or after 500 iterations. It embeds the resulting active-support plan into an exactly zero matrix on inactive rows and columns.

The generalized scaling follows unbalanced transport (Chizat et al., 2018). To derive the product-reference kernel, differentiate the objective of Equation 2 at a positive entry:

$$C_{ij} + \varepsilon \log \frac{T_{ij}}{a_i b_j} + \tau \log \frac{(T\mathbf{1})_i}{a_i} + \tau \log \frac{(T^{\top}\mathbf{1})_j}{b_j} = 0. \tag{13}$$

Writing $T = \mathrm{diag}(u)K\,\mathrm{diag}(v)$ and rearranging gives Equation 12. Omitting $a_i b_j$ from K changes the objective: the entropy correction contains row- and column-dependent linear terms, which are not constant when marginals are relaxed. This distinction is numerically tested rather than absorbed into a normalization convention.

A.2 Raw optimum versus diagnostic supply cap

The raw optimizer may transport more than the source mass on a row. We therefore use the separately capped plan in Equation 3 for diagnostic readouts. The implementation exposes `cap_rows=False` to test the raw optimum independently. It does not describe the capped plan as the solution to an additional inequality-constrained UOT problem.

Proposition 1 (Carrier and score bounds)**.** *For one axis with supply vector a, both the capped carrier $m_A(x)$ and confidence-pooled parent carrier $\overline{m}_A(v)$ lie componentwise in $[0, a]$. If each nonempty axis has unit mass, $0 \le s(v) \le 2$; if one axis is absent, $s(v) \le 1$.*

Proof. Nonnegativity of T and the row cap imply $0 \le m_A(x)_i \le a_i$. The parent carrier is an integral over an interval of length one of vectors in this same box, including zero when the parent support is empty. Positive parts of differences between two such vectors are bounded by a. Summing their coordinates and then the two axes proves the result. □

The bound applies to a *single candidate*. Several writes can independently introduce the same wrong content, or lose it and reintroduce it. There is no assumption that the sum of all candidate scores is at most one. Normalization provides inspection weights; calibration against intervention success would require a separate experiment. Numerically, stored weights are zero if total raw score is at most 10^{-12}; ranking uses raw scores and abstains when all are exactly zero.

A.3 Pooling multiple parents without multiplying budgets

Let the distinct positive dependency confidences be $0 < c_1 < \cdots < c_L \leq 1$, with $c_0 = 0$, and let X_ℓ contain the carriers of parents with confidence at least c_ℓ. The exact finite implementation of Equation 4 is

$$\overline{m}_A(v) = \sum_{\ell=1}^{L} (c_\ell - c_{\ell-1}) m_A(X_\ell). \tag{14}$$

The omitted interval $(c_L, 1]$ has empty support and contributes zero. All parents at a level are pooled *before* token support selection, capacity normalization, and transport. This preserves one source budget per level, even at a merge. Identical parent texts with the same confidence add no new distinct tokens and therefore do not duplicate mass. Adding different text can change relative capacities and UOT scores; no general monotonicity claim is made for this operation.

A.4 Locality, copy invariance, and what normalization changes

Proposition 2 (Raw-score locality)**.** *Fix the anchors, carrier operator, candidate state, incoming parent states, and dependency confidences at a write v. Any modification elsewhere in the graph leaves $s(v)$ unchanged.*

Proof. Equations 4–8 depend only on the stipulated arguments. The graph contains no recursively attenuated source masses and no path-length factor in this readout. □

This statement includes inserting an unrelated file write into the temporal stream when it does not change the observed dependencies. It does not guarantee invariance when an insertion changes which version a later tool reads. Nor does it guarantee an unchanged normalized weight: a new positive-score candidate increases the denominator of $p(v)$. Relative raw rankings among the unchanged candidates are preserved, but the new candidate may itself rank above them.

Proposition 3 (Exact copy under a single full-confidence dependency)**.** *Suppose a fully known write has one confidence-one parent and exposes exactly the same carrier text. Then both introduction and drop are zero for that write.*

Proof. The parent integral equals the carrier vector of that text. Both differences in Equation 6 are identically zero. □

The proposition also applies when several full-confidence parents expose identical token sets. It does not cover arbitrary merge inputs: pooling other text may change the selected token capacity even when one input is copied verbatim. This is why the executable robustness evaluation measures copies and merges rather than treating all such operations as guaranteed invariances.

A.5 A conditional localization result

Proposition 4 (Unique local transition under exact carriers)**.** *Consider one wrong-content atom of unit mass, full-confidence dependencies, complete states, and an exact binary carrier. Suppose the pooled parent carrier of $v^\star$ is zero, its own carrier is one, and every other write with carrier one has pooled parent carrier one. With no other anchors, $v^\star$ is the unique positive-score candidate. The analogous statement for a required-content atom holds when $v^\star$ is the only write whose pooled parent carrier is one and whose own carrier is zero.*

Proof. The introduction contrast at $v^\star$ is $[1 - 0]_+ = 1$. Any other carrying write has pooled exact carrier one by assumption and therefore contrast zero; a noncarrying write also has zero introduction. For loss, substitute the stipulated binary before/after carriers into the reversed difference. □

These are conditional guarantees about identifiable carrier transitions, not assumptions automatically satisfied by natural-language traces. Independent reintroductions, incomplete provenance, or

semantically ambiguous anchors create multiple plausible transitions. The soft UOT carrier supplies a graded score in that setting; the proposition is not extended to it without additional separation assumptions. A finite operational enumeration—2,048 fault attempts over field subsets, copy/projection/union operations, and deletion/replacement faults, of which 256 satisfy the inclusion conditions—checks the proposition under its exact-carrier assumptions with zero violations (Appendix I.3).

A.6 Why recursive residuals and hard events are different

Removing an explicit survival weight does not by itself remove positional attenuation. If a backward recurrence retains a fixed fraction $r \in (0, 1)$, then $d_{i-1} = rd_i$ and

$$\text{residual}_i = (1 - r)r^{n-i}d_n. \tag{15}$$

Thus earlier residuals shrink even without a separate β. The primary method instead compares every local transition using the same fixed anchor supply.

For a scalar product-reference problem with positive a, b and cost c, differentiation gives

$$t^\star = (ab)^{(\varepsilon+\tau)/(\varepsilon+2\tau)} \exp\left[-c/(\varepsilon + 2\tau)\right]. \tag{16}$$

When $a = b = 1$ and $c > 0$, the $\varepsilon \to 0$ limit is $\exp[-c/(2\tau)] \in (0, 1)$, not a binary match. Consequently, hard lexical introduction/drop is a separate operator, not the entropy-zero limit of the reported UOT objective. The implementation tests this scalar solution and a multidimensional optimum against numerical minimization.

B Provenance and Implementation Details

Role: implementation detail supporting reproduction; no experimental claim.

B.1 Parsing and resources

The graph builder consumes observed history, not evaluation labels. It parses structured command arrays, tool-call argument JSON, and assistant content containing an outer JSON command block. A JSON decoder determines object boundaries so embedded code fences inside string payloads do not split the outer command. Tool templates, placeholder commands, quoted messages from other agents, and user/system/tool messages do not create writes. Malformed JSON is counted and skipped rather than guessed.

Resources are canonical lexical paths or explicit field identifiers. Normalization respects an explicitly supplied working directory, removes redundant path components, and never resolves paths against this machine's filesystem. An absolute path and a bare basename are not assumed equal. Explicit read/write fields can describe unsupported tools, but a shell command without resource metadata is not interpreted as an observed file mutation. The evaluator never executes dataset code.

At each read event, the current producer and evidence are stored in an agent-specific pending set. At that agent's next write, the pending sources form the incoming dependencies and are consumed. Resource versions are published only after dependencies and old states have been read, preventing self-loops. One command producing several resources is one write node with per-resource state metadata; its designated primary output supplies the scoring carrier.

B.2 Reconstructed states and artifact selection

Full writes replace state; create without a payload denotes a known empty file. Append requires a known predecessor. Insertion is one-based, with explicit line bounds and newline handling. String replacement requires a nonempty exact match that is unique unless multiple replacement was explicitly requested. Line-range replacement additionally validates the quoted first and last original lines. Generative tools or ambiguous edits may establish a write dependency without establishing complete output text.

Table 3: Primary local-carrier configuration, fixed across reported real-trace evaluations.

Setting	Value	Meaning
ε	0.05	product-reference entropy
τ	0.5	relaxed marginal penalty
semantic / lexical cost	0.7 / 0.2	cosine distance / character overlap
carrier threshold / cap	0.4 / 40	relevant distinct tokens
anchor cap	80 per axis	fixed content-word support
explicit / narrative confidence	1.0 / 0.5	observed / supplementary evidence
temporal fallback	disabled	available confidence 0.1
Sinkhorn iterations / tolerance	500 / 10^{-8}	log-scaling updates
bootstrap replicates / seed	10,000 / 42	paired record resampling

Unknown state remains `None` with `state_known=False`. Observed payload may supply positive-content evidence but is not promoted to a complete final artifact for judging. At an unknown current state, omission is disabled. A known empty state is not replaced by a nonempty payload. The reference proxy and no-reference judge both abstain when the final selected artifact cannot be reconstructed.

The primary artifact policy is last code-resource write, otherwise last nonempty payload write. All candidates are truncated at that exact command rather than the end of its dialogue step. The artifact choice, removal count, tool coverage, and unknown-state count are recorded per trace. Post-artifact writes are retained in the builder's full output but excluded consistently from every method in the benchmark.

B.3 Carrier implementation

The embedding model is locally cached `sentence-transformers/all-MiniLM-L6-v2`, snapshot `1110a243fdf4706b3f48f1d95db1a4f5529b4d41`. Evaluation uses CPU and offline Hugging Face settings. Embeddings are cached by input string and reused across methods. There is no attribution-model training.

The implemented lexical cost for nonempty strings is

$$L(x, y) = 1 - \frac{|\operatorname{chars}(x) \cap \operatorname{chars}(y)|}{\max(|x|, |y|)}, \tag{17}$$

where chars is a set but string length counts repeated characters. This is a custom overlap penalty, not Dice distance or a metric: for example $L(\texttt{cache}, \texttt{cache}) = 0.2$. All reported UOT runs use this same fixed cost; no identity-zero or metric property is assumed in the transport derivation.

The tokenizer lowercases text and replaces non-alphanumeric characters with spaces. Real-trace anchor extraction retains alphabetic non-stopwords of length at least four, prioritizing words unique to one sentence and falling back to frequent words when fewer than eight such words exist. At most 80 words are retained per axis. This choice excludes operators and many numeric literals from semantic anchors; the separate typed-declaration experiment preserves explicit values instead. UOT and cosine use at most 40 relevant carrier tokens; lexical inclusion uses the full token set and no embedding calls.

C Evaluation Protocols and Label Alignment

Role: evaluation protocol and audit. The reference-proxy tables are secondary responsibility-to-write audits, not primary evidence.

C.1 Released labels and candidate-only targets

TracerTraj coding has 127 records with task, reference, history, mistake agent, step, and reason. The predictor receives an observation whitelist; target labels are consulted only after graph construction and anchor selection. The primary candidate support is every retained write, regardless of whether its state is known or a method can score it.

For original label z, define V_z as the set of all candidate writes at that integer step. If V_z is nonempty it is the visible target, including multiple writes by different agents at that step. Aligned targets preserve this set. Otherwise, among candidate steps written by the labeled agent, select the closest to z, breaking equal-distance ties toward the later step, then take all candidate writes at the selected step. If no such step exists, the target set is empty. Alignment changes only evaluation, never the scoring graph. It is a responsibility-to-write proxy, not a semantic annotation that the chosen payload is erroneous.

The visible cohort contains 25 records; the aligned cohort contains 110. Full-visible and full-aligned retain all 127 records, counting empty target sets, missing predictions, and abstentions as misses. Opposite-tie sensitivity uses the same candidates and records. This candidate-only rule differs from an older mapping that chose a write from the complete history before applying a different payload cutoff.

C.2 Ranking and uncertainty

Hit@1 accepts any target-set member at rank one; top-3 and reciprocal rank use the first target-set member in the ranking. Positive-score ties follow the fixed candidate order. All-zero evidence yields an empty ranking, not a chronology-based prediction. Out-of-candidate predictions are misses and are never snapped to a convenient nearby candidate. Every method uses the same cohort and target set.

Paired bootstrap resamples record IDs with replacement for the natural-trace benchmark, computing differences in hit@1, top-3, and MRR. The interval is the 2.5th–97.5th percentile of 10,000 replicates. Repeated controlled injections instead require source-workflow clusters, retaining all variants of each sampled workflow. No significance claim is transferred from one comparison or cohort to another.

C.3 Secondary reference-proxy audit

The natural-trace tables in this section are secondary responsibility-to-write alignment audits, not write-level correctness gold. The primary evidence for the zero-attribution-training inspectable intervention interface comes from controlled interventions and the direct comparisons in Sections E, F, and G. Section J separately assesses the feasibility of observable write-level review; it does not supply new method-accuracy estimates.

The reference proxy is available on 114 complete final states. Its UOT-provenance aligned difference against cosine is -0.9 percentage points, with paired 95% interval $[-5.5, 3.6]$; against last-write it is $+4.5$ points, interval $[-6.4, 15.5]$. These comparisons characterize the reference-conditioned lexical task without asserting a resolved transport advantage. The contribution of dependency structure is assessed through controlled interventions and locality rather than inferred from this table alone.

C.4 No-reference local-carrier results

The judge cache covers all 127 record indices with 128 historical entries, including one pilot timeout followed by a successful explicit retry. It contains 107 complete responses; all have stop finish reasons and JSON objects containing exactly the two requested fields. Twenty report an empty defect pair. Six further responses fail exact-quote validation and one is truncated. Eight unknown final states and five absent artifacts trigger no request. Complete entries have verified task/candidate hashes, requested model, prompt version, and quotations. The local evaluator accepts these 107 entries without using references or repairing rejected responses.

Panel (b) of Table 4 reports this regime. UOT and cosine each abstain on 40 records; lexical matching abstains on 43. The two wrong-only UOT variants abstain on 89. These counts are not removed from accuracy denominators. UOT-provenance and cosine have the same aligned hit@1, with paired 95% difference interval $[-2.7, 2.7]$ points; their MRR difference is .0015, interval $[-.0118, .0151]$. Complete predictions are still diagnoses, not verified semantic fault labels. Position priors are unchanged from Table 4.

Table 4: Unified natural-trace evaluation on identical candidates and targets, with reference-proxy (a) and no-reference (b) anchors. Percentages are hit@1 and top-3; Full denotes full-aligned hit@1. Bold in (b) highlights UOT's top-3 against the fresh LLM ranker in Table 5, not a column maximum.

Method	Visible (25)	Aligned (110)	Top-3	MRR	Full (127)
(a) Reference proxy					
First write	56.0	61.8	90.0	.760	53.5
Last write	28.0	53.6	84.5	.679	46.5
Lexical, provenance	52.0	59.1	82.7	.714	51.2
Cosine, provenance	40.0	59.1	85.5	.718	51.2
UOT, provenance	32.0	58.2	86.4	.712	50.4
Lexical, chain	52.0	58.2	85.5	.721	50.4
Cosine, chain	44.0	65.5	88.2	.759	56.7
UOT, chain	32.0	62.7	86.4	.732	54.3
UOT provenance, correct only	24.0	55.5	82.7	.689	48.0
UOT provenance, wrong only	40.0	47.3	70.9	.598	40.9
UOT chain, correct only	24.0	55.5	84.5	.691	48.0
UOT chain, wrong only	32.0	49.1	71.8	.609	42.5
(b) No reference					
UOT, provenance	16.0	46.4	**66.4**	.564	40.2
UOT, chain	24.0	48.2	67.3	.572	41.7
Cosine, provenance	20.0	46.4	66.4	.562	40.2
Cosine, chain	24.0	47.3	66.4	.567	40.9
Lexical, provenance	16.0	45.5	63.6	.545	39.4
Lexical, chain	24.0	47.3	63.6	.554	40.9
UOT provenance, correct only	16.0	47.3	68.2	.573	40.9
UOT provenance, wrong only	12.0	23.6	31.8	.277	20.5
UOT chain, correct only	20.0	48.2	68.2	.579	41.7
UOT chain, wrong only	12.0	23.6	31.8	.277	20.5

Table 5: LLM predictions on current common candidates. Legacy names are source protocol labels, not verified exposure records; their original prompts/responses are unavailable. They return one unordered step rank, so top-3 and MRR equal hit@1 and cannot recover unseen ranks. Fresh ranker metrics use actual write-ID rankings.

Source	Visible (25)	Aligned (110)	Top-3	MRR	Full (127)
Cached vanilla, with_g	36.0	50.0	50.0	.500	43.3
Cached Self-Refine, with_g	32.0	47.3	47.3	.473	40.9
Cached CRITIC, with_g	36.0	48.2	48.2	.482	41.7
Cached vanilla, without_g	36.0	47.3	47.3	.473	40.9
Fresh no-reference ranker	12.0	48.2	52.7	.507	41.7

C.5 Fresh ranker, remapped baselines, and measured usage

The fresh no-reference ranker receives task, complete final state, every retained write's ID, agent, step, resource sets, observed payload and reconstructed state, plus dependency edges and confidences. It receives no reference, defect-judge anchors, or responsibility labels. Thus the pipelines share observation availability and candidates, not identical model inputs: the defect judge sees only task/final artifact and delegates localization to the local carrier. The requested model is `deepseek-v3.2`, temperature zero, output cap $4{,}096$, timeout 300 seconds, with four workers and no automatic retries. The full run reuses three complete pilot entries.

All 114 eligible requests yield valid complete responses; 13 records with missing final state trigger no request. Twenty-three complete responses abstain. Across all 127 records, rank lengths $0, 1, 2, 3, 4$ occur $36, 71, 17, 2, 1$ times; on aligned-110 they occur $27, 65, 15, 2, 1$ times. Partial rankings are permitted by the fixed prompt, with no padding or remapping to nearby writes. UOT supplies all candidates in descending score order whenever evidence is positive, so top-3 compares the returned shortlist policies, not forced rankings of equal length.

Legacy predictions are mapped only to their exact current candidate step, grouping simultaneous writes as one rank. Their positional record correspondence cannot be proven from source caches without record digests; they are audits, not observation-matched no-reference baselines. All observation-matched LLM comparisons within this secondary audit use the fresh ranker. Relative to that ranker, no-reference UOT-provenance has aligned differences -1.8 hit@1 points ($[-10.9, 7.3]$), $+13.6$ top-3 points ($[3.6, 23.6]$), and $+.0567$ MRR ($[-.0323, .1438]$). Full-127 top-3 is 57.5% versus 45.7%, a paired gain of 11.8 points ($[3.1, 20.5]$). These intervals use the unchanged record denominator, not shared successful responses.

Table 6: Actual provider-reported token usage across all attempts, including rejected judge responses. Bold highlights the smaller prompt-token total among requests with known usage. One timed-out judge request has unknown usage and is not assigned zero cost. No request is made for missing final states.

Pipeline	Requests	Known usage	Prompt tokens	Completion tokens	Total tokens
Defect judge	115	114	**78,452**	399,588	478,040
Write ranker	114	114	302,193	346,786	648,979

On the 107 records with complete, usage-bearing responses from both models, the medians of per-record ranker/judge ratios are 2.978 for prompt tokens, $.906$ for completion tokens, and 1.206 for total tokens. This shared set is used only for cost ratios. The judge's 107 complete responses account for $446{,}407$ reported tokens; its all-attempt total also includes seven rejected responses. Sum of request durations is $8{,}142.5$ seconds for the judge and $6{,}970.4$ for the ranker, not parallel wall time. Returned model IDs are absent for both; requested IDs are retained. Reasoning usage is counted as reported by the provider. No currency, local embedding cost, equal-compute comparison, or latency reduction is inferred.

C.6 Position and tie sensitivity

The aligned cohort contains 59 last-write targets and 51 non-last-write targets. UOT-provenance hit@1 is $42/59$ and $22/51$, respectively. First-write obtains $28/59$ and $40/51$, while last-write obtains $59/59$ and $0/51$. Some traces have both first and last writes in the target set, so these strata should not be interpreted as disjoint first-versus-last classes.

Changing alignment ties from later to earlier gives UOT-provenance $59/110 = 53.6\%$, compared with $64/110 = 58.2\%$ for later ties. First-write changes from $68/110$ to $80/110$ and last-write from $59/110$ to $49/110$. The primary rule is fixed in advance of method comparison; the sensitivity is reported rather than selecting the better tie policy per method. Per-agent and macro-averaged metrics are included in the result JSON.

D Structural and Historical Audits

Role: audit of provenance coverage and of a historical denominator; no method comparison.

D.1 Observed provenance coverage

Across all 127 coding histories, the parser encounters $1{,}357$ JSON blocks, of which 37 are malformed, and materializes 674 write nodes. Of these, 142 have unknown complete states. It recognizes all 714 valid Editor commands; 48 terminal commands have unsupported shell semantics and are not guessed into writes. There are 122 selected artifacts: 121 code-resource selections and one last-payload fallback. Of these, 114 have complete reconstructed states and eight have unknown states; five traces have no selected artifact.

After artifact truncation there are 631 writes and 869 edges: 335 explicit, 19 external, 234 narrative-derived, and 281 structural root edges. Eight graphs contain a non-root fork and a merge; 76 contain a merge. Counts include evidence-root structure and should not be interpreted as a count of proven causal mechanisms. All constructed graphs are acyclic by version order.

To test that this coverage is not specific to one trajectory format, we admit a second coding-agent framework, OpenHands, through a thin format adapter that renames its single `str_replace_editor` tool—dispatched on an inner `command` field—and its payload keys onto the canonical write vocabulary, leaving the audited provenance core unchanged. Across 300 SWE-bench-derived sessions (1,420 trajectories) the parser materializes 16,857 write nodes from 99,310 commands, of which 10,805 (64.1%) have complete reconstructed states. It recognizes all `view`, `create`, `str_replace`, and `insert` editor commands; eight rare editor commands (six `undo_edit`, two `append`) and 56,461 shell and agent-control commands (49,731 `execute_bash`, 3,868 `think`, 1,673 `task_tracker`, 1,189 `finish`) have unsupported semantics and are not guessed into writes. Every trajectory yields one selected artifact. Admitting the framework required no change to the provenance core, only a format adapter, indicating that the canonical write vocabulary generalizes across coding-agent tool schemas.

D.2 Legacy protocol denominator audit

Older caches used a broader payload extractor and a full-history remapping followed by artifact cutoff. They retain 31 visible and 117 aligned targets, not the primary protocol's $25/110$. Reusing the cached single-step LLM predictions on their original truncated candidates gives the audit below. It is separate from a rerun of the new local-carrier method.

Table 7: Legacy candidate-checked cached predictions, shown only to disambiguate historical denominators. Counts are hit@1 numerators.

Method	Visible (31)	Aligned (117)	Full (127)
First write	6	9	9
Last write	9	60	60
Vanilla LLM with reference	11	56	56
Self-Refine with reference	10	56	56
CRITIC with reference	10	57	57
Vanilla LLM without reference	9	51	51

In particular, $56/117 = 47.9\%$ is the candidate-checked vanilla-reference result under this legacy cutoff. A result with denominator 125 cannot be relabeled as $n = 117$. Older no-reference caches from `run_trace_backward_diff` implement hard two-axis matching, not UOT; their truncated `defect_text` is not reused as an anchor. An old cached target rank alone does not recover the full ranking, zero-score abstention, or best rank among several same-step writes. The audit preserves those distinctions explicitly.

E Controlled Executable Workflows

Role: primary evidence for scope, locality, and replay cost.

E.1 Generation, injection, and safety boundary

The fixed generation seed is 731829. Sixty source workflows each contain a four-field policy (threshold, inclusivity, rate, offset), two to four PRD transitions, one to three configuration transitions per branch, a merge, two to four program transitions, and a final program. Independent review and maintenance writes remain in the graph. Three program templates compute a thresholded bonus, tariff, or allocation. A random topological scheduling order changes where branches and independent writes occur without changing their dependencies.

For each workflow, the generator samples three to five distinct eligible writes uniformly without replacement, then samples an applicable fault type at each position. Threshold mutation adds one; condition mutation flips inclusivity; omission removes the offset field. Applicability depends on whether that field exists at the chosen PRD, config, or code write, not on a model's score. The complete plan and its SHA256 are fixed before any prediction. Fault payloads contain ordinary changed values or missing fields, not injection markers.

An independent arithmetic oracle supplies tests around the threshold, at negative and zero inputs, and at distant positive inputs. The executor accepts a program only if it exactly equals one of the self-authored templates for a validated bounded primitive policy. It launches Python in isolated mode with a five-second timeout. It never executes source from the released datasets, arbitrary model output, or a string supplied by a user as code. Both clean and injected programs are run, and all attempts are retained.

There are 246 attempts: 73 threshold changes, 71 condition flips, and 102 omissions. Of these, 222 are clean-pass/injected-fail pairs: 68, 63, and 91 by fault type, and 74 PRD, 96 config, and 52 code faults by stage. The remaining 24 attempts are asymptomatic under the test oracle. Valid positions span write ordinals 3–20 and all five relative-position quintiles, with counts $8, 58, 53, 66, 37$. No attempt is filtered using a localization score.

E.2 Predictor observations and controls

The predictor sees only a failed graph and anchors obtained from a reference program versus the failed final program. Raw anchors are token-set differences including numeric tokens, unlike the bounded alphabetic anchors in the real-trace benchmark. Typed anchors retain changed key/value pairs obtained by safe JSON or Python-literal parsing. The predictor receives neither the clean intermediate trajectory nor the injected node identifier. The identifier enters evaluation only after rankings are fixed.

All writes are candidates, including independent review, notes, final, and post-artifact writes. This stress-test policy differs explicitly from the natural-trace cutoff. UOT and cosine use the same default costs and supports as the primary implementation. Lexical carriers use exact token inclusion. Typed-carrier variants replace each carrier with canonical key/value declarations. Their embedding-based versions retain the inherited key-segment embedding behavior and therefore do not implement an exact value comparison; no tuned value-aware OT cost is added after observing results. The typed first-conflict control tests explicit unequal values or a parent-present/child-absent field in a nonempty typed carrier, with chronological ties. It does not use the injection location or a branch schema to repair its predictions.

The same methods are reported on both cohorts in Table 11.

The 95% source-cluster interval for UOT-provenance hit@1 is $[8.9, 19.3]\%$. Its paired difference against UOT-chain is 9.0 points, interval $[4.1, 14.3]$, for hit@1, and .040, interval $[-.001, .082]$, for MRR. The top-3 difference has the opposite sign: -14.0 points, interval $[-23.7, -4.8]$. Hence the claim about graph structure is metric-specific, not a claim of uniform rank dominance. All method differences, fault-type strata, and stage strata are stored in the result JSON.

E.3 Descendant replay and regeneration cost

For each candidate, the replay procedure regenerates that write and its provenance descendants in topological order. It uses the same fixed generator as forward materialization without repeating the exogenous injection. Only seed generators read the original policy; other generators consume their declared current parents. A selected downstream node can therefore reproduce an inherited fault. All non-descendants are checked to remain byte-identical, and the resulting final program is executed against the same independent oracle.

Every candidate intervention is actually executed. This gives an exact uniform-selection expectation rather than a single random draw, and permits each attribution method to select from the same intervention outcomes. Known-location replay passes all 222 cases and regenerates a mean 7.23 writes. First-write passes 23.4% at 3.90 writes; uniform selection passes 36.4% at 5.39; UOT-provenance passes 14.0% at 1.92; lexical provenance passes 45.0% at 3.91; typed lexical provenance passes 59.5% at 4.86; typed first-conflict passes 78.8% at 8.51. Ancestor selections can succeed without matching the injected node, so repair success is not inferred from hit@1. Total program executions, including baseline and noise checks, are 5,376.

These are measured outcomes of controlled deterministic replay, not a comparison under equal regeneration budgets and not learned repair. The experiment does not reproduce a real OpenHands execution environment. Its value is an auditable intervention contract: the selected write determines

which state is regenerated, dependency closure determines its propagation, and an executable oracle determines success.

E.4 Noise tests and the scope of locality

Four paired perturbations are generated without looking at injection position or prediction: three independent maintenance writes, an exact-copy node whose existing consumers are rewired to it, synonymous comments, and all three combined. In 16 attempts per copy-containing condition, the selected source is a dangling review branch with no consumers. There are 984 noisy injected trajectories across all attempts and corresponding clean-workflow checks. Every perturbation preserves the observed test outputs. Comment changes use a reference with the same comment style, keeping behavioral anchor differences fixed.

Table 8: UOT-provenance robustness on 222 valid pairs. Score and rank preservation compare the injected write with its original value; ranks exclude newly inserted candidates for the preservation check, but not for MRR. Bold denotes preservation in every valid pair.

Perturbation	Hit@1 (%)	MRR	Score preserved (%)	Rank preserved (%)
None	14.0	.311	—	—
Unrelated writes	14.0	.243	**100.0**	**100.0**
Identical copy	14.0	.311	**100.0**	**100.0**
Synonymous comments	14.0	.310	32.4	94.6
Combined	14.0	.243	32.4	94.6

Raw-score locality concerns unchanged local arguments, not immunity to every new candidate or semantically equivalent wording. The unrelated-write condition preserves all original fault scores but adds competing notes. The root-to-note loss signal in $182/222$ base cases also demonstrates that task context is not an output-specific requirement assignment. This behavior is retained in the evaluation rather than removed using a ground-truth-dependent candidate filter.

F Output Applicability

Role: primary evidence for output-ancestor scope, declared field responsibility, and transfer to natural traces.

F.1 Fixed rules and observation boundary

Output relevance and field responsibility are separate inputs to attribution. The original score uses every parent as context. The *observed-loss* switch (O) instead pools only non-root explicit or narrative parents with known, non-null reconstructed states for the correct-content loss axis. It preserves their confidence weights and leaves wrong-content introduction unchanged. A known empty child can still lose content; an unknown parent snapshot cannot establish observed loss. The *artifact* switch (A) zeroes scores outside the selected artifact's reverse explicit/narrative ancestor closure, including the artifact itself and excluding the task root. OA combines the switches. The original default (All), anchors, carrier costs, thresholds, and candidate order are unchanged.

All writes remain in each score matrix, including zero-scored writes, and chronological ties are retained. If any score is positive, the full sorted list includes zero-score candidates; an all-zero matrix abstains. Thus A changes score eligibility, not the evaluation denominator, and is not equivalent to simply deleting candidates. Its closure is computed from the failed observed graph, with no clean trajectory or target label. An absent artifact gives an empty closure. The interpretation requires sufficient provenance: a missing dependency can suppress a genuinely responsible write.

The development experiment reuses the original 60 workflows and their 3,846 executed candidate interventions. Every original score and ranking is reproduced before applying a switch. Fresh validation fixes 60 new workflow draws, their random eligible injections, and the complete plan hash at seed 20260911 before prediction. It yields 244 attempts, 226 valid failures, and 18 asymptomatic attempts. All 4,069 candidate replays are executed, giving 4,373 total executions including clean

and injected checks. Validation is a same-generator replication, not an unseen-domain test. Both cohorts retain every attempt; neither fault labels nor localization scores select the valid-pair subset.

F.2 Controlled replication and equally scoped controls

Table 11 reports fresh validation. For every matcher, O, A, and OA have identical reported aggregate metrics in both controlled cohorts, although their complete rankings differ. For example, UOT O and A differ on $127/222$ development rankings and $106/226$ validation rankings. We therefore group their aggregates in this table, not their algorithms. The frozen primary contrast was O minus All for UOT hit@1; A is a separate control, not a retrospectively preregistered primary method. Intervals are pointwise descriptive source-cluster intervals, without a best-setting significance claim.

That table lists the scoped raw, typed, chain, and prior rows together, with N and abstention A reported only for the scoped control runs.

On development, UOT exact hits increase from 31 to 121 ($+40.5$ points, 95% cluster interval $[33.9, 46.8]$); on validation, they increase from 26 to 115 ($+39.4$, $[31.2, 48.1]$). Neither cohort loses an original hit. The injected write's own score is unchanged in all $222 + 226$ pairs: the improvement comes from removing competing inapplicable evidence, not increasing the target's score. Root-only UOT selections change from 182 to 0 on development and 189 to 0 on validation. With O/A/OA, development hit@1 is 31.1% for cosine, 66.2% for lexical, 16.7% for typed UOT and typed cosine, and 75.7% for typed lexical. Full score matrices, strata, paired differences, and replay outcomes remain in the result files.

Artifact-scoped UOT repairs $166/222$ development and $164/226$ validation failures, with mean regeneration costs 7.12 and 7.34. Localization and replay differ because ancestor selections can also succeed. All-candidate and artifact-uniform controls use exact expectations over executed interventions. The first artifact ancestor repairs every case but regenerates 13.77 development or 14.51 validation writes; this separates output relevance from choosing an economical repair location. Costs are not matched across methods.

F.3 Transfer to parsed natural traces

Both natural anchor regimes are recomputed on the original 127 records, retaining the same 110 aligned targets. Original candidate IDs, targets, artifacts, anchors, scores, and rankings are checked against the archived outputs. Judge anchors come from the existing hash-bound cache; no model is called. This is exploratory transfer on previously inspected traces. Table 9 reports all scope switches and equally scoped priors; per-record MRR and paired intervals are also retained in the result files.

Artifact scoring preserves aggregate no-reference UOT hit@1, top-3, and MRR, while changing $13/127$ full rankings; reference-proxy rankings change in $21/127$ records. Eleven aligned target sets have no node in the parsed artifact closure. O minus All changes aligned hit@1 by -13.6 points (paired interval $[-20.9, -7.3]$) with reference anchors and -18.2 ($[-26.4, -10.9]$) without a reference. These observations support keeping artifact relevance and task-context suppression as separate controls, rather than replacing the natural-trace default with a rule validated only on complete generated provenance.

F.4 Declared branch-field responsibility

This mechanism probe supplies an additional write-bound contract, rather than claiming to infer responsibilities from the released logs. At fixed seed 20260912, six new workflow draws cover all three program families. Before materialization or injection, each operation exports its resource, serialization format, schema version, and required output fields from the generator's field declaration. The manifest contains no expected values or fault locations, is hashed with the fixed plan, and is checked unchanged after every injection. The generator itself is unchanged: its field list projects present values; the predictor separately interprets the exported list as an output obligation.

Each workflow has three targeted config variants: remove offset at the first amount-branch config write, increment threshold at the first eligibility config write, and flip inclusivity at that same eligibility write. These are designated mechanism locations, not random-position coverage. All 18 attempts are clean-pass/injected-fail pairs. Predictors receive only the failed graph, reference/final

Table 9: Natural-trace applicability transfer (%). Full/aligned hit@1 use $127/110$ records; top-3 uses 110. All missing predictions and targets remain misses. O, A, and OA are distinct on these incompletely observed graphs. Priors use no defect anchors.

		Reference proxy		No reference	
Method	Scope	Full/aligned	Top-3	Full/aligned	Top-3
UOT	All	50.4/58.2	86.4	40.2/46.4	66.4
UOT	O	38.6/44.5	71.8	24.4/28.2	48.2
UOT	A	49.6/57.3	86.4	40.2/46.4	66.4
UOT	OA	38.6/44.5	71.8	23.6/27.3	44.5
Cosine	All	51.2/59.1	85.5	40.2/46.4	66.4
Cosine	O	37.8/43.6	69.1	22.0/25.5	43.6
Cosine	A	49.6/57.3	87.3	40.2/46.4	66.4
Cosine	OA	37.8/43.6	70.0	19.7/22.7	40.9
Lexical	All	51.2/59.1	82.7	39.4/45.5	63.6
Lexical	O	40.9/47.3	67.3	20.5/23.6	40.9
Lexical	A	51.2/59.1	83.6	37.8/43.6	62.7
Lexical	OA	41.7/48.2	66.4	18.9/21.8	38.2
First write	All	53.5/61.8	90.0	53.5/61.8	90.0
First write	A	51.2/59.1	90.0	51.2/59.1	90.0
Last write	All	46.5/53.6	84.5	46.5/53.6	84.5
Last write	A	46.5/53.6	88.2	46.5/53.6	88.2

defect anchors, and the fixed manifest. In particular, they never receive the clean intermediate trajectory or the injected write ID. Contract acquisition and no-reference anchor diagnosis are outside this experiment.

All modes share strict JSON, literal-assignment, or declaration-line parsing and the same typed anchors, candidates, observed-loss scope, and artifact closure. A valid empty policy is known empty; an invalid or unavailable state is unknown, not proof of deletion. The three modes differ only in the loss gate: *off* leaves D^{O} unchanged; *declared* retains an anchor's loss iff its key is a required output; *current keys* uses keys present in the current output as a negative control. Unspecified responsibility retains the original loss, whereas an explicitly empty obligation permits no field loss. The small experiment uses fully specified manifests. Introduction, carrier costs, and per-axis anchor budgets never change with the gate. Boundedness is preserved because each loss coordinate is multiplied by zero or one; locality additionally conditions on the same manifest and artifact-scope membership.

An exact non-OT transition control uses the same observations: a wrong value is introduced when present now and absent from all known explicit parents; a correct value is lost when present in a parent but absent or unequal now. Every non-root dependency in these workflows is explicit with confidence one, so the comparator and carriers use the same parent evidence. It receives the same three gates. The inherited semantic typed UOT/cosine operators match key segments, whereas lexical and exact predicates compare values. Off disables responsibility gating, not the shared manifest-backed parsing.

The clean diagnostic scores each clean graph with its paired failed-case anchors. This deliberately nonempty probe measures legitimate projection loss; it is not end-to-end no-defect detection. Each pair contains one measured clean projection of the corrected field onto a branch not responsible for that field. Table 10 reports every method/mode. Declared responsibility removes all 18 projection false losses while preserving all six actual offset-deletion losses. Current-key gating also removes the false losses, but erases the six true deletion signals: an omitted field cannot be allowed to erase its own obligation.

The two additional declared-mode hits resolve chronological ties between real omissions and legitimate projection boundaries. For the 12 value/condition changes, lexical and exact controls retain the same positive introduction at the fault under every gate; semantic typed UOT/cosine have zero introduction because their keys do not change. Lexical and exact scores and rankings agree throughout. Neither reintroduces wrong content on the 74 downstream checks with nonempty wrong-anchor budgets; the additional 43 omission-descendant checks have empty wrong axes. UOT/cosine also give zero introductions, so their downstream zeroes alone do not demonstrate value discrimination.

Table 10: Field results on the same 18 faults and six workflows. (a) Responsibility gates: FP counts measured clean projection losses out of 18, Del counts true deletion signals retained out of 6, and Hit/Pass count exact localizations and executed repairs out of 18. (b) Direction and scope controls: H1, top-3, N, A, and Pass are counts, the uniform row is an exact expectation, and W averages regenerated writes including abstentions. "—" means replay coverage is incomplete, not zero. Bold marks declared-mode projection suppression with deletion preservation, not a universal matcher ranking.

Typed matcher	Gate	FP ↓	Del ↑	Hit	Pass	W
UOT	Off	18	6	4	4	8.06
UOT	Declared	**0**	**6**	6	6	2.72
UOT	Current keys	0	0	0	0	0.00
Cosine	Off	18	6	4	4	8.06
Cosine	Declared	**0**	**6**	6	6	2.72
Cosine	Current keys	0	0	0	0	0.00
Lexical	Off	18	6	16	16	7.39
Lexical	Declared	**0**	**6**	18	18	7.50
Lexical	Current keys	0	0	12	12	4.78
Exact transition	Off	18	6	16	16	7.39
Exact transition	Declared	**0**	**6**	18	18	7.50
Exact transition	Current keys	0	0	12	12	4.78

Carrier	Control	H1	Top-3	N	A	Pass	W
Slice	First	0	0	18	0	—	—
Slice	Nearest	0	0	18	0	—	—
Slice	Uniform	1.28	3.85	16.72	0	—	—
Raw lexical	Unsigned	12	16	4	2	12	6.61
Raw lexical	Directional	12	16	4	2	12	6.61
Raw lexical	Original/A	12	16	4	2	12	6.61
Raw UOT	Unsigned	6	16	10	2	—	—
Raw UOT	Directional	9	16	7	2	—	—
Raw UOT	Original/A	9	16	7	2	—	—
Typed lexical	Unsigned/off	16	18	2	0	16	7.39
Typed lexical	Directional/off	16	18	2	0	16	7.39
Typed lexical	Dir./declared	18	18	0	0	18	7.50
Typed exact	Unsigned/off	16	18	2	0	16	7.39
Typed exact	Directional/off	16	18	2	0	16	7.39
Typed exact	Dir./declared	18	18	0	0	18	7.50
Typed UOT	Unsigned/off	4	6	14	0	4	8.06
Typed UOT	Directional/off	4	6	14	0	4	8.06
Typed UOT	Dir./declared	6	6	0	12	6	2.72

Their hit@1 changes from $4/18$ to $6/18$, while MRR is .3385 off and .3333 declared. Lexical/exact hit@1 changes from $16/18$ to $18/18$, with MRR .9444 to 1. These counts isolate responsibility gating from value representation rather than establishing an OT advantage.

The 56 actual program executions comprise six clean finals, 18 injected finals, and 32 unique per-case interventions selected by the methods or the known fault. Every known-location replay passes; non-descendants are checked byte-identical. Unlike the randomized cohorts, this probe executes the union of selected interventions, not every candidate, and claims no uniform-selection expectation. Mean work includes abstentions, so lower work alone is not repair efficiency. The scope is a six-workflow mechanism check with oracle-reference anchors and supplied contracts, not a population accuracy or general repair claim.

G Direct Slicing, Direction, and Contract Controls

Role: primary evidence for slicing, direction, and contract controls.

G.1 OFFLINE COMPARISON CONTRACT

These controls test an inspectable intervention interface: observed dependencies delimit relevance and regeneration closure, fixed defect anchors define the evidence being traced, and a local transition selects a write without attribution training. They distinguish this interface from a claim of general agent repair or a requirement that transport dominate exact rules. The control protocol fixes seed 20260914 and reuses archived development (222 valid pairs from 246 retained attempts), validation (226 from 244), and field (18 from 18) cohorts. Valid-pair status is inherited from the earlier executed clean/fault checks, not selected by the new methods. Each randomized cohort has 60 source workflows; the field cohort has six. Validation here reuses the same-generator replication, not another fresh sample.

Every method retains the same write IDs and order, including unrelated writes. The backward slice is the selected artifact's reverse explicit/narrative dependency closure, including the artifact and excluding the task root. It is a read/write provenance slice, not a language-semantic program slicer. We report its coverage separately from three selection policies: first chronological slice write, nearest write by shortest dependency distance to the artifact, and the exact uniform expectation over slice writes. The nearest policy includes the artifact at distance zero. All intact slices contain the injected write; coverage alone does not identify it. Carrier methods zero scores outside this shared slice, retain chronological ties, and abstain on all-zero evidence. A nonempty ranking still includes zero-score candidates, so ordinary top-3 need not be positive-evidence top-3.

The unsigned control measures absolute changes in the same wrong/correct anchor-support banks; the directional control retains wrong-content introduction and correct-content loss. Both use identical carriers, anchors, and known observed loss parents. Direction is compared with the responsibility gate *off*; there are no unsigned-declared arms. The original artifact-scoped raw score is also retained as a separate comparator. Typed methods all share manifest-backed parsing, including off mode. Their contracts are additional generator-derived inputs specifying resource, format, schema version, and required output fields, with no expected values or fault labels. Declared gating changes only directional loss coordinates; it is not inferred from the failed output. Missing obligations (`required_fields=None`) retain resource/serialization metadata and the ungated loss fallback, unlike an explicitly empty obligation.

All new official control and boundary runs record `actual_program_executions=0` and `network_calls=0`. Control replay results are lookups of previously executed interventions on intact graphs, not new executions or estimates from localization accuracy. Work W averages regenerated writes over the entire valid cohort, assigning abstentions zero work. Unavailable interventions are not assigned zero success or work, and no replay mean is inferred over just the available field cases. Paired uncertainty uses $2{,}000$ source-workflow bootstrap replicates (seed 20261011), retaining variants within each sampled workflow; intervals are pointwise descriptive 95% intervals, not winner-selection guarantees. Field results are six-workflow mechanism counts without population confidence intervals.

G.2 WHAT THE DIRECT COMPARISONS ESTABLISH

Table 11 separates slice coverage, evidence direction, and field obligation. The first slice write repairs every validation case but never exactly identifies the injection and regenerates 14.51 writes on average; nearest-slice selection repairs none. Directional raw UOT identifies $115/226$ injections versus unsigned UOT's $67/226$: $29.6\% \to 50.9\%$, a paired $+21.2$ percentage points ($[16.3, 26.9]$). Development changes from $60/222$ to $121/222$, or $27.0\% \to 54.5\%$ ($+27.5$ points, $[20.2, 34.6]$). These direction intervals come from `framework_direction_contrasts.json`, a later descriptive completion of *all* frozen same-carrier direction contrasts, not a preregistered winning contrast or a new parameter choice. The originally enumerated contrasts in the control summaries are retained separately.

Unsigned and directional lexical/exact outcomes coincide in this generator; typed UOT's gate-off direction contrast is also zero. Thus direction's raw-UOT gain is not claimed for every carrier. Declared responsibility is a separate comparison: typed lexical and exact both improve from $172/226 = 76.1\%$ off to $226/226 = 100\%$ declared ($+23.9$ points, $[19.2, 28.6]$), while development improves from 75.7% to 100%. This is an advantage of supplied obligations with value-aware evidence, not evidence that OT is necessary for exact declarations. Typed UOT retains the inher-

Table 11: Complete direct-control matrix on the controlled cohorts (222 development and 226 validation valid pairs, 60 source-workflow clusters each). All keeps every parent as context, Chain is the chain baseline, A restricts scoring to output ancestors, and K adds supplied field obligations; U/D mark unsigned/directional change, written U/D where the two aggregates coincide. Every A row is the original method under output-ancestor scope except A+K, which also gates loss by declared obligations. Dev. and Val. are hit@1; Top-3 and MRR are validation. N, abstention A, and archived replay pass are reported only for the scoped control runs, so "—" means not evaluated under that protocol, not zero; where they are reported, H1+N + A = 100% before rounding. W is mean regenerated writes including abstentions. Bold marks within-carrier direction and contract gains, not column maxima.

Carrier	Variant	Dev. H1	Val. H1	Top-3	MRR	N	A	Pass	W
Slice	First	0.0	0.0	17.7	.190	100.0	0.0	100.0	14.51
Slice	Nearest	0.0	0.0	17.7	.187	100.0	0.0	0.0	1.00
Slice	Uniform	8.0	7.5	22.6	.240	92.5	0.0	50.7	7.25
Raw UOT	All	14.0	11.5	20.4	.289	—	—	11.5	1.73
Raw UOT	Chain	5.0	5.3	29.6	.256	—	—	11.5	2.56
Raw UOT	A+U	27.0	29.6	89.4	—	65.5	4.9	60.2	7.29
Raw UOT	A+D	**54.5**	**50.9**	95.1	.719	44.2	4.9	72.6	7.34
Raw cosine	All	14.0	10.6	15.9	.273	—	—	10.6	1.69
Raw cosine	Chain	5.4	4.9	23.5	.245	—	—	11.1	2.46
Raw cosine	A+D	—	23.9	95.1	.595	—	—	53.5	7.54
Raw lexical	All	45.0	41.2	50.4	.536	—	—	41.2	3.44
Raw lexical	Chain	36.0	33.6	58.4	.507	—	—	33.6	2.91
Raw lexical	A+U/D	66.2	64.2	95.1	.796	31.0	4.9	82.3	7.58
Typed UOT	All	0.5	0.4	7.1	.154	—	—	0.4	1.06
Typed UOT	A+U/D	16.7	17.7	42.9	.356	82.3	0.0	52.2	7.83
Typed UOT	A+D+K	41.0	41.6	41.6	—	0.0	58.4	41.6	3.09
Typed cosine	All	0.5	0.4	7.1	.154	—	—	0.4	1.06
Typed cosine	A+D	—	17.7	42.9	.356	—	—	52.2	7.83
Typed lexical	All	59.5	58.8	65.5	.682	—	—	58.8	4.54
Typed lexical	A+U/D	75.7	76.1	100.0	.881	23.9	0.0	88.1	7.58
Typed lexical	A+D+K	**100.0**	**100.0**	100.0	—	0.0	0.0	100.0	7.16
Typed exact	A+U/D	75.7	76.1	100.0	—	23.9	0.0	88.1	7.58
Typed exact	A+D+K	**100.0**	**100.0**	100.0	—	0.0	0.0	100.0	7.16
Typed first-conflict	All/A	43.7	43.8	100.0	.719	—	—	78.3	8.56
Prior	First write All	0.0	0.0	2.7	.115	—	—	35.4	5.95
Prior	Last write All	0.0	0.0	17.7	.181	—	—	0.0	1.00
Prior	Uniform All	5.8	5.6	16.8	.195	—	—	38.0	5.72

ited key-only representation without a value-aware cost or parameter retrofit: declared mode yields $94/226 = 41.6\%$ hit@1 and $132/226 = 58.4\%$ abstention. Its zero non-target rate must be read with that abstention rate. Gate-off typed UOT top-3 is 42.9%, but positive-evidence top-3 is 41.6%; zero-score ties account for the difference.

Panel (b) of Table 10 applies the direct slicing, direction, and contract controls to the same 18 cases.

These counts preserve the six-workflow mechanism scope of Section F.4. Raw-UOT direction adds three exact hits ($6 \rightarrow 9$), while lexical/exact gate-off direction adds none. Declared typed lexical/exact adds two hits ($16 \rightarrow 18$). The earlier field archive executed a union of selected interventions, not every candidate; consequently newly selected slice and raw-UOT interventions do not all have archived replay outcomes. All 18 methods, full rankings, positive-evidence metrics, paired contrasts, and replay-availability flags remain in the frozen control JSONs.

H BOUNDARY OF INCOMPLETE AND CORRUPTED OBSERVATIONS

Role: robustness of the primary localization results under incomplete and corrupted observations.

H.1 FIXED STRUCTURAL PERTURBATIONS

The boundary protocol uses seed 20260914 and rates $r \in \{0.1, 0.3\}$. For each source workflow, structural IDs are hash-ordered with the seed, base identity, and perturbation type, sharing the order across fault variants. The 0.1 selections are prefixes of the 0.3 selections. Neither target IDs nor scores enter this ordering. Missing-edge corruption removes $\lceil r|E| \rceil$ original non-root explicit/narrative edges. Spurious-edge corruption adds the same requested number from absent forward-ordered candidate pairs, capped by available pairs; its denominator is the *original edge count*, not the number of absent pairs. Forward ordering preserves acyclicity. Read/write metadata remain recorded, but scorers do not reconstruct the corrupted edges from them.

Unknown-state corruption selects $\lceil r|V| \rceil$ candidate writes and removes their complete state, observed payload, and per-resource states. Missing-contract corruption selects that many writes and sets only required fields to `None`; resource, format, and schema metadata persist. Wrong-contract corruption instead toggles one hash-preselected field's membership in each selected obligation set, either adding or removing it. Neither contract perturbation changes the graph or any expected value. All candidate IDs and the selected artifact ID remain fixed. Reference/final anchors are frozen *before* corruption, even when the final state is subsequently hidden: this is localization conditional on an existing diagnosis, not an end-to-end missing-final-artifact experiment.

All 244 validation attempts remain archived; every validation boundary metric uses the same 226 valid pairs, including misses and abstentions. Non-target selection compares top-1 with the injected-write identity, not with a claim that every other write is causally innocent. For a selected write $\hat{v}$, closure mismatch is

$$C = \mathbf{1}\{\mathrm{Desc}_{\mathrm{observed}}(\hat{v}) \neq \mathrm{Desc}_{\mathrm{true}}(\hat{v})\}, \tag{18}$$

where each closure includes the selected write itself. Both closures concern the *same selected write*, not the gold fault; abstention has two empty closures and hence zero mismatch. Missing/extra closure sizes are retained in JSON. The tables report unconditional H1/N/A/C, with paired source-cluster intervals and all remaining methods in the frozen JSONs. No corrupted-graph replay success is claimed or inferred from the intact intervention archive.

H.2 READING LOCALIZATION AND CLOSURE TOGETHER

Missing edges place the true fault outside the observed slice in $134/226$ cases at rate .1 and $193/226$ at .3. At .3, raw lexical has only 8.4% H1 despite 0% closure mismatch, with 54.4% abstention; typed exact has 11.5% H1 and 38.1% abstention with the same zero mismatch. These small closure errors reflect changed selection and empty abstention closures, not robustness to absent dependencies. Conversely, typed exact retains 100% H1 under spurious edges while its closure mismatch grows to 8.0% at .3: selecting the right write does not establish the right regeneration boundary.

State and obligation perturbations leave graph edges intact, so zero closure mismatch there says nothing about localization. At rate .3, unknown states yield 59.3% H1 for typed exact and 28.3% for raw UOT. Missing versus wrong obligations yield typed-exact H1 93.4% versus 94.7%, with different non-target/abstention partitions; neither ordering is a general ranking of corruption severity. Typed UOT's 58.4% intact abstention changes to 22.6% under unknown states and 39.8% under missing obligations, accompanied by non-target selections. Reporting H1, non-target selection, abstention, and closure jointly prevents a reduction in abstention or work from being mistaken for improved diagnosis.

In the field probe, missing edges exclude the fault from the slice in $16/18$ cases at .1 and all 18 at .3, even though the four displayed methods have zero selected-write closure mismatch. Table 13 retains all five perturbation types and both rates rather than extrapolating from this small probe to population robustness. The two boundary JSONs retain all 18 methods for each of the 11 intact/corrupted conditions, including slice priors, unsigned controls, original artifact scores, and typed gate-off and lexical variants.

Table 12: All five boundary types and both fixed rates on validation (226 pairs, 60 clusters). Entries are percentages: H=hit@1, N=non-target selection, A=abstention, C=selected-write closure mismatch. Raw methods are directional; typed methods are directional/declared. All four metrics retain the full denominator.

		Raw lexical				Raw UOT			
Perturbation	Rate	H	N	A	C	H	N	A	C
Intact	—	64.2	31.0	4.9	0.0	50.9	44.2	4.9	0.0
Missing edges	.1	27.4	33.2	39.4	5.8	14.6	52.2	33.2	8.0
Missing edges	.3	8.4	37.2	54.4	0.0	4.4	42.0	53.5	0.4
Spurious edges	.1	64.2	31.0	4.9	6.6	50.0	45.1	4.9	6.2
Spurious edges	.3	63.3	31.9	4.9	14.2	45.1	50.0	4.9	13.3
Unknown states	.1	58.0	27.9	14.2	0.0	30.5	62.8	6.6	0.0
Unknown states	.3	47.3	24.3	28.3	0.0	28.3	56.2	15.5	0.0
Missing contracts	.1	64.2	31.0	4.9	0.0	50.9	44.2	4.9	0.0
Missing contracts	.3	64.2	31.0	4.9	0.0	50.9	44.2	4.9	0.0
Wrong contracts	.1	64.2	31.0	4.9	0.0	50.9	44.2	4.9	0.0
Wrong contracts	.3	64.2	31.0	4.9	0.0	50.9	44.2	4.9	0.0
		Typed exact				Typed UOT			
Perturbation	Rate	H	N	A	C	H	N	A	C
Intact	—	100.0	0.0	0.0	0.0	41.6	0.0	58.4	0.0
Missing edges	.1	35.8	34.5	29.6	6.6	16.4	52.7	31.0	13.3
Missing edges	.3	11.5	50.4	38.1	0.0	6.2	55.8	38.1	1.3
Spurious edges	.1	100.0	0.0	0.0	4.0	41.6	0.0	58.4	0.9
Spurious edges	.3	100.0	0.0	0.0	8.0	41.6	0.0	58.4	3.5
Unknown states	.1	81.0	7.1	11.9	0.0	35.4	43.4	21.2	0.0
Unknown states	.3	59.3	17.3	23.5	0.0	26.5	50.9	22.6	0.0
Missing contracts	.1	99.1	0.9	0.0	0.0	40.7	7.1	52.2	0.0
Missing contracts	.3	93.4	6.6	0.0	0.0	35.0	25.2	39.8	0.0
Wrong contracts	.1	96.9	1.3	1.8	0.0	38.5	4.0	57.5	0.0
Wrong contracts	.3	94.7	2.2	3.1	0.0	36.3	7.1	56.6	0.0

I Observation Certificates: Identifiability, Mechanisms, and Stability

Role: diagnostic. The identifiability ceilings and stability certificates bound what any method can recover; they are not achieved-accuracy results.

The studies in this section were fixed by a written protocol before execution. The selection policy admits all archived valid controlled cases and all 12 public review packets, with no winner-dependent inclusion or parameter search. Runs are local and offline, reuse unchanged carrier implementations, create exclusively new output files, and verify source and input hashes before and after execution; the public packets are additionally bound by a pre-goal evidence baseline. Independent post-run audits recompute the stored arithmetic directly from the frozen artifacts (`observation_results_audit.json`: 856 observation-class checks across 8 cohort variants, 2,048 operational attempts, 256 admissible states and scores, 108 certificate checks, 1,539 stress samples, 7,353 node-bound checks, 684 fixture samples; status passed with input hashes unchanged).

I.1 Structural identifiability ceilings

Two observation regimes are compared. The *structural* observation retains the canonical root, the selected artifact, ordered node name/step/agent/read/write/path/state-availability fields, and ordered dependency tuples with kinds and confidences; it excludes payloads, reconstructed states, task text, case identifiers, and fault labels. The *diagnosis-augmented* observation adds the complete frozen anchor object and is reported separately, not assumed equivalent. Within each regime, cases sharing an exact observation fingerprint form an equivalence class; the attainable hit@1 ceiling aggregates the maximum target multiplicity of each class over all retained cases. This is a finite-data combi-

Table 13: Field boundary mechanism counts, 18 cases on six workflows, with the same H/N/A/C definitions and method settings as Table 12. No population intervals or corrupted-graph replay outcomes are inferred.

		Raw lexical				Raw UOT			
Perturbation	Rate	H	N	A	C	H	N	A	C
Intact	—	12	4	2	0	9	7	2	0
Missing edges	.1	1	7	10	0	1	7	10	0
Missing edges	.3	0	8	10	0	0	8	10	0
Spurious edges	.1	12	4	2	1	9	7	2	1
Spurious edges	.3	12	4	2	2	9	7	2	1
Unknown states	.1	12	4	2	0	6	10	2	0
Unknown states	.3	10	5	3	0	3	12	3	0
Missing contracts	.1	12	4	2	0	9	7	2	0
Missing contracts	.3	12	4	2	0	9	7	2	0
Wrong contracts	.1	12	4	2	0	9	7	2	0
Wrong contracts	.3	12	4	2	0	9	7	2	0
		Typed exact				Typed UOT			
Perturbation	Rate	H	N	A	C	H	N	A	C
Intact	—	18	0	0	0	6	0	12	0
Missing edges	.1	2	10	6	0	2	10	6	0
Missing edges	.3	0	12	6	0	0	12	6	0
Spurious edges	.1	18	0	0	1	6	0	12	1
Spurious edges	.3	18	0	0	2	6	0	12	2
Unknown states	.1	18	0	0	0	6	6	6	0
Unknown states	.3	14	2	2	0	6	10	2	0
Missing contracts	.1	18	0	0	0	6	2	10	0
Missing contracts	.3	18	0	0	0	6	4	8	0
Wrong contracts	.1	17	0	1	0	5	0	13	0
Wrong contracts	.3	17	0	1	0	5	1	12	0

Table 14: Observation-equivalence ceilings on the archived controlled cohorts. "Structural" uses the dependency-structure observation only; "+anchors" adds the frozen defect-anchor object. Mixed classes contain both target and non-target cases and are exactly the classes where a restricted predictor can err. Ceilings are attainable hit@1 for any predictor restricted to the stated observation.

			Structural		+ frozen anchors		
Archive	Attempts	Valid	Classes	Ceiling	Classes	Mixed	Ceiling
Controlled (development)	246	222	60	27.0%	140	66	63.1%
Applicability (validation)	244	226	60	26.5%	144	64	63.7%
Field responsibility	18	18	6	66.7%	18	0	100.0%
Combined	508	466	126	28.3%	302	130	64.8%

natorial identity for predictors restricted to that observation: no bootstrap interval applies, and no claim is made about payload-aware diagnosis or language-semantic slicing.

Every structural class is target-mixed ($466/466$ cases): dependency structure alone never isolates the injected write within these cohorts, capping any structure-only predictor at 28.3% combined hit@1. Frozen anchors split the cases into 302 classes of which 130 remain mixed (294 cases), raising the ceiling to 64.8%; the field-responsibility cohort becomes fully identified ($18/18$ single-target classes). The gap between the two ceilings is the information that payload-anchored diagnosis adds over dependency structure; the residual mixed mass under anchors is information no anchor-based method can recover from these observations.

I.2 Mechanism and numerical fixtures

Direction fixtures use fixed value-aware binary carriers: a $7 \to 8$ deterioration and an $8 \to 7$ correction separate the directional score from its unsigned counterpart at the level of a single transition. Obligation fixtures hold graph, states, and anchors fixed while swapping which of two branches carries the required-field contract; exact directional loss plus external obligations moves the selected responsible branch, while gating by current output keys erases the deletion signal, since a deletion cannot erase its own obligation. These are self-authored finite mappings; no public source executes, and the fixtures establish mechanisms under stated exact-carrier assumptions rather than a soft-carrier localization theorem. Numerical fixtures exercise empty supports, unknown states, duplicate parent text, confidence-order crossings, zero/one boundaries, and differing anchor-bank sizes across fixed lexical and bounded numerical UOT carriers with unchanged ε, τ: 684 samples, $4{,}788$ bound checks, zero violations.

I.3 Exact operation semantics

A finite enumeration operationally checks Proposition 4: all four subsets of two source fields at each of four copy/projection/nonconflicting-union operations, crossed with four fault writes and deletion/replacement faults ($2{,}048$ attempts). Inclusion requires that the correct value reach the clean output, that the selected write require it and receive correct predecessor support, and that nonconflicting unions leave the defect in the final output; every ineligible, output-unrequired, conflicting-union, and nonpersistent attempt is recorded before scoring (64, $1{,}664$, 32, and 32 respectively; 256 admissible, split evenly between deletion and replacement faults). Among admissible states the exact typed carrier identifies the unique fault-positive write in all 256 fault cases, finds no positive on any healthy case, and records zero violations. The check binds the exact value-aware carrier with declared loss gating and artifact scope; it is not a guarantee for lexical or embedding carriers, and not a natural-trace benchmark.

I.4 Confidence-stability certificates

For fixed texts, anchors, edge identities and kinds, state availability, obligations, and candidate/output eligibility, pooled carrier coordinate changes are bounded by $a_i \min(1, \sum_{\text{incoming}} |\Delta c|)$, and raw directional score changes by the total nonempty axis budget times the same quantity. A *certificate* holds when a positive top raw score exceeds every competitor and the abstention alternative by more than both worst-case radii; it is a universal sufficient bound over all admissible fixed-edge confidence perturbations within a radius, including strict positivity against abstention. All-zero ties are not certified. Each incoming confidence may move independently by at most the radius, clipped to $[0, 1]$; edge identities persist even at zero confidence, so no edge-deletion, eligibility, or closure robustness is implied.

On all 12 public packets with frozen no-reference anchors and no field contracts (primary readout: observed-loss artifact-scope directional scores; lexical, cosine, and UOT carriers all retained), radii $\{0.01, 0.05, 0.1\}$ give 108 certificate checks. Exactly one packet yields a certified winner—UOT and cosine at radii 0.01 and 0.05, lexical at 0.01; every other packet/carrier/radius combination abstains or lacks a strict margin, so certificates are rare on natural traces rather than routine. Deterministic stress sampling—all-lower, all-upper, alternating, and 16 hash-seeded sign patterns per radius and case (seed 20260915)—produces $1{,}539$ samples with zero measured bound violations and zero ranking changes, alongside $7{,}353$ node-bound checks. Embedding-backed carriers prewarm from the complete historical cohort per anchor regime (127 records; $2{,}285$ strings without reference, $2{,}351$ with) in separate banks. Certificates concern the local numerical readout: a stable or numerically verified prediction can still be semantically wrong, and no calibrated probability is claimed.

J Observation-Only Write-Review Feasibility Pilot

Role: feasibility audit of observable write-level review; it supplies no method-accuracy estimate.

Table 15: All 12 pilot cases. Both reviews agree on defect and attribution statuses. Targets are first-observable-introduction IDs; a subsequent user-reported confirmation retains every status and target. Reasons summarize static evidence, not executed outcomes.

Case	Defect status	Target	Provisional reason / unresolved evidence
P01	Ambiguous	—	Callable versus standard-input interface is unresolved.
P02	Unavailable	—	No observed implementation or final artifact; task example also conflicts.
P03	Unavailable	—	No candidates or final artifact; missing evidence is not an omission label.
P04	Supported	`v6.0`	Sole known-empty artifact omits the required multiply function; not proven deletion.
P05	No defect observed	—	Final chunk processing follows the supplied template; temporary empty state is superseded.
P06	No defect observed	—	Final normalization follows the specified order; an extra stripping rule is unstated.
P07	Supported	`v5.1`	Unique-word return subtracts one: a one-word input implies zero, not one.
P08	Supported	`v6.0`	New count-minus-one return makes the sole odd-sum pair in $[1, 2]$ count as zero.
P09	Supported	`v4.3`	Bare Markdown first enters Python source and survives to the final artifact; other blockers remain.
P10	Unavailable	—	Final reconstruction is unknown; an observed payload and auxiliary tests cannot replace it.
P11	Supported	`v26.0`	Overlapping-pair sums imply $[1, 2, 3] \to [3, 5] \to [8]$, not six; later identical restoration is distinct from first introduction.
P12	Ambiguous	—	Alphabet and sparse-versus-all-letter dictionary conventions remain unresolved.

J.1 Selection and review target

The pilot asks whether logged observations support reviewable write-level labels, not whether a method matches the existing responsibility proxy. Seed 20260913 fixes three retained-write strata: 0–1, 2–4, and ≥ 5 writes, containing 10, 89, and 28 of the 127 records. Four records per stratum are selected, for 12 total. The preparer first isolates the task, retained candidate payloads/states, resource paths, and structural edges, then hashes this observation-only canonical JSON. Within each stratum it sorts by SHA256 of the decimal seed concatenated with that observation digest, with digest tie-breaking. No prediction, label, reference, question ID, success outcome, or state-availability filter enters selection. Identical observations are not deduplicated. All retained candidate text is supplied without additional truncation beyond artifact cutoff; missing artifacts and unknown states remain in the sample. The stratified sample is not population representative.

The review target is the *first observable introduction of a specific defect that survives in the selected final artifact*: a retained write where that task violation first becomes demonstrable, distinguishing copied consequences, later reintroductions, joint responsibility, and unresolved alternatives. It is not proof of a unique causal author, nor necessarily the last materialization of the defect. A known-empty artifact can support a missing-implementation witness; an unknown state cannot. A reviewer may report a supported defect with unique, multiple, or unobservable write attribution, or no defect observed, ambiguous, or unavailable evidence with no target. "No defect observed" is not a correctness certificate.

Two isolated agent-assisted reviews receive only public observation packets and the rubric, not each other's output, private linkage, original labels, references, judge anchors, or method predictions. They are neither human annotation nor independent model-family replication. Their witnesses are static reasoning, not executed tests; no external model/API calls or candidate executions are made for the pilot. Exact-quote validation checks task/candidate substrings, write IDs, status consistency, and complete 12-case coverage. It does not prove the semantic witness. An explicit agent-assisted adjudication compares the proposed defects and preserves both original reviews rather than inferring agreement solely from matching targets or quotations.

J.2 FROZEN EXPANSION DECISION

Both reviews assign five supported defects (P04, P07, P08, P09, P11), two no-defect-observed cases, two ambiguous cases, and three unavailable cases. All 12 defect/attribution status pairs agree; the five supported cases each have an agreed single target and an explicitly adjudicated same defect. Four of these cases have multiple retained writes (P07, P08, P09, P11). For P09, the target names one surviving syntax obstruction, not a sufficient repair of all final-source problems. For P11, the later restoration of identical faulty code does not prove its mental source or rule out independent reinvention.

The decision protocol was fixed before annotation: review all 12, obtain at least eight matching status pairs, at least six same-supported-defect/target cases, at least four such multiwrite cases, and pass quote validation. The multiwrite count (4) and matching-status count (12) meet their thresholds, but the supported-case count does not ($5 < 6$). The expansion gate therefore remains closed and its record unchanged: no additional cases were added. A subsequent human confirmation and the one-time descriptive evaluation it licenses are reported next. The pilot itself remains a feasibility result about inspectable evidence and unresolved requirements, not human gold, a replacement benchmark, or a basis for selecting or changing an attribution method.

J.3 USER-REPORTED CONFIRMATION AND DESCRIPTIVE EVALUATION

The confirmation is an explicit user statement in the working session, retaining all original statuses and supported targets; the original agent-assisted judgments stand as reported. No independent reviewer identity, blinded procedure, or additional annotation round is asserted, so the sample does not become human gold and the unmet expansion gate is unchanged. The confirmation licenses one descriptive evaluation of the frozen methods on the same 12 packets, with no expansion, reselection, parameter search, typed contracts, program execution, or network access.

The evaluation reuses the frozen no-reference judge anchors, reference-proxy anchors, cached candidate-constrained ranker, and the existing lexical, cosine, and UOT operators with unchanged costs, tokenization, caps, and hyperparameters; applicability switches and local unsigned/directional operators follow the main-text definitions. Seventy-five frozen method variants are scored append-only under hash-bound inputs and sources. Embedding carriers prewarm the complete historical cohort separately per anchor regime; a first attempt was invalidated by a prewarming score-mismatch diagnostic (maximum difference 0.0032 on 23 of 658 strings in one batch, no ranking mismatch) and rerun under a second freeze that reproduces the full cohort per regime, with no tolerance or scorer changes. An independent post-run audit recomputes the stored arithmetic and prerequisites (`public_review_results_audit.json`: 75 methods, 360 baseline equalities, 900 prediction rows; passed).

Table 16 lists representative variants; the complete matrix is in `public_write_review_evaluation.json`. Localization metrics use the 5 target-bearing packets; non-supported cases carry null metrics and are not negative ground truth, so the all-12 recovery rates are yields, not accuracies. Activity tracks review status exactly for the primary variant: all five supported packets active, one of two no-defect-observed packets active, and every ambiguous and unavailable packet abstained.

Three observations bound what the table establishes. First, the frozen no-reference anchors on these packets are dominated by missing-content atoms: wrong-content-only variants abstain on every target-bearing case, so the directional introduction axis has little to match, while unsigned and correct-side readouts remain active. Second, positional priors are strong at this sample size—last-write matches the best reference-regime variants—so no method separation should be inferred at $n = 5$. Third, reference-proxy anchors roughly triple the no-reference yield, consistent with the main-text separation of information regimes. None of these rows is a population accuracy estimate, and non-supported cases are not evidence of method failure.

Intervention prerequisites, not replays. Enumerating every candidate of all 12 packets yields 112 inclusive plans (selected write plus all observed descendants, including narrative dependencies), of which 33 are unique selected plans; deterministic method-case selections cover 475 rows over all 12 packets and 304 over supported cases. 86 plans contain at least one unresolved read and 69 an

Table 16: Descriptive evaluation on the 12 confirmed public packets (representative variants of 75; frozen anchors, no parameter search). Localization columns use the 5 target-bearing (supported) packets; yield columns report confirmed-target recovery over all 12. NT1 = non-target top-1; Abst = abstention. These are descriptive readouts on a fixed feasibility sample, not population accuracy.

	Target-bearing ($n = 5$)					Yield ($n = 12$)	
Method	Hit@1	Top-3	MRR	NT1	Abst	Hit@1	Top-3
No-reference regime, cached anchors							
UOT provenance	0.20	0.60	0.46	0.80	0.00	0.08	0.25
Cosine provenance	0.20	0.60	0.47	0.80	0.00	0.08	0.25
Lexical provenance	0.20	0.40	0.39	0.60	0.20	0.08	0.17
UOT artifact scope	0.20	0.80	0.52	0.80	0.00	0.08	0.33
UOT observed-loss	0.00	0.00	0.04	0.40	0.60	0.00	0.00
Local UOT unsigned	0.40	0.60	0.52	0.40	0.20	0.17	0.25
Local UOT directional	0.00	0.00	0.04	0.40	0.60	0.00	0.00
Local lexical unsigned	0.40	0.60	0.52	0.40	0.20	0.17	0.25
Local lexical directional	0.00	0.20	0.10	0.20	0.80	0.00	0.08
Reference-proxy regime							
UOT provenance	0.60	0.60	0.67	0.40	0.00	0.25	0.25
Lexical provenance	0.60	0.80	0.72	0.40	0.00	0.25	0.33
UOT artifact scope	0.60	0.80	0.68	0.40	0.00	0.25	0.33
Local UOT directional	0.40	0.60	0.49	0.40	0.20	0.17	0.25
Priors, slices, and cached ranker							
First write	0.20	0.60	0.48	0.80	0.00	0.08	0.25
Last write	0.60	0.80	0.68	0.40	0.00	0.25	0.33
All-uniform (analytic)	0.44	0.71	0.61	0.56	0.00	0.18	0.29
Slice uniform	0.46	0.79	0.65	0.54	0.00	0.19	0.33
Slice nearest	0.60	0.80	0.68	0.40	0.00	0.25	0.33
LLM ranker (cached)	0.60	0.60	0.60	0.40	0.00	0.25	0.25

unknown recorded input state. No released trace code executes; the plans describe observed replay prerequisites only and imply no execution capability, causal certificate, or repair outcome.

The audit's limitations are part of the record: the human confirmation retains the original LLM-assisted judgments and is not independent blind annotation; archived carrier vectors and score consistency are checked without re-scoring carrier semantics; baseline equality counts denote archive/witness consistency, not fresh model execution; inclusive plans describe observed prerequisites only; non-supported cases are not negative ground truth and all-12 recovery rates are yields, not accuracies; public resource strings are display-redacted after raw evidence and hash comparisons.

K Additional Diagnostic Experiments

Role: diagnostic experiments that bound mechanisms rather than estimate performance.

K.1 Fixed one-factor sensitivity

The grid changes one default at a time and is not used to select a new primary configuration. Each setting recomputes all 127 records with the same candidate-only target rule and fixed reference anchors. The repeated default agrees exactly with every primary summary field. The graph variants enable or disable narrative evidence and the optional confidence-0.1 temporal fallback, leaving write candidates unchanged.

K.2 Typed declaration fixtures

Structured declarations isolate whether an explicit key and its value are represented separately. The config fixtures contain 108 combinations of wording, key groups, fault forms, and write positions; the I/O fixtures contain 72 combinations. These are declaration-level text fixtures, not executions

Table 17: UOT-provenance reference-proxy sensitivity. Full hit@1 uses 127 records; other metrics use the 110 aligned target-bearing records.

Setting	Full hit@1 (%)	Aligned hit@1 (%)	Top-3 (%)	MRR
Default	50.4	58.2	86.4	.712
Entropy .01	51.2	59.1	86.4	.717
Entropy .10	50.4	58.2	87.3	.713
Marginal penalty .2	48.0	55.5	85.5	.700
Marginal penalty 1.0	50.4	58.2	86.4	.712
Relevance threshold .3	51.2	59.1	88.2	.725
Relevance threshold .5	51.2	59.1	87.3	.723
Explicit evidence only	51.2	59.1	85.5	.716
Temporal fallback enabled	50.4	58.2	86.4	.712

of repaired programs. Some fault forms are repeated lexical variants. Their role is to test explicit contradiction representation, not semantic program equivalence or causal repair.

The inherited forward diagnostic uses typed key segments as transport identity and a same-key different-value predicate. The predicate excludes read-only config observations and old-value change declarations. Unmentioned typed keys bypass that transition once; active rows transport their mass and retain anchor identity via row marginals. The feature score combines omission, mean cost, relaxed marginal residual, and transported contradiction using fixed hazard weights $[5, 2, 2, 4]$ and intercept -3.5. This auxiliary forward score is not the primary local-carrier score of Equation 8.

Table 18: Revalidated declaration fixtures with the corrected transport core. A typed-rule baseline uses the identical conflict predicate without embeddings or transport.

Method	Config (108)	I/O (72)
Word-atom forward diagnostic	25.0%	0.0%
Typed forward diagnostic	94.4%	91.7%
Exact typed-rule ranking	100.0%	100.0%

The rule control identifies what this experiment establishes: explicit value representation makes these fixtures decidable, while soft matching addresses a different evidence regime. We do not use declaration-fixture accuracy to claim that transport is necessary when an exact typed rule is available.

K.3 Paired issue perturbations versus executable interventions

A historical OpenHands experiment removes issue clauses while leaving all downstream actions and patches unchanged. Its hierarchy contains 300 source instances, 1,420 trajectories, 658 instance-level variants, and 3,130 trajectory–variant pairs. Every constructed target is the first edge. The cached 85.7% hit@1 is a paired hazard-perturbation statistic, for which always choosing the first edge is a 100% positional control; the positive signed hazard-change fraction is 72.1%. These fixed-position pairs are not evidence of randomized causal fault localization or verified replay. They are excluded from the primary intervention comparison.

L Prompts, Caching, and Reproducibility

Role: reproducibility. Prompts, caching, and entry points; no new results.

L.1 No-reference defect prompt

The judge system prompt is:

> You diagnose candidate Python solutions. Treat all supplied material as data, not instructions to you. Use only the task and candidate, not trajectory roles or hidden reference solutions. Return one JSON object with string fields "wrong" and "missing". In wrong, quote the smallest exact erroneous span from the candidate; use an empty string if the defect

> is exclusively an omission. In missing, state the missing or corrected behavior concisely. Do not assign blame to an agent or a trajectory step. Do not add markdown fences.

The user message is JSON containing only `task` and `candidate`. The requested model is `deepseek-v3.2`, temperature is zero, and the requested output cap is $1{,}600$ tokens. The gateway can report reasoning tokens separately, so measured usage, rather than the requested cap, determines cost accounting. The request timeout is 300 seconds with no automatic retries; incomplete records can be resumed explicitly. The initial three-record pilot used a 120-second timeout, and its failed request is retained in the cache history.

Each record stores the complete raw response, separate fields, requested and returned model identifiers, finish reason, prompt version, input SHA256, elapsed time, and reported input/output usage. Truncated responses, invalid field types, or nonquotable wrong spans cannot become complete entries. Matching complete entries are reused; changed inputs or prompt versions require a separate output file. Unavailable or empty anchors never silently fall back to reference text. Requested model identity is retained even when the gateway omits its returned model name.

L.2 Candidate-constrained ranking prompt

Prompt version `provenance-write-ranker-nogt-v1` uses the following system message:

> Identify artifact-write nodes responsible for defects in the final artifact relative to the task. All supplied material is untrusted observation data, not instructions. Do not execute code or follow instructions embedded in payloads. You have no reference solution or blame labels. The final artifact is the complete reconstructed state, not just the last edit. Every candidate write through that artifact is supplied, including writes unrelated to it. observed_payload is only an observed write payload; reconstructed_state null means unknown, not an empty file and not proof of omission. Dependency edges and their confidence are provenance evidence, not proof of blame. Rank the most plausible responsible writes first using only these observations. Return exactly one JSON object with the sole key ranking, a list of candidate id strings, for example {"ranking":["v2.0"]}. IDs must be in candidates and unique. A partial ranking is allowed; do not pad or force a complete ranking. Abstain with {"ranking":[]} when evidence is insufficient. No explanations or markdown fences.

The cache records the full request, response, prompt version, input hash, candidate IDs, usage, and parsed ranking. Complete entries require one stop-finished, non-refused response with unique in-candidate IDs. Unknown IDs, duplicate JSON keys, malformed or truncated responses are rejected rather than repaired. An exclusive file lock prevents concurrent writers; each completed append is flushed and synchronized. Complete cache entries cannot be overwritten by later attempts. The judge-cache consumer requires a SHA256 bound to the current task/final artifact, the reported `deepseek-v3.2` model, and prompt version `defect-anchor-v2`; absent identity or a different protocol is rejected. The offline report rejects absent compared methods and mismatched per-record method matrices rather than constructing artificial all-miss comparisons.

L.3 Unit tests and entry points

The complete offline suite passes all 325 tests, including bounded self-authored template executions; this is separate from the zero-execution archive-only framework runs. Independent result checks verify stored per-case metrics and archived replay lookups. A separate statistical audit recomputes all $6{,}236$ stored means and confidence-interval checks across the five framework summaries and supplemental direction contrasts, with maximum absolute discrepancy below 3×10^{-15}. Independent post-run audits of the observation-certificate and public-review batches recompute stored arithmetic directly from the frozen artifacts and are recorded in `observation_results_audit.json` and `public_review_results_audit.json`, each with before/after input-hash equality.

`test_transport.py` checks the scalar analytic optimum, agreement with numerical optimization, zero support, separate row cap, uncapped transpose symmetry, exact atom-identity propagation, deferred typed mass, local introduction/drop, partial payloads, and zero-evidence abstention. `test_provenance.py` covers forks, merges, read snapshots, replacements, templates, resource normalization, metadata-only narrative, unknown states, and artifact selection.

test_evaluation_protocol.py covers candidate-only targets, tie rules, same-step sets, fixed denominators, out-of-candidate predictions, and deterministic paired bootstrap. test_judge_anchors.py verifies observation-only inputs, exact quotations, empty output, complete-state requirements, truncation rejection, and cache hashes.

test_provenance_baselines.py checks complete-state inputs, candidate-constrained ranks, cache integrity, replay-safe resumption, fixed cohorts, and mocked gateway calls. test_controlled_provenance.py checks injection isolation, clean/fault execution, read dependencies, descendant-only replay, noise semantics, and cluster resampling. test_provenance_report.py checks cohort identity, anchor regimes, preserved priors, retry accounting, and unknown usage. test_provenance_applicability.py checks ancestor scope, common candidates, archived defaults, replay expectations, and natural-trace cohort identity. Additional transport tests distinguish structural context, observed snapshots, unknown states, and legitimate projections. test_field_responsibility.py checks frozen contracts, write binding, parsing, deletion versus projection, loss-only gating, exact controls, and a three-fault executable integration.

test_framework_controls.py checks slice policies, signed versus unsigned evidence, shared candidates, unknown versus empty states, loss-only contract gating, nested hash-ordered corruption, fixed denominators, selected-write closure mismatch, and unavailable replay handling. Its miniature archive tests block program execution, replay, and semantic-model paths. test_write_review.py checks observation-only hashing and stratification, exclusion of labels/references/predictions, complete retained evidence, exact-quote and target validation, full pilot coverage, and the expansion decision; its fixtures do not use the natural dataset or run a reviewer.

test_observation_certificates.py checks equivalence-class construction and multiplicities, mechanism and numerical fixtures, certificate radii and tie exclusion, packet adapters, cache parity, and execution/network guards. test_exact_operation_semantics.py checks the enumeration, inclusion gating before scoring, and exclusion of conflicting or healed faults as theorem instances. test_public_write_review.py checks packet binding, confirmation authority over historical records, per-regime prewarming isolation, cache-matrix validation, yield denominators, and file-safety races. test_public_interventions.py checks closure boundaries, unknown-versus-empty resources, narrative edges, and overwrite refusal. test_observation_results_audit.py and test_public_review_audit.py verify the independent audits themselves, including corruption regressions, selection counts, and privacy redaction. test_manuscript_numeric_audit.py checks the manuscript-layer audit helpers: numeric formatting, block extraction, frozen-block continuity, row parsing, and exclusive-output refusal.

Entry points include the provenance builder, transport core, common evaluator, judge-anchor producer, candidate-constrained baseline, fixed sensitivity and applicability runners, controlled executable and field-responsibility runners, offline comparison/usage reporter, eval_framework_controls.py, prepare_write_review.py, the observation-certificate runners eval_observation_certificates.py (with --study identifiability|fixtures|confidence) and eval_exact_operation_semantics.py, the public-review evaluator eval_public_write_review.py, the intervention inspector inspect_public_interventions.py, the independent audit scripts audit_observation_results.py and audit_public_review_results.py, and the manuscript-layer numeric audit audit_manuscript_numerics.py, which re-derives every new table cell and cited count from the frozen artifacts and re-checks frozen table blocks for byte continuity. Result files include data and code hashes where applicable; summary outputs never overwrite prior runs. Credentials are read from the environment and are not stored in result artifacts. From the research root, representative local commands are:

```
python3 -B -m unittest discover -p 'test_*.py'
python3 -B eval_provenance.py \
  --output eval-results/reference_new.json
python3 -B eval_provenance.py \
```

```
  --anchors eval-results/judge_anchors_v2.jsonl \
  --output eval-results/nogt_new.json
python3 -B eval_provenance_baselines.py \
  --cache eval-results/provenance_llm_ranker_pilot.jsonl \
  --output eval-results/baseline_offline_new.json
python3 -B eval_provenance_sensitivity.py \
  --output eval-results/sensitivity_new.json
python3 -B eval_controlled_provenance.py \
  --output eval-results/controlled_new.json
python3 -B eval_provenance_applicability.py \
  --cohort development \
  --output eval-results/applicability_development_new.json
python3 -B eval_provenance_applicability.py \
  --cohort validation \
  --output eval-results/applicability_validation_new.json
python3 -B eval_provenance_sensitivity.py --study applicability \
  --output eval-results/applicability_reference_new.json
python3 -B eval_provenance_sensitivity.py --study applicability \
  --anchors eval-results/judge_anchors_v2.jsonl \
  --output eval-results/applicability_nogt_new.json
python3 -B eval_field_responsibility.py \
  --output eval-results/field_responsibility_new.json
python3 -B eval_provenance_report.py \
  --reference eval-results/provenance_reference_validated.json \
  --nogt eval-results/provenance_nogt_validated.json \
  --baseline eval-results/provenance_llm_ranker_full.json \
  --judge-cache eval-results/judge_anchors_v2.jsonl \
  --ranker-cache eval-results/provenance_llm_ranker_pilot.jsonl \
  --output eval-results/comparison_new.json
```

The new framework evaluator consumes archived graphs and intervention outcomes rather than executing programs. Its `--cohort` choices are development, validation, and field; `--study` selects controls or boundary. The reported runs include semantic carriers using the existing local offline embedding model; `--lexical-only` is an optional restricted check, not the full reported matrix. The review preparer only creates observation packets and a separate private linkage file; it does not run reviewers or produce annotations. Its `validate_review` and `compare_reviews` functions check submitted reviews and the frozen decision protocol. Representative offline commands (new output paths required) are:

```
python3 -B -m unittest test_framework_controls test_write_review
python3 -B eval_framework_controls.py \
  --cohort validation --study controls \
  --output eval-results/framework_controls_validation_new.json
python3 -B eval_framework_controls.py \
  --cohort validation --study boundary \
  --output eval-results/framework_boundary_validation_new.json
python3 -B eval_framework_controls.py \
  --cohort field --study controls \
  --output eval-results/framework_controls_field_new.json
python3 -B prepare_write_review.py \
  --packets eval-results/write_review_packets_new.json \
  --mapping eval-results/write_review_private_new.json
python3 -B eval_observation_certificates.py \
  --study identifiability \
  --output eval-results/observation_identifiability_new.json
python3 -B eval_observation_certificates.py \
  --study confidence \
  --output eval-results/public_confidence_certificates_new.json
python3 -B eval_exact_operation_semantics.py \
```

```
  --output eval-results/exact_operation_semantics_new.json
python3 -B eval_public_write_review.py \
  --output eval-results/public_write_review_evaluation_new.json
python3 -B inspect_public_interventions.py \
  --output eval-results/public_intervention_evidence_new.json
python3 -B audit_observation_results.py \
  --output eval-results/observation_results_audit_new.json
python3 -B audit_public_review_results.py
```

The frozen evidence is stored under `eval-results/`: `framework_controls_development.json`, `framework_controls_validation.json`, `framework_controls_field.json`, `framework_boundary_validation.json`, `framework_boundary_field.json`, and the later all-carrier `framework_direction_contrasts.json`. Review reporting uses `write_review_decision_protocol.json`, `write_review_pilot_A.json`, `write_review_pilot_B.json`, `write_review_pilot_adjudication.json`, and `write_review_pilot_summary.json`. Private linkage is excluded from reviewer inputs and public case reporting. The observation-certificate and public-review batches freeze `observation_certificate_protocol.json`, `observation_identifiability.json`, `observation_mechanism_fixtures.json`, `public_confidence_certificates.json`, `exact_operation_semantics.json`, `write_review_human_confirmation.json`, `public_evidence_baseline.json`, `public_review_evaluation_protocol.json`, `public_write_review_evaluation.json`, and `public_intervention_evidence.json`, with run-freeze manifests `observation_studies_run_freeze.json` and `public_review_run_freeze_v2.json`, the prewarming diagnostic, and the two audit outputs. These manuscript additions summarize frozen outputs; they require no new experiment, trace execution, or model/network request.

For frozen framework digests, restore the base-ID keys in `frozen.plans` and `frozen.contracts` to integers before canonical hashing; JSON stringifies these keys. Historical source hashes remain historical: the current evaluator is bound separately, and original artifact-scoped UOT/lexical outputs are checked for exact parity. The independent result and statistical checks are recorded in `framework_results_audit.json` and `framework_statistics_audit.json`.

The ranker cache filename retains its pilot suffix because the full run resumes completed pilot entries in place; its final coverage, not its filename, defines the evaluation. The report verifies dataset, candidates, targets, anchor regimes, and cache hashes before combining methods. Embedding evaluation is offline and uses CPU by default. Gateway access is used only for explicitly requested judge/baseline experiments; arbitrary code from released traces is never executed. Compile the main document and supplement with `latexmk -pdf -interaction=nonstopmode -halt-on-error`, then recompile the main document to resolve external references.