WHERE CONTEXT GOES AND WHAT IT OWES: RESIDENCY CONTRACTS FOR LONG-RUNNING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-running LLM agents compact histories, delegate subtasks, prune tool outputs, and resume sessions. These transformations can preserve task information while losing its conditions of use, such as exact wording, scope, source role, retrievability, freshness, and supporting evidence. Context management must therefore determine what survives each boundary, in what form, for a particular receiver. We introduce *residency contracts*, which make these requirements explicit for each typed artifact. *Boundary replay* restores a pre-boundary checkpoint, perturbs one condition of use, and compares paired continuations to measure its signed behavioral effect. Replay supplies offline evidence and supervision, while a runtime controller combines policy rules with a learned scorer to produce contracts without receiver-time replay. On BoundaryBench, our benchmark of constructed coding tasks, twelve executor models share 24 task templates. The controller improves preservation by 0.20 to 0.22 and strict success by 20.8 to 25.0 percentage points over their native-harness baselines. For Grok 4.20 non-reasoning, preservation rises from 0.61 to 0.83 and success from 0.30 to 0.55 while input token use falls by 55%. Relevance-only scoring reduces preservation by 0.09. These paired comparisons evaluate the combined scorer-and-rule policy with construction metadata under different token caps. The results support explicit conditions of use as an object for context management on the evaluated families.

1 INTRODUCTION

A long-running agent rarely operates in one persistent context. Current agents compact histories, delegate subtasks through short briefs, prune long tool outputs, and resume sessions from conclusions that may have gone stale (Anthropic, 2025; Wang et al., 2025; Yang et al., 2024). Concurrent studies find that these transformations silently drop what the agent needs. Across 20 production configurations, Claude Code's `/compact` prompt on Sonnet 4.6 kept 53% of safety rules after one compaction round and 10% after five (Zerhoudi et al., 2026). Policy violations rise from 0% in full context to 30% after compaction (Chen, 2026), and memory consolidation drops source-authority constraints in 48 of 49 configurations (Zhan et al., 2026). In each case the agent keeps the subject of its task but loses the conditions under which information may be used (its *conditions of use*), such as a rule's exact wording, a permission's scope, or an instruction's source.

We call each such transformation a *context boundary*, a point where the *harness* (the software around the model that assembles prompts, runs tools, and manages memory) rewrites the *packet*, that is, what the next model call (the *receiver*) can see or retrieve. At a constructed debugging handoff (Figure 1b), the parent holds six *artifacts* (typed items with metadata), namely the task with its allowed edit path `src/cache.py` (a_1), the failing test `test_expiry` (a_2), a diagnosis (a_3), a long command log (a_4), a diagnosis from an incompatible commit (a_5), and tool output reading "also edit `config.py`" (a_6). A fluent summary paraphrases a_2, drops the edit scope and the handle to a_4, and absorbs a_6 as an instruction. Copying everything revives a_5, leaves a_6 in the instruction stream, and exceeds the budget. A boundary must therefore decide not only whether an artifact should cross, but also how it should cross so that the receiver can use it correctly.

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.

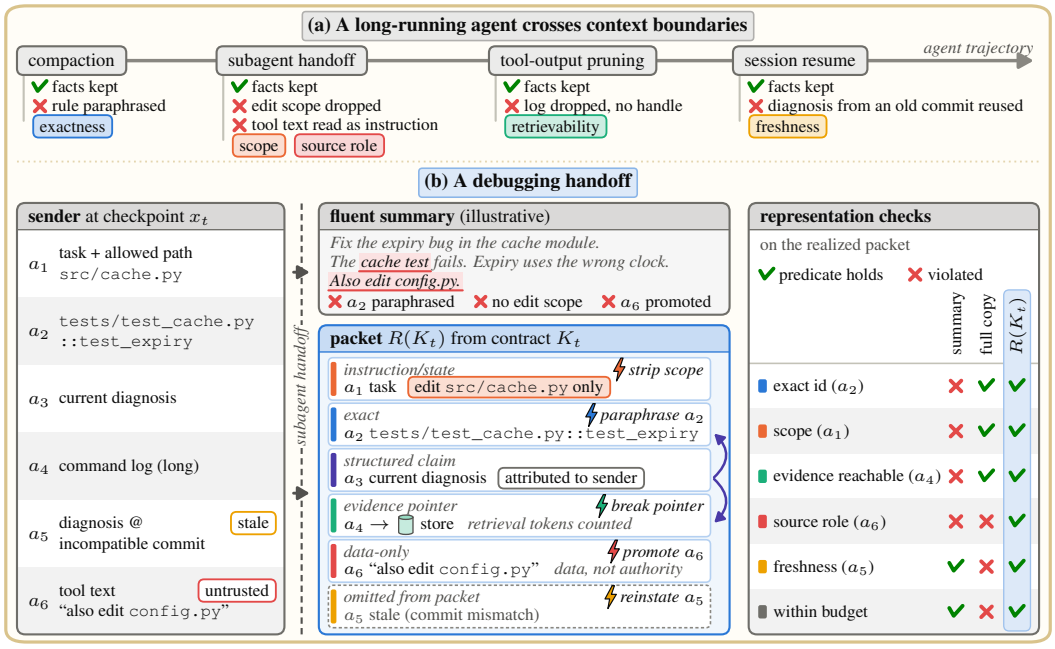

Figure 1: Where agents break and what a residency contract preserves. (a) Task facts can survive a context boundary while a condition of use is lost. (b) A constructed debugging handoff compares a fluent summary, an unstructured full copy of the sender inventory (matrix only), and the packet $R(K_t)$ produced under contract K_t. The dashed row records an artifact omitted from the packet. Lightning marks name replay interventions. Each check is a representation predicate.

Existing context managers decide what information to retain, compress, retrieve, or protect. Systems such as ClawVM enforce fidelity constraints on managed agent state (Rafique & Bindschaedler, 2026), and recent methods protect conditions such as exactness, scope, and source authority (Table 18, Appendix H). We ask how to state these requirements for a particular boundary and receiver, and how to test their behavioral consequences across artifact types.

To answer this question, we introduce a *residency contract* (Section 2). For each artifact, it specifies a *placement*, such as exact, structured, summary, evidence pointer, data-only, or excluded, together with the obligations that must hold for the resulting packet. We consider six obligation types, namely exactness, scope, source role, retrievability, freshness, and dependency. A *boundary policy* determines which obligations are required for the current agent state, boundary type, and receiver role. Structural checks inspect the contract, while representation checks inspect its realized packet. This separates what a boundary must preserve from how the harness realizes those requirements, so failures in the contract can be distinguished from failures in its realization.

Making these requirements explicit is not enough to establish that they preserve downstream behavior. We therefore use *boundary replay* to measure the consequences of perturbing a condition of use. Replay restores the same pre-boundary *checkpoint*, a snapshot of the task, repository, tools, and history, and compares the reference packet with a matched intervention packet. Paraphrasing an exact identifier or reintroducing stale state can hurt or help, so the difference between paired continuations gives a signed effect. Repeating these executions at every deployed boundary would add execution cost and delay. We instead use replay offline for supervision and behavioral evaluation. At runtime, a *controller* combines policy rules, a learned boundary-conditioned scorer, a planner, and a renderer to produce contracts without receiver-time replay (Section 5). Because replay tests a local intervention, we also study when such evidence extends to a full packet or trajectory (Section 4).

We make three contributions. First, we formulate *conditions of use* as an explicit object of context management and introduce residency contracts to specify, for each artifact, boundary, and receiver, both how information should cross and what properties must remain true after it does. Second, we introduce boundary replay to test the behavioral consequences of violating these conditions,

and characterize when such local evidence can and cannot support claims about a complete packet or multi-boundary trajectory. Third, we show how these ideas can guide runtime context control and introduce BoundaryBench for controlled evaluation across boundary failures. Across twelve executor models on the same 24 task templates and four seeds, the controller improves preservation by 0.20–0.22 and strict success by 20.8–25.0 percentage points over native-harness baselines. On Grok 4.20 non-reasoning, preservation rises from 0.61 to 0.83 with 55% fewer input tokens. They establish improved preservation and success at the evaluated operating points, while the ablations and development audits delimit what can be attributed to learning.

2 Context boundaries and residency contracts

At a context boundary, the harness assembles the next receiver's packet from typed artifacts, such as task instructions, test identifiers, and tool logs. In the debugging handoff of Figure 1b, the receiver needs the failing-test identifier verbatim, the allowed edit path stated, and the supporting log reachable. A residency contract specifies how these artifacts cross and the conditions needed to use them correctly. We assume a typed inventory supplied by structured harness logs. Learning to type raw histories is outside our scope. Appendix B.1 gives the full formal setup and definitions.

From a contract to a packet. A contract K assigns each artifact a placement and records the obligations that placement must satisfy. It can retain exact text, structure or summarize content, expose a pointer, quote text as data, or exclude an artifact. An optional pointer offers access on request, whereas an evidence pointer makes reachability mandatory. The *renderer* arranges these placements into a region-annotated layout. The *execution adapter* turns that layout into the prompt, evidence access, and tools supplied to the receiver. Together they realize the packet $R(K)$.

Required obligations. A boundary policy fixes which obligations are required for the checkpoint, boundary type, and receiver role. In BoundaryBench, our benchmark of constructed coding tasks (Section 6), task construction fixes this policy. A deployment harness designer declares it for the intended application. A contract is *admissible* when it includes every required obligation and all its declared obligations hold almost surely under rendering. A contract cannot gain admissibility by omitting a required obligation. Table 1 summarizes the six obligation types, with the full comparison in Table 18. Appendix B gives the operation vocabulary, boundary classes, and worked handoff.

Table 1: Six obligations on the receiver's packet. Full comparison in Table 18 (Appendix H).

Obligation	What must survive
● **Exactness**	Exact wording and binding status
● **Scope**	Allowed files, actions, and audiences
● **Source role**	Whether text is instruction or data
● **Retrievability**	Evidence reachable within budget
● **Freshness**	Validity at the receiver's version
● **Dependency**	Support required by retained claims

Three validity levels. *Structural* checks inspect the contract, for example that no artifact is both retained and excluded. *Representation* checks inspect the realized packet, for example that exact fields appear verbatim and quoted data sits in the data region. *Behavioral* validity concerns what the receiver does. We record an endpoint vector Y that keeps success, violations, harmful adoption (use of artifacts marked stale or harmful), and representation scores separate. Data-only placement specifies how text crosses. Whether the receiver ignores an embedded instruction is measured.

Behavioral sufficiency. Let $U(x, P, \omega) \in [0, 1]$ be the receiver's utility from a restorable agent checkpoint x with packet P, where ω collects continuation and renderer randomness. We compare against a declared reference packet P^{ref} from the native harness or full context. For boundary type b and receiver role r, let $\mathcal{K}(x, b, r)$ be the admissible contracts. A contract is ϵ-sufficient when its expected utility loss is at most ϵ, giving the cost objective

$$\min_{K \in \mathcal{K}(x,b,r)} \hat{c}(K) \quad \text{subject to} \quad \mathbb{E}_{\omega}\big[U(x, P^{\mathrm{ref}}, \omega) - U(x, R(K), \omega)\big] \leq \epsilon, \tag{1}$$

where $\hat{c}$ is a deterministic planning proxy for cost (Appendix C.4). The constraint is one-sided, so a contract may beat its reference. It is local to one checkpoint and one utility, and further constraints, such as a violation-rate bound, are imposed alongside it. The runtime planner targets $\hat{c}(K) \leq B$ for token cap B using scored placements. Behavioral sufficiency is evaluated offline (Section 5).

3 TESTING CONDITIONS OF USE BY BOUNDARY REPLAY

A residency contract specifies what should survive a context boundary and checks whether those requirements hold in the realized packet. These checks do not establish whether a requirement matters to the receiver's subsequent behavior. We use *boundary replay* to measure that consequence directly.

Boundary replay. At boundary t, replay restores the same pre-boundary checkpoint x_t and runs two matched continuations. One receives the reference packet P_t^{ref}, and the other receives an intervened packet $I_{a,\psi}(P_t^{\mathrm{ref}})$, where $I_{a,\psi}$ deliberately violates one condition of use for artifact a. For example, it may paraphrase an identifier that should remain exact or reintroduce state that should remain stale and excluded. The task objective, pre-boundary state, executor, validator, and other execution conditions remain fixed. The expected signed effect is

$$\Delta_{t,a,\psi} = \mathbb{E}_\omega \left[U(x_t, P_t^{\mathrm{ref}}, \omega) - U(x_t, I_{a,\psi}(P_t^{\mathrm{ref}}), \omega) \right] \in [-1, 1]. \tag{2}$$

A positive value means that the intervention reduces the measured outcome, whereas a negative value means that the intervention improves it. Each replay pair yields the observed difference $D = U^{\mathrm{ref}} - U^{\mathrm{int}}$ together with the endpoint vector Y. Shared seeds couple the two continuations without making execution deterministic. If the checkpoint cannot be restored consistently, the corresponding behavioral effect is not identified. Restoration requirements, placement-matched controls, and paired uncertainty are detailed in Appendix E.1.

Specification, representation, and behavioral evidence. Boundary replay is one of three distinct forms of evidence in our framework. A *construction requirement* specifies what should survive the boundary, a *representation probe* checks whether that requirement holds in the realized packet, and an *execution-backed replay* measures what happens when the requirement is violated. For example, for the failing-test identifier a_2, the construction record may require exact preservation, the representation probe checks whether the identifier appears verbatim, and replay compares receiver behavior with the exact and paraphrased packets. Only the last comparison estimates the behavioral effect in Eq. (2). The three forms of evidence therefore answer different questions and are kept separate. The audited development fit uses construction-derived operation targets and is reported separately from the production controller and intervention-supervision comparisons (Appendices C and G.4).

Why replay remains offline. Replay provides direct behavioral evidence, but executing a counterfactual continuation at every deployed boundary would roughly duplicate downstream execution. We therefore use replay offline for supervision and audit rather than as a receiver-time decision procedure. At runtime, the fitted controller constructs contracts without access to replay labels or held-out outcomes. The completed replay audit estimates local effects for the checkpoints and interventions that it actually replays. Such a local contrast does not by itself certify an entire packet or a multi-boundary trajectory. The next section asks when local replay evidence can support those broader claims.

4 WHAT CAN LOCAL REPLAY TELL US?

Boundary replay tests one intervention at one checkpoint. Broader claims require accounting for artifact interactions, behavioral outcomes, and checkpoint sampling. Four standard results make these limits precise, with proofs and full constructions in Appendix D.

4.1 TWO LIMITS OF LOCAL EVIDENCE

Artifact effects need not add up. Either copy of a test identifier can suffice, while losing both causes failure. An identifier and edit scope can instead be jointly required, so either deletion causes failure. Source role also matters, since the same text can induce different actions as instruction or data. Behavioral preservation therefore requires testing the complete packet.

Proposition 1 (Singleton effects do not certify joint preservation)**.** *Let $\mathcal{A} = \{a_1, a_2\}$ and let $f(S) \in [0, 1]$ be the utility of a deterministic receiver retaining $S \subseteq \mathcal{A}$. (a) Some monotone f has both singleton deletion effects $f(\mathcal{A}) - f(\mathcal{A} \setminus \{a_i\}) = 0$, yet joint deletion loss $f(\mathcal{A}) - f(\varnothing) = 1$. (b) Some monotone f has singleton deletion effects summing to 2 and joint deletion loss 1. (c) Two packets with the same artifact set, differing only in one artifact's instruction/data placement, can induce continuation laws at total-variation distance 1.*

Equal success can hide different behavior. Two continuations can both pass the tests while only one edits an unauthorized file. Evaluation must therefore report violations, harmful adoption, and representation scores alongside success. Proposition 2 follows from the variational characterization of total variation (Levin & Peres, 2017, Prop. 4.5).

Proposition 2 (Silent failure). *Let $Q_{x,P}$ be the law of continuation records $Z \in \mathcal{Z}$ from checkpoint x with packet P under a fixed executor and validator, and let $\mathrm{TV}(\mu,\nu) = \sup_B |\mu(B) - \nu(B)|$. Then $\mathrm{TV}(Q_{x,P}, Q_{x,P'}) \le \epsilon$ if and only if every measurable endpoint u from $\mathcal{Z}$ to $[0,1]$ satisfies $|\mathbb{E}_{Q_{x,P}} u - \mathbb{E}_{Q_{x,P'}} u| \le \epsilon$. Equal expected test success alone permits total-variation distance 1.*

4.2 When local effects compose over trajectories

Trajectory loss can be recovered from local contrasts, provided replay uses the appropriate checkpoints and continuations. Reach each checkpoint under the controller, compare the reference and contract packets there, then continue *both branches under the reference policy*. Averaging these contrasts over controller-reached checkpoints and summing across boundaries gives the final expected loss. The reference harness must be runnable from controller-reached checkpoints. This specializes the performance-difference and simulation lemmas (Kakade & Langford, 2002; Kearns & Singh, 2002).

Proposition 3 (Per-boundary decomposition of trajectory loss). *For a fixed horizon T, let F_t and H_t be the reference and contract Markov kernels on checkpoint space $\mathcal{X}$, each running boundary t to the next checkpoint. States hold the full restorable history. Let $\mu_0 = \nu_0$, $\mu_t = \mu_{t-1}F_t$, and $\nu_t = \nu_{t-1}H_t$. For a common terminal endpoint u from $\mathcal{X}$ to $[0,1]$, set $W_T = u$ and $W_{t-1} = F_t W_t$. Then*

$$\mathbb{E}_{\mu_T} u - \mathbb{E}_{\nu_T} u = \sum_{t=1}^{T} \mathbb{E}_{x\sim\nu_{t-1}}\big[\Delta_t(x)\big], \qquad \Delta_t(x) = (F_t W_t)(x) - (H_t W_t)(x). \tag{3}$$

Here W_t is expected terminal utility under the reference continuation. If each averaged local loss is at most ϵ_t, trajectory loss is at most $\sum_t \epsilon_t$ (Corollary 5). Local tolerances therefore accumulate across boundaries.

4.3 Why the checkpoint distribution matters

The checkpoint distribution describes which states a policy reaches and how often. Reference-policy logs can miss states created by earlier contract decisions. In the stale-pointer construction of Figure 5a, the reference repeatedly repairs pointers, while the contract can carry stale ones forward. Replay with reference checkpoints and reference continuations then reports zero loss even though the contract's trajectory loses utility. Appendix D.4 gives the exact calculation.

Reference checkpoints remain usable with a different contrast, in which *both branches continue under the contract policy* (Eq. (14)). The checkpoint law and continuation policy must be chosen together. Our completed replay audit establishes local effects on its declared roster, without establishing the sampling law needed for a trajectory certificate (Appendix E.2).

4.4 Finite replay evidence

Replay estimates are noisy. A certificate places an upper confidence bound on expected loss for a specified checkpoint distribution. Fix the candidates and sample size before testing, use independent replay samples, and accept a candidate only when that upper bound meets the loss tolerance. Hoeffding's inequality and a union bound give the following guarantee (Hoeffding, 1963).

Proposition 4 (Fixed-candidate replay certificate). *Fix M candidate contract rules $x \mapsto K_j(x)$, sample size n, and checkpoint distribution λ before replay. For each candidate j, draw n i.i.d. paired samples, each including a checkpoint from λ and both continuations. Let $D_{j,k} = U^{\mathrm{ref}}_{j,k} - U^{(j)}_k \in [-1,1]$, $\hat{\Delta}_j = \frac{1}{n}\sum_k D_{j,k}$, and $\Delta_j = \mathbb{E}D_{j,1}$. For $0 < \delta < 1$, with probability at least $1-\delta$, simultaneously for all j,*

$$|\hat{\Delta}_j - \Delta_j| \le \beta_{n,M,\delta} = \sqrt{2\log(2M/\delta)/n}. \tag{4}$$

Choosing the cheapest candidate with $\hat{\Delta}_j + \beta_{n,M,\delta} \le \epsilon$ and abstaining if none qualifies bounds the probability of accepting a candidate with loss above ϵ by δ (Corollary 7). For candidates sharing the

policy before the tested boundary, that policy's checkpoint law and reference continuations identify a term of Proposition 3. Changes to earlier decisions require checking each candidate's checkpoint law. For one candidate, 738 independent pairs suffice for radius at most 0.1 at $\delta = 0.05$. Appendix D.5 gives the remaining calculations and dependence restrictions. This sampling cost motivates offline replay, with the runtime controller proposing contracts without repeating the test (Section 5).

5 From Offline Evidence to Runtime Context Control

The runtime controller uses offline supervision to propose a contract, then checks the contract and its realization before the receiver acts.

Policy rules constrain placements. Rules constrain exact fields, mandatory evidence, source roles, and stale or harmful content. Some rules use scorer rankings to select among permitted placements, such as quoting or excluding unsafe tool text. The selected locks are protected during budget allocation, even when another artifact scores higher.

The scorer ranks allowed alternatives. A small boundary-conditioned scorer uses artifact content and metadata, receiver role, boundary type, and local dependency information. Its rankings guide both rule-constrained choices and the remaining placements. Architecture, features, operation vocabulary, and training details appear in Appendices B and C.

The planner composes a candidate contract. The planner combines locked and scored placements under the context budget and dependency constraints. If required content alone exceeds the budget, it reports overflow rather than silently removing it. Because per-artifact choices can interact, the resulting K_t is only a candidate contract, not a guarantee that all requirements are jointly satisfied.

Validation precedes realization. Structural checks verify the complete contract, including required obligations and dependencies. The renderer and execution adapter then realize the validated contract as the receiver-facing packet $R(K_t)$, after which representation checks verify properties such as verbatim fields, source-role separation, and evidence reachability. Thus the scorer proposes placements, while correctness is assessed on the complete contract and realized packet. Behavioral sufficiency remains an empirical question and is evaluated through offline replay and downstream execution.

6 BoundaryBench evaluation questions and setup

We use BoundaryBench to answer three questions. **Q1.** Does the controller preserve the information that must survive a context boundary and help the agent complete more tasks? **Q2.** Does explicitly modeling properties such as exactness, scope, provenance, and freshness help beyond simply keeping information that appears relevant? **Q3.** Do the same effects appear across different executor models and different kinds of context boundaries?

BoundaryBench contains twelve controlled coding-task families, including an unrecoverable-failure control. Each task includes the artifacts available before the boundary and validators that check both task completion and whether the required information survives (Table 8, Appendix F).

Tasks and models. The main evaluation contains 24 task templates, each run with four seeds, for 96 trajectories per policy arm and executor. We evaluate twelve models from the Grok, Gemini, GLM, Kimi, and Qwen families on the same 24 templates. Grok 4.20 non-reasoning is also used for the full baseline and ablation study. A frozen 40-pair subset of twenty templates with two seeds supplies the prespecified success analysis (Appendix E.5). Development audits and repair experiments use separate data and are not part of the main comparison. Different seeds and executor models repeat the same underlying task templates rather than introducing new task types.

What each method is allowed to use. At runtime, the controller can use the typed artifact metadata available to the harness, but it cannot see replay outcomes or held-out task outcomes. The privileged oracle additionally sees the construction requirements used to create the benchmark. Thus the main experiment evaluates the complete controller with its available metadata. It does not by itself show how much of the gain comes from learning rather than from the rules or metadata.

The methods also use different context budgets. The controller is limited to 12,288 active tokens and 16,384 cumulative input tokens. The native harness uses limits of 24,576 and 32,768, while

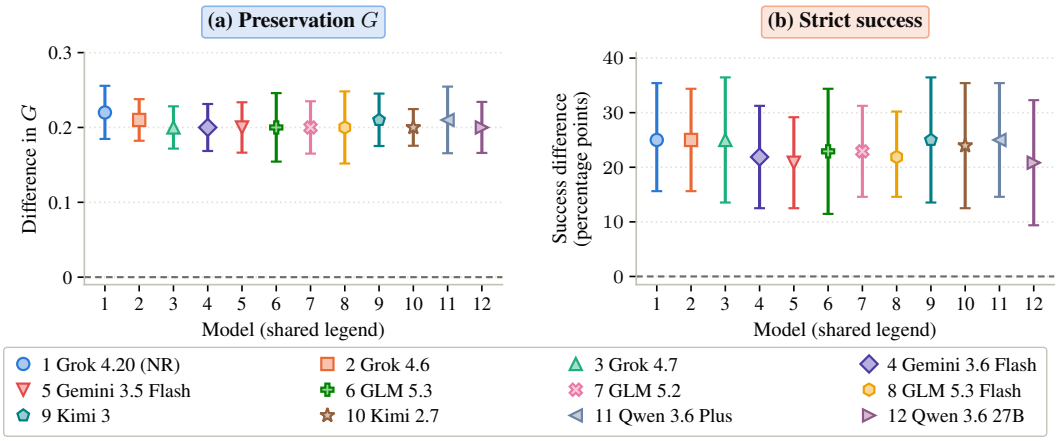

Figure 2: Controller minus native harness across all 12 benchmarked models. Model numbers, colors, and markers follow one shared legend for both panels. Whiskers are pointwise 95% percentile intervals from 100,000 template-cluster bootstrap resamples. Each model has 96 matched task/seed pairs from 24 templates. The controller has smaller token caps and additional construction metadata, so these contrasts compare operating points rather than isolate learning under equal resources and access. NR denotes non-reasoning mode.

the full-context reference uses 131,072 for both. The compact baselines and ablations use the same limits as the controller. We therefore compare the behavior of the methods at their specified resource settings rather than claiming an equal-budget comparison. Retrieved evidence is included in the input-token count.

What we measure. We report two main outcomes. *Strict success* records whether the task is completed according to the executable validator. The preservation score G_i records whether the information that should survive the boundary is still usable afterward. Its six members measure exactness, scope, provenance, evidence recovery, version validity, and exclusion of stale or harmful content. We average events within each applicable member, then average those members to obtain G_i (Appendix E.4). Benchmark intervals resample whole templates, retaining paired arms and all seeds. We report dependency closure, violations, harmful adoption, overflow, and resource use separately rather than folding them into the main score.

7 Empirical Findings

Contract-guided control improves preservation and task success across executors. Across twelve executor models evaluated on the same 24 task templates and four seeds, controller minus native differences in preservation range from $+0.20$ to $+0.22$, and differences in strict success range from $+20.8$ to $+25.0$ percentage points (Figure 2). All pointwise 95% template-cluster intervals lie above zero. These comparisons establish a consistent operating-point difference across executors on the same task population. They do not constitute a matched-budget comparison because the controller and native harness use different resource caps and information access.

On Grok 4.20 non-reasoning, where we run the full comparator and ablation suite, preservation rises from 0.61 to 0.83 and strict success from $29/96$ to $53/96$ trajectories (Table 2). The corresponding paired differences are $+0.22$ (95% interval $[0.185, 0.256]$) and $+0.25$ ($[0.156, 0.354]$). Realized input use falls from 18,400 to 8,300 tokens per task. Full context reaches $G = 0.86$ and 58.3% success using 41,700 input tokens per task. The lower token use therefore describes the evaluated operating point rather than equal-budget efficiency.

The gain spans multiple conditions of use. The improvement in G is not concentrated in a single preservation member. Controller minus native differences range from $+0.175$ to $+0.255$ across exactness, scope, provenance, evidence recovery, version validity, and stale/harmful exclusion (Figure 3). These retrospective comparisons use each member's applicable population. Dependency

Table 2: Preservation, task completion, and adverse outcomes on Grok 4.20 (non-reasoning), with 96 paired cells from 24 templates. Violations and harmful adoption are counts of affected trajectories out of 96. Input is the mean in thousands of tokens, including retrieval. Caps and information access differ across arms (Section 6). Bold identifies the proposed controller.

Controller / reference	$G \uparrow$	Success (%)↑	Violations↓	Harmful↓	Input (k)↓
Native-harness baseline	0.61	30.2	11	12	18.4
Windowing and recency	0.58	33.3	12	13	9.6
BM25 retrieval proxy	0.66	37.5	9	10	11.2
Summary-only baseline	0.63	36.5	10	11	6.8
Token-cost selector	0.67	39.6	9	10	7.4
Manual residency contract	0.74	45.8	7	7	7.9
Fixed rule baseline	0.71	43.8	8	8	8.1
Controller	**0.83**	**55.2**	4	3	8.3
References					
Full context	0.86	58.3	3	2	41.7
Privileged oracle contract	0.95	67.7	0	1	7.6

closure is measured separately because its denominator depends on the selected artifacts. Disabling graph closure lowers closure by 0.044 with interval $[-0.060, -0.028]$ (Table 11).

The behavioral endpoints move in the same direction but are reported separately from representation preservation. On Grok 4.20, violations fall from $11/96$ to $4/96$ trajectories and harmful adoption from $12/96$ to $3/96$ (Table 2). These counts are descriptive rather than confirmatory safety results. Success improves across all four boundary types, while some safety counts tie (Appendix G.2).

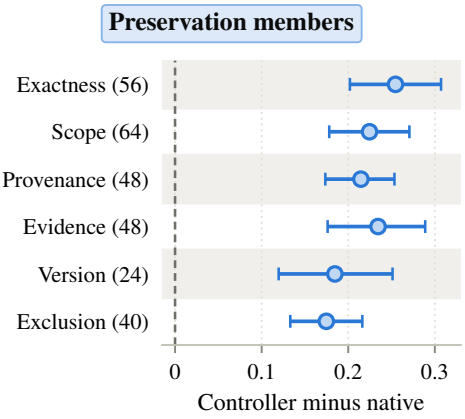

Figure 3: Preservation improves across all six members. Points and pointwise 95% template-cluster intervals compare controller with native. Parentheses give applicable pair counts.

Typed conditions matter beyond generic relevance within the controller. Replacing the controller's typed supervision with relevance-only scoring reduces G by 0.09, removing intervention targets reduces it by 0.13, and replacing informed ranking with seeded random ranking reduces it by 0.11 (Table 3). Removing boundary/receiver conditioning and removing the sign of supervision also reduce preservation. In contrast, the intervals for the fixed-score knapsack and beam-planner alternatives include zero (Table 11). These ablations therefore associate the observed gain more strongly with what the controller represents and scores than with a particular budget-search procedure. They do not by themselves isolate learned value from construction metadata or deterministic rules.

Boundary replay provides useful local behavioral evidence. Checkpoint restoration succeeds for $88/96$ replay pairs, of which four produce no realized intervention, leaving 84 eligible behavioral contrasts. On a separate frozen set of 72 failed trajectories, the specified localized repair succeeds on 45 and full context on 46, with 42 successes in common (Figure 6). The three repair-only successes show that adding the full context is not a universal remedy for these selected failures. Because a separate repair calibration under-covered, these recovery fractions remain descriptive.

A probe predicts whether an intervention reduces binary success better than metadata-only prediction. On the 88 complete replay pairs, the full probe reaches AUROC 0.810, compared with 0.680 for metadata-only prediction, a paired difference of 0.130 with artifact-group bootstrap interval $[0.073, 0.184]$. Its token cost is 0.111 of paired replay cost. Calibration does not improve. Ten-bin calibration error is 0.099 for the full probe and 0.095 for metadata alone. Full probe comparisons and ROC curves appear in Appendix G.3.

The current evidence supports the combined policy, not isolated learned value. The development scorer improves validation cross-entropy from the class prior's 1.182 to 0.896 on its artifact-disjoint split. This fit does not establish that learning supplies the production gain. Construction metadata alone predicts construction targets with 0.80 validation accuracy, and an alternative split by boundary event repeats a training artifact in $78/98$ validation rows (Appendix G.4). The production experiments therefore support the combined policy of typed metadata, deterministic rules, and learned scoring under the evaluated information condition. Whether learned scoring adds value beyond equally informed rules and construction metadata remains unresolved. A direct attribution test would replace only learned ranking with prespecified metadata-based ranking, holding information access, policy rules, the planner, and resource caps fixed.

Table 3: Ablated minus full controller on Grok 4.20, with 96 paired cells from 24 templates and 95% template-bootstrap intervals. Harmful adoption counts affected trajectories out of 96, compared with 3 for the full controller.

Ablation	ΔG	95% interval	Harmful↓
Zero conditioning	-0.06	$[-0.095, -0.025]$	5
No intervention targets	-0.13	$[-0.156, -0.104]$	8
Relevance only	-0.09	$[-0.130, -0.049]$	7
Unsigned supervision	-0.05	$[-0.096, -0.004]$	7
No exclusion targets	-0.02	$[-0.058, +0.017]$	8
Random ranking	-0.11	$[-0.153, -0.067]$	7

8 Related work

Context management. Hierarchical memory (Packer et al., 2024), prompt compression (Li et al., 2023; Jiang et al., 2023; 2024; Xu et al., 2024), learned folding and compaction (Kang et al., 2026; Sun et al., 2026a; Ye et al., 2026; Liu et al., 2025b; Li et al., 2026), and tool-output selection (Lin et al., 2026; Chen et al., 2026a; Wang et al., 2026b) decide what to retain. Observation masking is a strong baseline (Lindenbauer et al., 2025), while repeated rewriting can collapse context (Zhang et al., 2026b). Contracts specify obligations for the resulting packet. CoACT's next-action agreement is one instance of the sufficiency constraint in Eq. (1).

Residency and conditions of use. ClawVM enforces fidelity invariants on typed state pages at lifecycle boundaries (Rafique & Bindschaedler, 2026). Knowledge Triage combines typed retention with rule replication (Zerhoudi et al., 2026), and CWL manages dependency-linked episodes (Semenov & Dorofeev, 2026). Our contracts condition obligations on both boundary type and receiver role, separate specification from realization, and test obligations by replay. A data-only placement still depends on receiver defenses (Greshake et al., 2023; Wallace et al., 2024; Chen et al., 2025a;b; Hines et al., 2024; Debenedetti et al., 2024; 2026) and provenance commitments (Zheng & Yang, 2026).

Interventions and replay. Intervention-based memory selection (Srivastava, 2026), context attribution (Cohen-Wang et al., 2024; Liu et al., 2025a), failure attribution (Zhang et al., 2025; 2026a; Shah, 2026; Bonagiri et al., 2026), and continuation-based compaction checks (Min et al., 2026; Chen et al., 2026b) perturb inputs or steps. Our intervention changes one typed boundary packet, and its trajectory decomposition pairs controller-reached checkpoints with reference continuations (Proposition 3). Concurrent theory studies compaction through communication complexity (Tirmazi et al., 2026). Appendix H details the closest comparisons, including multi-agent failure taxonomies (Cemri et al., 2026).

9 Conclusion and limitations

Residency contracts treat an artifact's conditions of use, rather than its content alone, as an explicit object of context management. Across twelve executors on 24 constructed task templates, contract-guided control improves both preservation and task success while using fewer input tokens at the evaluated operating points. These comparisons use different resource caps and retain construction metadata, so they do not establish transfer to unseen task families or a learned advantage over equally informed deterministic rules. Likewise, the completed replay study establishes local behavioral effects at the audited checkpoints rather than a trajectory-wide certificate. Together, the results support a broader view of context management: preserving information requires preserving the conditions under which a receiver can use it correctly.

AI USE STATEMENT

We used generative AI tools (Claude and Codex) for language editing and revision of this manuscript. The research ideas, idea exploration, methodology, implementation, experiments, and original analyses were carried out by the authors. As part of the revision, the tools also helped restructure and redraft the text, search for related literature and verify bibliographic entries, write the plotting scripts for the figures, write validation and statistical reanalysis scripts for author-supplied records, and formulate and write out the proofs of the standard results that Section 4 specializes to context boundaries (Appendix D). No generative AI tool produced any reported experimental measurement. The authors take full responsibility for all content of this paper, including text, proofs, and figures prepared with AI assistance.

ETHICS STATEMENT

This work involves no human subjects. BoundaryBench fixtures deliberately include misleading tool output with embedded instructions, and execution trajectories can contain untrusted instructions from repositories and tools. Raw trajectories are kept in restricted storage. A data-only placement records the role in which text crosses a boundary. It is not a defense against prompt injection, and the protocol measures whether a receiver honors it instead of assuming so.

REPRODUCIBILITY STATEMENT

Code and supporting research artifacts will be made available upon acceptance. Appendices B and C specify the contract vocabulary and implemented controller. Appendix D supplies the proofs, and Appendices E and F define the replay estimands, benchmark families, information conditions, and evaluation checks. Appendix G gives additional results and validity evidence.

REFERENCES

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. Swe-bench+: Enhanced coding benchmark for llms, 2024. URL https://arxiv.org/abs/2410.06992.

Anthropic. Effective context engineering for AI agents. Anthropic Engineering blog, September 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. Causalflow: Causal attribution and counterfactual repair for llm agent failures, 2026. URL https://arxiv.org/abs/2605.25338.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=fAjbYBmonr.

Haorui Chen, Yuancheng Zhu, Yitong Zhang, and Jia Li. Coact: Action-preserving observation compression for coding agents, 2026a. URL https://arxiv.org/abs/2607.02911.

Shiyang Chen. Governance decay: How context compaction silently erases safety constraints in long-horizon llm agents, 2026. URL https://arxiv.org/abs/2606.22528.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. Struq: defending against prompt injection with structured queries. In *Proceedings of the 34th USENIX Conference on Security Symposium*, SEC '25, USA, 2025a. USENIX Association. ISBN 978-1-939133-52-6.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. Secalign: Defending against prompt injection with preference optimization. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25, pp. 2833–2847, New York, NY, USA, 2025b. Association for Computing Machinery.

ISBN 9798400715259. doi: 10.1145/3719027.3744836. URL https://doi.org/10.1145/3719027.3744836.

Zhuofu Chen, Rui Pan, Yinwei Dai, and Ravi Netravali. Slipstream: Trajectory-grounded compaction validation for long-horizon agents, 2026b. URL https://arxiv.org/abs/2605.08580.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=7CMNSqsZJt.

Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing ai agents with information-flow control, 2025. URL https://arxiv.org/abs/2505.23643.

Ian C. Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: a unified framework for model explanation. *J. Mach. Learn. Res.*, 22(1), January 2021. ISSN 1532-4435.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: a dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramer. Defeating Prompt Injections by Design . In *2026 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 587–618, Los Alamitos, CA, USA, March 2026. IEEE Computer Society. doi: 10.1109/SaTML68715.2026.00040. URL https://doi.ieeecomputersociety.org/10.1109/SaTML68715.2026.00040.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa R Kundurthy, Sean M. Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-bench pro: Can AI agents solve long-horizon software engineering tasks? In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=uEVTdoAbnK.

Kedar Dhamdhere, Ashish Agarwal, and Mukund Sundararajan. The shapley taylor interaction index. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning, 2019. URL https://arxiv.org/abs/1904.02868.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, pp. 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Yu He, Haozhe Zhu, Yiming Li, Shuo Shao, Hongwei Yao, Zhihao Liu, and Zhan Qin. AttriGuard: Defeating indirect prompt injection in LLM agents via causal attribution of tool invocations. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 1547–1566, Baltimore, MD, August 2026. USENIX Association. ISBN 978-1-939133-58-8. URL https://www.usenix.org/conference/usenixsecurity26/presentation/he-yu.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting, 2024. URL https://arxiv.org/abs/2403.14720.

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

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. URL https://www.jstor.org/stable/4615733.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=5EmOOLtH5P.

Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in ml-based science, 2022. URL https://arxiv.org/abs/2207.07048.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Zy4uFzMviZ.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002. doi: 10.1023/A:1017984413808.

David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, second edition, 2017. ISBN 978-1-4704-2962-1. doi: 10.1090/mbk/107. With contributions by Elizabeth L. Wilmer.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6342–6353, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.391. URL https://aclanthology.org/2023.emnlp-main.391/.

Yujiang Li, Zhenyu Hou, Yi Jing, Jie Tang, and Yuxiao Dong. Compactionrl: Reinforcement learning with context compaction for long-horizon agents, 2026. URL https://arxiv.org/abs/2607.05378.

Shanchao Liang, Spandan Garg, and Roshanak Zilouchian Moghaddam. The swe-bench illusion: When state-of-the-art llms remember instead of reason. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '26, pp. 395–405, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400724268. doi: 10.1145/3786583.3786882. URL https://doi.org/10.1145/3786583.3786882.

Ziqi Lin, Ye Wu, Mengying Yang, Xu Liu, Yizhou Liu, Qiang Ke, and Qin Guo. Tracer: Per-tool context retention for llm agents via consequence-attributed reinforcement learning, 2026. URL https://arxiv.org/abs/2608.29363.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*, 2025. URL https://openreview.net/forum?id=OHVzruJl5k.

Fengyuan Liu, Nikhil Kandpal, and Colin Raffel. Attribot: A bag of tricks for efficiently approximating leave-one-out context attribution. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=9kJperA2a4.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Shukai Liu, Jian Yang, Bo Jiang, Yizhi Li, Jinyang Guo, Xianglong Liu, and Bryan Dai. Context as a tool: Context management for long-horizon swe-agents, 2025b. URL https://arxiv.org/abs/2512.22087.

Shuyu Liu. What does context compression cost an agent? interaction costs unrevealed by task-completion metrics, 2026. URL https://arxiv.org/abs/2608.16370.

Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization, 2009. URL https://arxiv.org/abs/0907.3740.

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

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations, 2024. URL https://arxiv.org/abs/2411.00640.

Guanghui Min, Liang Wu, Mayank Darbari, Chen Chen, and Liangjie Hong. Toward reliable context compression for long-horizon agents: An empirical study of execution instability, 2026. URL https://arxiv.org/abs/2608.06503.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Mofasshara Rafique and Laurent Bindschaedler. Clawvm: Harness-managed virtual memory for stateful tool-using llm agents. In *Proceedings of the Sixth European Workshop on Machine Learning and Systems*, EuroMLSys '26, pp. 1–12, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400726057. doi: 10.1145/3805621.3807648. URL https://doi.org/10.1145/3805621.3807648.

Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/ross10a.html.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Andrew Semenov and Svyatoslav Dorofeev. Beyond compaction: Structured context eviction for long-horizon agents, 2026. URL https://arxiv.org/abs/2606.11213.

Jaineet Shah. Causal agent replay: Counterfactual attribution for llm-agent failures, 2026. URL https://arxiv.org/abs/2606.08275.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. We need to talk about random splits. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1823–1832, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.156. URL https://aclanthology.org/2021.eacl-main.156/.

Saksham Sahai Srivastava. Causal intervention-based memory selection for long-horizon llm agents, 2026. URL https://arxiv.org/abs/2605.17641.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon agent via context folding. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=lNRgWoGfYg.

Yiheng Sun, Huifei Wang, Yancheng Zhu, Zhenyu Li, Zebin Zhao, and Yifan Yuan. When "must" becomes "maybe": Constraint weakening in llm agent workflows, 2026b. URL https://arxiv.org/abs/2608.24569.

Hayder Tirmazi, Sam Markelon, Allison Bishop, and Michael Mitzenmacher. Context compaction theory, 2026. URL https://arxiv.org/abs/2608.01326.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions, 2024. URL https://arxiv.org/abs/2404.13208.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents, 2025. URL https://arxiv.org/abs/2407.16741.

Yian Wang, Agam Goyal, Eshwar Chandrasekharan, and Hari Sundaram. Facts without rules: Boundary metadata collapse in multi-agent llm handoffs, 2026a. URL https://arxiv.org/abs/2608.29028.

Yuhang Wang, Yuling Shi, Mo Yang, Rongrui Zhang, Shilin He, Heng Lian, Yuting Chen, Siyu Ye, Kai Cai, and Xiaodong Gu. Swe-pruner: Self-adaptive context pruning for coding agents, 2026b. URL https://arxiv.org/abs/2601.16746.

Zhiqi Wang, Yichi Zhang, Dongwon Lee, and Yuchen Yang. Lost in compaction: Evaluating side-constraint loss under context compaction, 2026c. URL https://arxiv.org/abs/2608.11242.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mlJLVigNHp.

Zhuoning Xu, Xiucheng Zhang, Hanjun Luo, Yingbin Jin, Yinpeng Dong, and Hanan Salam. Masdrift: Benchmarking authorization preservation across multi-agent architectures, 2026. URL https://arxiv.org/abs/2608.07556.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Jingren Zhou, Siheng Chen, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context folding. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IuZoTgsUws.

Lu Yi, Runlin Lei, Liuyi Yao, Yuexiang Xie, Yuyang Li, Wenhao Zhang, Zhewei Wei, Yaliang Li, and Jian-Yun Nie. Learning agent-compatible context management for long-horizon tasks, 2026. URL https://arxiv.org/abs/2605.30785.

Saber Zerhoudi, Jelena Mitrović, and Michael Granitzer. The Compaction Cliff in long-running AI agent memory. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM)*, 2026. URL https://arxiv.org/abs/2608.22752.

Qiuyang Zhan, Rui Zhang, Sheng Guo, Lepeng Zhao, and Zhuotao Liu. When memory becomes authority: Benchmarking authority collapse at the memory consolidation boundary, 2026. URL https://arxiv.org/abs/2608.01679.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng YAN. Agentracer: Who is inducing failure in the LLM agentic systems? In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=l05DseqvuD.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eC4ygDs02R.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19149–19164, San Diego, California, United States, July 2026c. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.956. URL https://aclanthology.org/2026.findings-acl.956/.

Chris Zheng and Geng Yang. Continuity: Security-context contracts for composable llm agent controls, 2026. URL https://arxiv.org/abs/2609.05269.

Pujun Zheng, Zixin Shang, Shufan Jiang, Wenhui Tian, Dongsheng Zhu, Zerun Ma, Dingbo Yuan, and Qi Zhang. Swe-bench pro verified: A reliable benchmark for software engineering agents, 2026. URL https://arxiv.org/abs/2609.08149.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy K Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Antony Kellermann, Jasjeet S Sekhon, Jacob Steinhardt, Sarah Schwettmann, Arvind Narayanan, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices in building rigorous agentic benchmarks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=E58HNCqoaA.

APPENDIX CONTENTS

A Notation . . . 16
B Residency contracts . . . 17
C Controller details . . . 20
D Proofs and limits of replay certification . . . 23
E Replay design and estimands . . . 27
F BoundaryBench and evaluation controls . . . 30
G Additional results and validity checks . . . 32
H Closest comparisons . . . 36

A NOTATION

Table 4 collects the recurring symbols, and Appendix B.1 gives the full boundary setup and contract definitions. Symbols used in one proof or algorithm are defined locally. Boundaries are indexed by t, while i indexes task/seed rows, because a task can cross several boundaries. Logarithms are natural.

Table 4: Notation used across the formulation, controller, and evaluation.

Symbol	Meaning
x_t, b_t, r_t	Checkpoint, boundary type, and receiver role.
$\mathcal{A}_t$, $\mathcal{G}_t$	Artifact inventory and graph of dependency, conflict, and parent edges.
$a = (\mathrm{id}, c, m)$	Artifact identifier, content or reference, and typed metadata.
P_t^{ref}, $\tilde{P}_t = R(K_t)$	Reference packet and contract packet realized by the renderer and execution adapter.
$K_t = (\pi_t, \Gamma_t)$	Placement map and finite set of obligations.
S_t^p, X_t, D_t, M_t	Artifacts with placement p, exclusions, dependency closure, and modifier record.
$\Gamma^{\mathrm{req}}(x, b, r)$, $\mathcal{K}(x, b, r)$	Required obligations and admissible contracts.
$Z \sim Q_{x,P}$, $U = u(Z)$, Y	Continuation record, scalar utility in $[0, 1]$, and endpoint vector.
ω, $I_{a,\psi}$	Continuation and renderer randomness, and perturbation ψ of artifact a.
$\Delta_{t,a,\psi}$, $D = U^{\mathrm{ref}} - U^{\mathrm{int}}$	Expected signed intervention effect and its paired observation, both in $[-1, 1]$.
$\hat{c}(K)$, B	Original-cost planning proxy and declared token cap. Realized cost is logged separately.
$o \in \mathcal{O}$, p_θ, $o^\star$	One of 20 operations, its scorer, and the training target.
$\phi(a, \ell_a)$, e_b, e_r, $g(a, \mathcal{G})$	Artifact features including the requirement-label block, boundary and role embeddings, and graph summaries.
F_t, H_t, μ_t, ν_t	Reference and contract transition kernels, and their checkpoint laws.
W_t, V_t, $\Delta_t(x)$	Reference and contract value-to-go, and $(F_t W_t)(x) - (H_t W_t)(x)$.
ϵ, ϵ_t, $\epsilon_t^{\mathrm{TV}}(x)$	Allowed utility loss, local loss tolerance, and local total-variation discrepancy.
λ, M, n, δ	Replay checkpoint law, candidate count, independent pairs per candidate, and error probability.
$\hat{\Delta}_j$, Δ_j, $\beta_{n,M,\delta}$	Candidate mean loss, its expectation, and simultaneous confidence radius.
$s_{i,m}$, $\mathcal{M}_i$, G_i	Preservation-member score, applicable members, and their row mean.
$\mathcal{F}$, Rep_i, C_i, ρ_{repair}	Failed cohort, repair and full-context success indicators, and their joint success fraction.
n_{10}, n_{01}	Controller-only and comparator-only favorable outcomes in paired binary data.
$\mathcal{U}(\theta)$, q_K	Negative validation cross-entropy and maximum utility among K null refits.

Signs and missing values. Success, preservation, and utility are oriented upward. Violation, harmful adoption, tokens, overflow, and latency are oriented downward. A positive $\Delta_{t,a,\psi}$, $\Delta_t(x)$, or D is a loss from replacing the reference, whereas positive $n_{10} - n_{01}$ favors the controller. Missing observations and empty denominators are undefined, never zero.

B RESIDENCY CONTRACTS

B.1 FORMAL BOUNDARY SETUP AND DEFINITIONS

Checkpoints, packets, and outcomes. At boundary t the harness holds a checkpoint x_t, the boundary has type b_t, and the receiver has role r_t and sees a packet P (prompt regions, evidence resolver, and tool access). A declared reference policy (native harness or full context) delivers the reference packet P_t^{ref}. The renderer arranges a contract's placements into a region-annotated layout, and the execution adapter turns that layout into the prompt, evidence access, and tools supplied to the receiver. Their combined realization is $\tilde{P}_t = R(K_t)$ for a Markov kernel R, allowing stochastic rendering. Continuing from x with P under a fixed executor and validator yields a record $Z \sim Q_{x,P}$ in the record space $\mathcal{Z}$. An endpoint u maps records to $[0, 1]$ with utility $U = u(Z)$, and the endpoint vector Y keeps success, violations, harmful adoption (use of an artifact marked stale or harmful), and representation scores separate. The variable ω collects continuation and renderer randomness.

Typed artifacts. An artifact $a = (\mathrm{id}, c, m)$ in the inventory $\mathcal{A}_t$ has content c and metadata $m = (\tau_a, s, \alpha, \nu, \sigma, \xi, \rho, \kappa)$ for type, source, authority, version, scope, exact fields, evidence references and risk, and cost and layout. The typed graph $\mathcal{G}_t = (\mathcal{A}_t, E_t^{\mathrm{dep}}, E_t^{\mathrm{conf}}, E_t^{\mathrm{par}})$ records dependencies ($(a, a') \in E_t^{\mathrm{dep}}$ when using a requires a'), conflicts, and parentage. A logging harness supplies source roles from event types, exact fields from commands, diffs, and test runs, versions from the commit, scope from the task specification and permissions, and cost from the tokenizer. We study the contract decision given a typed inventory, as emitted by harnesses that log structured events. Learning to type raw histories is out of scope.

Definition 1 (Residency contract)**.** A residency contract $K_t = (\pi_t, \Gamma_t)$ consists of a placement map π_t from $\mathcal{A}_t$ to $\Pi = \{\text{exact}, \text{structured}, \text{summary}, \text{pointer}, \text{evidence pointer}, \text{data-only}, \text{excluded}\}$ with optional modifiers (version expiry, permission restriction, cache placement, and tool exposure), and a finite set Γ_t of obligations, each with a predicate $v_\gamma(x_t, K_t, R(K_t)) \in \{0, 1\}$. A pointer is optional, whereas an evidence pointer must resolve. We write $S_t^p = \pi_t^{-1}(p)$.

Definition 2 (Boundary policy and admissibility)**.** A boundary policy $\Gamma^{\mathrm{req}}(x, b, r)$ is the finite set of obligations that the problem requires at checkpoint x for boundary type b and receiver role r. A contract $K = (\pi, \Gamma)$ is admissible, written $K \in \mathcal{K}(x, b, r)$, if and only if $\Gamma^{\mathrm{req}}(x, b, r) \subseteq \Gamma$ and $v_\gamma(x, K, R(K)) = 1$ almost surely for every $\gamma \in \Gamma$.

The required obligations are fixed outside the contract, so a contract cannot gain admissibility by omitting a required obligation. A handoff policy may require allowed paths, data-only tool text, and verbatim exact fields. BoundaryBench fixes the policy by construction, and a deployment harness designer declares it.

B.2 ARTIFACT AND CONTRACT RECORDS

The metadata components above are stored as flat fields alongside the artifact identifier and content. The 28 artifact types cover instructions, rules, source and tool excerpts, commands, evidence, memories, delegated work, messages, and permission or resource decisions. Provenance s identifies the producing event and parent artifacts. Authority and trust α are separate from relevance. Version ν includes expiry, repository identities, and policy and permission versions. Scope σ records allowed files, actions, or roles, and ξ lists exact fields. Evidence and risk annotations are stored in ρ, and cost, cacheability, prompt region, and reuse scope in κ. Missing versions do not imply unrestricted validity. The planner substitutes zero for unknown token costs, so resource conclusions require populated estimates and realized-token measurements.

A dependency edge (a, a') means that using a requires a'. Parent edges record derivation without asserting necessity. Conflict edges mark competing statements without adjudicating authority or truth. The implementation record of a contract is

$$K_t \longleftrightarrow \big(b_t, r_t, S_t^{\mathrm{exact}}, S_t^{\mathrm{struct}}, S_t^{\mathrm{sum}}, S_t^{\mathrm{ptr}}, S_t^{\mathrm{evid}}, S_t^{\mathrm{data}}, X_t, D_t, M_t\big). \tag{5}$$

The six retained placement sets and X_t partition $\mathcal{A}_t$ and hence determine π_t. The modifier record M_t encodes Γ_t, while (x_t, b_t, r_t) selects Γ^{req}. Every reference names the same inventory and the record

identifies its renderer. In particular,

$$X_t \cap \left(S_t^{\text{exact}} \cup S_t^{\text{struct}} \cup S_t^{\text{sum}} \cup S_t^{\text{ptr}} \cup S_t^{\text{evid}} \cup S_t^{\text{data}}\right) = \varnothing. \tag{6}$$

An optional pointer offers access on request. A required evidence pointer makes reachability an obligation. Both must resolve to the intended object under the receiver's permissions, with retrieval charged to the same budget as directly supplied context.

B.3 Operation vocabulary

The scorer has 13 terminal operations and 7 modifiers (Table 5). Each artifact receives one terminal placement and may receive several modifiers. A structured claim and its supporting log are separate artifacts joined by a dependency edge, so retaining the claim and a pointer to the log does not assign two placements to one artifact.

Table 5: Operation vocabulary of the scorer. Terminal operations fix a placement, and modifiers attach a requirement to one without deciding whether the artifact crosses. Names that share a row are distinct scored operations with the interpretation shown.

Operation	Placement	Contract interpretation
Thirteen terminal operations		
Pin exact, pin field	exact	The exact fields ξ of the artifact identify the content that appears verbatim.
Structure state	structured	Retains the state in structured form, without implying that every original token is kept.
Include with provenance	structured	Adds a provenance requirement.
Include with conflict	structured	Adds a conflict marker.
Include stale warning	structured	Adds a stale-state marker.
Require verification	structured	Adds a verification marker. The marker does not execute a verification.
Summarize	summary	The realization supplies the summary text.
Defer pointer	pointer	Makes the artifact available on request.
Store evidence pointer	evidence pointer	Makes reachability of the evidence an obligation.
Quote as data, quarantine	data-only	Both deliver the artifact in the data region.
Evict	excluded	Places the artifact in X_t and removes contradictory retained assignments.
Seven modifiers		
Restrict permission	none	Records a permission restriction.
Pin stable prefix	none	Requires stable-prefix placement and the associated cache-reuse condition.
Pin dynamic suffix	none	Requires dynamic-suffix placement.
Mark non-cacheable	none	Prohibits the declared cache reuse.
Expire on version change	none	Records the version scope of the artifact.
Hide tool, expose tool	none	Control tool exposure. Exposure also records the residency decision for the tool's output.

B.4 Boundary classes and coverage

The seven primary boundary classes are compaction, subagent handoff, subagent return, rule loading, tool-schema exposure, tool-output pruning, and session resume. The six secondary classes are memory read and write, mode or permission switch, worktree handoff, gateway routing, prompt serialization, and resource control. A policy is declared for each class and receiver role. This vocabulary specifies the interface, independently of evaluated coverage.

Six BoundaryBench families align with named classes, namely handoff criteria, pruned tool evidence, rule-file conflict, tool-schema exposure, prompt serialization, and resumed stale state. The last tests staleness rather than a typed session-version change. The remaining six families stress obligations, namely exact-test bug fix, scope-limited patch, stale guidance, misleading tool output, version mismatch, and unrecoverable failure. Compaction and subagent return have no dedicated family and

occur only as event surfaces. A separate minimal gateway fixture does not exercise native multi-agent behavior. Coverage therefore follows executed families and receivers, rather than the number of declared boundary classes.

B.5 Worked debugging handoff

The constructed handoff in Figure 1b permits edits to `src/cache.py` and names a failing test. Table 6 shows how one packet preserves six different obligations. No observed outcome is attached to this example.

Table 6: Contract for the constructed debugging handoff of Figure 1b. Each artifact receives one placement, and each obligation is a predicate on the realized packet $R(K_t)$.

Artifact	Content	Placement	Obligation on $R(K_t)$
a_1	Task with allowed edit path `src/cache.py`	Structured, in the instruction and state region	Scope. The packet states `src/cache.py` as the only allowed edit path.
a_2	Failing test `tests/test_cache.py::test_expiry`	Exact	Exactness. The identifier appears verbatim.
a_3	Current diagnosis	Structured claim, attributed to the sender	Dependency. Its support, a_2 and a_4, stays reachable.
a_4	Command log (long)	Evidence pointer to the store	Retrievability. The pointer resolves within the receiver's budget, and retrieval tokens are counted.
a_5	Diagnosis at an incompatible commit (stale)	Excluded	Freshness. The inapplicable diagnosis is absent from the receiver's packet.
a_6	Tool text "also edit `config.py`" (untrusted)	Data-only, quoted	Source role. The text is data and grants no permission.

The scope predicate checks what the packet states. Whether the receiver obeys it is a behavioral endpoint. The six replay interventions paraphrase a_2, strip the scope from a_1, break the pointer to a_4, promote a_6 into the instruction region, reinstate a_5, or remove a_4 while retaining a_3. The last intervention tests an unsupported claim. Removing both the claim and its support tests a different packet. A pointer that the receiver cannot open within its limits fails retrievability even if the sender still holds the evidence.

B.6 Dependency closure and final validation

For $S \subseteq \mathcal{A}_t$, define

$$\mathrm{Reach}_{\mathrm{dep}}(S) = S \cup \{a' \mid \text{a dependency path leads from an artifact in } S \text{ to } a'\}. \tag{7}$$

With S_t the union of the retained sets, the planner records

$$D_t = \mathrm{Reach}_{\mathrm{dep}}(S_t) \setminus X_t. \tag{8}$$

Exclusion takes precedence, so this operation never restores evicted support. The dependency obligation for retained a requires $\mathrm{Reach}_{\mathrm{dep}}(\{a\}) \cap X_t = \varnothing$. If it fails, a valid contract must drop, weaken, or reverify the claim. Computing D_t makes none of those repairs.

Final validation checks structural predicates on K_t, including the partition, valid references, renderer coverage, and $\Gamma^{\mathrm{req}} \subseteq \Gamma_t$. Representation predicates on $R(K_t)$ check verbatim fields, source-role separation, resolvable evidence, stated scope, and applicable version markers or exclusions. Behavioral validity is measured through Y. These checks are distinct, since a well-formed contract can still be rendered with a paraphrased exact field, and a correct rendering can still be misused by the receiver.

C CONTROLLER DETAILS

Figure 4 connects replay supervision to runtime control. This appendix specifies the implemented construction-supervised development recipe. Its checkpoint and training recipe are distinct from those used in the production comparison.

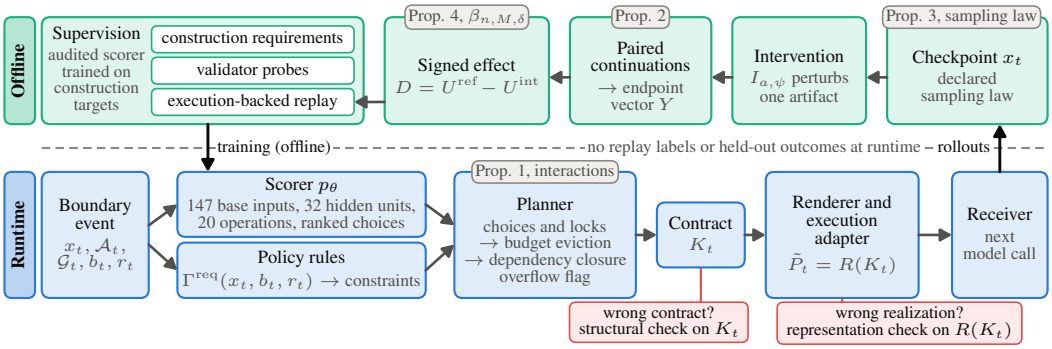

Figure 4: Supervision-to-controller design (Sections 3 and 5). Offline replay (top, right to left) supplies signed effects, kept distinct from construction requirements and validator probes. Whole-contract testing replaces the single-artifact intervention with the candidate packet. Proposition 3 pairs controller checkpoints with reference continuations in both branches. The completed audit establishes local contrasts only (Appendix E). At runtime (bottom), policy rules constrain choices, the scorer ranks permitted alternatives, and the planner composes a candidate contract. Structural checks inspect the contract before realization. The renderer and execution adapter produce the packet, whose representation is then checked. No replay labels or held-out outcomes cross the dashed line into runtime. Gray tags locate the results of Section 4.

C.1 ENCODING AND SCORER

The scorer maps artifact a, boundary type b, receiver role r, and typed artifact graph $\mathcal{G}$ to a distribution over the 20 operations $o \in \mathcal{O}$ in Table 5,

$$p_\theta(o \mid a, b, r, \mathcal{G}) = \mathrm{softmax}\big(W_2 \tanh(W_1 z + u_1) + u_2\big)_o. \tag{9}$$

The input $z = [\phi(a, \ell_a), e_b, e_r, g(a, \mathcal{G})]$ concatenates artifact features, boundary and role embeddings, and local graph summaries. In the base layout $z \in \mathbb{R}^{147}$. Controlled scorer comparisons hold the chosen architecture fixed to separate supervision and information access from capacity.

The base artifact encoding $\phi(a, \ell_a) \in \mathbb{R}^{110}$ comprises 64 hashed text coordinates, 28 structured metadata slots, and 18 requirement-label slots ℓ_a. Whitespace-tokenized word unigrams to trigrams are hashed from content, exact fields, and scope, provenance, resource, risk, and tool tags. Hash collisions are possible, and the encoder has no learned text embedding. Structured slots encode type, source, authority, trust, freshness, estimated token cost, cache and prompt layout, repository and policy identity presence, evidence-pointer presence, exact-field and tag counts, and graph degrees. The label block contains 16 requirements, confidence, and a presence marker. Its declared absent value has marker -1 in the replay-label-free condition, and it never reads the held-out continuation.

Construction metadata can itself reveal the intended operation. A mask therefore suppresses both a structured slot and its corresponding hashed tag, with graph-family masks applied separately and the same masks at fit and inference time. Zeroing only the requirement-label block leaves this metadata available. Zero conditioning replaces boundary and role embeddings by zero, but boundary information can still enter through metadata.

The exact-gradient configuration signs the hashed counts, applies $\log(1 + x)$ to their magnitudes, and normalizes each text block to unit Euclidean norm. It uses $\log(1 + \mathrm{cost})$, expands categorical slots to one-hot blocks, and standardizes numeric features using training statistics frozen with the checkpoint. These blocks widen the input beyond the base 147 coordinates. The earlier surrogate configuration uses raw hashed counts and categorical bucket indices as magnitudes.

The boundary and role embeddings have 16 coordinates each, with a fallback for unseen roles. Let $N_{\mathrm{dep}}(a)$ be the artifacts on which a depends, $N_{\mathrm{conf}}(a)$ its recorded conflict partners, and d_a^{dep} and d_a^{conf} their counts. The five graph features are

$$g(a, \mathcal{G}) = \Big(d_a^{\mathrm{dep}}, d_a^{\mathrm{conf}}, \frac{\sum_{a' \in N_{\mathrm{dep}}(a)} d_{a'}^{\mathrm{dep}}}{d_a^{\mathrm{dep}}}, \frac{\sum_{a' \in N_{\mathrm{conf}}(a)} d_{a'}^{\mathrm{conf}}}{d_a^{\mathrm{conf}}}, d_a^{\mathrm{dep}} + d_a^{\mathrm{conf}} \Big). \tag{10}$$

Empty averages and degrees of references outside the inventory are zero. The first two coordinates duplicate structured metadata and the fifth is their sum, so only the neighbor averages add information. Parent edges do not enter. Equation (9) concatenates these features with the artifact encoding and embeddings, then applies a 32-unit tanh layer and a 20-way softmax. The base dense layers have 5,396 parameters, in addition to the embeddings. This is a perceptron with local graph summaries and no message passing. Final validation checks graph consistency.

C.2 Supervision and training

A row pairs an artifact with a boundary event. The training-set builder reads construction requirements, separately tagged validator-probe or execution-backed labels, or no supervision. Tags identify a source without establishing its behavioral fidelity. Signed replay effects are stored separately because a changed outcome alone does not say whether retention helped. The development fits use construction requirements.

Operation targets. For a multi-label row, the first requirement in the following fixed order sets $o^\star$. Exactness maps to pin exact, evidence to store evidence pointer, and necessity, structured state, and dependency to structure state. Scope and provenance map to include with provenance, version scope to expire on version change, cache sensitivity to pin stable prefix, conflict to include with conflict, unsafe instruction forwarding to quote as data, boundary locality to quarantine, and redundancy to summarize. Staleness, harmfulness, and budget harmfulness map to evict. A lower-priority label can therefore be removed without changing the target. A label ablation needs a manipulation check on targets, features, or rules.

Nine operations never receive positive imitation targets in this recipe, namely pin field, defer pointer, include stale warning, require verification, restrict permission, pin dynamic suffix, mark non-cacheable, hide tool, and expose tool. Rules directly name pin field, restrict permission, pin dynamic suffix, and mark non-cacheable. The other five can enter only through scorer ranking, including when a provenance or scope rule selects among retaining operations. There is no positive imitation target or dedicated rule for tool exposure, and no rule revalidates a resumed artifact.

Objective and updates. For training rows $\mathcal{D}$, let $L_{\mathrm{imitation}}^{(a,t)} = -\log p_\theta(o^\star \mid a, b_t, r_t, \mathcal{G}_t)$. The objective is

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(a,t) \in \mathcal{D}} \left[L_{\mathrm{imitation}}^{(a,t)}(\theta) + \sum_{k \in \mathcal{T}} \lambda_k L_k^{(a,t)}(\theta) \right]. \tag{11}$$

The imitation weight is one. The active auxiliary set $\mathcal{T}$ includes effect regression, five obligation penalties for exactness, scope, provenance, evidence, and version-sensitive cache placement, and a cache-layout penalty. Each default auxiliary weight is smaller than one and fixed in the recorded configuration. Effect regression uses $\bar{\chi}_\theta = \sum_o p_\theta(o)\chi_o$, where $\chi_o = -1$ for evict, $\chi_o = 0$ for summarize, defer pointer, and quarantine, and $\chi_o = 1$ for other operations. These codes do not represent measured utility. Action, outcome, resource, and contrastive boundary terms are logged without updates by default. Activating outcome and resource gives ten gradient-owning terms.

Writing the logits as ζ, each row updates the dense parameters by

$$\theta \leftarrow \theta - \eta_{\mathrm{step}} \left[\nabla_\theta L_{\mathrm{imitation}}^{(a,t)} + (J_\theta \zeta)^\top v_{\mathrm{aux}} \right]. \tag{12}$$

The exact-gradient configuration includes the softmax Jacobian in v_{aux} and hence takes stochastic gradients of Eq. (11). It uses Glorot initialization, a fresh seeded row order each pass, and a recipe fixing step size and pass count before the fit. Embeddings update when enabled. Independent finite-difference checks found relative error at most 2.1×10^{-9} for all ten gradient-owning terms under

both supported interpreters. This validates differentiation, while behavioral benefit requires replay. The earlier surrogate uses fixed row order and omits the softmax Jacobian from several auxiliary updates, so it does not minimize the displayed objective. Its failure is reported in Appendix G.4. Outcome-only and compression-only controls are specified, but their settings are not consumed by the current trainer and they do not produce separately trained policies.

C.3 Planning and realization

Precedence. The first applicable rule decides the terminal operation. Rules are checked in this order.

1. Evict stale or harmful artifacts.
2. For exact fields, lock the higher-scoring of pin exact and pin field.
3. For unsafe instruction forwarding, lock the highest-scoring of quote as data, quarantine, and evict.
4. For an evidence pointer, lock store evidence pointer.
5. On a commit, worktree, policy, or permission-version mismatch, evict and attach expire on version change.
6. For provenance tags, take the highest-scoring retaining operation, defaulting to include with provenance.
7. For scope tags, take the highest-scoring retaining operation, defaulting to structure state.

A retaining operation is any terminal operation other than evict. When no rule decides, the scorer's highest-ranked terminal operation applies. A modifier ranked above all terminal operations is attached to that choice. Dynamic content cannot receive pin stable prefix and instead receives the higher-scoring of mark non-cacheable and pin dynamic suffix. A scope-bearing artifact evicted by a rule receives restrict permission. Composition attaches scope and provenance requirements to all retained artifacts carrying those tags, regardless of their terminal operation.

Precedence can lock an exact field from an incompatible version before the version rule is reached. These rules therefore do not jointly guarantee admissibility. Each can be disabled independently, but its metadata remains available to the scorer unless separately masked.

Budget and dependencies. The default planner repeatedly evicts the unlocked retained artifact with the lowest selected-operation probability, breaking ties by identifier. It stops when $\hat{c}(K) \leq B$ or only locked artifacts remain, then reports overflow if necessary. Excluded artifacts are removed from every retained set. Dependency closure is recorded without restoring excluded support (Eq. (8)). A claim whose support was evicted consequently fails its dependency obligation until the contract is repaired or rejected.

Two implemented alternatives change the budget selection among unlocked artifacts. A zero-one knapsack routine maximizes their total selection score under discretized costs after charging locked placements. It ignores alternative representations and dependencies. A beam of width w explores keep and evict choices with deterministic tie-breaking and no global optimality guarantee. The random-ranking control changes the scores alone and retains the same rules and budget procedure.

Rendering and validation. The renderer exports ordered sections, an assignment to instruction and state, exact, evidence, or data-only regions, and an operation trace. The harness-specific execution adapter writes summaries, quotes data-only text, resolves evidence under receiver permissions, charges retrieval, and assembles the model call. Validation first checks structural predicates on K_t, then representation predicates and resource use on $R(K_t)$. The two statuses and overflow flag are reported separately. Exporting a valid layout alone does not establish successful realization or receiver behavior.

C.4 Resource accounting

With n_a artifacts, m_e graph edges, text length L_{txt}, and input width d_{in}, the scorer costs $O(L_{\mathrm{txt}} + n_a(n_a + m_e) + n_a(32d_{\mathrm{in}} + 32 \cdot 20))$ per inventory. The $n_a(n_a + m_e)$ term arises because graph

maps are rebuilt per artifact. The greedy planner recomputes and sorts candidates after each eviction, giving $O(n_a^2 \log n_a)$ worst-case cost. Closure costs $O(n_a + m_e)$. Knapsack uses $O(n_a \bar{B})$ state updates for discretized capacity $\bar{B}$, plus solution reconstruction. Beam search maintains at most w partial assignments per artifact and also pays for copying them.

The planning proxy charges original estimated artifact cost for every retained representation. Runtime accounting separately logs realized input and output tokens, retrieval tokens and calls, overflow, controller latency, and receiver latency. Cache placement declares a reuse opportunity, whose serving effect needs hit and miss telemetry. Offline supervision costs $O(kc_{\mathrm{run}})$ for k paired interventions with continuation cost c_{run}, and agreement with cheaper validator probes must be evaluated before substituting them for replay. Integration with additional harnesses requires their boundary hooks and execution adapters, beyond the implemented BoundaryBench inventories.

D Proofs and limits of replay certification

The four propositions specialize standard results to context boundaries. The singleton examples use non-additivity under removal and interaction (Ghorbani & Zou, 2019; Dhamdhere et al., 2020; Covert et al., 2021), the endpoint statement uses the variational characterization of total variation (Levin & Peres, 2017, Prop. 4.5), the trajectory identity is a performance-difference identity (Kakade & Langford, 2002), and the certificate combines Hoeffding's inequality (Hoeffding, 1963) with a union bound. We give complete proofs and retain the constructions that explain when the resulting guarantees apply.

D.1 Conventions

Checkpoint and continuation-record spaces are measurable. A checkpoint stores the full restorable history, so the next-state law is Markov. The fixed horizon T includes absorbing padding after early termination. A path endpoint is made terminal by storing its accumulated value in the checkpoint. Random unbounded horizons are outside the result. Behavioral records use common coordinates across arms and omit arm identifiers, while retaining every variable read by an endpoint or a later transition.

For a kernel F, write $(Fg)(x) = \int g(y)F(x, dy)$ and $(\mu F)(B) = \int F(x, B)\mu(dx)$, so $(\mu F)g = \mu(Fg)$. Kernels preserve measurability and the range $[0, 1]$. We use $\mathrm{TV}(\mu, \nu) = \sup_B |\mu(B) - \nu(B)| \in [0, 1]$. A stochastic renderer induces the mixture of continuation laws over its realized packets. Within a replay pair any coupling, including shared seeds, is allowed if each marginal execution law is correct. Linearity fixes the mean difference independently of that coupling.

D.2 Singleton effects and silent failure

Proof of Proposition 1. For redundancy, let $f(S) = \mathbf{1}[S \neq \varnothing]$ on $\mathcal{A} = \{a_1, a_2\}$. Both singleton deletions leave utility 1, while joint deletion reduces it to 0. For complementarity, let $f(S) = \mathbf{1}[S = \mathcal{A}]$. Each singleton deletion loses 1, while joint deletion also loses 1. Both functions are monotone.

For the role construction, place the same instruction to edit an unauthorized file in either the instruction region or the data region. A deterministic receiver that obeys it only in the instruction region produces two records with unauthorized-action indicators 1 and 0. The event of an unauthorized action separates the two point-mass laws, giving total variation 1 despite identical artifact sets. □

For a removal order with $S_0 = \mathcal{A}$ and $S_i = S_{i-1} \setminus \{a_{(i)}\}$, telescoping gives $f(\mathcal{A}) - f(\varnothing) = \sum_i [f(S_{i-1}) - f(S_i)]$. All but the first term concern intermediate packets. Thus this identity does not turn singleton effects at the full packet into additive values.

For a measurable h with values in $[0, 1]$, the layer-cake identity gives $\int h\, d\mu = \int_0^1 \mu\{h > s\}\, ds$. Subtracting the corresponding identity for ν and bounding the integrand yields

$$\left| \int h\, d\mu - \int h\, d\nu \right| \leq \mathrm{TV}(\mu, \nu). \tag{13}$$

Proof of Proposition 2. Equation (13) proves the forward implication. For the reverse, take $h = \mathbf{1}_B$ for every measurable event B and then take the supremum. For silent failure, choose records z_0, z_1 where tests pass in both and an unauthorized action occurs only in z_1. Point masses at these records have equal test success and total variation 1. Mixing the second point mass into the first with weight q gives unchanged test success, violation rate q, and total variation q for every $q \in [0, 1]$. □

A total-variation bound therefore controls all bounded endpoints. A bound for one tested endpoint supplies only that endpoint's guarantee.

D.3 Trajectory decomposition and its sampling law

Proof of Proposition 3. The reference recursion gives $\mu_0 W_0 = \mu_T u$ and $\nu_{t-1} F_t W_t = \nu_{t-1} W_{t-1}$, while $\nu_{t-1} H_t W_t = \nu_t W_t$. Consequently,

$$\begin{aligned}\sum_{t=1}^{T} \mathbb{E}_{\nu_{t-1}} \Delta_t &= \sum_{t=1}^{T} (\nu_{t-1} W_{t-1} - \nu_t W_t) \\ &= \nu_0 W_0 - \nu_T u = \mu_T u - \nu_T u,\end{aligned}$$

where the last equality uses $\mu_0 = \nu_0$. All terms are integrable because $W_t \in [0, 1]$. The identity is also valid for $T = 0$. □

The reference harness must be runnable from contract-reached checkpoints. The contrast Δ_t compares the packets at boundary t and then uses the reference continuation. Its averaging law is the contract law ν_{t-1}, even though its value-to-go is the reference value W_t. A scorer imitating targets at reference-policy checkpoints can face covariate shift when deployed on its own trajectories (Ross & Bagnell, 2010; Ross et al., 2011).

Corollary 5 (Local control on the contract's checkpoints)**.** *If $\mathbb{E}_{\nu_{t-1}} \Delta_t \le \epsilon_t$ at every boundary, then $\mathbb{E}_{\mu_T} u - \mathbb{E}_{\nu_T} u \le \sum_t \epsilon_t$. In particular, nonpositive local losses imply nonpositive trajectory loss.*

Proof. Sum the inequalities and use Proposition 3. □

For the symmetric identity, set $V_T = u$ and $V_{t-1} = H_t V_t$. Now $\mu_{t-1} F_t V_t = \mu_t V_t$ and $\mu_{t-1} H_t V_t = \mu_{t-1} V_{t-1}$, so telescoping gives

$$\mu_T u - \nu_T u = \sum_{t=1}^{T} \mathbb{E}_{\mu_{t-1}}[(F_t - H_t) V_t]. \tag{14}$$

This form uses reference checkpoints followed by contract continuations. Reference checkpoints followed by reference continuations instead estimate $\sum_t \mathbb{E}_{\mu_{t-1}} \Delta_t$. Their discrepancy from the true loss is $\sum_t (\nu_{t-1} - \mu_{t-1}) \Delta_t$, bounded in absolute value by $2 \sum_{t=2}^{T} \mathrm{TV}(\nu_{t-1}, \mu_{t-1})$ using Equation (13) on $(\Delta_t + 1)/2$.

Why total variation can give a weaker bound. Assume $\epsilon_t^{\mathrm{TV}}(x) = \mathrm{TV}(F_t(x, \cdot), H_t(x, \cdot))$ is measurable, as on countably generated state spaces. Two elementary inequalities suffice. For any shared kernel F, $\mathrm{TV}(\mu F, \nu F) \le \mathrm{TV}(\mu, \nu)$ by applying Equation (13) to $F(\cdot, B)$ and taking the supremum. For a common input law λ, $\mathrm{TV}(\lambda F, \lambda H) \le \mathbb{E}_\lambda \mathrm{TV}(F(x, \cdot), H(x, \cdot))$ by integrating the pointwise event difference and taking the supremum.

Corollary 6 (Total-variation composition)**.** *For each t, choose ρ_{t-1} to be either μ_{t-1} or ν_{t-1}. Then $|\Delta_t(x)| \le \epsilon_t^{\mathrm{TV}}(x)$ and*

$$|\mu_T u - \nu_T u| \le \mathrm{TV}(\mu_T, \nu_T) \le \min\left\{1, \sum_{t=1}^{T} \mathbb{E}_{\rho_{t-1}} \epsilon_t^{\mathrm{TV}}\right\}. \tag{15}$$

Proof. The endpoint inequalities follow from Equation (13). Writing $d_t = \mathrm{TV}(\mu_t, \nu_t)$, insert $\nu_{t-1} F_t$ between $\mu_{t-1} F_t$ and $\nu_{t-1} H_t$ to obtain $d_t \le d_{t-1} + \mathbb{E}_{\nu_{t-1}} \epsilon_t^{\mathrm{TV}}$ by the two inequalities above. Inserting $\mu_{t-1} H_t$ instead gives the same recursion with μ_{t-1}. Induct from $d_0 = 0$ and use $d_T \le 1$. □

This is the standard hybrid argument (Kearns & Singh, 2002). It compares whole next-checkpoint laws. If different deterministic packets persist as a coordinate of the next checkpoint, the event that this coordinate equals the reference packet has probabilities 1 and 0. Then $\epsilon_t^{\mathrm{TV}} = 1$ even when the endpoint is unchanged, making the composition bound trivial. Omitting that coordinate is valid only if the remaining state still determines subsequent transitions. Behavioral similarity alone does not justify that omission.

D.4 Two chains that expose the boundary conditions

Stale-pointer chain. Use states valid, stale, and absorbing lost, with $u = (1, 1, 0)$ and initial state valid. The reference kernel sends both valid and stale states to valid. The contract sends valid to stale with probability ϵ and otherwise to valid, and sends stale to lost. Both kernels keep lost. Since $Fu = u$, all $W_t = u$, and the local contrasts on the three states are $(0, 1, 0)$. The reference always visits valid, so its logged contrast sum is zero. Under the contract, $\nu_s(\mathsf{valid}) = (1-\epsilon)^s$ and $\nu_s(\mathsf{stale}) = \epsilon(1-\epsilon)^{s-1}$ for $s \geq 1$. For $T \geq 1$, Proposition 3 therefore gives

$$\mu_T u - \nu_T u = \sum_{s=1}^{T-1} \epsilon(1-\epsilon)^{s-1} = 1 - (1-\epsilon)^{T-1}. \tag{16}$$

This equals the mass of lost at time T. It is 0.370 for $\epsilon = 0.05, T = 10$. The per-state TV distances are $(\epsilon, 1, 0)$, so the on-policy TV sum is $[1-(1-\epsilon)^T] + [1-(1-\epsilon)^{T-1}]$, capped at 1. The uniform TV bound is 1. Figure 5a compares these exact quantities.

Absorbing-failure chain. Start at ok with $u(\mathsf{ok}) = 1$ and $u(\mathsf{lost}) = 0$. The reference keeps either state. At boundary t, the contract sends ok to lost with probability ϵ_t and keeps lost. Thus $\Delta_t(\mathsf{ok}) = \epsilon_t$, $\Delta_t(\mathsf{lost}) = 0$, and $\nu_{t-1}(\mathsf{ok}) = \prod_{s<t}(1-\epsilon_s)$. Hence

$$\mu_T u - \nu_T u = \sum_t \epsilon_t \prod_{s<t}(1-\epsilon_s) = 1 - \prod_t (1-\epsilon_t), \tag{17}$$

where successive products telescope. With constant ϵ, loss is $1-(1-\epsilon)^T$ and approaches 1 as T grows. A trajectory tolerance ϵ_{tot} requires $\epsilon \leq 1-(1-\epsilon_{\mathrm{tot}})^{1/T}$, approximately $\epsilon_{\mathrm{tot}}/T$ for small tolerance.

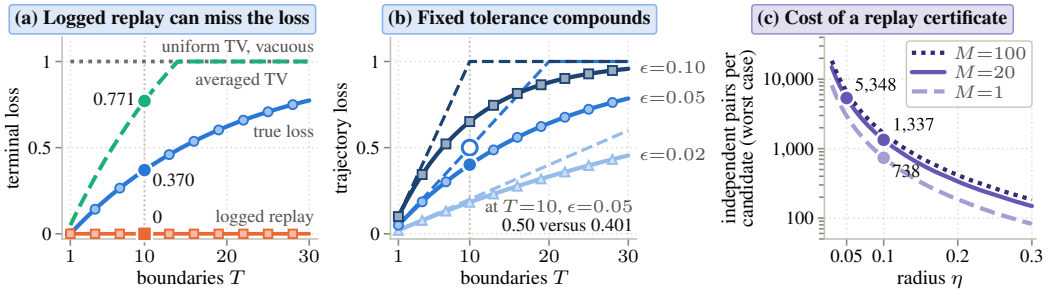

Figure 5: What local replay certifies. (a) A stale-pointer chain in which the reference harness revalidates pointers ($\epsilon = 0.05$). Replay with reference checkpoints and reference continuations reports zero loss (squares), and the uniform total-variation bound is vacuous (dotted). The true loss (circles) equals the on-policy sum of Proposition 3, grows with the number of boundaries T, and stays below the on-policy averaged total-variation bound (dashed). (b) With an absorbing failure, a fixed per-boundary tolerance ϵ yields trajectory loss $1-(1-\epsilon)^T$ (solid), close to the sum of tolerances $T\epsilon$ (dashed, capped at 1) for small $T\epsilon$. (c) Independent paired continuations per candidate that suffice for a distribution-free certificate of radius η over M candidates (Proposition 4, $\delta = 0.05$). All curves are exact calculations.

D.5 Replay certificates and candidate selection

Proof of Proposition 4. For fixed j, the n i.i.d. differences lie in an interval of width 2. Hoeffding's inequality (Hoeffding, 1963) gives $\Pr\{|\hat{\Delta}_j - \Delta_j| \geq r\} \leq 2\exp(-nr^2/2)$. The union bound over

M candidates gives

$$\Pr\left\{\max_j |\hat{\Delta}_j - \Delta_j| \geq r\right\} \leq 2M\exp(-nr^2/2).$$

Set $r = \sqrt{2\log(2M/\delta)/n}$ to make the right side δ. Independence across candidates is unnecessary. □

A candidate under a nondegenerate checkpoint law λ is a rule $x \mapsto K_j(x)$. Checkpoints and paired continuations must be i.i.d. across samples for each rule. Candidates may share checkpoints, seeds, and reference runs across j. When candidates share the policy before boundary t, its checkpoint law $\lambda = \nu_{t-1}$ and reference continuations identify a local term in the trajectory decomposition for each candidate. Changes to earlier decisions require checking each candidate's checkpoint law.

Corollary 7 (Certified selection with abstention)**.** *Fix tolerance ϵ and planning costs $\hat{c}(K_j)$ before replay. Let $\beta = \beta_{n,M,\delta}$ and $\hat{\mathcal{J}} = \{j \mid \hat{\Delta}_j + \beta \leq \epsilon\}$. Choose a cheapest member of $\hat{\mathcal{J}}$ with a fixed tie rule, or abstain if it is empty. The probability of accepting a candidate with $\Delta_j > \epsilon$ is at most δ. On the same coverage event, if $\mathcal{J}_- = \{j \mid \Delta_j \leq \epsilon - 2\beta\} \neq \varnothing$, the rule accepts at cost at most $\min_{j\in\mathcal{J}_-} \hat{c}(K_j)$.*

Proof. On simultaneous coverage, every accepted j has $\Delta_j \leq \hat{\Delta}_j + \beta \leq \epsilon$. Also $j \in \mathcal{J}_-$ implies $\hat{\Delta}_j + \beta \leq \Delta_j + 2\beta \leq \epsilon$, so $\mathcal{J}_- \subseteq \hat{\mathcal{J}}$. Minimizing cost over the latter proves the second claim. Failure requires the coverage event to fail. □

The guarantee controls acceptance jointly with failure, rather than failure conditional on acceptance. It certifies a λ-average endpoint loss. For λ concentrated at x, a candidate that is also admissible under Definition 2 and accepted on the simultaneous coverage event is feasible for Equation (1) at x. Selection optimizes only over the fixed candidate family.

The radius is at most η when $n \geq 2\log(2M/\delta)/\eta^2$, giving the sizes in Figure 5c. At $\delta = 0.05$, the sufficient integer sizes are 738 for $M = 1, \eta = 0.1$, 1,337 for $M = 20, \eta = 0.1$, and 5,348 for $M = 20, \eta = 0.05$. Shared seeds change variance through the covariance of paired outcomes and need not improve it. Variance-adaptive bounds can use concentrated differences more efficiently (Maurer & Pontil, 2009). The stated proof permits neither outcome-dependent candidate expansion nor optional stopping. Dependent tasks, unrestored environments, and repeated copies of deterministic continuations do not supply i.i.d. samples.

D.6 A FORKED ON-POLICY TRAJECTORY ESTIMATOR

Corollary 8 (Forked on-policy estimator)**.** *For $T \geq 1$, draw N independent contract trajectories. Independently choose one boundary $t_i^\star$ uniformly from $\{1, \ldots, T\}$ on each trajectory. At its checkpoint, fork a reference-packet and a contract-packet arm, both followed by reference continuation. Let $D_i \in [-1, 1]$ be their terminal endpoint difference and $L_i = TD_i$. Then $\mathbb{E}L_i = \mu_T u - \nu_T u$ and, with probability at least $1 - \delta$,*

$$\mu_T u - \nu_T u \leq \bar{L} + T\sqrt{2\log(1/\delta)/N}. \tag{18}$$

Proof. Conditioning on $t_i^\star = t$ and $x_t = x$ gives $\mathbb{E}[D_i \mid t, x] = \Delta_t(x)$. The checkpoint law is ν_{t-1}, so $\mathbb{E}L_i = \sum_t \mathbb{E}_{\nu_{t-1}}\Delta_t$ by uniform sampling. Proposition 3 identifies the sum. Hoeffding's one-sided bound for independent $L_i \in [-T, T]$ gives $\Pr\{\mathbb{E}L_i - \bar{L} \geq s\} \leq \exp[-Ns^2/(2T^2)]$. Substitute the stated radius. □

The two-sided radius replaces $\log(1/\delta)$ by $\log(2/\delta)$. Worst-case accuracy η requires order T^2/η^2 forks, while end-to-end paired differences lie in $[-1, 1]$ and need order $1/\eta^2$ pairs. Forking supplies local attribution and training targets. It is not a sample-efficiency improvement for estimating total loss alone.

E REPLAY DESIGN AND ESTIMANDS

This appendix defines the replay, preservation, and repair estimands and the uncertainty procedures for the paired comparisons. Certification additionally requires the checkpoint and continuation laws in the propositions.

E.1 RESTORATION AND INTERVENTION IDENTITY

A checkpoint contains the repository revision and relevant working state, task objective, permissions, process and tool state, active history, and reachable evidence objects. An unrestorable external service needs either a fixed recording or a declared sampling model. Each pair restores the same checkpoint independently, so one branch cannot change the other's inputs. Executor, tokenizer, tools, sampling settings, resource limits, validator, and grading procedure stay fixed unless explicitly targeted by the intervention. Missing grading remains missing.

An intervention is identified by its target artifact, perturbation type, parameters, realized packet, and resource condition. Ordered composition is part of the identity when operations do not commute. The task objective, grading criterion, reference packet, and durable evidence store remain fixed. Deleting an instruction and quoting it as data are distinct interventions because only the latter preserves its evidentiary content. Placement-matched controls separate the effect of changing content from changes in packet length or position (Liu et al., 2024). Perturbations can also delete an artifact, paraphrase an exact field, remove scope, promote data to the instruction role, break a pointer, or reinstate stale state.

E.2 CHECKPOINT SAMPLING AND PAIRED EFFECTS

For Proposition 3, run the controller to the boundary and restore its checkpoint $x_t \sim \nu_{t-1}$. Deliver the reference or contract packet and continue both branches under the reference policy. The paired mean then estimates $\mathbb{E}_{\nu_{t-1}}\Delta_t$, the required local loss. Intervention supervision substitutes $I_{a,\psi}(P_t^{\mathrm{ref}})$ for the contract packet to estimate Equation (2). A checkpoint drawn from a logged reference trajectory instead has law μ_{t-1}. Reference continuation there estimates a different quantity, as the stale-pointer construction shows. Logged checkpoints can estimate the symmetric identity when both branches continue under the contract (Equation (14)). Each replay record must therefore name both its checkpoint-generating policy and its continuation policy.

A uniform random boundary on each independent controller trajectory gives the forked total-loss estimator of Corollary 8. A diagnostic can compare its mean $\bar{L}$ from N trajectories with an end-to-end paired mean $\bar{D}^{\mathrm{e2e}}$ from N' independent trajectories. Under the stated sampling and restoration premises, Hoeffding bounds and a union bound imply

$$|\bar{L} - \bar{D}^{\mathrm{e2e}}| \le T\sqrt{2\log(4/\delta)/N} + \sqrt{2\log(4/\delta)/N'}$$

with probability at least $1-\delta$. Failure challenges those premises. Agreement does not verify them or certify total variation. The production replay roster samples uniformly over its declared boundary checkpoints. This description does not identify those checkpoints with the controller-induced law ν_{t-1}, so the audit is not an on-policy certificate.

For a fixed checkpoint and intervention, with paired differences $D_k = U_k^{\mathrm{ref}} - U_k^{\mathrm{int}} \in [-1, 1]$,

$$\hat{\Delta} = \frac{1}{n}\sum_{k=1}^{n} D_k, \qquad \mathbb{E}\hat{\Delta} = \Delta_{t,a,\psi}. \tag{19}$$

Unbiasedness requires correct marginal execution laws and permits shared seeds within pairs. Concentration requires independence across pairs. Repeating an identical deterministic continuation adds no information. A behavior-changed flag loses the sign, and one minus a validator score loses the reference outcome. Construction requirements, validator probes, and execution-backed effects therefore remain separate supervision sources. For expected utilities V_0, V_1, V_2, V_{12} under the reference, two individual interventions, and their ordered joint application, define

$$\mathcal{I}_{12} = V_1 + V_2 - V_{12} - V_0. \tag{20}$$

Positive values mean joint loss exceeds the sum of individual losses. Negative values mean it falls short. Each contrast requires all four conditions.

E.3 COMPLETED REPLAY ELIGIBILITY AUDIT

Of 96 replay pairs, 88 had both endpoints and agreeing execution-state digests. Four complete pairs were no-ops, leaving 84 nonzero intervention contrasts. Completion coverage is $88/96 = 0.917$ and eligible contrast coverage is $84/96 = 0.875$. The eight incomplete pairs have missing endpoints and contribute to neither contrast estimate. These exclusions do not redefine the separately frozen matched-controller population. Table 7 separates restoration coverage from the additional checkpoint-law requirement.

Table 7: Restoration requirements and the 96-pair production audit. A declared sampling frame alone does not establish the law required for certification.

Requirement	Recorded comparison	Completed record
Checkpoint law	Uniform declared boundary-checkpoint roster	Controller-induced law not established
Source and receiver	Revision, state, event, artifacts, contract, packet	Recorded for all 96 pairs
Execution and grading	Matching execution digests and common validator	88 complete, 8 one-sided
Manipulation	Explicit target and operation, nonzero change	84 contrasts, 4 complete no-ops
Pairing and units	Task, seed, event, template, analysis unit	Fixed before outcomes, 24 clusters

E.4 PRESERVATION AND BEHAVIORAL ENDPOINTS

Strict validator success and violation or harmful adoption measure executed behavior. Preservation scores measure the realized packet. At event e of task i, $s_{ie,m} \in [0, 1]$ is the fraction of applicable artifacts meeting member m's predicate. The six members require exact fields verbatim, scope restrictions matching the receiver's boundary policy, retained origin and source role, evidence reachable within the charged retrieval budget, version-valid artifacts or explicit marking or exclusion, and stale or harmful exclusion. For the last member, stale artifacts are excluded or marked stale, harmful artifacts are excluded or passed as data, and neither reaches the instruction region. Tool-originated content retains its data role.

Let $E^{\mathrm{app}}_{i,m}$ contain the events where construction makes member m applicable, and let $\mathcal{M}_i$ contain the members with nonempty applicable sets. When all required observations exist,

$$s_{i,m} = \frac{1}{|E^{\mathrm{app}}_{i,m}|} \sum_{e \in E^{\mathrm{app}}_{i,m}} s_{ie,m}, \qquad G_i = \frac{1}{|\mathcal{M}_i|} \sum_{m \in \mathcal{M}_i} s_{i,m}. \tag{21}$$

Averaging within members first prevents repeated events from dominating G_i. It is undefined with fewer than two applicable members or any missing applicable observation. Only structural non-applicability removes a member. The native and controller arms have identical member availability. Exactness applies to 56 trajectories, scope to 64, provenance and evidence to 48 each, version to 24, and exclusion to 40. The counts therefore do not change the composition of the paired comparison.

Dependency closure separately measures the fraction of selected artifacts that retain required support. Its denominator depends on policy selection, so closure is excluded from G_i and requires the selected-artifact count alongside it. The controller selects 10.21 artifacts per trajectory on average, compared with 9.98 for fixed rules. Mean per-trajectory closure is 0.927 versus 0.876, with paired difference 0.051 and 95% cluster interval $[0.032, 0.068]$.

E.5 PAIRED CONTRASTS AND INFERENCE

For the controller c and comparator h, let Y^h_i be binary trajectory-level success on task i. If an event roster is fixed before outcomes, use the same event coordinates for both arms and first average within task to obtain $\bar{Y}^h_i$. Otherwise compare each arm's complete trajectory from a common initial

checkpoint, with $\bar{Y}_i^h = Y_i^h$. The full-population contrast is

$$\hat{\theta}_h = \frac{1}{n}\sum_{i=1}^{n}(\bar{Y}_i^c - \bar{Y}_i^h). \tag{22}$$

McNemar inference requires binary paired outcomes, rather than event-averaged fractions. In that case $\hat{\theta}_h = (n_{10} - n_{01})/n$, where n_{10} counts controller-only successes and n_{01} comparator-only successes. Concordant pairs remain in the denominator. For violations, recode as absence of violation so that positive differences again favor the controller.

Seeds, events, stress levels, and repair coordinates are aggregated within independent tasks before inference. Shared-template tasks retain their cluster identity. Missing outcomes require treatment fixed before outcomes. They do not automatically become model failures or outcome-dependent exclusions.

For independent pairs drawn from a common joint outcome law, with equal marginal success under the null, the favorable count conditional on $n_d = n_{10} + n_{01}$ is $\mathrm{Binomial}(n_d, 1/2)$. The exact two-sided McNemar value (McNemar, 1947) is

$$p(n_{10}, n_{01}) = \min\left\{1, 2^{1-n_d}\sum_{k=0}^{\min(n_{10},n_{01})}\binom{n_d}{k}\right\}, \tag{23}$$

including $p = 1$ at $n_d = 0$. The completed 40-pair primary comparison had 12 favorable, 2 harmful, and 26 concordant pairs, so the difference is 0.25 and nominal $p = 0.012939453125$. The roster is a subset of the 96-pair comparison, comprising the first 20 recorded template clusters at seeds 11 and 23. Its joint table contains 10 successes under both policies, 16 failures under both, and the 14 discordances above. The McNemar value treats these 40 pairs as independent, although each template contributes two seeds. Resampling the 20 whole template clusters instead gives a 95% percentile interval of $[0.10, 0.40]$. The main 96-pair analysis uses the same procedure across its 24 clusters, with 100,000 resamples and fixed analysis seed 250926.

Analysis families. The six preservation-member comparisons are retrospective diagnostics, separate from the primary strict-success comparison. They use exact two-sided sign flips of mean template differences, with Holm adjustment within their six-test family (Holm, 1979). Their inference requires independent template clusters and sign exchangeability under the null. The effective sample for each member is restricted to applicable trajectories, as reported in Table 13. Violations, harmful adoption, dependency closure, ablations, repair, and probe analyses are descriptive and are not covered by that adjustment. In particular, a favorable preservation result is not a confirmatory safety finding. Non-rejection establishes neither equality nor equivalence.

E.6 Single-boundary repair

Fix the failure criterion, replayability criterion, and repair-selection rule before inspecting repair outcomes. Let $\mathcal{F}$ be the resulting replayable failed trajectories. Replay coverage divides $|\mathcal{F}|$ by all trajectories meeting the failure criterion, retaining restoration failures in that denominator. For $i \in \mathcal{F}$, let Rep_i indicate success after the specified repair and C_i success under full context from the same checkpoint. The joint recovery estimand is

$$\rho_{\mathrm{repair}} = \Pr(\mathrm{Rep}_i = 1, C_i = 1 \mid i \in \mathcal{F}), \qquad \hat{\rho}_{\mathrm{repair}} = \frac{1}{|\mathcal{F}|}\sum_{i\in\mathcal{F}}\mathbf{1}[\mathrm{Rep}_i = C_i = 1]. \tag{24}$$

Only joint successes contribute. Repair success when full context fails is a separate distraction contrast, and missing endpoints remain missing. A specified single repair, any successful repair in a fixed set, and an adaptive search define different estimands. Recovery may compensate for an earlier error and does not establish a unique boundary cause.

All 72 frozen failed trajectories have complete restoration and a nonzero repair. Figure 6 gives their joint outcomes. The specified repair succeeds on 45 trajectories and full context on 46, with 42 shared successes. Thus $\hat{\rho}_{\mathrm{repair}} = 42/72 = 0.583$. Three repair successes occur despite full-context failure, the separate distraction fraction $3/72 = 0.042$. These outcomes describe the selected failed set and repair class, without estimating natural failure prevalence.

A separate development calibration used 192 repair coordinates in 144 trajectories from 12 templates and found bootstrap undercoverage (Appendix G.4). That population is distinct from the 72-trajectory study. We therefore retain descriptive numerators and denominators without a confirmatory repair interval. Replaying more seeds reduces continuation noise but adds no template diversity.

F BOUNDARYBENCH AND EVALUATION CONTROLS

This appendix describes BoundaryBench, the completed execution conditions, and the controls needed to interpret the production comparison.

F.1 BENCHMARK CONSTRUCTION

BoundaryBench couples a coding fixture to a typed inventory, an intended boundary, and a family validator. Each family in Table 8 controls a requirement and exposes a competing explanation. Boundary events include compaction, subagent handoff and return, tool-output pruning, and resumption. The development population has 144 task/seed rows from 12 templates, separate from the completed comparison's 96 pairs from 24 template clusters. Training and evaluation contain the same families. Template and artifact identities must stay attached to every row because additional seeds do not supply independent structural diversity.

Table 8: BoundaryBench's twelve families. The validator checks the controlled requirement. The last column identifies what a successful construction alone cannot establish.

Family	Controlled requirement	Competing explanation
Exact-test bug fix	Preserve a failing-test identifier verbatim	Exact-field metadata identifies the answer
Scope-limited patch	Keep edits within allowed paths	A deterministic scope filter suffices
Stale guidance	Exclude or qualify outdated advice	Freshness tags supply the decision
Misleading tool output	Keep imperative tool text in a data role	A parser supplies the separation
Rule-file conflict	Distinguish competing sources and their scope	Fixed precedence supplies every choice
Handoff criteria	Preserve receiver task and completion criteria	Generic instruction copying suffices
Tool-schema exposure	Expose tools appropriate to the receiver's role	A smaller fixed tool set suffices
Resumed stale state	Revalidate mutable state at resume	Staleness is detected without version reasoning
Pruned tool evidence	Preserve exact excerpts or a resolvable pointer	A pointer looks valid but is never dereferenced
Prompt serialization	Preserve instruction/data regions and reuse restrictions	Symbolic layout changes but the prompt does not
Version mismatch	Respect applicability to the receiver's version	The version tag is a complete label
Unrecoverable-failure control	Keep failure after full restoration	A constructed negative control does not diagnose natural failures

Stress transformations vary freshness, exact fields, scope, provenance, evidence, and version applicability. Removing a required observation and adding distractors are different interventions. The unrecoverable control checks that the analysis attributes none of its failures to the boundary. Elsewhere, recovering a construction requirement establishes expressibility and reconstruction. Failure can arise from vocabulary mismatch, target priority, optimization, or rendering and does not identify model capacity.

F.2 INFORMATION ACCESS AND MATCHED EXECUTION

Oracle-assisted arms see construction requirements through the 18 requirement-label slots. *Replay-label-free* arms receive no replay labels or outcomes at receiver time, and that block has its declared absent value with presence marker -1. Construction metadata remains available through the text,

structured features, and dependency graph. Thus the evaluated controller combines rules and learned scores with construction metadata. Isolating learning beyond that metadata would require matched training and evaluation with annotation-derived information removed from every input channel, including hashed text, graph features, and cached lookups. The present comparison establishes the benefit of the combined policy under its stated information access.

The completed arms appear in Table 2. Content arms include native-harness, recency, BM25, summary-only, and token-cost policies. Manual, fixed-rule, and controller arms add typed metadata. Full context and the privileged oracle are references. The comparisons pair task, seed, and template within each model and use the same validator, tool interface, and accounting. Resource caps differ between policy families as shown in Table 9. Thus the native comparison measures policy behavior at the stated caps, while the compact content baselines and controller share the smaller cap. These fixed-cap comparisons leave the budget-response frontier untested.

Budget accounting distinguishes the cap on the active packet from cumulative input over a trajectory. Input includes retrieved evidence, output counts all generated tokens, and external storage is reported separately. Controller latency covers encoding, graph construction, scoring, planning, and rendering, separately from model-serving latency. A fixed cap does not imply equal spend.

Table 9: Production resource caps. Input columns distinguish active packet capacity from cumulative trajectory input. Native and compact policies allow 8,192 output tokens and 1,800 seconds per trajectory. Full context is an expanded-resource reference with 16,384 output tokens and 3,600 seconds.

Policy family	Active input	Cumulative input
Native harness	24,576	32,768
Controller, compact baselines, ablations	12,288	16,384
Full-context reference	131,072	131,072

Benchmarked model roster. Table 10 gives the identities in the production manifest. Each model has 96 paired task/seed records per arm from 24 template clusters. The main ten-arm comparison and its ablations use Grok 4.20 non-reasoning. The cross-model panel compares native, controller, and full-context policies on the same roster within each model. The separate local pilot uses `Qwen/Qwen3-Coder-Next-FP8` (Appendix G.5).

Table 10: Serving identities recorded for the twelve-model comparison. Grok 4.20 uses the dated non-reasoning snapshot shown. Other identifiers preserve the serving names in the model manifest.

Reported model	Serving identifier
Grok 4.20 non-reasoning	`grok-4.20-0309-non-reasoning`
Grok 4.6	`grok-4.6`
Grok 4.7	`grok-4.7`
Gemini 3.6 Flash	`gemini-3.6-flash`
Gemini 3.5 Flash	`gemini-3.5-flash`
GLM 5.3	`glm-5.3`
GLM 5.2	`glm-5.2`
GLM 5.3 Flash	`glm-5.3-flash`
Kimi 3	`kimi-k3`
Kimi 2.7	`kimi-k2.7-code`
Qwen 3.6 Plus	`qwen3.6-plus`
Qwen 3.6 27B	`Qwen/Qwen3.6-27B`

F.3 Ablations and validity controls

An ablation requires a logged change in its target, plan, contract, or realized packet. A configuration switch alone does not establish a manipulation. Zeroing boundary and role embeddings tests those channels because other metadata can still reveal the transition. Learned graph features and deterministic dependency closure are separate mechanisms, as are learned features, training targets, and hard rules for each semantic property. Planner comparisons hold scorer outputs, rules, inventory,

and budget fixed. Random ranking substitutes seeded scores while retaining locks and budget. The completed effects in Table 3 use ablated-minus-full signs.

Table 11 separates targeted member losses from changes in the composite. Removing exactness or scope/provenance targets lowers the corresponding member score. Evidence and version target removals have negative point estimates whose intervals include zero. These endpoints have different applicability sets and cannot be summed into a change in G. Removing graph closure lowers the fraction of selected artifacts with their required support from 0.927 to 0.883. The selected-artifact means are 10.21 and 9.98 respectively, so this endpoint describes each policy's selected set. The fixed-score knapsack and beam alternatives have intervals spanning zero, so the comparison does not distinguish their performance at this precision. A same-artifact test of boundary dependence additionally requires the task and other policy inputs to remain fixed while the required contract changes.

Table 11: Additional target and planner ablations, with ablated-minus-full signs and 95% template-cluster intervals. The scope/provenance score averages its applicable members within each trajectory. Closure is the fraction of selected artifacts retaining required support. The final two rows evaluate fixed-score planner alternatives using the full preservation composite.

Ablation	Endpoint	Pairs / clusters	Difference	95% interval
No exactness targets	Exactness	56 / 14	-0.050	$[-0.099, -0.001]$
No scope/provenance targets	Scope/provenance	80 / 20	-0.039	$[-0.074, -0.005]$
No evidence targets	Evidence	48 / 12	-0.030	$[-0.072, 0.012]$
No version targets	Version	24 / 6	-0.020	$[-0.068, 0.028]$
No graph closure	Closure	96 / 24	-0.044	$[-0.060, -0.028]$
Knapsack planner	G	96 / 24	-0.010	$[-0.051, 0.031]$
Beam planner	G	96 / 24	-0.020	$[-0.047, 0.007]$

F.4 Validity gates and analysis status

Definition 3 (Learning gate)**.** On an artifact-disjoint validation split, let $\mathcal{U}(\theta)$ be the fitted scorer's negative cross-entropy, $\mathcal{U}_{\text{const}} = -\ln 20$ the uniform scorer's utility, and q_K the maximum utility among K scorers refit to permuted targets. The gate passes when $\mathcal{U}(\theta)$ exceeds both $\mathcal{U}_{\text{const}}$ and q_K. Its rank score is $\big(1 + \#\{k \mid \mathcal{U}(\theta^{(k)}) \geq \mathcal{U}(\theta)\}\big)/(K+1)$. A null refit refused by fit health is replaced at $\mathcal{U}_{\text{const}}$ and disclosed, after which the rank score is not an exact permutation p-value.

Fit health checks hidden-unit saturation and training cross-entropy against the class prior, with hidden sign patterns and within-task utility spread. Artifact disjointness excludes shared artifacts and boundary events between training and validation. A stronger claim of learning beyond metadata must also exceed the best prespecified constant, class-prior, recency, token-cost, manual, anchor, and random-ranking rule under the same information condition. A failed manipulation yields no effect estimate, and incomplete restoration does not support a behavioral contrast. The production comparisons establish the performance of the evaluated controller under its declared information access. Development split and fit diagnostics in Appendix G.4 address leakage and degenerate scoring. They do not by themselves establish transfer beyond the constructed benchmark families.

G Additional results and validity checks

The additional analyses separate model breadth, preservation mechanisms, and replay prediction. Development audits and single-arm pilots use distinct populations and provide complementary checks rather than additional controller comparisons.

G.1 Model breadth and preservation members

Table 12 gives the full cross-model panel underlying Figure 2. Each model contributes 96 paired records from 24 templates. The shared roster permits comparison within each executor. It does not turn the twelve repetitions of a template into twelve independent task populations.

Table 12: Per-model operating points on 96 task/seed pairs from 24 templates. Each cell reports native harness / controller / full context, respectively. Input is the mean cumulative input in thousands of tokens. The native and controller arms use different caps and information access. Full context uses a larger reference cap. NR denotes non-reasoning mode.

Model	$G \uparrow$	Success (%)↑	Input (k)↓
Grok 4.20 (NR)	0.61 / 0.83 / 0.86	30.2 / 55.2 / 58.3	18.40 / 8.30 / 41.70
Grok 4.6	0.64 / 0.85 / 0.88	33.3 / 58.3 / 61.5	18.90 / 8.50 / 42.60
Grok 4.7	0.66 / 0.86 / 0.89	35.4 / 60.4 / 63.5	19.20 / 8.60 / 43.20
Gemini 3.6 Flash	0.60 / 0.80 / 0.83	29.2 / 51.0 / 54.2	17.90 / 8.15 / 41.10
Gemini 3.5 Flash	0.58 / 0.78 / 0.81	27.1 / 47.9 / 51.0	17.50 / 7.95 / 40.40
GLM 5.3	0.62 / 0.82 / 0.85	31.2 / 54.2 / 57.3	18.30 / 8.30 / 41.80
GLM 5.2	0.61 / 0.81 / 0.84	30.2 / 53.1 / 56.2	18.10 / 8.20 / 41.40
GLM 5.3 Flash	0.59 / 0.79 / 0.82	28.1 / 50.0 / 53.1	17.70 / 8.05 / 40.80
Kimi 3	0.63 / 0.84 / 0.87	32.3 / 57.3 / 60.4	18.60 / 8.40 / 42.20
Kimi 2.7	0.62 / 0.82 / 0.85	31.2 / 55.2 / 57.3	18.20 / 8.25 / 41.60
Qwen 3.6 Plus	0.62 / 0.83 / 0.86	31.2 / 56.2 / 59.4	18.50 / 8.35 / 42.00
Qwen 3.6 27B	0.57 / 0.77 / 0.80	26.0 / 46.9 / 50.0	17.20 / 7.85 / 40.00

The composite gain is distributed across all six preservation members (Table 13). Applicability is fixed by construction, so each member uses its own paired denominator. The cluster sign-flip results with Holm adjustment remain favorable across this six-member family. This retrospective decomposition supports broad preservation improvement under the evaluated information condition. It does not establish that the scorer adds value after removing construction metadata.

Table 13: Controller minus native preservation on applicable production rows. Intervals resample whole template clusters. Two-sided sign-flip probabilities are adjusted over these six diagnostics with Holm's procedure.

Member	Pairs / clusters	Difference	95% interval	Adjusted p
Exactness	56 / 14	0.255	$[0.202, 0.307]$	0.00061
Scope	64 / 16	0.225	$[0.178, 0.271]$	0.00018
Provenance	48 / 12	0.215	$[0.174, 0.253]$	0.00195
Evidence	48 / 12	0.234	$[0.176, 0.289]$	0.00195
Version	24 / 6	0.185	$[0.120, 0.251]$	0.03125
Exclusion	40 / 10	0.175	$[0.133, 0.216]$	0.00391

The target ablations in Table 3 use ablated minus full signs. Disabling one semantic-rule group reduces its obligation score from 261/300 to 252/300 applicable events, a difference of -0.030. This event-level obligation score is distinct from the trajectory-level preservation composite and binary behavioral endpoints. Fixed-score knapsack and beam alternatives change G by -0.010 and -0.020, with intervals spanning zero (Appendix F.3). The embedding-retrieval sensitivity arm attains $G = 0.680$, between native control and the controller.

G.2 Behavior across boundary types

Table 14 separates the primary comparison by boundary type. Compaction and delegation each contribute eight templates, while pruning and resumption contribute four each, with four seeds per template. The preservation means are 0.83 for the controller and 0.61 for native control in each subgroup. Strict-success counts favor the controller for all four boundary types, but behavioral safety does not improve uniformly. Pruning has one harmful adoption among 16 trajectories under each policy, and resumption has two violations among 16 under each policy. Delegation records no harmful adoption under either policy. These subgroup comparisons are descriptive and use the same tasks as the aggregate result.

Table 14: Descriptive boundary breakdown for Grok 4.20 non-reasoning. Outcome entries give controller / native trajectory counts, with the shared denominator in the pairs column. Each template has four seeds. Mean preservation G is 0.83 / 0.61 in each boundary subset. These small subsets do not receive separate significance tests.

Boundary	Pairs / templates	Success↑	Violations↓	Harmful adoption↓
Compaction	32 / 8	22 / 15	0 / 3	1 / 9
Delegation	32 / 8	18 / 9	2 / 5	0 / 0
Pruning	16 / 4	6 / 4	0 / 1	1 / 1
Resumption	16 / 4	7 / 1	2 / 2	1 / 2

Table 15: Replay-label probe comparisons on the same 88 held-out pairs in 22 artifact groups. AUROC intervals use 100,000 whole-group bootstrap resamples. ECE uses ten equal-width probability bins. Cost is total probe tokens divided by total replay tokens.

Probe	AUROC	95% interval	ECE	Cost ratio
Full probe	0.810	$[0.724, 0.908]$	0.099	0.111
Metadata only	0.680	$[0.585, 0.801]$	0.095	0.050
Cheap baseline	0.590	$[0.463, 0.735]$	0.125	0.021

G.3 Predicting replay labels

The probe is evaluated on the 88 complete replay records for Grok 4.20 non-reasoning, including the four no-op controls. Its label is one when reference success exceeds intervention success. Thirty records are positive, and the test set contains 22 artifact groups. All three variants score the identical records. Table 15 reports AUROC, calibration, and aggregate token-cost ratios. Expected calibration error uses ten equal-width probability bins, weighting each absolute calibration gap by the bin's sample fraction. Paired AUROC differences use the same 100,000 whole-group bootstrap draws for each variant, retaining every record in a sampled artifact group.

Every fixed-state repeated score agrees with its original score. Figure 6 shows the empirical receiver-operating curves. The full probe improves AUROC by 0.130 over metadata only, with paired interval $[0.073, 0.184]$, and by 0.220 over the cheap baseline, with interval $[0.135, 0.316]$. Its calibration error is slightly higher than the metadata variant's. Thus the additional signal improves ranking, while replay remains the behavioral label of record. The analysis evaluates held-out predictions on the declared artifact-disjoint split. It does not independently establish training-set separation or transfer to other executors.

G.4 Scorer fit and alternative explanations

The development population comprises 472 artifact-at-boundary rows from 12 trajectories and 59 artifacts. These development measurements were independently reproduced. The earlier surrogate-update scorer emitted eviction on all rows, saturated every hidden unit, and had in-sample cross-entropy 2.675 against the class prior's 1.410. A constant scorer reproduced its historical evaluation records. A separately prepared checkpoint for a real-task controller study failed all four degeneracy checks and was not used. Neither failed checkpoint is the full-context model of the local pilot.

The exact-gradient fit used a recipe committed before fitting. Its trajectory split also separates artifacts and boundary events on the observed rows (Table 16). This latter separation is measured, rather than guaranteed by splitting trajectories alone. The training-to-validation gap limits the generalization claim despite improvement over both class priors. Saturation is the fraction of hidden pre-activations with magnitude above 3, where the $\tanh$ derivative is below 0.01. Sign patterns count distinct vectors of pre-activation signs. Fit health rejects training saturation above one half or training cross-entropy that is not at least 10% below the same-row class prior.

The learning gate compares validation utility -0.89571 with the uniform scorer's $-\ln 20 = -2.99573$ and the maximum utility -1.32206 from 19 permuted-target refits. One refused refit was replaced by the uniform scorer. The fit passes the gate, but the recorded rank score 0.05 is not an exact permutation p-value with that substitution.

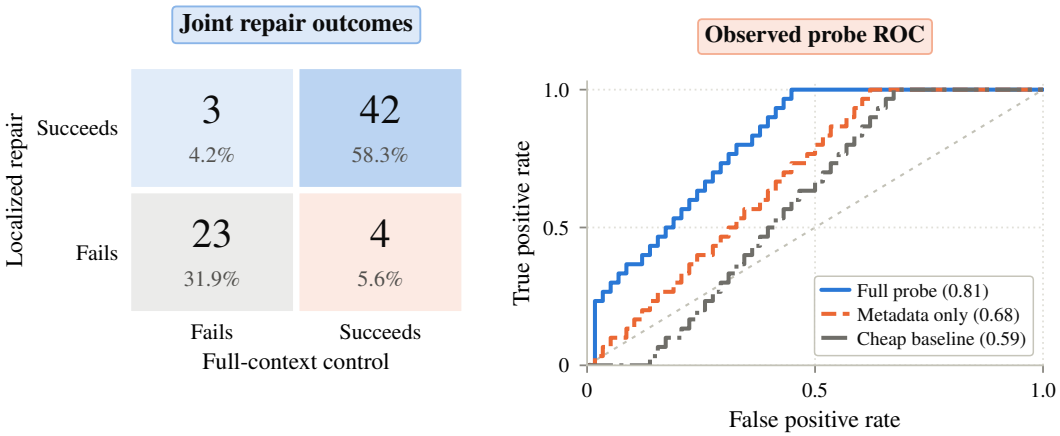

Figure 6: Repair and probe diagnostics on distinct cohorts. Left, localized repair succeeds on 45 of 72 failed trajectories and full context on 46, with 42 successes in common. Right, empirical ROC curves compare three predictors on the same 88 held-out replay pairs in 22 artifact groups. Parentheses give AUROC.

Table 16: Exact-gradient fit on an artifact-disjoint development split. The class prior uses target frequencies on the same rows as each scorer score.

Quantity	Training	Validation
Trajectories / artifacts / rows	9 / 44 / 352	3 / 15 / 120
Scorer cross-entropy	0.04481	0.89571
Class-prior cross-entropy	1.44645	1.18222
Saturation	0.00870	0.01250
Sign patterns	70	24

Artifact overlap. A separate gate test split development rows by boundary event into 96 training and 98 validation rows. Of the latter, 78 repeated a training artifact, so the gate refused the split. An artifact-carved control with 94 training and 95 validation rows had no overlap. This test concerns split validity, not the fit above. Recurring artifacts must be grouped across events to prevent leakage (Kapoor & Narayanan, 2022; Søgaard et al., 2021).

Metadata predictability. With requirement-label slots absent, the 110-slot base encoding takes 59 distinct values on the 472 rows, each associated with one target in sample. A 13-entry token-cost lookup reaches 0.800 validation accuracy against a 0.400 majority-class rate, despite complete encodings not recurring across the split. This association limits what imitation fit alone establishes. The production comparison evaluates the combined controller with its available metadata, rules, and learned scorer.

Repair calibration. A separate outcome-free calibration nests 192 repair coordinates in 144 trajectories from 12 templates, distinct from the completed set of 72 failed trajectories. A percentile bootstrap under-covered in 8 of 11 evaluable scenarios, with minimum conditional coverage 0.905 at nominal 0.95. Template dependence therefore matters for uncertainty (Miller, 2024), and repair is reported descriptively.

G.5 Local pilot configuration

The local full-context pilot used seed 42 on a fixed 20-instance SWE-Bench Pro roster (Deng et al., 2026). Table 17 retains the model and execution limits needed to interpret the run. There was no post-start retry, substitution, top-up, silent truncation, or paid fallback.

Eighteen instances were evaluated and five resolved. Two unavailable environments remain in the roster denominator and are not imputed as model failures. This local run is separate from the hosted pilot below.

Table 17: Local pilot settings. Limits are distinguished from measured use. This single-arm pilot does not estimate a controller effect.

Setting	Value
Model	`Qwen/Qwen3-Coder-Next-FP8`
Checkpoint	`da6e2ed27304dd39abadd9c82ef50e8de67bdd4c`
Serving	vLLM 0.19.0, eager mode, tensor parallelism 2, context 32,768
Decoding	Temperature 0, top-p 1, text-only chat completions
Request caps	30,720 input tokens, 2,048 output tokens, 300 seconds
Trajectory caps	32 requests, 262,144 input tokens, 16,384 output tokens, 1,800 seconds
Allocation caps	Two H100 GPUs, 24 GPU-hours, 36 hours overall wall time
Measured use	4,183,902 input and 231,446 output tokens, 21.5 GPU-hours, 31 wall-hours

G.6 Roster eligibility and hosted pilot

Reference-patch qualification on the 20-instance roster yielded 9 fresh passes and 6 ineligible instances. Five historical grading records raised eligibility to 14, below the prespecified gate of 16. Three compatibility outcomes adopted after the gate raised it to 17. Fresh, historical, and post hoc records remain separate. Qualification never counts as a model success.

The hosted full-context pilot attempted 16 of 20 instances, produced 12 patch-bearing predictions and 4 empty patches, and resolved 2 tasks. Its local ledger records 1,020 settled calls, no open reservations or ledger-rule violations, and a USD 2.26 estimate without invoice reconciliation. This historical single-arm ledger is separate from the named production executors. The pilots establish execution feasibility. They supply no estimate of a controller effect on SWE-Bench Pro, where local-file, Git-history, network, and task-metadata leakage also affect evaluation validity (Zheng et al., 2026).

H Closest comparisons

Typed retention and boundary failures. ClawVM is the closest systems precedent. It manages typed state pages with minimum-fidelity invariants, multiple representations under a token budget, and validated writeback at lifecycle boundaries (Rafique & Bindschaedler, 2026). Residency contracts instead specify the task information delivered to a receiver, condition obligations on its role and the boundary type, and evaluate them by replay. CWL's dependency-linked episodes and deterministic eviction policy provide a precedent for dependency-aware retention (Semenov & Dorofeev, 2026). Knowledge Triage combines typed retention with source-authority weights and replicated constraints (Zerhoudi et al., 2026). Contracts add explicit receiver conditions, realizations, and behavioral tests.

Several observations motivate these distinctions. ConstraintRot reports policy violations rising from 0% with full context to 30% after compaction over 1,323 episodes, with constraint pinning restoring compliance (Chen, 2026). Constraint-aware extraction similarly improves retention of session constraints (Wang et al., 2026c). Handoff compression can deactivate prerequisites, while restoring explicit state fields reduces forbidden actions (Sun et al., 2026b). Delegation can improve task completion while increasing unauthorized actions (Xu et al., 2026), and MAST includes lost history and withheld information among multi-agent failure modes (Cemri et al., 2026). These studies motivate measuring conditions of use separately from task success. Compression can also increase retrieval calls without a significant completion change, with effects that depend on the environment (Liu, 2026).

Learned context management and simple comparators. ACON uses downstream failures and successful compressed runs to optimize and distill compression guidelines (Kang et al., 2026). AgentFold and CAT protect structured task state (Ye et al., 2026; Liu et al., 2025b), while Context Folding trains branch-and-return behavior with scope-aware process rewards (Sun et al., 2026a). These methods motivate learning from behavior and retaining explicit structure. A contract exposes finer obligations that can be checked independently of when an agent chooses to compress or return. Learned editing and retention also optimize task outcomes and cost (Zhang et al., 2026c; Yi et al., 2026; Li et al., 2026; Lin et al., 2026). CoACT preserves next-action agreement (Chen et al., 2026a),

Table 18: Full comparison for the six obligation types in Table 1. Each row pairs a reported or constructed failure with related approaches, partial forms in today's harnesses, and the contract form and replay intervention that test it.

Obligation and what is lost	Reported or constructed failure	Related approach	Partial form in today's harnesses	Contract form and *test intervention*
• **Exactness**. Rule wording or binding status	Safety rules kept fall from 53% to 10% over one to five `/compact` rounds (Zerhoudi et al., 2026). 17% of constraints kept (Wang et al., 2026c). Violations rise from 0% to 30% (Chen, 2026). Blockers deactivated in 100% of compressed handoffs (Sun et al., 2026b).	Per-type retention, constraint extractor or pinning, restored state fields	Summary with preservation instructions, no checkable predicate (Anthropic, 2025)	Exact, or structured with a binding marker. *Paraphrase or delete the field*
• **Scope**. Audience or permission limit	Boundary markers fall from about 0.80 to 0.57 at a 25-word cap (Wang et al., 2026a). Unauthorized actions in 2.7 to 19.8% of tasks under centralized delegation versus 0.6 to 0.8% in peer networks (Xu et al., 2026).	Audience allowlist, re-anchoring to the request	Subagent brief with no scope field	Scope obligation for role r. *Strip the scope*
• **Source role**. Tool or memory text as authority	Authority collapse in 48 of 49 configurations (Zhan et al., 2026). Indirect prompt injection (Greshake et al., 2023).	Authority labels, instruction hierarchy, and prompt/data separation with structured queries (StruQ) (Chen et al., 2025a)	Tool output appended to the history	Data-only with provenance. *Promote to the instruction role*
• **Retrievability**. Evidence dropped without a handle	Retrieval calls rise in all six model-regime comparisons while completion changes are not significant (Liu, 2026).	Memory pointers (Packer et al., 2024), retrieval with pinned rules	Truncation, observation masking (Lindenbauer et al., 2025), condensers (Wang et al., 2025)	Evidence pointer resolvable within budget. *Break the pointer*
• **Freshness**. Stale conclusion reused	BoundaryBench stale-guidance, resumed-state, and version-mismatch families (constructed)	Revalidation at resume	Resume reloads the prior summary	Exclusion, stale marker, or version expiry. *Reinstate the stale artifact*
• **Dependency**. Claim kept without its support	Constructions of Proposition 1	Dependency-linked eviction (Semenov & Dorofeev, 2026)	None standard	Dependency closure. *Remove a dependency*

Knowledge Triage (Zerhoudi et al., 2026) addresses exactness, source authority, and in-scope rules jointly. Appendix B extends the mapping.

one choice of the utility in Equation (1). Our endpoint vector additionally separates task success from violations.

Observation masking is an essential cheap comparator. In the evaluated coding setting, masking older observations halves cost relative to the raw agent while matching LLM summarization's solve rate (Lindenbauer et al., 2025). Contracts therefore need comparisons at matched information access and budget, including deterministic rules. A richer output vocabulary alone establishes neither better compression nor learned value.

Interventions and continuation-based evaluation. Causal memory selection (Srivastava, 2026) and context attribution (Cohen-Wang et al., 2024; Liu et al., 2025a) use interventions to assess information. Agent failure attribution applies related logic to agents or steps (Zhang et al., 2025; 2026a; Shah, 2026; Bonagiri et al., 2026). Our intervention unit is a typed change to the packet crossing one boundary. Deletion, provenance removal, and summarization ask different questions, so their signed effects remain separate.

TRACE compares closed-loop continuations from the same environment state before and after compaction, then uses the preferences to optimize a compression prompt (Min et al., 2026). Boundary-local paired evaluation is thus shared ground. Contracts extend the intervention vocabulary to individual obligations and to handoff, pruning, and resume. Slipstream validates summaries against a continuation from the pre-compaction state (Chen et al., 2026b). Our test executes both packets and measures endpoints. Trajectory composition requires matching checkpoint and continuation policies (Section 4). Deployment can change the states reached, consistent with the covariate-shift concern of Ross et al. (2011). Compaction theory based on communication complexity addresses a different question, the representation budget needed for a query set (Tirmazi et al., 2026).

Enforcement and benchmark integrity. A data-only placement specifies a source role. Receiver defenses must enforce that role, for example through instruction hierarchy, separated prompt and data channels, or system-level information flow (Wallace et al., 2024; Chen et al., 2025a;b; Debenedetti

et al., 2026; Costa et al., 2025). AttriGuard checks tool calls by re-execution under attenuated observation control (He et al., 2026), while provenance contracts carry security commitments across agent components (Zheng & Yang, 2026). Residency contracts provide no security guarantee of their own.

Long-context position effects and distraction motivate placement-matched controls (Liu et al., 2024; Shi et al., 2023). SWE-Bench Pro supplies executable task outcomes (Deng et al., 2026), while its verified variant identifies local-file, Git-history, network, and metadata leakage routes (Zheng et al., 2026). Contamination and weak tests can otherwise inflate coding-agent results (Aleithan et al., 2024; Liang et al., 2026). These findings support separate checks of evidence access, grading integrity, and resource use (Kapoor et al., 2025; Zhu et al., 2026).