OBLIGATE: OBLIGATION-GATED RUNTIME REPAIR FOR LLM WORKFLOW OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM workflow optimization has largely focused on selecting or constructing workflows before execution, while failures after partial execution are commonly handled by retrying or regenerating the failed output. We study how to transition an already-executed workflow from a failed state to a verified state while preserving valid computation. We formulate post-failure repair as a transactional state transition over a versioned artifact DAG: runtime evidence opens a typed obligation that restricts legal interventions; candidate testing occurs on cloned state; dependency-closed replay restores consistency; and verifier improvement determines promotion. On strict MBPP+ (378 tasks, three seeds), ObliGate reaches $80.95 \pm 0.70\%$ raw *pass*@1 at 325 task-action tokens, reducing tokens by 50.5% relative to fixed execution. In controlled latent-root failures, it recovers 76.9% of actionable cases versus 28.1% for draft-only repair. Across 228 naturally occurring failures, the complete arm records 33.3% score-improving repairs and the study quantifies replay-scope costs; controlled faults provide the cross-artifact recovery test. The same evidence-to-action interface is instantiated for code, mathematics, and QA under adapter-registered, single-writer workflows.

1 INTRODUCTION

LLM workflow optimization concerns executable combinations of model calls, retrieval, tools, and verification. Optimizers commonly search a reusable template or select a task-specific graph before execution. Once a workflow has produced artifacts, however, a verifier failure creates a different optimization problem: which existing artifact may be changed, what evidence authorizes that change, which derived results become stale, and when can the replacement become live?

A verifier-feedback loop efficiently repairs a faulty draft. Stateful recovery poses two further decisions: which artifact to intervene on, and how to propagate that intervention without discarding unaffected work. Permission to edit more artifacts supplies capability, but does not select an effective repair; a local improvement also needs consistent descendants before it can become live. We separate *intervention selection* from *dependency-consistent execution*. Runtime evidence constrains admissible actions, cloned probes test candidate interventions, and the executor evaluates the resulting state before promotion. The optimization object is a repair transition over an already realized workflow.

ObliGate separates this transition into evidence-constrained intervention selection (Figure 1) and dependency-consistent repair execution (Figure 2). Typed obligations and registry checks determine admissible actions; the controller proposes roots and replay plans. The executor tests each candidate on cloned, versioned state, replays registered dependencies, and uses verifier improvement to determine promotion. Reservations and usage traces account for resource consequences. The framework is inference-only: it performs no weight update, prompt optimization, or task-specific fine-tuning.

Our contributions connect three responsibilities: **(1) Evidence-constrained intervention selection:** typed obligations admit compatible actions, and configurable checkpoint-local probes test authorized interventions. **(2) Dependency-consistent repair execution:** versioned lineage, isolated candidates, replay plans, and verifier-gated promotion produce coherent replacements, with sufficient conditions for preservation outside the dependency closure. **(3) Measured decision and execution value:** feedback-controlled comparisons, replay-scope contrasts, and candidate traces distinguish terminal quality, recomputation, and promotion outcomes with explicit resource accounting. The evidence separates conditional invocation, intervention coverage, and execution scope: Trace-Repair supplies

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.

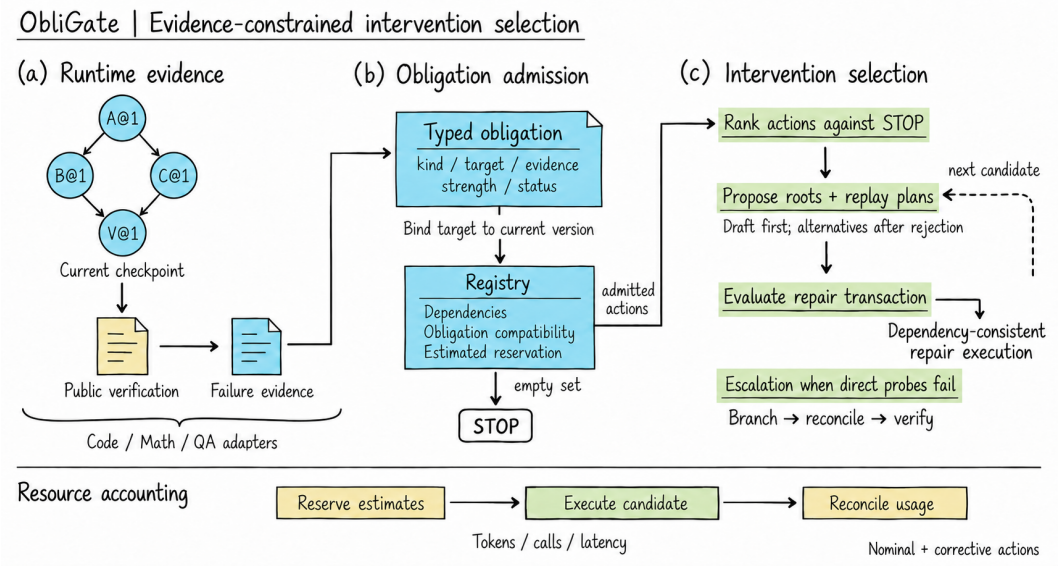


Figure 1: Evidence-constrained intervention selection. Runtime evidence opens a typed obligation. Registry checks admit actions for ranking against STOP; candidate roots and replay plans pass to dependency-consistent repair execution (Figure 2). Failed direct probes permit escalation. Reservations are reconciled with actual usage.

a feedback baseline, controlled faults test cross-artifact coverage, and the natural-failure factorial measures replay-policy cost and recovery operating points.

2 Related Work

Workflow optimization. AFlow and AdaptFlow optimize workflow structure or reusable initialization; FlowReasoner, Workflow-R1, and LLM-as-Scheduler adapt task-level policies or execution schedules (Zhang et al., 2025; Zhu et al., 2025; Gao et al., 2025; Kong et al., 2026; Xiang et al., 2026). ObliGate addresses a complementary optimization object: the post-failure transition of materialized artifacts. Its decision couples admissible interventions with the downstream work and verification required to make them useful.

Verification and localized recovery. QualityFlow uses quality checks during program synthesis, while Doctor-RAG localizes failed RAG steps and reuses a valid prefix (Hu et al., 2025; Jiao et al., 2026). DenoiseFlow studies uncertainty-aware computation and recovery (Yan et al., 2026). ObliGate makes authorization, checkpoint-local version tracking, replay closure, promotion, and resource admission explicit in one cross-domain transaction. Trace-Repair provides the shared verifier-feedback comparison.

Governance and dependency state. POLARIS governs typed execution, ContextWeaver organizes dependency-aware memory, and ALAS and Mnemosyne expose transactional state and recovery (Moslemi et al., 2026; Wu et al., 2026; Geng and Chang, 2025; Chang et al., 2026). ObliGate makes the post-failure choice of intervention and replay scope explicit within this stateful setting. Its contract links admissibility, checkpoint versions, candidate verification, and resource records; the evaluation measures the resulting quality–computation trade-offs.

3 METHOD AND EXECUTION CONTRACT

3.1 EVIDENCE-CONSTRAINED INTERVENTION SELECTION

State $S_t = (G_t, Z_t, O_t, B_t, H_t)$ contains a registered artifact DAG, current versions, open obligations, resource allowances, and execution history. Artifacts include evidence, drafts, parsed answers, and verifier records, with producer and parent-version identifiers. The replay graph distinguishes data dependencies from historical repair-feedback links. Runtime information $u(x)$ is separate from evaluation-only information $h(x)$: hidden tests and gold answers are consulted only after termination.

An adapter converts verifier evidence into

$$o = (\text{kind}, \text{target}, \text{evidence}, \text{strength}, \text{status}).$$

The target names a logical artifact; the serialized executor operates on its current checkpoint version. The registry checks obligation status, kind, and adapter-local evidence strength, while the repair plan supplies the concrete roots. Public execution failures admit code repair; structured failures admit answer correction; incomplete QA support admits evidence acquisition. Infrastructure exceptions trigger infrastructure retries, not task obligations. Obligations and candidate plans are evaluated within the current checkpoint; concurrent writers require a separate version-validation protocol.

For action a, the registry checks dependencies, obligation compatibility, and its reservation:

$$\mathcal{A}_{\text{legal}}(S_t) = \{a : \text{deps}(a, S_t) \wedge \text{auth}(a, O_t) \wedge \widehat{\mathbf{c}}(a) \preceq B_t\}.$$

The controller ranks only this set against STOP. Optional support-conditioned lower-confidence-bound ranking uses prior strength 3.0 and uncertainty penalty 0.04; support is disjoint from target tasks. Ranking cannot override the registry. This separation makes an evidence-specific permission enforceable even when several candidate actions have similar predicted utility.

3.2 DEPENDENCY-CONSISTENT REPAIR EXECUTION

The adapter supplies root candidates with executable replay plans, with $K = 2$ in the main configuration. The draft-root candidate is tried first; further candidates are tested if it fails, and an accepted improvement ends probing. Each candidate runs on a clone and refreshes the downstream artifacts required by its registered plan. Promotion requires $V(S') - V(S_t) \geq \epsilon_{\text{prom}}$; otherwise the live checkpoint is retained. The action-versus-STOP and promotion margins are both 0.015, but apply to different quantities. The natural-failure suite uses its separately reported candidate limits.

For example, a failed public assertion on `draft@1` opens a draft obligation. A clone produces `draft@2` and refreshes its dependent verdict before promotion. The configured public tests and acceptance rule remain fixed: refreshing a verification artifact is not permission to relax the test. If direct probes fail, a branch–reconcile cycle can combine alternatives before re-verification. Multiple obligations can coexist, but edits commit serially.

Dependency-consistency proposition. Let $D(R; G_t)$ be the registered descendants of edited roots R. Assume complete read/write registration, an acyclic graph, one live writer, a closure-complete topological replay plan, and pure, idempotent, or appropriately compensated replay. Rejection preserves live artifact versions. Acceptance refreshes $R \cup D(R; G_t)$ and preserves versions outside it; a promoted state contains no stale registered descendant of an accepted root. *Proof.* Probe writes are isolated. Topological replay visits descendants after their parents; induction therefore establishes consistent parent versions throughout the closure. Serialized promotion installs that candidate, while rejection discards it. The proposition covers any DAG with such a plan; the implementation uses adapter-supplied plans. It establishes dependency consistency, with semantic quality assessed separately by the fixed verifier. Appendix J derives the corresponding fixed-edit cost condition.

3.3 RESOURCE ADMISSION AND ACCOUNTING

The executor reserves estimated tokens, calls, and latency before each action and reconciles provider-reported usage afterward. The main experiments use a reserved-correction contract: corrective actions become admissible only after a failure obligation opens. Executed nominal and corrective workflow

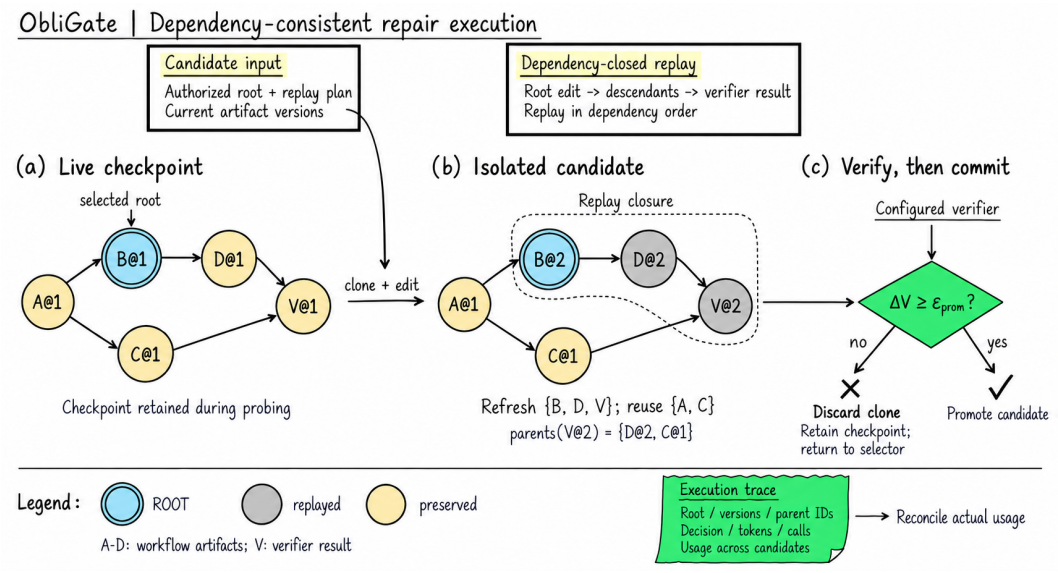


Figure 2: Dependency-consistent repair execution. An authorized root edit runs on an isolated checkpoint. Editing B refreshes B, D, V while reusing A, C and recording the new parent versions. The configured verifier promotes a completed candidate when $\Delta V \geq \epsilon_{\text{prom}}$; otherwise the checkpoint is retained. The trace records decisions and usage across candidates.

actions, including rejected probes, contribute to C_{task}; provider-transport and hidden-evaluator retries enter a separate infrastructure ledger. Deterministic tests consume latency but no provider tokens.

Estimated admission and hard-reserve execution are distinct modes. The former stops and records an episode when realized usage exceeds its nominal ceiling. The hard-reserve mode blocks actions whose reservation plus reserve cannot fit and rejects overlong results before downstream execution. Its accepted-action ledger excludes these rejected outputs, whose usage is retained in separate trace events. The primary matrix measures realized reservation violations; the separate reserve audit measures commit and termination enforcement. "Strict MBPP+" denotes the hidden EvalPlus evaluation protocol.

4 Experimental Design

The primary experiment uses 378 strict MBPP+ tasks, DeepSeek-V4-Pro at temperature zero, seeds 13/42/2026, and token ceilings 1200/1800/2400, with at most eight calls, 60 seconds of admitted latency, and four workers. Five internal configurations share the backend, prompts, manifest, public verifier, hidden evaluator, and action interface: One-Pass constructs once; Fixed-Deep follows the deep schedule; Trace-Repair performs bounded draft repair with verifier feedback; Value-Routed uses soft action ranking; Obligation-Gated enables the complete transaction. Trace-Repair and Draft Repair denote the same draft-local control family; the natural-failure study uses the latter label.

Raw *pass*@1 counts every task. Resolved *pass*@1 excludes only exhausted evaluator errors/timeouts after bounded retries; raw quality remains primary. Tables report seed means and sample SD. Independent selector, support, sensitivity, and trigger suites retain their own calibration identities (Appendix A); contrasts are computed within each suite. Natural-failure repair intervals use case-bootstrap resampling. Public tests, parser outputs, and retrieved support guide execution; no hidden outcome enters runtime decisions.

Mechanism studies comprise latent-root faults, known-root replay, a controlled 20-record rerun comparison, and 228 natural failure checkpoints (169 draft-local, 22 parser, 34 retrieval/coverage, 3 compound). Five continuation arms restore each saved checkpoint; full rerun restarts the same task. Natural repair denotes offline terminal-score improvement, distinct from online acceptance and full-state correctness. Costs include the recorded prefix and corrective usage; correction-exempt

Table 1: Strict MBPP+ on 378 tasks, three seeds, and five internal configurations. Raw and resolved scores are mean ± sample SD; resolved excludes exhausted evaluator timeouts/errors. Tokens include executed workflow actions; Viol. reports realized nominal-ceiling violations under reserved-correction admission.

Budget	Method	Raw *pass*@1(%)	Res. *pass*@1(%)	Tokens	Calls	p95 (s)	Viol. (%)
1200	One-Pass	76.98 ± 1.06	77.95 ± 0.67	235	1.00	1.96	0.00
	Trace-Repair	78.57 ± 1.06	79.92 ± 0.66	277	1.10	2.72	0.44
	Value-Routed	78.04 ± 0.70	79.03 ± 0.67	476	1.62	4.20	1.23
	Fixed-Deep	77.69 ± 2.25	78.72 ± 1.10	596	2.02	4.04	1.94
	Obligation-Gated	$\mathbf{79.90 \pm 0.26}$	$\mathbf{80.18 \pm 0.38}$	282	1.11	3.74	0.79
1800	One-Pass	76.81 ± 0.40	77.15 ± 0.30	236	1.00	2.70	0.00
	Trace-Repair	79.98 ± 0.81	80.55 ± 0.72	290	1.11	3.30	0.53
	Value-Routed	79.72 ± 0.81	80.14 ± 0.81	644	2.10	5.72	0.88
	Fixed-Deep	80.16 ± 0.53	80.59 ± 0.53	656	2.13	6.08	1.50
	Obligation-Gated	$\mathbf{80.95 \pm 0.70}$	$\mathbf{81.86 \pm 0.71}$	325	1.20	4.13	1.59
2400	One-Pass	76.37 ± 0.40	76.57 ± 0.41	236	1.00	2.29	0.00
	Trace-Repair	79.72 ± 0.61	80.35 ± 0.44	289	1.12	3.17	0.09
	Value-Routed	79.72 ± 0.31	80.21 ± 0.38	691	2.17	7.18	1.06
	Fixed-Deep	80.78 ± 0.15	81.28 ± 0.14	701	2.19	6.81	1.68
	Obligation-Gated	$\mathbf{81.13 \pm 0.15}$	$\mathbf{81.70 \pm 0.14}$	337	1.21	4.15	1.32

execution makes this a realized-cost comparison. Appendix G specifies arm limits. Local scans cover 131 HumanEval, 341 MBPP, 1,055 GSM8K, 486 MATH, and 800 HotpotQA tasks, using code accuracy, math exact match, and QA F1. The seed-13 QualityFlow bridge shares the strict manifest and evaluator.

5 Results

5.1 Quality and computation

Table 1 retains all 15 primary cells. Relative to Fixed-Deep, ObliGate improves raw *pass*@1 by $2.21/0.79/0.35$ points and reduces task-action tokens by $52.7/50.5/51.9\%$. More importantly for incremental value, Trace-Repair already uses failure-contingent feedback: ObliGate adds $1.33/0.97/1.41$ points for $5/35/48$ tokens. These gains evaluate the complete repair policy beyond conditional invocation. The mechanism studies examine how that policy uses its additional computation.

The action ledger explains this operating point (Figure 3). At budget 2400, every configuration spends approximately 236 tokens constructing a draft. Fixed-Deep adds 358 tokens of evidence acquisition and unconditional later stages; ObliGate instead allocates 101 tokens to classification, probes, and selective escalation. With initial correction probability q, conditional cost is approximately $C_{\text{construct}} + C_{\text{verify}} + qC_{\text{correction}}$. Candidate testing adds a state-level check absent from draft-only repair. The main matrix reports the aggregate policy effect; controlled and natural studies then separate coverage, replay scope, and cost.

5.2 Authorization, promotion, and escalation

Table 2 tests component removals at budget 1800, seed 13. Removing admission lowers quality by 5.55 points and adds 416 tokens; removing escalation lowers quality by 2.38 points. The clone–probe–replay path uses 94 fewer tokens than the no-replay variant at a comparable score. The natural factorial separately varies replay scope. Global memory retains the score with 68 additional tokens; support memory is an optional ranking aid.

The matched trigger suite is a routing diagnostic: it preserves typed legality, probing, and promotion while replacing kind/strength filtering with any-failure routing. Signed differences ($+0.18/-0.88/+0.88$ points) vary by budget, so the result supports auditability and admissibility, not a stable quality gain from typing. A separate strength-aware development check improves 38.55% of repairs and degrades 8.43%; the full diagnostics are in Appendix D. Across the main cells, 129 probes are promoted, 191 fail the improvement test, and 210 are infeasible. Thus 75.7% of considered candidates never replace live state; hard action-precondition violations are zero.

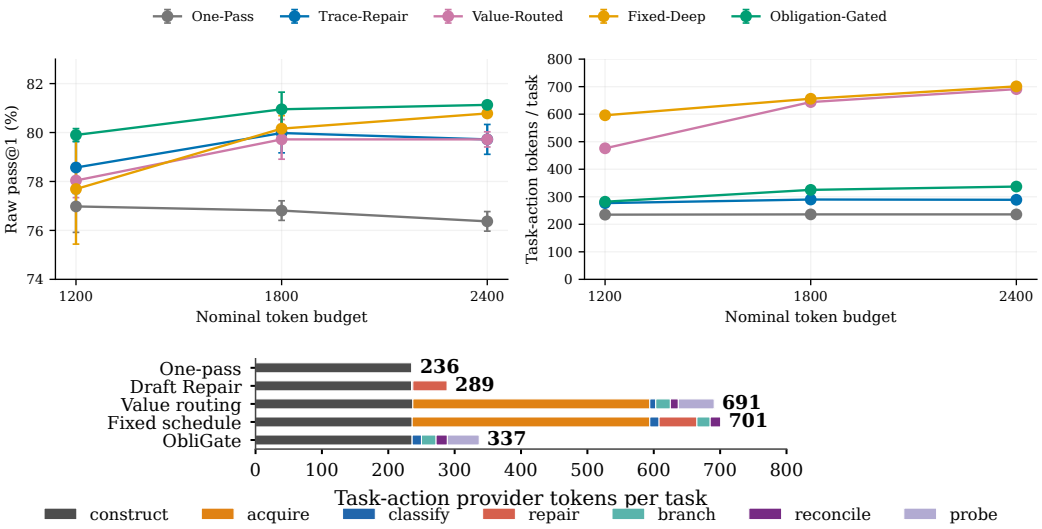


Figure 3: Top: strict MBPP+ quality and task-action tokens for the five internal configurations. Bottom: the budget-2400 action ledger, including probes and escalation. Deterministic public verification uses no provider tokens.

Table 2: Component ablation at budget 1800 (seed 13). All variants share the task manifest, backend, and evaluator.

Variant	Raw *pass*@1 (%)	Tokens	Calls
Obligation-Gated	80.95	315	1.20
No local replay	80.69	409	1.25
No branch escalation	78.57	401	1.28
Global action memory	80.95	383	1.25
No obligation gate	75.40	731	2.13

5.3 Recovery beyond the current draft

The latent-root matrix tests 1,344 actionable cases, 288 clean controls, and 288 unrecoverable controls per configuration. ObliGate recovers 76.9% versus 28.1% for draft repair (paired difference 48.8 points; 95% interval 46.3–51.5; McNemar $p < 10^{-190}$). Upstream, verifier-record, and compound recovery reaches 50.0%, 100.0%, and 66.7%. Expanded edit scope enables these interventions; disabling the local-repair path reduces recovery to 61.9% and removing escalation to 41.7%. These bundled controls test the recovery path; the natural factorial separately varies replay scope. Any-failure replay ties ObliGate, and fixed/value-routed execution saturates on the recoverable construction (Appendix C).

Table 3 measures controlled recovery cost: local correction uses 55.9% fewer tokens than restart, or 14.6% fewer including its nominal prefix. The verifier-root construction concerns an affected verdict record under a fixed evaluator, not changing the trusted checking rule. Natural failures next test realized policy costs without injecting faults.

5.4 Natural failures and mechanism decomposition

Table 4 isolates replay scope under a shared checkpoint. Closed-draft replay yields 76/228 score improvements at 7,934.9 tokens, matching the complete ObliGate arm (76/228) at 19,549.2 tokens; probe-enabled replay changes the policy operating point but does not add aggregate recoveries in this corpus. Its main measured benefit is reduced unnecessary recomputation when a fixed probe policy is retained. Direct draft repair and full rerun improve 72/228 and 47/228, respectively.

Table 3: Complementary controlled recovery studies. Known-root values are means over three seeds; the executed comparison measures both alternatives on the same 20 records.

Study / strategy	Repair (%)	Tokens	Other outcome
Known-root replay (251 records)	100.0 ± 0.0	807 ± 29	Three-seed recovery
Known-root reservation reference	—	2,090	Replay saving $61.39 \pm 1.38\%$
Executed full rerun (20 records)	90.0	1,715.75	4.80 calls
Executed local replay (20 records)	100.0	756.80[†]	3.90 calls

[†] Replay-stage tokens exclude the nominal prefix; end-to-end local replay averages 1,465.45 tokens.

Table 4: Natural-failure factorial on 228 cases. Improvement and false repair are scored on initially incorrect answers; false repair means a changed answer without score improvement. Abstention means an unchanged draft. All rates retain the 228-case denominator; brackets give bootstrap 95% intervals. E2E tokens include recorded probes and replay; recomputation is an artifact ratio. The final four arms cross root probing and replay scope.

Method	Score improved (%)	False repair (%)	Abstain (%)	E2E tokens	Unnec. recompute
Full rerun	20.6 [15.4,25.9]	20.2	46.5	1,920.5	1.000
Draft repair	31.6 [25.4,37.3]	8.3	47.8	5,826.5	0.671
Replay full draft	32.5 [25.9,39.0]	10.5	52.6	7,867.7	0.167
Replay closed draft	33.3 [27.2,39.5]	10.5	51.8	7,934.9	0.000
Replay full probe	33.8 [27.2,39.9]	8.3	56.1	24,985.1	0.167
ObliGate (full)	33.3 [27.2,39.5]	8.3	57.0	19,549.2	0.167

Averaged over both root policies, closed replay saves 2,684 tokens and lowers normalized unnecessary recomputation by 0.083. Under the fixed probe policy it saves 5,435.9 tokens (paired 95% interval: 3,722–7,185 fewer); under draft-root repair it changes cost by only 67.2 tokens and two improvements. Thus replay scope has the clearest natural-failure effect. Probing defines a more conservative, higher-cost operating point whose quality benefit is not established by this corpus.

All 76 ObliGate improvements occur in draft-local cases; the 37 retrieval/coverage and compound cases show no measured improvement. The full stratum decomposition is retained in Appendix G; in the main text, the natural study is used to quantify replay-scope cost, while controlled faults provide the cross-artifact recovery result.

5.5 Resource enforcement and controller robustness

The hard-reserve audit uses 16 fixed synthetic-overrun tasks at a 3,200-token budget (Table 5). A minimum-safe-cost check and post-action rejection reduce accepted-action ledger violations from 12.5% to zero while retaining *pass*@1 at 68.8%; 6.25% of episodes are terminated after an overlong result is recorded separately. This provides a measured bounded-execution mode alongside the reservation-based primary matrix.

Three-seed selector and calibration tests show that the governed operating point does not depend on one ranking rule (Table 6). LCB is within 0.18 points of budgeted ReAct with 7.8% fewer tokens. Utility-only loses 5.47 points and nearly doubles tokens. Requested support sizes $0/32/64/128$ retain $0/1/3/11$ examples after overlap filtering; LCB stays in the same quality band. The independent sensitivity blocks vary margins, probe count, and step cap: tested margins span 0.27 points, and a third probe adds 48.1 tokens without improving the mean score. All absolute sensitivity results are retained in Appendix D.

An additional provider-free threshold suite uses disjoint support/evaluation tasks in code, math, and QA. Scores remain constant across margins $0/0.015/0.05$; the math adapter reduces mean usage from 1,227.2 to 1,193.3 tokens. A compound-fault rerun recovers all designed cases with the complete and fixed controllers, compared with none for feedback-only repair, exercising the one-cycle escalation path. Registered write conflicts are rejected. Appendix E retains these threshold, contract, and compound checks.

Table 5: Reserve-and-reject audit on 16 synthetic-overrun tasks at budget 3,200. Tokens, calls, and violations use the accepted-action ledger; Trunc. records output rejection and termination, with rejected usage stored separately. Scores and rates are percentages.

Variant	Pass@1	Tok.	Calls	Viol.	Trunc.
Original	68.8	2,430	7.06	12.5	0.0
Reserve 32	68.8	2,384	6.88	0.0	6.25
Reserve 64	68.8	2,384	6.88	0.0	6.25

Table 6: Selector and support tests at budget 1800 (three seeds). Selector runs retain 11 of 128 requested support examples. Support rows show requested→retained counts. *Two empirical-mean seeds cold-start and stop on every target; dashes indicate unreported metrics.

Factor	Setting	Raw *pass*@1 (%)	Tokens	Calls	p95 (s)
Selector	Lower confidence bound	80.51 ± 0.61	370.1 ± 3.1	1.23	5.41
	Budgeted ReAct rule	80.69 ± 0.46	401.6 ± 4.9	1.23	5.76
	Prior only	80.95 ± 1.21	388.2 ± 6.1	1.27	6.50
	UCB	80.16 ± 0.26	378.6 ± 4.2	1.24	5.91
	Thompson sampling	80.86 ± 0.40	395.2 ± 18.6	1.28	5.87
	Utility only	75.04 ± 3.27	708.6 ± 66.1	2.14	5.98
	Empirical mean*	26.37 ± 45.67	125.8 ± 217.9	0.41	1.91
Support	$0 \rightarrow 0$	79.45 ± 1.22	396.9	—	—
	$32 \rightarrow 1$	80.95 ± 0.46	397.5	—	—
	$64 \rightarrow 3$	80.25 ± 0.55	387.9	—	—
	$128 \rightarrow 11$	80.51 ± 0.61	370.1	—	—

5.6 SAME-BACKEND WORKFLOW COMPARISONS

Table 7 reports local DeepSeek-V4-Pro runs, with the internal policy named for each ObliGate operating point. The external workflows retain their prompts and schedules, so scores are paired with realized tokens; no average mixes code accuracy, math exact match, and QA F1. HumanEval gains 0.76 points over SWE-Agentic with 6.23 times fewer tokens. The original-MBPP and math rows demonstrate high-quality operating points under the shared adapter interface. HotpotQA's Fixed-Deep row evaluates QA coverage, while strict MBPP+ provides the replicated resource-controlled comparison.

Table 7: Locally executed DeepSeek-V4-Pro workflow operating points. Internal ObliGate policies are named per dataset; external methods retain their workflow prompts and schedules. Code and math scores are accuracy; HotpotQA reports answer F1. AgentPrune uses a 100-task math subset; other rows use the full local manifests.

Domain / dataset	Score (%)	Tok/T	Framework / policy
Code generation (HumanEval / MBPP)			
SWE-Agentic	**96.95**/ – –	4,227 / –	PEGV execution gate
ObliGate	**97.71**/**95.31**	**679**/**422**	failure-gated / trace-only
AFlow	91.60 / 61.00	705 / 562	one-pass workflow
Direct	80.92 / 57.48	741 / 560	zero-shot
Mathematical reasoning (GSM8K / MATH)			
ObliGate	**96.40**/**79.01**	581 / 1,253	artifact-replay / static-minimal
Direct	87.58 / 36.01	506 / 961	zero-shot
AgentPrune (100 tasks)	80.00 / 39.00	2,670 / 4,195	five-agent debate
Open-domain QA (HotpotQA)			
ObliGate/Fixed-Deep	**70.06**	3, 115	selected QA operating point
AFlow	60.18	1,418	default workflow
Direct	18.12	**1, 064**	zero-shot
SC3	15.81	3,216	three-sample consistency

The protocol-matched QualityFlow bridge scores 79.63% at 558 tokens on seed 13, compared with the corresponding seed-13 ObliGate cell at 80.95% and 315 tokens. The replicated three-seed endpoint reported above is 80.95±0.70% at 325 tokens. Published cross-dataset values retain their original model and evaluation settings in Appendix F. Hidden-evaluator timeouts and errors are tracked

separately from public verifier failures; trace storage averages 1.33–1.35 KB per task. The evaluator audit and trace cases are preserved in Appendix H.

6 Discussion and Conclusion

ObliGate makes post-failure workflow optimization a decision over state, not only a decision to spend another model call. Trace-Repair establishes the efficient feedback baseline; the complete policy adds terminal-quality gains. Replay ablations establish execution value within the configured policy; controlled faults demonstrate recovery beyond draft editing, while natural failures quantify replay-scope costs and recovery operating points. The design principle is to separate evidence-constrained intervention selection from dependency-consistent execution: candidate states are evaluated before commitment, and replay limits the work required to realize them.

The consistency result requires registered dependencies, complete replay plans, and one live writer; adapter-local verification separately assesses candidate quality. The main evaluation uses one backend, and natural strata and the hard-reserve audit retain their reported sample sizes. The results establish quality gains over feedback-only repair, a replay-scope cost advantage under a fixed probe policy, and controlled cross-artifact recovery; natural failures do not by themselves establish an additional root-probing quality gain. Together they support a practical, inspectable approach to optimizing recovery while preserving unaffected computation.

AI use statement

Generative AI tools assisted manuscript language editing, framework-figure preparation, and LaTeX formatting. The reported LLM workflow evaluations are described in the experimental protocol and appendices.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *International Conference on Learning Representations*, 2024.

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

Edward Y. Chang, Longling Geng, and Emily J. Chang. Mnemosyne: Agentic transaction processing for validating and repairing AI-generated workflows. *arXiv preprint arXiv:2607.00269*, 2026.

Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Varun Gandhi, Jaewook Lee, Shantanu Todmal, Franck Dernoncourt, Ryan Rossi, Zichao Wang, and Andrew Lan. GRASP: Granularity-aware search policy for agentic RAG. *arXiv preprint arXiv:2607.10463*, 2026.

Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin, and Tianyu Pang. FlowReasoner: Reinforcing query-level meta-agents. *arXiv preprint arXiv:2504.15257*, 2025.

Hector Garcia-Molina and Kenneth Salem. Sagas. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 249–259, 1987.

Longling Geng and Edward Y. Chang. ALAS: Transactional and dynamic multi-agent LLM planning. *arXiv preprint arXiv:2511.03094*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021.

Yaojie Hu, Qiang Zhou, Qihong Chen, Xiaopeng Li, Linbo Liu, Dejiao Zhang, Amit Kachroo, Talha Oz, and Omer Tripp. QualityFlow: An agentic workflow for program synthesis controlled by LLM quality checks. *arXiv preprint arXiv:2501.17167*, 2025.

Yunfan Jeong et al. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of NAACL*, 2024.

Shuguang Jiao, Chengkai Huang, Shuhan Qi, Xuan Wang, Yifan Li, Quanchi Weng, Lingchuan Liu, Xunliang Cai, and Lina Yao. Doctor-RAG: A failure-aware repair framework for agentic retrieval-augmented generation, 2026. URL https://arxiv.org/abs/2604.00865.

Mingze Kong, Zikun Qu, Zhongquan Zhou, Pengyu Liang, Xiang Li, Zhiwei Shang, Zhi Hong, Kaiyu Huang, Zhiyong Wang, and Zhongxiang Dai. Workflow-R1: Group sub-sequence policy optimization for multi-turn workflow construction. *arXiv preprint arXiv:2602.01202*, 2026.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. LLM-as-a-Verifier: A general-purpose verification framework. *arXiv preprint arXiv:2607.05391*, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023.

Zahra Moslemi, Keerthi Koneru, Yen-Ting Lee, Sheethal Kumar, and Ramesh Radhakrishnan. POLARIS: Typed planning and governed execution for agentic AI in back-office automation, 2026. URL https://arxiv.org/abs/2601.11816.

Nils Palumbo, Sarthak Choudhary, Jihye Choi, Guy Amir, Prasad Chalasani, and Somesh Jha. Formal policy enforcement for real-world agentic systems, 2026. URL https://arxiv.org/abs/2602.16708.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, Xinyuan Song, Lewei He, and Yang Jingsong. Difficulty-aware agentic orchestration for query-specific multi-agent workflows, 2025. URL https://arxiv.org/abs/2509.11079.

Jiaqing Wu, Tong Wu, Manqing Zhang, Yunwei Dong, and Bo Shen. Runtime execution traces guided automated program repair with multi-agent debate. *arXiv preprint arXiv:2604.02647*, 2026.

Yating Wu, Yuhao Zhang, Sayan Ghosh, Sourya Basu, Anoop Deoras, Jun Huan, and Gaurav Gupta. ContextWeaver: Selective and dependency-structured memory construction for LLM agents, 2026. URL https://arxiv.org/abs/2604.23069.

Dawei Xiang, Kexin Chu, Wenyan Xu, Wenhui Zhang, and Wei Zhang. LLM-as-scheduler: Agentic workflow dynamic scheduling. In *Proceedings of ACL*, 2026.

Jiawei Xu, Arief Koesdwiady, Sisong Bei, Yan Han, Baixiang Huang, Dakuo Wang, Yutong Chen, Zheshen Wang, Peihao Wang, Pan Li, and Ying Ding. Rethinking the value of multi-agent workflow: A strong single agent baseline, 2026. URL https://arxiv.org/abs/2601.12307.

Yandong Yan, Junwei Peng, Shijie Li, Chenxi Li, Yifei Shang, Can Deng, Ruiting Dai, Yongqiang Zhao, Jiaqi Zhu, and Yu Huang. DenoiseFlow: Uncertainty-aware denoising for reliable LLM agentic workflows. *arXiv preprint arXiv:2603.00532*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, 2018.

Ling Yue, Kushal Raj Bhandari, Ching-Yun Ko, Dhaval Patel, Shuxin Lin, Nianjun Zhou, Jianxi Gao, Pin-Yu Chen, and Shaowu Pan. From static templates to dynamic runtime graphs: A survey of workflow optimization for LLM agents. *arXiv preprint arXiv:2603.22386*, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *International Conference on Learning Representations*, 2026.

Runchuan Zhu, Bowen Jiang, Lingrui Mei, Fangkai Yang, Lu Wang, Haoxiang Gao, Fengshuo Bai, Pu Zhao, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. AdaptFlow: Adaptive workflow optimization via meta-learning. In *Findings of EMNLP*, 2025.

A PROTOCOLS AND EXPERIMENTAL IDENTITIES

Table 8: Experimental identities at budget 1800. Each suite preserves its own support pool and selector; within-suite contrasts separate ranking, support, and trigger effects.

Suite	Seeds	Support req.→kept	Selector / changed factor	Purpose	Endpoint	Comparison
Main	13/42/2026	$86 \to 9$	state-conditioned, confidence-penalized value rule	primary quality–computation matrix	80.95 ± 0.70, 325; S13: 80.95, 315	five main controls at matched budget
Ablation	13	$86 \to 9$	remove one mechanism from the primary rule	component contribution	main S13 reference: 80.95, 315	each removal versus complete method
Selector	13/42/2026	$128 \to 11$	explicit LCB, ReAct, prior, UCB, Thompson, utility, or mean	ranking robustness	LCB diagnostic anchor: 80.51 ± 0.61, 370.1	selectors inside this suite only
Support	13/42/2026	$0 \to 0$, $32 \to 1$, $64 \to 3$, $128 \to 11$	explicit LCB fixed	support sparsity	Table 6	support rows inside this suite only
Sensitivity	13/42/2026	$128 \to 11$	explicit LCB; vary tied margins, probes, or steps	local parameter robustness	absolute rows in Table 18	rows within the same factor block
Trigger	13/42/2026	$0 \to 0$	typed versus Boolean trigger; replay fixed	trigger specificity	Figure 7	matched rows within each budget

Table 9: Execution structure of the internal controls and route-trigger diagnostic. Probes and promotion operate on cloned state; only accepted candidates replace live artifacts.

Internal configuration	Execution structure	Enabled control	Effect and experimental role
One-pass (`static_minimal`)	construct → verify → stop	no gate, probe, replay, or promotion	Cheapest floor; cannot recover a failed draft.
Fixed schedule (`static_full`)	acquire → verify → repair → branch/reconcile → verify	fixed actions; no runtime admission	Computation reference; spends deep-path actions whether or not failure evidence requires them.
TRACE-REPAIR (`trace_only`)	construct → verify → bounded repair–verify retries	shared construction, verifier, preconditions, and repair loop; no typed gate, root probe, replay, or clone promotion	Closest internal configuration; efficient when the failed draft is the true root, but it cannot test or repair upstream artifacts.
Length rule (`v2_like`)	choose fixed schedule when prompt+context > 500, otherwise one-pass	pre-execution size rule	Auxiliary router; avoids controller overhead but cannot adapt after observing a failure.
Value routing (`evidenceflow`)	budget-filtered action ranking with one cloned local candidate	replay and promotion; no typed admission	Soft-routing control; can localize work but may admit unnecessary acquisition or branching.
ObliGate (`failure_gated`)	verify → typed obligation → two root probes → optional branch/reconcile	typed admission, replay, promotion	Complete method; tests interventions and commits only a verifier-improving candidate.
Any-failure trigger (`boolean_trigger`)	ObliGate execution, but every failed obligation opens the same route	Boolean route trigger; typed registry, replay, promotion retained	Isolates trigger filtering; does not remove typed action authorization.

A.1 DEFAULT CONFIGURATION

The default main arm is the LCB rule. Selector controls replace only the ranking rule, while the executor, action legality, task manifest, provider, and terminal evaluation remain unchanged. This isolates ranking sensitivity from changes in workflow content.

The main matrix requests 86 support examples and retains nine after overlap filtering. The independent selector and sensitivity suites request 128 and retain 11. Trigger runs use no retained support. All target trajectories terminate before hidden evaluation; neither same-task nor cross-target hidden feedback updates the frozen controller.

Table 10: Adapter evidence and legal transitions. Promotion compares verifier improvement within an adapter; hidden tests and gold answers remain evaluation-only.

Adapter-visible event	Strength	Obligation target and legal next action	Promotion condition
Code public-test assertion or execution error	Deterministic	current draft; compatible repair or one branch–reconcile cycle	public verifier improves by the adapter-mapped commit margin
GSM8K/MATH parse or normalized-equivalence failure	Structured	parsed answer; format correction or task-specific repair	adapter-native parse/equivalence verdict improves; no claim about proof correctness
HotpotQA incomplete evidence bridge	Support	evidence artifact; retrieve or check support, not destructive answer repair	newly acquired context must improve the adapter-local support check
HotpotQA answer–evidence inconsistency with a complete bridge	Structured	draft; evidence-conditioned answer repair	answer-support check improves without consulting gold answer
Provider/evaluator timeout or exception	Infrastructure	no task artifact; bounded infrastructure retry only	never opens or promotes a task-failure edit

Table 11: Default controller configuration and the corresponding sensitivity arm.

Parameter	Default	Sensitivity values	Meaning
Maximum steps	8	4, 8, 12	Episode safety cap.
Promotion/STOP threshold ϵ	0.015	0, 0.015, 0.05	Minimum verifier improvement.
Repair probes	2	1, 2, 3	Candidate roots tested before escalation.
Prior strength κ	3.0	fixed	Shrinkage toward the runtime heuristic.
Uncertainty penalty λ	0.04	fixed	Conservative LCB ranking.
Exploration rate	0.12	fixed	Seeded support-time exploration.
Utility cost weight	0.04	fixed	Soft comparator only; hard gate remains executor-enforced.

Table 12: Task-action provider-token ledger at budget 2400 (three-seed means). The ledger separates construction, evidence control, repair, escalation, and root probes; deterministic verification consumes no provider tokens.

Configuration	Construct	Acquire	Classify	Repair	Branch	Reconcile	Probe	Total
One-Pass	236	0	0	0	0	0	0	236
Trace-Repair	235	0	2	52	0	0	0	289
Value-Routed	237	357	9	0	22	12	54	691
Fixed-Deep	236	358	14	57	20	16	0	701
Obligation-Gated	236	0	14	0	22	17	48	337

B Task-action Ledger and Reservation Audit

At budget 2400, the fixed schedule spends 358 tokens reacquiring evidence and 57 on unconditional repair, whereas ObliGate spends 14 on classification, 22 on branching, 17 on reconciliation, and 48 on probes after a failure opens an obligation. Across the main cells, 129 probe candidates are promoted, 191 are rejected by the improvement test, and 210 are infeasible; hard action-precondition violations are zero. An evaluator-free reservation audit records the estimation boundary: evidence acquisition underestimates by 6.9 tokens on average (46.7% of paired events) and reconciliation by 127.0 tokens (89.7%), while construction and direct repair are conservatively overestimated by 364.0 and 234.5 tokens. These deltas explain why the primary matrix reports reservation violations rather than hiding them.

Table 13: Current endpoint contrasts derived from Table 1. Differences are ObliGate minus the named internal reference; token reduction is relative to Fixed-Deep.

Budget	Reference	Score Δ (points)	ObliGate tokens	Reduction vs fixed
1200	Fixed-Deep	+2.21	282	52.7%
1800	Fixed-Deep	+0.79	325	50.5%
2400	Fixed-Deep	+0.35	337	51.9%
1200	Trace-Repair	+1.33	282	—
1800	Trace-Repair	+0.97	325	—
2400	Trace-Repair	+1.41	337	—

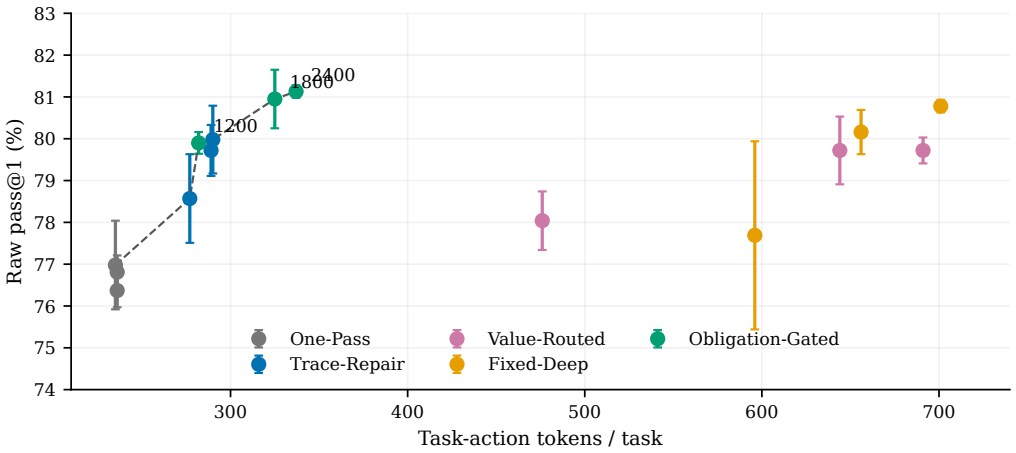


Figure 4: Quality–task-action-token operating points for all 15 cells of the current strict MBPP+ matrix. Error bars show between-seed sample SD; the dashed line joins non-dominated mean operating points.

C Controlled Fault Coverage and Executed Recovery

Draft-local faults are repaired by Trace-Repair at 100% and 1,467.2 tokens. Its upstream, verifier-root, and compound rates are zero; ObliGate reaches 50.0%, 100.0%, and 66.7%. Without escalation, verifier-root and compound rates are 0% and 11.5%. Fixed-Deep and Value-Routed reach 100% on the recoverable actionable pool. All configurations pass the 288 clean controls and fail the 288 unrecoverable controls. On clean cases, ObliGate uses 891.7 tokens versus 693.6 for Trace-Repair. These controls distinguish recovery scope, clean execution, and failure termination. The executed 20-record comparison uses 756.80 replay-stage tokens (1,465.45 end-to-end) versus 1,715.75 full-rerun tokens, 3.90 versus 4.80 calls, and 100% versus 90% repair. The replay-stage ratio-of-means saving is 55.9%; mean record-wise saving is 58.6%.

Table 14: Controlled latent-root fault coverage over 1,344 actionable cases per configuration, with three seeds. Unnecessary is the share of correction actions missing the injected root. Root labels are hidden from the controller.

Internal configuration	Cases	Success (%)	Tokens/T	Tokens/success	Calls/T	Unnecessary (%)
One-pass	1,344	2.7	**692.0**	25,835.4	**2.00**	0.0
TRACE-REPAIR	1,344	28.1	3,237.0	11,539.8	10.42	40.8
No local replay	1,344	61.9	3,369.2	5,442.5	9.48	53.4
Any-failure replay	1,344	**76.9**	3,286.7	**4,276.2**	11.62	53.0
No branch escalation	1,344	41.7	2,360.8	5,655.9	9.05	**31.6**
ObliGate	1,344	**76.9**	3,286.7	**4,276.2**	11.62	53.0

The complete route exceeds TRACE-REPAIR by 48.8 points over the actionable pool (task-paired bootstrap 95% CI: 46.3–51.5; exact McNemar $p < 10^{-190}$), with a 49.7-token mean difference whose 95% interval crosses zero (-11.3 to 110.2). The matched trigger and natural-failure studies complement this controlled coverage result by measuring routing selectivity and end-to-end recovery. Family-level results and both negative controls are reported in the artifact.

Table 15: Known-root recovery by seed. Root-set precision, recall, exact-set match, and repair are percentages. Saving uses the 2,090-token reservation reference.

Seed	Records	Prec.	Recall	Exact set	Repair	Tokens / saving
13	84	74.40	94.05	54.76	100.0	810 / 61.26
42	88	73.30	94.32	52.27	100.0	777 / 62.82
2026	79	72.78	94.94	50.63	100.0	834 / 60.08
Mean $\pm$ SD	N/A	73.50 ± 0.83	94.43 ± 0.46	52.56 ± 2.08	100.0 ± 0.0	807 ± 29 / 61.39 ± 1.38

D ROUTE TRIGGERS, SUPPORT, AND SENSITIVITY

Table 16: Matched route-trigger control on strict MBPP+ (378 tasks; seeds 13/42/2026). Both internal configurations retain typed obligations, legality, probing, promotion, and stopping; only the trigger predicate changes. Scores are raw *pass*@1; tokens are primary-path task-action tokens per task.

Budget	Route trigger	S13	S42	S2026	Mean $\pm$ SD	Tokens
1200	Kind/strength	82.54	82.28	82.80	82.54 ± 0.26	251.8
	Any failed	82.28	82.54	82.28	82.36 ± 0.15	246.5
1800	Kind/strength	82.28	82.54	82.28	82.36 ± 0.15	250.5
	Any failed	83.33	83.07	83.33	83.24 ± 0.15	248.1
2400	Kind/strength	83.60	83.33	83.60	83.51 ± 0.15	250.7
	Any failed	82.54	82.80	82.54	82.63 ± 0.15	248.5

Filtered-minus-any-failure: $+0.18/-0.88/+0.88$ points at budgets 1200/1800/2400; filtered routing adds 5.3/2.4/2.2 task-action tokens, respectively.

Table 17: Support-size robustness at budget 1800 (three seeds per row).

Support	LCB (%)	Utility (%)	LCB tok.
0	79.45 ± 1.22	75.66 ± 0.53	396.9
32	80.95 ± 0.46	76.37 ± 0.61	397.5
64	80.25 ± 0.55	76.19 ± 0.46	387.9
128	80.51 ± 0.61	75.04 ± 3.27	370.1

The four-step sensitivity arm is an independent run with $80.95 \pm 0.70\%$ raw score; its eight-step reference is $80.16 \pm 0.53\%$. Their mean difference is 0.79 points. Promotion-margin settings span 0.27 points; the two-probe setting has the highest measured mean in its block. These local contrasts keep the identity of each independently generated block intact.

Table 18: Independent one-factor controller sensitivity at budget 1800 (three seeds). Scores and tokens are mean ± sample SD.

Factor	Setting	Raw *pass*@1 (%)	Tokens	p95 (s)
Promotion ϵ	0	80.16 ± 0.70	382.0 ± 7.7	5.51
	0.015 (default)	80.34 ± 0.55	376.8 ± 11.4	5.52
	0.05	80.07 ± 0.55	376.3 ± 1.9	5.70
Repair probes	1	79.98 ± 0.31	379.9 ± 3.8	5.69
	2 (default)	80.69 ± 0.26	374.7 ± 3.8	6.12
	3	80.07 ± 0.81	422.8 ± 6.4	7.25
Maximum steps	4	80.95 ± 0.70	373.7 ± 6.2	5.74
	8 (default)	80.16 ± 0.53	379.5 ± 3.4	5.42
	12	80.51 ± 1.00	374.1 ± 6.2	5.30

E Boundary and Threshold Checks

Table 19: Threshold rerun on disjoint synthetic support/evaluation tasks (three seeds; $n = 16$ evaluation tasks per seed and domain). Scores and end-to-end tokens are means over seeds with sample SD. The suite is a boundary diagnostic and is not pooled with the strict MBPP+ endpoint.

Domain	ϵ	Score mean ± SD	E2E tokens mean ± SD	Violation mean ± SD
Code	0.000	0.4167 ± 0.0361	1390.7 ± 47.6	0.0000 ± 0.0000
	0.015	0.4167 ± 0.0361	1390.7 ± 47.6	0.0000 ± 0.0000
	0.050	0.4167 ± 0.0361	1390.7 ± 47.6	0.0000 ± 0.0000
Math	0.000	0.3542 ± 0.0955	1227.2 ± 41.5	0.0000 ± 0.0000
	0.015	0.3542 ± 0.0955	1193.3 ± 74.9	0.0000 ± 0.0000
	0.050	0.3542 ± 0.0955	1193.3 ± 74.9	0.0000 ± 0.0000
QA	0.000	0.4583 ± 0.0955	1357.9 ± 72.9	0.0000 ± 0.0000
	0.015	0.4583 ± 0.0955	1357.9 ± 72.9	0.0000 ± 0.0000
	0.050	0.4583 ± 0.0955	1357.9 ± 72.9	0.0000 ± 0.0000

The threshold suite uses 16 evaluation tasks per seed and domain, disjoint from support, under the timestamped provider-free rerun. The contract audit rejects registered parallel write conflicts and records unknown parents for inspection. The compound-fault rerun gives the complete and fixed controllers score 1.0 in all three domains, versus 0.0 for feedback-only repair; code and math use one branch–reconcile cycle, while QA resolves without escalation. These interventions exercise the declared replay and escalation contract.

F Published Workflow Context

Table 20: Literature-reported workflow results under the original models and evaluation protocols, plus the local strict MBPP+ QualityFlow bridge. N/R means not reported.

Work	Model / setting	Code	Math	QA	Tok./T	Source
Published original protocols						
AFlow (Zhang et al., 2025)	GPT-4o-mini	94.70 / 83.40 / N/R	93.5 / 56.2	73.5	N/R	paper
AdaptFlow (Zhu et al., 2025)	GPT-4o-mini / GPT-4.1	94.70 / 84.00 / N/R	94.6 / 61.5	73.8	N/R	paper
OneFlow (Xu et al., 2026)	GPT-4o-mini	92.10 / 81.40 / N/R	93.3 / 54.1	73.5	N/R	paper
QualityFlow (Hu et al., 2025)	Claude Sonnet-3.5-v2	97.56 / 94.20 / 79.89	N/R	N/R	N/R	paper
FlowReasoner-14B (Gao et al., 2025)	o1-mini worker	97.26 / 92.15 / N/R	N/R	N/R	N/R	paper
LLM-as-Scheduler (Xiang et al., 2026)	Doubao scheduler	94.50 / 93.70 / N/R	95.1 / N/R	N/R	N/R	paper
Additional MBPP+ context						
DeepSeek-Coder-V2-Instruct	reported setting	N/R / 89.40 / 76.20	N/R	N/R	N/R	paper (diff. model)
QualityFlow local bridge	DeepSeek-V4-Pro, seed 13	N/R / N/R / 79.63	N/R	N/R	558	reproduction

Local comparisons in the main text use DeepSeek-V4-Pro and explicitly named internal configurations. Published values above preserve the original model and evaluator. The local QualityFlow strict bridge uses 378 targets, with 301 passes (79.63%), 211,039 total tokens, and 778 calls in seed 13.

G NATURAL-FAILURE PROTOCOL AND ADDITIONAL PLOTS

Shared cases and accounting. The corpus contains 228 runtime-failure checkpoints and 1,368 records, one per case and method. Continuation arms restore identical initial answer digests and artifact checkpoints; full rerun starts the same task afresh. The saved checkpoint has a nominal 2,400-token/eight-call budget, with corrective execution exempt from that ceiling. Direct draft repair permits up to three repair attempts; replay arms permit up to three root candidates, each with up to three repair–verify attempts, followed by the configured escalation path. The comparison measures complete policies with shared operators, not equal numbers of proposals. E2E tokens include the saved prefix and recorded correction trace for continuation, and the fresh trajectory for full rerun.

Outcome definitions and strata. For initial score s_0, final score s_1, and answer digest h, the logged improvement indicator is

$$I_{\text{improve}} = \mathbf{1}[s_0 < 0.999999 \ \wedge \ s_1 > s_0 + 10^{-9}],$$

excluding infrastructure failures from the success indicator but retaining every case in the reported denominator. False repair additionally requires a changed answer and no improvement; it is distinct from an online acceptance event. Abstention means the answer digest is unchanged. The cohort contains 77 initially correct answers flagged by runtime checks. Of ObliGate's 76 improvements, 74 reach unit terminal score and two are partial improvements; these metrics evaluate answers, not a complete multi-artifact acceptance test. Strata describe failure symptoms: all 22 parser cases carry draft-root annotations, whereas retrieval/coverage and compound identify 37 upstream-involving cases. Offline annotations do not guide runtime root selection. One full-draft record has an infrastructure error and remains in its stratum denominator.

Table 21: Complete natural-failure decomposition. Entries are improvement count / false-repair count / mean recorded E2E tokens. Stratum denominators are fixed across all six methods.

Method	Draft-local ($n = 169$)	Parser ($n = 22$)	Retrieval ($n = 34$)	Compound ($n = 3$)
Full rerun	41 / 35 / 1,815	3 / 10 / 2,169	3 / 1 / 2,255	0 / 0 / 2,267
Draft repair	70 / 13 / 4,332	2 / 6 / 5,804	0 / 0 / 12,675	0 / 0 / 12,551
Replay full draft	73 / 19 / 5,815	1 / 3 / 7,144	0 / 2 / 17,671	0 / 0 / 17,716
Replay closed draft	74 / 19 / 5,725	2 / 3 / 8,533	0 / 2 / 17,671	0 / 0 / 17,716
Replay full probe	77 / 17 / 18,155	0 / 0 / 22,832	0 / 2 / 57,737	0 / 0 / 54,325
ObliGate	76 / 17 / 15,370	0 / 0 / 19,748	0 / 2 / 38,306	0 / 0 / 40,933

Paired evidence. ObliGate versus direct draft repair yields 68 cases with both improving, eight with only ObliGate improving, four with only draft repair improving, and 148 with neither improving. Against closed replay from the draft root, the counts are 72/4/4/148. The paired differences are $+1.75$ points (95% interval $[-1.32, 4.82]$) and 0.00 points ($[-2.20, 2.63]$), respectively. Equal aggregate rates need not represent identical repaired cases. These intervals and the token-difference interval use 10,000 paired case-bootstrap draws, seed 20260919, and linearly interpolated percentile endpoints. They estimate complete-policy differences rather than attributing every improvement to typed filtering.

The full-probe minus closed-probe token difference is $24{,}985.1 - 19{,}549.2 = 5{,}435.9$; the full-draft minus closed-draft difference is $7{,}867.7 - 7{,}934.9 = -67.2$. Their average is the reported 2,684-token replay effect. The probe-enabled cost reduction is 21.8%; this is a conditional replay-scope contrast, not evidence that probing improves natural recovery. Candidate generation is independent across arms, so this is a policy-level scope contrast. The complete arm has 0.167 unnecessary recomputation and the closed-draft arm 0.000; artifact ratios and actual tokens are reported separately.

Table 22: Natural failure strata. Cells give score-improving repairs / false-repair count / mean recorded E2E tokens; denominators are fixed across methods. Parser is a symptom stratum with draft-root annotations; retrieval/coverage and compound comprise the 37 upstream-involving cases.

Method	Draft-local ($n = 169$)	Parser ($n = 22$)	Retrieval ($n = 34$)	Compound ($n = 3$)
Full rerun	41/35/1,815	3/10/2,169	3/1/2,255	0/0/2,267
Draft repair	70/13/4,332	2/6/5,804	0/0/12,675	0/0/12,551
Replay full probe	77/17/18,155	0/0/22,832	0/2/57,737	0/0/54,325
ObliGate	76/17/15,370	0/0/19,748	0/2/38,306	0/0/40,933

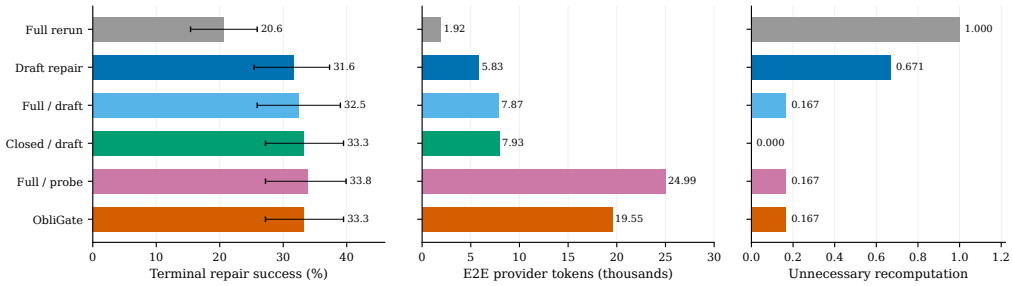


Figure 5: Expanded natural-failure outcomes on the same 228 source cases. Continuation arms restore the saved checkpoint; full rerun restarts. Repair denotes offline score improvement; error bars are case-bootstrap 95% intervals. The second panel includes recorded prefix and probe tokens.

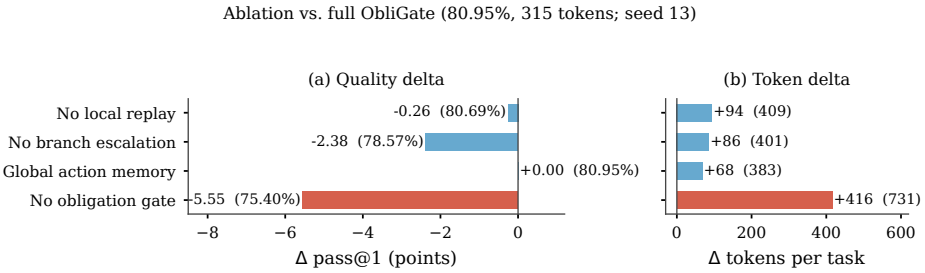


Figure 6: Component deltas from the seed-13 complete-method row (80.95%, 315 tokens).

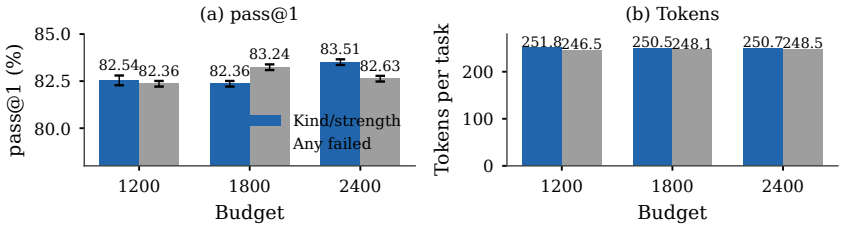


Figure 7: The matched kind/strength versus any-failure route-trigger diagnostic.

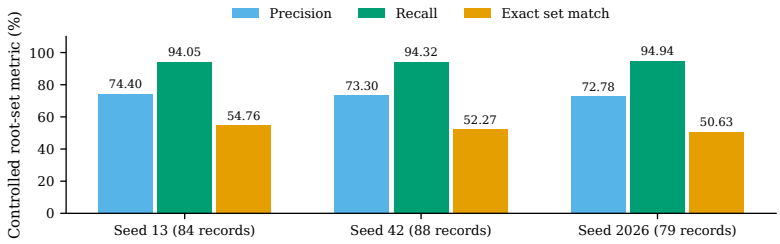


Figure 8: Known-root precision, recall, and exact-set-match diagnostics across the controlled-recovery seeds.

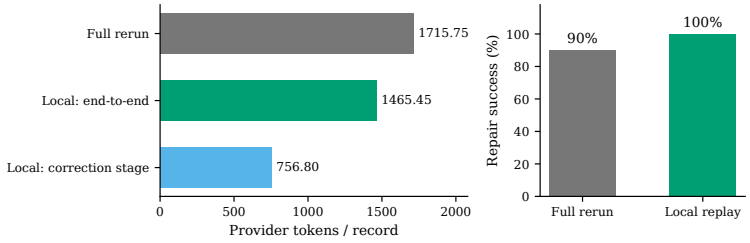


Figure 9: Executed 20-record recovery comparison. The cost panel distinguishes the complete local pipeline from its correction stage; repair rates compare the two executed strategies.

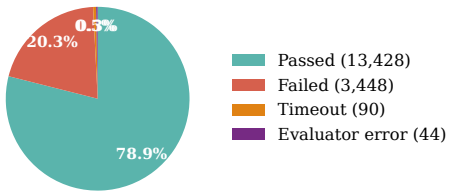


Figure 10: Archived evaluator-status audit. Infrastructure statuses are recorded separately from runtime verifier failures.

H Evaluator Audit and Trace Cases

Table 23: Archived hidden-evaluator audit over 15 configurations per seed, retained as a separate evaluator-status check. Evaluator errors and timeouts are infrastructure outcomes and never open a runtime repair obligation.

Seed	Passed	Failed	Timeout	Evaluator error
13	4,461	1,136	29	44
42	4,490	1,150	30	0
2026	4,477	1,162	31	0
Total	13,428	3,448	90	44

Raw scores count all terminal statuses; resolved scores remove only exhausted evaluator timeout/error records. These statuses never become runtime obligations. A successful repair (task 135, seed 13, budget 1200) follows construct, verify, repair, verify, stop. A rejected repair (task 101) preserves the checkpoint and escalates through branch and reconcile. An overrun (task 109) stops after the realized provider delta exceeds the remaining estimate. Serialized traces average 1.33–1.35 KB per task and 2.28–2.50 KB at p95, covering versions, parents, digests, obligations, and resource deltas.

I Runtime Algorithm and Action Contract

```
Input: RuntimeTask x, budget B, controller pi, verifier V
Initialize EvidenceState s0:
    empty artifact graph and obligations
while not stopped:
    legal <- ActionRegistry.legal_actions(s)
    a <- pi.select(s, legal)
    if a == STOP:
        export terminal artifact; break
    result <- Executor.execute(a, s, B)
    update artifact DAG, trace, resource deltas, obligations
    if result is verifier failure:
        create typed failure obligation
    if repair rejected and branch is legal:
        enable branch -> reconcile -> verify
After all target workflows terminate:
    run hidden final evaluation.
Never return hidden result to pi.
```

Action fields. Each registered action stores the following fields in the implementation: a name and action kind; artifact read and write sets; required artifact names; required obligation kinds; estimated token, call, latency, and cost usage; cacheability; replayability; and a parallelizable flag. Obligations also store evidence strength (deterministic, structured, or support). The legality check is evaluated before provider allocation. In particular, `REPAIR` and `BRANCH` require a failed deterministic or structured runtime-visible obligation, while `RECONCILE` requires both candidate artifacts. This contract is what allows the controller's empirical value estimate to remain subordinate to hard preconditions.

J Execution Value and Probe Cost Conditions

Fixed-edit replay proposition. Fix a checkpoint, root set R, and replacement contents before choosing the replay scope. Let $A = R \cup D(R; G)$ and let U be the nodes refreshed by a complete replay but outside A. Assume the dependency-consistency conditions in Section 3, valid artifacts outside A, deterministic replay under fixed external inputs, and the same final acceptance rule. Closed and complete replay then produce the same semantic contents for every required output and the same acceptance result, although refreshed version identifiers may differ.

Proof. A node outside A has neither a changed root nor a changed ancestor. Re-executing it under the fixed inputs reproduces its prior contents. In topological order, each node in A consequently receives the same replacement or parent contents in both executions and produces the same output. The common verifier sees identical required contents. Both plans record their actual parent versions, so each resulting state is internally version-consistent.

If node executions have additive costs $c_v \geq 0$ under these identical inputs, and executor overheads are h_{full} and h_{closed}, then

$$C_{\text{full}} - C_{\text{closed}} = \sum_{v \in U} c_v - (h_{\text{closed}} - h_{\text{full}}).$$

Closed replay is strictly cheaper when preserved computation exceeds its extra bookkeeping: a larger closure leaves less reusable work. The different natural policy costs are consistent with this mechanism, but do not instantiate the fixed-edit assumptions because their repairs are generated independently.

When an additional probe is useful. For terminal quality Q, recorded cost C, and deployment weight $\lambda > 0$, an extra probe improves expected utility precisely when

$$\mathbb{E}[Q_{\text{probe}} - Q_{\text{direct}}] > \lambda \mathbb{E}[C_{\text{probe}} - C_{\text{direct}}].$$

This break-even condition separates the value of a new root hypothesis from the cost of testing it; it is an interpretation of the operating points, not an additional fitted controller. A direct-first order and stopping after an accepted improvement limit spending on easy repairs. Wider root exploration is useful only when expected verifier improvement exceeds its cost; the natural corpus does not meet that condition reliably. The observed contrasts motivate adaptive probing as a conditional policy choice, while closure replay reduces the propagation cost of a chosen intervention.

K Reproducibility and Claim Map

Freeze the provider identifier, manifest digest, prompts, target order, concurrency, and seed for each cell. Keep support and targets disjoint; retain candidates and evaluator statuses. Record nominal, correction, and infrastructure usage separately. The primary matrix measures quality beyond conditional feedback; component removals measure replay and escalation within the configured policy; trigger tests vary filtering while retaining typed legality. Controlled faults establish cross-artifact intervention coverage; natural comparisons establish observed replay-scope costs and recovery operating points. The fixed-edit proposition establishes sufficient conditions for execution equivalence; the hard-reserve audit concerns its accepted-action ledger. Cross-domain rows retain native metrics and named internal policies.