# GRAPHADMIT: EVIDENCE-CONDITIONED MUTATION ADMISSION FOR REPOSITORY AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A validator that discovers a broken import after execution arrives one state transition too late: files, dependency state, or an external system may already have changed. GRAPHADMIT turns that transition into an evidence-conditioned admission decision. For each structurally risky action, it materializes a dry-run graph and complete diff, intersects the requested scope with a host-frozen authorization envelope, and evaluates a model-independent universe of repository requirements, invariants, and refutations. Resolved evidence activates bounded sandbox probes; weaker evidence contracts the checked predicate set while unresolved obligations remain in the support denominator. Rejections return entity-level causes for bounded replanning, making GRAPHADMIT a complete execution policy rather than a proposal-only filter. Across eight paired seeds on a fixed 300-task SWE-bench Verified manifest (2,400 runs), the policy raises resolution from 29.4% to 31.2% while reducing harmful-edit tasks from 13.6% to 11.3%, high-severity harm from 4.0% to 2.8%, and rollbacks from 1.6 to 1.1 per task relative to a risk-matched scope-only policy. The paired effects exclude zero: $+1.8$ resolution points (95% CI $[1.1, 2.5]$), $-2.3$ harmful-edit points ($[-3.2, -1.4]$), $-1.2$ severe-harm points ($[-1.7, -0.7]$), and $-0.5$ rollbacks ($[-0.7, -0.3]$). On strong-support tasks, resolution improves by 3.2 points as harmful edits fall by 2.6 points. Common-threshold sweeps, four repair backbones, a 300-task nine-language transport manifest, and blinded adjudication preserve the severe-harm reduction. These policy-level results establish evidence-conditioned admission with structured feedback as a practical control point that redirects repository agents before structural damage without sacrificing task completion.

## 1 INTRODUCTION

A three-file rename can pass through an agent as one tool call yet leave a fourth caller stale, rewrite a lockfile, or execute a command outside the intended repair. If a later test catches the failure, the repository transition has already occurred. Modern repository agents create exactly this exposure because they combine localization, cross-file editing, package management, and command execution in long trajectories (Jimenez et al., 2024; Pan et al., 2025; Yang et al., 2024; Xia et al., 2025; Chen et al., 2024a; Zhang et al., 2024; Wang et al., 2024). Patch scoring measures the final repair; it does not decide whether a particular state-changing action should be allowed to alter the working state.

GRAPHADMIT decides at this boundary. The host freezes an authorization envelope before mutation, the checker represents the live repository as a typed graph, and a dry run exposes the complete materialized change set. A deterministic miner constructs candidate requirements, invariants, and refutations independently of the agent's certificate. Evidence coverage selects a full sandbox probe set, a partial graph-and-metadata check, or scope containment. Unsupported obligations remain visible in the denominator, so omission cannot promote an action to a stronger mode. A rejection identifies the failing entities and gives the agent a bounded opportunity to replan.

Generation grammars constrain the form of a proposal, prerequisite collectors delay editing until observations arrive, and post-hoc validators inspect a candidate or failure after it exists. The baselines evaluated here do not jointly make the materialized repository transition—including generated files, dependency state, and host authorization—the checked object under explicitly incomplete evidence.



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.

GRAPHADMIT supplies that admission object and a mode semantics that remains defined when only part of the repository evidence resolves.

The evaluation measures the complete execution policy because admission feedback changes later proposals. Every policy begins from the same checkout, prompt stack, risk trigger, model-call cap, and seed; the primary estimand is the paired task-level outcome difference, not a proposal-level replay after trajectories diverge. A component staircase compares scope-only, scope plus refutations, scope plus repository requirements, and full evidence-conditioned admission, while a component diagnostic measures sensitivity to post-action observations. This design tests whether repository evidence at the mutation boundary improves both safety and completion.

*Admission property.* For a protected mutation, GRAPHADMIT admits execution only when the materialized dry-run change is contained in both the host-authorized envelope and the agent's requested subset, and every check selected by the observed-support mode passes on its designated pre- or post-mutation state. The evaluated actions are scope-expanding edits, dependency mutations, edits spanning at least three files, and externally side-effecting commands.

At the default operating point, GRAPHADMIT improves resolution by 1.8 points over scope-only while reducing harmful-edit tasks by 2.3 points, high-severity harm by 1.2 points, and rollbacks by 0.5 per task; every paired interval excludes zero. The joint gain reaches $+3.2$ resolution and $-2.6$ harmful-edit points on the pre-policy strong-support stratum. Shared-threshold sweeps, four backbone configurations, a nine-language transport manifest, and independent labels test whether the safety result depends on one operating point, model, language family, or harness label.

This work contributes:

- A repository-specific admission object: the complete dry-run mutation checked against independently authorized scope and a fixed obligation universe with explicit current-state and dry-run-state predicate semantics.
- An evidence-conditioned policy that retains unresolved obligations, selects probe sets by support, and converts failed checks into entity-level replanning feedback without changing the execution budget.
- A paired policy staircase over 2,400 primary runs, component and support analyses, four repair backbones, shared operating-point sweeps, nine-language transport, and blinded labels; together they show lower severe harm and rollback cost with preserved or improved resolution.

## 2 PROBLEM FORMULATION

We model repository state at time $t$ as a typed directed graph $\mathcal{G}_t = (V_t, E_t, \tau)$ over files, symbols, tests, build targets, packages, and permission scopes; edges encode imports, calls, coverage, build membership, and dependency relations. Supported action types are `edit_file`, `create_file`, `delete_file`, `run_command`, `install_dep`, and `refactor`. Before mutation begins, the harness freezes a localized region from the task statement and localization phase, then intersects it with the host permission policy to obtain $\mathcal{S}^{\mathrm{host}}$. Opening or naming a file after this freeze does not expand authorization. The agent may request a narrower $\mathcal{S}^{\mathrm{ask}}$, but cannot enlarge $\mathcal{S}^{\mathrm{host}}$. A proposal is structurally high-risk when its dry-run change leaves the frozen region, mutates dependency state, affects at least three files, or invokes an externally side-effecting command.

The failure model centers on accidental and underspecified agent behavior. The agent may read all visible repository content and propose any supported mutation; trusted inputs are the task statement, current checkout, frozen localizer output, parser output, dependency metadata, host permission policy, and bounded read-only probes. The enforcement boundary is the atomic transition from the current checkout to the proposed mutated state. The protected property is structural admissibility relative to this evidence: containment of the materialized dry-run change set, satisfaction of available repository requirements, preservation of checked invariants, and absence of a triggered refutation. Semantic repair correctness is measured independently at the task endpoint.

The harness first materializes an isolated preview $\Delta_t(a_t)$ without changing the working state. Let $h(a_t, \Delta_t) = 1$ when the action leaves the frozen region, mutates dependency state, affects at least

three files, or invokes an externally side-effecting command. The learned risk score is

$$\rho(a_t, \mathcal{G}_t, \Delta_t) = \sigma\big(w^\top \phi(a_t, \mathcal{G}_t, \Delta_t)\big) .$$

An action is gated iff $h(a_t, \Delta_t) = 1$ or $\rho(a_t, \mathcal{G}_t, \Delta_t) \geq \theta$. Here $\phi$ contains requested-versus-realized scope distance, dependency mutation, touched-file count, command side-effect class, permission sensitivity, graph fan-out, and probe availability. Logistic weights are fit once from the operational harmful-action labels defined in Appendix A.2 on repository-disjoint calibration snapshots, and both $w$ and the default threshold are frozen before evaluation. Thus every protected class enters admission independently of learned-score recall, while $\rho$ identifies additional suspicious proposals. The local operating points use $\theta \in \{0.45, 0.55, 0.65\}$, and the global frontier adds $0.35$ and $0.75$. All main results use $0.55$, and no evaluation task or label participates in calibration.

For each gated action $a_t$, a deterministic miner constructs a candidate universe $\mathcal{O}_t$ from the task and frozen checkout. Public interfaces emit `exists`, `type_is`, and compile-preservation candidates; import and call edges emit reachability candidates; direct coverage emits selected-test candidates; dependency metadata emits satisfiability candidates; and checkout-truncated history emits refutations for repeated structural breakage. Candidates are canonicalized by predicate and entity identity before any certificate is read. The agent emits $c_t = (\texttt{req}, \texttt{scope}, \widehat{o}_t, \texttt{inv})$, where $\widehat{o}_t$ is serialized as `exp` and states expected post-action outcomes. It binds candidates to graph entities but cannot delete candidates from $\mathcal{O}_t$. The checker materializes a dry-run graph $\mathcal{G}_t^a$ and complete diff $\Delta_t(a_t)$, including generated files. For mode $m$, acceptance is

$$\begin{aligned} \mathcal{V}_m(\mathcal{G}_t, \mathcal{G}_t^a, a_t, c_t) &= \mathbf{1}[C_m], \\ C_m \equiv \mathcal{S}^{\text{host}} \supseteq \mathcal{S}^{\text{ask}} &\supseteq \text{scope}(\Delta_t) \\ &\wedge \bigwedge_{p \in \mathcal{P}_m^{\text{pre}}} p(\mathcal{G}_t) \wedge \bigwedge_{p \in \mathcal{P}_m^{\text{post}}} p(\mathcal{G}_t^a) \\ &\wedge \bigwedge_{r \in \mathcal{R}_m} \neg r(\mathcal{G}_t, \mathcal{G}_t^a). \end{aligned}$$

The predicate vocabulary is `exists`, `imports`, `tests_cover`, `type_is`, `compiles`, `test_passes`, `dep_satisfies`, and `perm_held`. Existence, authorization, and source-edge predicates in $\mathcal{P}_m^{\text{pre}}$ read $\mathcal{G}_t$; compile, dependency, selected-test, import, and type-preservation predicates in $\mathcal{P}_m^{\text{post}}$ read $\mathcal{G}_t^a$ inside an isolated bounded sandbox. After an admitted action executes, the observed outcome $o_t$ may support later replanning but never determines admission of that action; $\widehat{o}_t$ is only its pre-action expectation. Candidates are normalized by predicate and entity identity, and linked by exact symbol/path identity followed by unique graph resolution. Contradictory, post-checkout, or ambiguous signals are marked unsupported.

Let $q_t$ be the fraction of the fixed, deduplicated $\mathcal{O}_t$ whose operands resolve uniquely and whose evidence is current. Strong support ($q_t > 0.8$) selects FULL: scope plus all resolved graph predicates, sandbox compile/dependency/type/selected-test probes, and applicable refutations. Weak support ($0.4 \leq q_t \leq 0.8$) selects PARTIAL: scope plus all resolved graph and metadata predicates and only sandbox probes whose complete target set is available. Missing support ($q_t < 0.4$) selects SCOPE: host/requested-scope containment and permission checks. These boundaries encode probe completeness rather than an outcome-tuned confidence score and are frozen before evaluation. Unresolved candidates stay in the denominator and cannot be omitted by the certificate; every mode, candidate status, probe result, and decision is recorded. Acceptance establishes the predicates evaluated in that mode on their stated current or dry-run state.

For the task-level heterogeneity analysis, we construct a proposal-independent universe $\mathcal{O}_i^0$ from task $i$'s statement and base-checkout interfaces, imports, direct test coverage, dependency metadata, and checkout-truncated history. Applying the same resolver to $\mathcal{O}_i^0$ defines $\bar{q}_i$ before any rollout; no proposed action, dry run, or policy trajectory enters this quantity. The thresholds above partition the 300 tasks into 124 strong-, 96 weak-, and 80 missing-support tasks. Outcome comparisons within a stratum therefore use a label fixed from the common checkout.

## 3 OBLIGATION-AWARE MUTATION ADMISSION

GRAPHADMIT factorizes mutation control into a risk gate, support classification, pre-execution checks, post-execution observation, and structured replanning feedback. Scope-only retains the com-

mon risk trigger, host authorization, and materialized-diff containment; Scope+`inv` adds refutations; Scope+`req`+`inv` adds repository obligations; and full GRAPHADMIT adds support-conditioned probes and $o_t$-conditioned replanning. These are complete episode policies, not interchangeable action classifiers. They share initial task state and budget, while their trajectories may diverge after an admission decision; accordingly, "full minus scope" denotes the effect of the complete admission-and-feedback policy.

The checker combines graph queries, dry-run scope simulation, affected-test selection from coverage metadata, and bounded compile, dependency, permission, or type probes. A failed check returns one of six structured codes—scope violation, missing dependency, broken import, test regression, permission error, or type mismatch—together with the failing graph entities. The agent may replan within its unchanged call budget. The reported 0.82s median is the deterministic graph-and-predicate evaluation after dry-run materialization and certificate construction. For each task run, controller latency is 100 times total controller wall-clock seconds divided by the 1,800-second task cap, averaged across runs; it additionally includes dry runs, certificate generation, sandbox startup and probes, and rejected-action replanning.

**Algorithm 1** GRAPHADMIT support-aware admission

**Require:** Repository graph $\mathcal{G}_t$, proposed action $a_t$, threshold $\theta$
Materialize $(\mathcal{G}_t^a, \Delta_t) \leftarrow \mathrm{dryrun}(a_t)$
Compute $h(a_t, \Delta_t)$ and $\rho(a_t, \mathcal{G}_t, \Delta_t)$
**if** $h(a_t, \Delta_t) = 0$ and $\rho(a_t, \mathcal{G}_t, \Delta_t) < \theta$ **then**
    Execute $a_t$ and return
**end if**
Build fixed candidate universe $\mathcal{O}_t$; bind certificate $c_t$
Resolve every candidate and compute support $q_t$
Select $m \in \{\textsc{Full}, \textsc{Partial}, \textsc{Scope}\}$ from $q_t$
Evaluate host/requested scope and mode-specific pre/post predicates
Record $m$, unsupported fields, probes, and structured outcomes
**if** every check required by $m$ passes **then**
    Execute $a_t$ and record observed outcome $o_t$
**else**
    Reject $a_t$ and return structured failure reasons
**end if**

Obligation acquisition is deterministic before certificate binding. Mining uses repository-disjoint training material plus checkout-visible tests, type interfaces, import edges, dependency files, and history truncated at the task base commit. Hidden tests, gold patches, target repair commits, and later history are excluded. Template instantiation requires an exact path or symbol key; unique graph resolution may then complete the link. Templates are canonicalized by predicate and entity identity, deduplicated within repository, and retained in $\mathcal{O}_t$ when linkage is ambiguous; ambiguity and contradiction lower support rather than deleting a denominator item. Tests provide 42.1% initial high-risk-action support; types, imports, and history raise cumulative support to 56.8%, 67.4%, and 73.2%.

The prompted-schema configurations use constrained JSON generation over the repository-graph summary, task requirements, candidate action, dry-run summary, and mined obligations. The frozen DeepSeek configuration instead uses a lightweight certificate and failure-type head over the same serialized fields; this head is trained on 15K procedurally constructed tool-interaction worlds with exact action labels followed by 8.2K repository-disjoint snapshots. The cross-backbone panel matches every configuration to its own scope-only controller while holding the checker interface, risk trigger, support policy, and execution budget fixed.

Consider a cross-file rename from `service.fetch_data` to `service.fetch_records`. Before mutation, the host envelope contains `client.py`, `service.py`, and `tests/test_service.py`; the candidate universe includes the current import edge, callers, directly covering tests, and compile preservation. If the materialized dry run also touches `cli.py`, the checker rejects with `scope_escape(cli.py)` regardless of what the agent requested. If coverage metadata is missing but the import graph is current, the action receives partial graph checks rather than being promoted to full support.

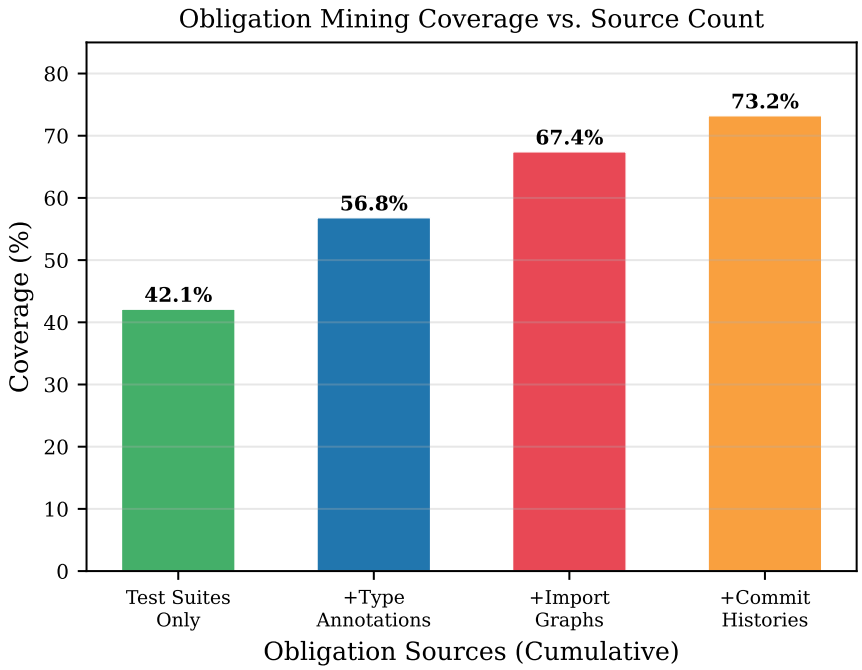



Figure 1: Cumulative coverage of high-risk-action obligations with additional mining sources drawn from outside the evaluation repositories.

Figure 1 shows why one checker mode is insufficient. Tests support 42.1% of high-risk actions; types and imports raise coverage to 56.8% and 67.4%; checkout-truncated history reaches 73.2%. The remaining unsupported candidates stay explicit and select weaker probe sets.

## 4 Evaluation

The primary benchmark is a fixed 300-task SWE-bench Verified manifest spanning all 12 Python repositories. Starting from the official 500 tasks, a pre-policy initialization gate requires checkout, environment creation, test discovery, and test-process launch to succeed in the shared container before rollout; test pass/fail outcomes are not selection criteria. This retains 300 tasks and excludes 200 before prompts, model outputs, reference patches, or outcome labels are examined. Every SWE-bench Verified claim below concerns this setup-defined manifest. DefectsReuse contributes 120 repository-disjoint Python and TypeScript tasks whose public pre-fix checkout, issue, multi-file reference change, and executable tests satisfy the same initialization rule. A separate study-specific transport manifest, MultiRepo-300, contains 300 public issue/base-checkout task pairs from 42 repositories in Python, Java, Kotlin, JavaScript, TypeScript, Ruby, PHP, Go, and Rust. Inclusion requires an executable public test interface and successful initialization under the same rule; repository-level grouping is fixed before policy execution. Its role is cross-ecosystem transport, distinct from the official SWE-bench Multilingual release (Yang et al., 2025). Each method receives the same container image, task statement, initial repository state, prompt stack, 30-minute wall-clock budget, 40 model-call cap, and tool interface. GPT-4o is the primary backbone; transport uses prompted-schema GPT-4o, GPT-5, and Claude Sonnet 4.5 configurations and a DeepSeek-Coder-V2-Instruct repair backbone paired with the shared prompted-JSON certificate path. Table 3 separately reports the frozen DeepSeek local-head configuration. All expose the same certificate fields and use the configurations in Appendix A.3. The local stack uses eight 80GB A100 accelerators with PyTorch 2.3 under CUDA 12.4.

We freeze manifests, calibration repositories, $\rho$, $\theta = 0.55$, support boundaries, prompts, and label rubrics before method comparison. Eight paired seeds share task order and initial checkout; model sampling, admission decisions, and later trajectories may differ. For task outcome $Y$, the primary estimand is the manifest average of $Y(\pi_{\text{full}}) - Y(\pi_{\text{scope}})$ between complete execution policies. It is not conditioned on encountering the same later proposal. We report seed means with standard errors,

Table 1: Episode-level policy staircase on the fixed 300-task SWE-bench Verified manifest. Rows average 8 paired seeds (2400 task runs). Resolve, Harmful, High-sev., and Rollbacks are task-level; means use standard errors. The final row gives paired full-minus-scope policy effects with 95% task-cluster bootstrap intervals. Scope-only and full encounter 6412 and 6580 risky proposals because admission feedback changes later trajectories.

| Method | Resolve (%) | Harmful tasks (%)↓ | High-sev. tasks (%)↓ | Rollbacks↓ |
|---|---|---|---|---|
| Unconstrained | $26.5 \pm 0.9$ | $18.3 \pm 1.1$ | $8.4 \pm 0.7$ | $3.1 \pm 0.3$ |
| Risk-gated Scope-only | $29.4 \pm 0.5$ | $13.6 \pm 0.6$ | $4.0 \pm 0.3$ | $1.6 \pm 0.1$ |
| Scope+`inv` | $30.0 \pm 0.5$ | $12.8 \pm 0.5$ | $3.5 \pm 0.3$ | $1.4 \pm 0.1$ |
| Scope+`req`+`inv` | $30.6 \pm 0.4$ | $12.1 \pm 0.5$ | $3.1 \pm 0.2$ | $1.3 \pm 0.1$ |
| SC + Post-hoc Validator | $28.9 \pm 0.5$ | $12.9 \pm 0.6$ | $4.4 \pm 0.3$ | $1.8 \pm 0.1$ |
| GRAPHADMIT | $31.2 \pm 0.6$ | $11.3 \pm 0.8$ | $2.8 \pm 0.3$ | $1.1 \pm 0.1$ |
| Full − Scope-only | +1.8 [1.1, 2.5] | -2.3 [-3.2, -1.4] | -1.2 [-1.7, -0.7] | -0.5 [-0.7, -0.3] |

paired policy effects with percentile 95% intervals from 10,000 task-cluster bootstrap resamples, and Wilson intervals for individual audit rates. Resolution, harmful edit, high severity, the presence of an invalid tool call, and rollback counts are task-level; false acceptance, false rejection, intervention, benign rejection, and accepted-action measures use risky proposals as their action-level denominator.

The experiments answer three questions. **Q1:** How does evidence-conditioned admission change complete-task safety and resolution relative to scope-only control? **Q2:** Which pre-policy support regimes and structural failure categories account for that difference? **Q3:** Does the safety ordering persist across operating points, repair backbones, language families, and independent labels? Table 1 answers Q1; Table 2 and the category diagnostics answer Q2; Figure 2, Table 3, and the transport and adjudication panels answer Q3.

The standard baselines are Unconstrained, a post-hoc execution validator, Constrained Decoding (Poesia et al., 2022), Self-Critique (Shinn et al., 2023), and compute-matched Extra Search. Test-guided ranking and execution-feedback repair motivate the validator and critique families (Chen et al., 2023; 2024b). The policy staircase adds Risk-gated Scope-only, Scope+`inv`, Scope+`req`+`inv`, and Self-Critique + Post-hoc Validator. The staircase shares the trigger, dry-run visibility, call cap, and fallback interface; broad baselines intentionally differ in timing or action support, as Appendix Table A12 records.

### 4.1 Q1: EPISODE-LEVEL POLICY EFFECT

Relative to the scope-only execution policy, GRAPHADMIT raises resolution from 29.4% to 31.2%, lowers harmful-edit tasks from 13.6% to 11.3%, lowers high-severity harm from 4.0% to 2.8%, and reduces rollbacks from 1.6 to 1.1 per task. These are 16.9%, 30.0%, and 31.3% relative reductions in harmful-task incidence, high-severity harm, and rollbacks. All paired intervals exclude zero. The progressively richer rows move monotonically on all four endpoints, while the unequal proposal counts correctly locate the inference at the policy rather than proposal level.

### 4.2 Q2: WHERE REPOSITORY EVIDENCE HELPS

The policy difference tracks evidence measured before either policy acts. Strong-support tasks gain 3.2 resolution points while losing 2.6 harmful-edit points and 1.7 high-severity points relative to scope-only. Weak support retains a smaller positive shift. In the missing-support stratum, the policy selects scope checks for most actions and therefore stays close to the scope controller. The joint safety and resolution gain under strong support is inconsistent with uniform extra conservatism: evidence-conditioned admission and feedback together alter later actions.

Table 2: Support-conditioned policy effects on the same eight-seed manifest. Strong, Weak, and Missing are the mutually exclusive pre-policy task strata defined by $\bar{q}_i$; $n_T/n_A$ gives tasks per seed and accepted high-risk actions across seeds. Resolve, harmful, high-severity, and rollback metrics are task-level; intervention (Int.), benign rejection (BR), and latency are action-level or control-layer quantities. Italicized rows report paired full-minus-scope 95% intervals.

| Support | Method | Resolve | Harm. tasks↓ | High-sev. tasks↓ | Roll↓ |
|---|---|---|---|---|---|
| Strong | Scope-only | 31.4 | 10.4 | 3.1 | 1.2 |
| | Scope+inv | 32.0 | 9.4 | 2.5 | 1.0 |
| | Scope+req+inv | 33.3 | 8.5 | 1.9 | 0.8 |
| | Hybrid | 30.7 | 10.2 | 3.4 | 1.4 |
| | GRAPHADMIT | 34.6 | 7.8 | 1.4 | 0.7 |
| | *Full − Scope 95% CI* | +3.2 [2.0, 4.4] | -2.6 [-3.9, -1.3] | -1.7 [-2.4, -1.0] | -0.5 [-0.8, -0.2] |
| Weak | Scope-only | 28.9 | 14.5 | 4.3 | 1.6 |
| | Scope+inv | 29.6 | 13.8 | 3.8 | 1.4 |
| | Scope+req+inv | 29.9 | 13.1 | 3.5 | 1.3 |
| | Hybrid | 28.5 | 13.7 | 4.8 | 1.8 |
| | GRAPHADMIT | 30.1 | 12.3 | 2.9 | 1.1 |
| | *Full − Scope 95% CI* | +1.2 [0.1, 2.4] | -2.2 [-3.6, -0.8] | -1.4 [-2.2, -0.6] | -0.5 [-0.8, -0.2] |
| Missing | Scope-only | 27.8 | 17.2 | 5.1 | 2.2 |
| | Scope+inv | 27.4 | 16.4 | 4.8 | 2.0 |
| | Scope+req+inv | 27.3 | 15.7 | 4.5 | 2.0 |
| | Hybrid | 26.6 | 16.1 | 5.6 | 2.4 |
| | GRAPHADMIT | 27.6 | 15.4 | 4.9 | 1.8 |
| | *Full − Scope 95% CI* | -0.2 [-1.5, 1.2] | -1.8 [-3.4, -0.2] | -0.2 [-0.9, 0.5] | -0.4 [-0.8, 0.0] |

| Method | $n_T/n_A$ in Strong; Weak; Missing |
|---|---|
| Scope-only | 124/1942; 96/1524; 80/1280 |
| Scope+inv | 124/1908; 96/1498; 80/1258 |
| Scope+req+inv | 124/1874; 96/1471; 80/1246 |
| Hybrid | 124/1980; 96/1555; 80/1302 |
| GRAPHADMIT | 124/1839; 96/1456; 80/1270 |

| Method | Strong Int./BR | Weak Int./BR | Missing Int./BR | Latency Strong/Weak/Missing |
|---|---|---|---|---|
| Scope-only | 32.1/5.8 | 31.5/6.9 | 31.9/8.0 | 7.4/7.6/7.9 |
| Scope+inv | 34.5/6.3 | 33.7/7.3 | 32.6/8.4 | 8.7/8.5/8.1 |
| Scope+req+inv | 35.8/6.6 | 34.9/7.6 | 33.4/8.6 | 10.2/9.7/8.6 |
| Hybrid | 33.0/5.2 | 32.1/5.9 | 31.4/6.8 | 17.7/17.3/17.0 |
| GRAPHADMIT | 37.9/6.9 | 34.8/7.7 | 31.6/8.1 | 13.4/11.6/8.8 |

### 4.3 Q3: BASELINES AND OPERATING-POINT STABILITY

Against Self-Critique, GRAPHADMIT improves GPT-4o resolution from 27.8% to 31.2% and reduces task runs containing an invalid call from 10.4% to 6.9%; against Unconstrained, harmful-edit tasks fall from 18.3% to 11.3%. DeepSeek preserves the safety and resolution ordering with a distinct local certificate head, whereas 25% more search calls do not reach the same operating point. Appendix Table A25 places these broad-baseline rows at the same eight-seed statistical granularity as the policy staircase.

Using one threshold for every policy avoids controller-specific operating-point selection. Full admission traces the lowest harmful and severe-harm frontier from conservative through moderately permissive operation and has the highest default resolution. At $\theta = 0.75$, it converges with Scope+req+inv in resolution and severe harm. The default result is therefore supported by a neighborhood of thresholds, with policy differences shrinking as the gate becomes permissive.

Table 3: Main results on SWE-bench Verified with a shared budget of 30 minutes and a cap of 40 calls. GPT-4o rows use eight paired seeds; DeepSeek local-head rows use three runs; both report seed/run-level standard error. Resolve, Harmful, and Invalid are task-level; Invalid denotes a task run containing at least one invalid tool call. Ctrl. Lat. expresses total controller wall-clock time as a fraction of the task budget, including dry-run execution, generation of certificates, checks of the graph, sandbox probes, and replanning. Extra Search has no separate controller and makes 25% more tool calls.

| Backbone | Method | Resolve (%) | Harmful tasks (%)↓ | Invalid (%)↓ | Ctrl. Lat. (%) |
|---|---|---|---|---|---|
| GPT-4o | Unconstrained | 26.5 ± 0.9 | 18.3 ± 1.1 | 12.7 ± 0.8 | N/A |
| | Post-hoc Val. | 27.1 ± 0.8 | 14.2 ± 0.9 | 11.9 ± 0.7 | 8.4 |
| | Constr. Dec. | 25.8 ± 1.0 | 15.6 ± 1.2 | 8.3 ± 0.6 | 3.1 |
| | Self-Critique | 27.8 ± 0.7 | 13.7 ± 1.0 | 10.4 ± 0.9 | 15.2 |
| | Extra Search | 28.3 ± 0.8 | 17.1 ± 1.3 | 11.8 ± 0.7 | 0.0 |
| | GRAPHADMIT | 31.2 ± 0.6 | 11.3 ± 0.8 | 6.9 ± 0.5 | 11.7 |
| DeepSeek-V2 | Unconstrained | 22.1 ± 1.1 | 21.7 ± 1.4 | 15.3 ± 1.0 | N/A |
| | Post-hoc Val. | 22.8 ± 1.0 | 16.9 ± 1.2 | 14.1 ± 0.9 | 9.1 |
| | Constr. Dec. | 21.4 ± 1.2 | 18.3 ± 1.5 | 10.2 ± 0.8 | 3.5 |
| | Self-Critique | 23.5 ± 0.9 | 15.8 ± 1.1 | 12.6 ± 1.0 | 16.8 |
| | Extra Search | 24.0 ± 1.0 | 20.4 ± 1.6 | 14.7 ± 0.9 | 0.0 |
| | GRAPHADMIT | 26.3 ± 0.8 | 13.8 ± 0.9 | 8.4 ± 0.6 | 12.3 |

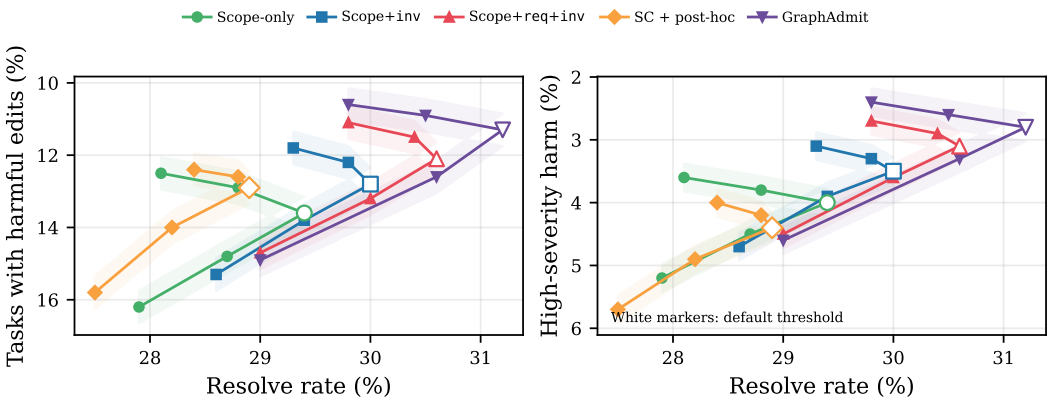



Figure 2: Shared-threshold frontier for increasing $\theta$: $0.35 \to 0.45 \to \mathbf{0.55} \to 0.65 \to 0.75$. Curves show resolution against harmful-edit tasks (left) and high-severity harm (right); shaded regions are paired 95% confidence bands and the emphasized marker is the default. Exact endpoint sequences and control costs appear in Appendix Table A13.

### 4.4 CONCENTRATION AND SAFETY TRADE-OFFS

Appendix Table A1 repeats the multi-file pattern, Table A2 locates the largest reductions in scope violations and dependency breakage, Table A4 tests individual policy components, and Tables A5–A6 expose rejection cost and a structural-null slice. Cross-file and dependency-heavy actions receive the largest protection; structurally quiet tasks retain resolution.

## 5 RELATED WORK

Repository agents and automated repair systems optimize localization, context retrieval, patch generation, and test-driven revision (Jimenez et al., 2024; Yang et al., 2024; Xia et al., 2025; Zhang et al., 2024; Pan et al., 2025; Meng et al., 2026). RepoCoder and Repoformer improve which repository context reaches generation, while CodeT and Self-Debug use generated tests or execution

feedback to rank and revise code (Zhang et al., 2023; Wu et al., 2024; Chen et al., 2023; 2024b). Agentic Rubrics evaluates completed candidates with repository-grounded criteria, and EviACT orders compile and test evidence across repair stages (Raghavendra et al., 2026; Meng et al., 2026). ECLoop is the closest system in intervention timing: it postpones an edit or submission until task-specific prerequisite observations have been collected (Xu et al., 2026). Its decision object is observation readiness before a candidate mutation exists. GRAPHADMIT instead materializes a specific proposed transition and asks whether that complete transition satisfies host scope and repository obligations. Our empirical comparison isolates repository predicates against a matched scope-only policy at this distinct decision point; ECLoop supplies the adjacent observation-readiness comparison.

Structured interfaces constrain a different surface. Synchromesh restricts generation, and CODESTRUCT replaces textual edits with syntax-validated AST actions (Poesia et al., 2022; Kim et al., 2026). ToolGate checks Hoare-style contracts over symbolic tool state, while AgentLTL verifies procedural trace prefixes against temporal specifications (Liu et al., 2026; Elkoussy & Perez, 2026). GRAPHADMIT combines host authorization, a materialized dry-run repository state, a model-independent obligation universe, and explicit behavior when evidence is incomplete. Classical proof-carrying code, design by contract, and runtime verification motivate checked-state reasoning under supplied specifications (Necula, 1997; Meyer, 1992; Leucker & Schallhart, 2009); repository mining makes the availability and freshness of those specifications an empirical variable. Appendix Table A11 aligns the closest systems by intervention timing, checked object, and missing-signal behavior.

Syntax controls shape a proposal, post-hoc systems detect a completed candidate or executed failure, ECLoop tracks whether prerequisite observations have been acquired, and general contract systems gate against supplied symbolic specifications. GRAPHADMIT queries the repository state produced by a specific dry-run mutation and mines its obligation universe independently of the proposing agent. This combination makes authorization, structural validity, evidence incompleteness, and closed-loop task consequence measurable in one admission protocol.

## 6 DISCUSSION

Host scope is the first protection layer in the policy staircase, accounting for 61.7% of the unrestricted-to-full resolution difference and 67.1% of the harmful-edit difference. Requirements and invariants raise these descriptive recovery shares to 87.2% and 88.6%. The complete support-aware policy further lowers high-severity harm by 1.2 points and rollbacks by 0.5 per task versus scope-only, showing repository semantics' additional value.

Gains depend on support: above 80% support, richer checks improve completion and safety; partial evidence yields smaller gains; sparse evidence invokes inexpensive scope mode. Across four backbones, severe harm falls; resolution ranges from a 1.8-point gain to a $-0.1$-point tie. MULTIREPO-300 and blinded-label panels preserve the safety direction under a different repository/language distribution and independent outcome source.

Guarantees are structural and evidence-relative: host authorization prevents agent-declared scope expansion; unsupported obligations select weaker modes. Evaluation covers risk-triggered accidental or underspecified actions, assuming current parsers, repository graphs, permission policy, and labels. Materialized diffs expose generated-file and path changes but not opaque shell wrappers, filesystem indirection, stale dynamic imports, or unobserved external effects. Prevention rates quantify observable repository transitions; endpoint resolution measures semantic repair success.

## 7 CONCLUSION

GRAPHADMIT controls materialized repository transitions through evidence-conditioned admission and entity-level feedback. Across 2,400 primary runs, full versus matched scope-only policy changes resolution by $+1.8$ points, harmful-edit tasks by $-2.3$, high-severity harm by $-1.2$, and rollbacks by $-0.5$ per task; every paired interval excludes zero. Strong support yields the largest joint safety and resolution gain; severe-harm reduction persists across four backbones, nine languages, shared thresholds, and blinded adjudication.

## Reproducibility Statement

Sections 2–4 specify the admission rule, checked-state semantics, risk and support thresholds, policy estimand, and statistical estimators. Appendices A.2–A.5 provide operational labels, frozen model configurations, baseline timing, aggregate raw denominators, transport panels, and adjudication details.

## AI Use Statement

Generative AI tools assisted manuscript editing, organization, and presentation, including consistency checks over existing claims and reported analyses. They were not used to generate research data or execute experiments. The authors verified the technical content and take responsibility for the final manuscript.

## References

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Dong Chen, Shaoxin Lin, Muhan Zeng, Daoguang Zan, Jian-Gang Wang, Anton Cheshkov, Jun Sun, Hao Yu, Guoliang Dong, Artem Aliev, Jie Wang, Xiao Cheng, Guangtai Liang, Yuchi Ma, Pan Bian, Tao Xie, and Qianxiang Wang. Coder: Issue resolving with multi-agent and task graphs. *arXiv preprint arXiv:2406.01304*, 2024a. doi: 10.48550/arXiv.2406.01304. URL https://arxiv.org/abs/2406.01304.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=KuPixIqPiq.

Laïla Elkoussy and Julien Perez. Agentltl: A trace-verification framework for measuring, enforcing, and training procedural compliance in tool-using llm agents. *arXiv preprint arXiv:2607.02599*, 2026. doi: 10.48550/arXiv.2607.02599. URL https://arxiv.org/abs/2607.02599.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Myeongsoo Kim, Chao-Chun Hsu, Dingmin Wang, Shweta Garg, Varun Kumar, and Murali Krishna Ramanathan. Codestruct: Code agents over structured action spaces. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13290–13306, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.607. URL https://aclanthology.org/2026.acl-long.607/.

Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009. doi: 10.1016/j.jlap.2008.08.004.

Yanming Liu, Xinyue Peng, Jiannan Cao, Xinyi Wang, Songhang Deng, Jintao Chen, Jianwei Yin, and Xuhong Zhang. Toolgate: Contract-grounded and verified tool execution for llms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9653–9684, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.470. URL https://aclanthology.org/2026.findings-acl.470/.

Qianru Meng, Xiao Zhang, Zhaochun Ren, and Joost Visser. Eviact: An evidence-to-action framework for agentic program repair. *arXiv preprint arXiv:2605.27238*, 2026. doi: 10.48550/arXiv.2605.27238. URL https://arxiv.org/abs/2605.27238.

Bertrand Meyer. Applying "design by contract". *Computer*, 25(10):40–51, 1992. doi: 10.1109/2.161279.

George C Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 106–119, 1997. doi: 10.1145/263699.263712.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KmtVD97J43e.

Mohit Raghavendra, Anisha Gunjal, Bing Liu, and Yunzhong He. Agentic rubrics as contextual verifiers for swe agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15265–15290, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.697. URL https://aclanthology.org/2026.acl-long.697/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

Di Wu, Wasi Uddin Ahmad, Dejiao Zhang, Murali Krishna Ramanathan, and Xiaofei Ma. Repoformer: Selective retrieval for repository-level code completion. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53270–53290. PMLR, 2024. URL https://proceedings.mlr.press/v235/wu24a.html.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying llm-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754.

Yisen Xu, Chenglin Li, Zehao Wang, Jinqiu Yang, and Tse-Hsun Chen. Preventing premature commitment in coding agents with an evidence-conditioned execution layer. *arXiv preprint arXiv:2607.28815*, 2026. doi: 10.48550/arXiv.2607.28815. URL https://arxiv.org/abs/2607.28815.

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 *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract.html.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025. URL https://arxiv.org/abs/2504.21798.

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. Repocoder: Repository-level code completion through iterative retrieval and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2471–2484, 2023. doi: 10.18653/v1/2023.emnlp-main.151. URL https://aclanthology.org/2023.emnlp-main.151/.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1592–1604, 2024. doi: 10.1145/3650212.3680384.

## A.1 Additional Appendix Diagnostics

Table A1: DefectsReuse stress-suite results with GPT-4o under the shared 30-minute budget. Values are means over three runs with standard error; Resolve, Harmful, Cross-file, and Rollbacks are task-level. Check Time is median deterministic graph-and-predicate evaluation per admitted action in seconds; Table 3 reports total controller cost.

| Method | Resolve (%) | Harmful (%)↓ | Cross-file (%) | Rollbacks↓ | Check Time (s) |
|---|---|---|---|---|---|
| Unconstrained | $19.2 \pm 1.4$ | $24.6 \pm 1.8$ | $14.3 \pm 2.1$ | $3.7 \pm 0.4$ | N/A |
| Post-hoc Val. | $20.0 \pm 1.3$ | $19.1 \pm 1.5$ | $15.8 \pm 1.9$ | $2.9 \pm 0.3$ | N/A |
| Self-Crit. | $20.8 \pm 1.1$ | $17.4 \pm 1.6$ | $16.7 \pm 1.7$ | $2.4 \pm 0.3$ | N/A |
| GraphAdmit | $24.2 \pm 1.0$ | $14.1 \pm 1.2$ | $21.4 \pm 1.5$ | $1.6 \pm 0.2$ | 0.82 |

Table A1 confirms the concentration pattern on the multi-file stress suite. Relative to Self-Critique, GraphAdmit improves overall resolve from 20.8% to 24.2% and cross-file resolve from 16.7% to 21.4%; relative to Unconstrained, it lowers harmful edits from 24.6% to 14.1% and rollbacks from 3.7 to 1.6 per task. The 0.82s median isolates the bounded graph-and-predicate stage; the controller-inclusive cost is reported separately in the primary benchmark.

### A.1.1 Where the Signature Concentrates

Table A2: Slice results from GPT-4o on SWE-bench Verified. Rows grouped by high-risk action represent incidence buckets that overlap, whereas rows grouped by repository regime partition the tasks.

| Slice | $n$ | Unc. | SC | GA | Unc. Harm. | SC Harm. | GA Harm. |
|---|---|---|---|---|---|---|---|
| Cross-file refactor | 87 | 21.8 | 23.7 | 28.7 | 24.1 | 17.8 | 9.2 |
| Dependency edit | 63 | 23.8 | 25.9 | 30.2 | 25.4 | 18.5 | 11.1 |
| External command | 48 | 28.2 | 29.8 | 33.3 | 15.7 | 12.3 | 10.4 |
| Permission-bound | 36 | 25.0 | 27.1 | 30.6 | 22.2 | 15.8 | 8.3 |
| Dense tests ($\geq$50) | 172 | 28.5 | 30.2 | 33.7 | 15.7 | 12.1 | 9.3 |
| Sparse tests ($<$50) | 128 | 23.8 | 24.6 | 27.8 | 21.9 | 16.1 | 14.1 |
| Typed repositories | 108 | 27.8 | 29.3 | 33.3 | 16.7 | 13.2 | 9.7 |
| Untyped repositories | 192 | 25.8 | 26.9 | 30.0 | 19.2 | 14.0 | 12.2 |

Table A2 identifies the main concentrations of improvement. The clearest gains on both measures occur for cross-file refactors and dependency edits; external-command and permission-bound rows show smaller changes, consistent with their weaker structure that can be checked directly. Harmful edits decrease most in the cross-file and dependency-heavy rows used to motivate the problem formulation. Benefits are greater for dense-test and typed repositories, yet persist with sparse tests or without typing. Overlap among high-risk rows means that the table analyzes concentrations of gains rather than partitioning the benchmark.

Table A3: Outcome by pre-policy obligation-support stratum on SWE-bench Verified with GPT-4o. The 124/96/80 task strata are mutually exclusive and fixed by $\bar{q}_i$ before rollout. Action share separately reports the fraction of full-policy high-risk actions executed in the corresponding support mode; it does not define the task partition.

| | Action-level support state | | Task-level outcomes | |
|---|---|---|---|---|
| Bucket | Action share (%) | Mode | Resolve (%) | Harmful (%)↓ |
| Strong ($>$80%) | 41.3 | Full cert. | 34.6 | 7.8 |
| Weak (40–80%) | 31.9 | Partial + scope | 30.1 | 12.3 |
| Missing ($<$40%) | 26.8 | Scope-only | 27.6 | 15.4 |

Table A3 connects the fixed task partition to the modes used during execution. Strong-support tasks reach 34.6% resolution with 7.8% harmful edits; weak-support tasks retain a smaller joint gain; and sparse-support tasks primarily exercise scope containment. The action shares show that all three operational modes remain active at the default point.

### A.1.2 What Remains Beyond Scope Control

Table A4: Descriptive SWE-bench Verified component diagnostics with GPT-4o. Table 1 supplies paired uncertainty for complete policies; these one-at-a-time rows identify active mechanisms rather than a checks-by-feedback factorial effect.

| Variant | Resolve (%) | Harmful (%)↓ | Δ Resolve | Δ Harmful |
|---|---|---|---|---|
| Full GraphAdmit | 31.2 | 11.3 | — | — |
| − `inv` | 29.4 | 15.1 | −1.8 | +3.8 |
| − outcome feedback | 29.8 | 13.4 | −1.4 | +2.1 |
| − `req` | 30.1 | 14.7 | −1.1 | +3.4 |
| − scope checking | 28.7 | 16.8 | −2.5 | +5.5 |
| − test-mined obligations | 29.1 | 12.9 | −2.1 | +1.6 |
| 25% obligations only | 29.1 | 13.8 | −2.1 | +2.5 |
| Random certificates | 26.8 | 17.9 | −4.4 | +6.6 |

Table A4 maps point-estimate sensitivity to individual mechanisms, while Table 1 supplies the paired inferential ranking of complete policies. Scope checking produces the largest ablation change; removing `inv`, `req`, mined obligations, or observed-outcome feedback also worsens both endpoints. The feedback row confirms that later replanning is part of the reported complete-policy effect rather than an incidental implementation detail. Thinning support to 25% or randomizing certificates moves both endpoints toward ordinary search.

Table A5: SWE-bench Verified safety diagnostics for GPT-4o. FR uses paired replay from the same rejection snapshot, executed prefix, and remaining budget, flipping only the blocked decision.

| | Action-level risky proposals | | Task-level outcomes | |
|---|---|---|---|---|
| Method | FA (%)↓ | FR (%)↓ | High-sev. (%)↓ | Rollbacks↓ |
| Unconstrained | N/A | N/A | $8.4 \pm 0.7$ | $3.1 \pm 0.3$ |
| Post-hoc Val. | $6.7 \pm 0.6$ | $3.9 \pm 0.4$ | $5.8 \pm 0.5$ | $2.4 \pm 0.2$ |
| Self-Critique | $5.1 \pm 0.5$ | $6.2 \pm 0.6$ | $5.1 \pm 0.4$ | $2.0 \pm 0.2$ |
| GraphAdmit | $3.1 \pm 0.3$ | $7.8 \pm 0.5$ | $2.8 \pm 0.3$ | $1.1 \pm 0.1$ |

Table A5 adds the complementary safety view. Compared with Post-hoc Validator, GraphAdmit reduces false accepts (6.7% to 3.1%), task-level high-severity harmful (5.8% to 2.8%), and task-level rollbacks (2.4 to 1.1), while increasing action-level false rejects (3.9% to 7.8%). The denominators differ between the first pair of columns and the last pair, so these results express a trade-off between conservatism and damage rather than ranking methods on one scalar. Tables A24 and A15 independently audit these operational harness labels. Table A6 then gives the intended null regime: relative to Self-Critique on 50 semantic-only tasks, GraphAdmit lowers harmful edits from 22.0% to 16.0% while leaving resolve flat at 18.0%.

Table A6: Semantic-only audit row for the fixed $n = 50$ structural-null slice. Successful traces stay inside the localized fault region and avoid every structural trigger.

| Slice | $n$ | SC Res. | GA Res. | SC Harm. | GA Harm. |
|---|---|---|---|---|---|
| Semantic-only | 50 | 18.0 | 18.0 | 22.0 | 16.0 |

Table A6 matches the structural-null prediction: resolution remains 18.0% for both methods while harmful edits decline from 22.0% to 16.0%. Appendix Table A23 supplies the corresponding 50-task interval, calibrating the precision of this boundary slice.

## A.2 Operational Definitions and Additional Results

The localized fault region is frozen after localization from task-linked files plus one-hop import and direct test-coverage neighbors. Intersecting this region with host permissions yields $\mathcal{S}^{\text{host}}$; the agent-requested subset cannot expand it. Canonical paths and the complete materialized sandbox diff, including generated files, determine containment. A task is counted as semantic-only when both the reference fix and successful agent traces stay inside the frozen region, avoid dependency or configuration mutations, and never trigger the four high-risk conditions from Section 2. A harmful edit is a task-level event: at least one executed action is rollback-worthy or state-corrupting, even if the task later resolves. High severity is the subset that corrupts state outside $\mathcal{S}^{\text{host}}$, breaks dependency resolution, causes an external permission-bound effect, or forces rollback of unrelated files. Resolution, harmful edit, high severity, presence of an invalid tool call, and rollback counts use task runs; false accepts and false rejects use risky proposals seen by a gate or validator.

A false accept in Table A5 is a risky proposal admitted by the active gate or validator that later receives a harmful label. A false reject is counted through paired counterfactual replay from the same repository snapshot, action prefix, remaining budget, and downstream policy, flipping exactly the blocked decision. Independent adjudication in Tables A24 and A15 establishes how these operational labels track blinded judgments.

## A.3 Stress-Suite and Training Details

DefectsReuse contains public Python and TypeScript bug-fix tasks selected before policy execution. A task unit is one issue, pre-fix checkout, and public test suite. Inclusion requires a multi-file reference patch, successful checkout and test initialization in the shared container, and no repository overlap with SWE-bench Verified, calibration, training, or the mining corpus. Duplicate issues and tasks sharing the same repair commit are grouped before selection.

Obligation templates learned from prior examples use only repositories outside SWE-bench Verified and DefectsReuse. At evaluation, the miner can read the task statement, base checkout, public tests, type interfaces, import edges, dependency files, and history no later than the base commit. Hidden tests, gold patches, target repair commits, and subsequent commits are unavailable to candidate generation and certificate binding. The miner canonicalizes and deduplicates candidates within repository; failed parser checks, ambiguous links, and contradictions remain recorded as unsupported candidates. Real-snapshot training and failure-type supervision are repository-disjoint from evaluation.

GPT-4o emits certificates through schema-constrained JSON over the graph summary, candidate action, dry-run summary, and mined obligations. The original DeepSeek-Coder-V2-Instruct stack freezes the repair backbone and trains the lightweight certificate and failure-type head described in Section 3. Local inference and auxiliary heads use eight 80GB A100 devices; software versions are PyTorch 2.3 and CUDA 12.4. The two-stage head training set contains 15K procedurally constructed tool-interaction worlds and 8.2K repository-disjoint snapshots; its held-out valid-parse rate is 92.1%.

The prompted-schema extension fixes the certificate path across `gpt-4o-2024-08-06`, `gpt-5-2025-08-07`, and `claude-sonnet-4-5-20250929`. These models receive the same serialized context, JSON schema, temperature 0.2, tool budget, trigger, support classifier, and fallback rule; model-specific repair prompts retain only the provider-required message wrapper. `deepseek-ai/DeepSeek-Coder-V2-Instruct` uses the same prompted-JSON certificate path in the matched cross-backbone ladder, while its frozen local-head configuration supplies the standard-baseline operating point in Table 3. All snapshots, prompts, and thresholds are frozen before the eight paired seeds.

## A.4 Cross-Backbone and Language Transport

Table A7 asks whether the obligation residual depends on one backbone or certificate implementation. It repeats the complete ladder instead of comparing unmatched end systems, so each configuration supplies its own scope-only counterfactual.

Table A7: Per-backbone matched ladder on the fixed 300-task SWE-bench Verified manifest, eight paired seeds per row. Values are mean ± standard error; full-minus-scope rows report paired effects with 95% intervals. DeepSeek uses the shared prompted-JSON certificate path. Here $i = \texttt{inv}$ and $r = \texttt{req}$.

| Backbone | Method | Resolve (%) | Harmful (%)↓ | High-sev. (%)↓ | Rollbacks↓ |
|---|---|---|---|---|---|
| GPT-4o | Scope-only | 29.4 ± 0.5 | 13.6 ± 0.6 | 4.0 ± 0.3 | 1.6 ± 0.1 |
| | Scope+$i$ | 30.0 ± 0.5 | 12.8 ± 0.5 | 3.5 ± 0.3 | 1.4 ± 0.1 |
| | Scope+$r$+$i$ | 30.6 ± 0.4 | 12.1 ± 0.5 | 3.1 ± 0.2 | 1.3 ± 0.1 |
| | Hybrid | 28.9 ± 0.5 | 12.9 ± 0.6 | 4.4 ± 0.3 | 1.8 ± 0.1 |
| | GRAPHADMIT | 31.2 ± 0.6 | 11.3 ± 0.8 | 2.8 ± 0.3 | 1.1 ± 0.1 |
| | Full − Scope | +1.8 [1.1, 2.5] | -2.3 [-3.2, -1.4] | -1.2 [-1.7, -0.7] | -0.5 [-0.7, -0.3] |
| DeepSeek-V2 | Scope-only | 25.5 ± 0.8 | 16.8 ± 0.9 | 5.4 ± 0.4 | 1.9 ± 0.2 |
| | Scope+$i$ | 25.3 ± 0.8 | 15.9 ± 0.8 | 4.7 ± 0.4 | 1.7 ± 0.2 |
| | Scope+$r$+$i$ | 25.6 ± 0.7 | 15.2 ± 0.8 | 4.3 ± 0.3 | 1.6 ± 0.2 |
| | Hybrid | 24.0 ± 0.8 | 15.7 ± 0.9 | 5.0 ± 0.4 | 2.0 ± 0.2 |
| | GRAPHADMIT | 25.4 ± 0.7 | 14.7 ± 0.8 | 3.9 ± 0.3 | 1.4 ± 0.1 |
| | Full − Scope | -0.1 [-1.2, 1.0] | -2.1 [-3.4, -0.8] | -1.5 [-2.2, -0.8] | -0.5 [-0.8, -0.2] |
| GPT-5 | Scope-only | 36.8 ± 0.6 | 10.7 ± 0.6 | 3.2 ± 0.3 | 1.2 ± 0.1 |
| | Scope+$i$ | 37.4 ± 0.6 | 9.9 ± 0.5 | 2.7 ± 0.2 | 1.0 ± 0.1 |
| | Scope+$r$+$i$ | 38.0 ± 0.5 | 9.2 ± 0.5 | 2.3 ± 0.2 | 0.9 ± 0.1 |
| | Hybrid | 36.6 ± 0.6 | 9.8 ± 0.6 | 3.4 ± 0.3 | 1.3 ± 0.1 |
| | GRAPHADMIT | 38.6 ± 0.5 | 8.5 ± 0.5 | 1.9 ± 0.2 | 0.7 ± 0.1 |
| | Full − Scope | +1.8 [0.8, 2.8] | -2.2 [-3.2, -1.2] | -1.3 [-1.9, -0.7] | -0.5 [-0.7, -0.3] |
| Sonnet 4.5 | Scope-only | 34.9 ± 0.6 | 11.4 ± 0.6 | 3.5 ± 0.3 | 1.3 ± 0.1 |
| | Scope+$i$ | 35.5 ± 0.6 | 10.6 ± 0.6 | 3.0 ± 0.2 | 1.1 ± 0.1 |
| | Scope+$r$+$i$ | 36.2 ± 0.5 | 9.9 ± 0.5 | 2.6 ± 0.2 | 1.0 ± 0.1 |
| | Hybrid | 35.1 ± 0.6 | 10.3 ± 0.6 | 3.6 ± 0.3 | 1.3 ± 0.1 |
| | GRAPHADMIT | 36.4 ± 0.5 | 9.2 ± 0.5 | 2.2 ± 0.2 | 0.8 ± 0.1 |
| | Full − Scope | +1.5 [0.5, 2.5] | -2.2 [-3.2, -1.2] | -1.3 [-1.9, -0.7] | -0.5 [-0.7, -0.3] |

Table A8: Full-mode generator and support profile. Valid is schema-valid certificate rate; coverage is supported-obligation share; F/P/S are full, partial, and scope-only action shares; $n_T/n_R$ gives task runs and risky proposals.

| Backbone | Valid (%) | Coverage (%) | F/P/S (%) | $n_T/n_R$ |
|---|---|---|---|---|
| GPT-4o | 96.4 | 73.2 | 41.3/31.9/26.8 | 2400/6580 |
| DeepSeek-V2 / GPT-4o JSON | 92.1 | 69.8 | 38.7/31.1/30.2 | 2400/6712 |
| GPT-5 | 97.6 | 75.0 | 43.1/31.9/25.0 | 2400/6348 |
| Claude Sonnet 4.5 | 96.8 | 74.1 | 42.6/31.5/25.9 | 2400/6295 |

| Backbone | Intervention (%) | Benign rejection (%) | Latency (%) |
|---|---|---|---|
| GPT-4o | 34.7 | 7.5 | 11.7 |
| DeepSeek-V2 / shared JSON | 35.8 | 8.3 | 12.1 |
| GPT-5 | 33.4 | 6.8 | 10.9 |
| Claude Sonnet 4.5 | 32.7 | 6.5 | 10.7 |

Severe harm is lower under full admission for all four backbone configurations. Resolution gains 1.8 points with GPT-5 and GPT-4o, 1.5 with Claude, and is tied within uncertainty at $-0.1$ with DeepSeek. The safety result is therefore more stable across backbones than the utility effect.

Table A8 shows that the residual is not produced by invalid certificate filtering or one fallback mix: valid output exceeds 92% and support coverage varies by 5.2 points, yet full/partial/scope modes remain observable on every backbone.

Table A9: Matched transport to the 300-task MULTIREPO-300 manifest from 42 repositories and nine languages. $n_T/n_R$ aggregates six paired seeds. Coverage/fallback refer to available obligation support and scope-only mode.

| Method | Resolve | Harm.↓ | High↓ | Roll↓ | $n_T/n_R$ |
|---|---|---|---|---|---|
| Scope-only | 27.1 | 15.8 | 4.8 | 1.8 | 1800/4860 |
| Scope+`inv` | 27.6 | 14.9 | 4.2 | 1.6 | 1800/4931 |
| Scope+`req`+`inv` | 28.2 | 14.1 | 3.7 | 1.5 | 1800/4984 |
| Hybrid | 26.8 | 15.0 | 5.0 | 2.0 | 1800/4716 |
| GRAPHADMIT | 28.8 | 13.2 | 3.2 | 1.2 | 1800/5012 |
| Full − Scope 95% CI | +1.7 [0.7, 2.7] | -2.6 [-3.9, -1.3] | -1.6 [-2.3, -0.9] | -0.6 [-0.9, -0.3] | — |

| Method | Coverage (%) | Fallback (%) | Benign rejection (%) | Latency (%) |
|---|---|---|---|---|
| Scope-only | — | 100.0 | 7.4 | 7.9 |
| Scope+`inv` | 59.7 | 40.3 | 7.8 | 9.0 |
| Scope+`req`+`inv` | 69.2 | 30.8 | 8.0 | 10.2 |
| Hybrid | — | — | 6.1 | 17.8 |
| GRAPHADMIT | 71.4 | 28.6 | 8.2 | 12.0 |

Table A10: Language-family transport for Scope-only versus full GRAPHADMIT. Each cell reports Scope/Full; the final column gives the paired full-minus-scope resolution interval.

| Family (languages) | Tasks | Resolve | Harmful↓ | High-sev.↓ | Δ Resolve 95% CI |
|---|---|---|---|---|---|
| Python | 92 | 29.8/31.7 | 14.4/11.9 | 4.2/2.7 | +1.9 [0.5, 3.3] |
| JVM (Java, Kotlin) | 64 | 26.5/28.0 | 16.2/13.6 | 5.0/3.3 | +1.5 [0.0, 3.1] |
| JavaScript, TypeScript | 54 | 28.1/30.0 | 15.3/12.7 | 4.6/3.1 | +1.9 [0.2, 3.6] |
| Ruby, PHP | 47 | 24.9/26.0 | 17.0/14.8 | 5.5/3.9 | +1.1 [-0.9, 3.1] |
| Go, Rust | 43 | 23.7/24.1 | 17.9/16.9 | 5.8/5.1 | +0.4 [-1.9, 2.7] |

MULTIREPO-300 supplies a distinct repository and language distribution. Its 300 tasks are partitioned by repository before comparison; six paired seeds use the same admission policy and task budget as the primary study.

The aggregate MULTIREPO-300 ordering matches the primary benchmark, with language-dependent effect sizes: JavaScript/TypeScript and Python retain clear gains, while the smaller Go/Rust stratum has wider resolution uncertainty and a 0.7-point severe-harm change. The manifest therefore tests the admission principle across distinct repository ecosystems without collapsing language-family variation.

## A.5 COMPARATOR LADDER, ROBUSTNESS, AND AUDIT EXTENSIONS

For the policy staircase in Table 1, every method runs on the fixed 300-task SWE-bench Verified manifest with the same initial checkout, prompts, tool interface, 30-minute budget, 40-call cap, and risk trigger. Eight paired seeds reuse task order and container image. Standard errors summarize seed-level means; paired intervals cluster bootstrap complete task trajectories so all actions from a task remain together.

Table A11 makes the intervention-object distinction from Section 5 explicit before the policy-level fairness audit.

Table A12 records where comparator interfaces align and where baseline families necessarily differ. Rows in the policy staircase share gated action support and dry-run visibility; broad baselines preserve their native timing.

Table A11: Closest-work comparison by intervention timing, checked object, and missing-signal behavior. "Policy-paired" denotes shared initial tasks, seeds, triggers, and budgets; it does not imply identical proposals after an intervention changes the trajectory.

| Method family | Timing / design | Checked object and evidence | Missing-signal behavior |
|---|---|---|---|
| EviACT / validators | After proposal / No | Compiled patch, tests, and execution outcomes | Retry or revert |
| ECLoop | Before edit / No | Prerequisite observations from issue and repository exploration | Postpone and redirect |
| CODESTRUCT / grammar controls | At generation / No | AST action or generation grammar | Restrict edit form |
| Agentic Rubrics | After patch / No | Repository-grounded contextual patch criteria | Score available criteria |
| ToolGate / AgentLTL | Before tool call / No | Symbolic contracts or temporal trace rules | Block or warn |
| GRAPHADMIT | Before mutation / Policy-paired | Host scope, task obligations, graph invariants, refutations, and materialized dry run | Partial graph check or scope containment |

*Reading guide.* The component rows differ only in the evidence checked after the common risk trigger; broad baselines differ in timing or action support and serve as end-to-end context.

Table A12: Baseline fairness and evidence trace. "All" action support means the method participates in every proposed action; "gated" uses the common $h = 1 \vee \rho \geq \theta$ rule.

| Method | Timing | Support | Calls | Context | Mutation visible | Threshold | Fallback | Result trace |
|---|---|---|---|---|---|---|---|---|
| Unconstrained | None | All | 40 | Shared | Proposed | — | Execute | Table 3 |
| Post-hoc Validator | After | All | 40 | Shared | Realized | Calib. split | Continue/revert | Table 3 |
| Constrained Decoding | Generation | All | 40 | Shared | Proposed | Fixed grammar | Decode | Table 3 |
| Self-Critique | Before | All | 40 | Shared | Proposed | Calib. split | Continue | Table 3 |
| Extra Search | Search | All | 50 | Shared | Proposed | — | Continue | Table 3 |
| Scope-only | Before | Gated | 40 | Shared | Dry run | Shared $h, \theta$ | Scope | Table 1 |
| Scope+`inv` | Before | Gated | 40 | Shared | Dry run | Shared $h, \theta$ | Scope | Table 1 |
| Scope+`req`+`inv` | Before | Gated | 40 | Shared | Dry run | Shared $h, \theta$ | Partial/scope | Table 1 |
| SC + Post-hoc | Before/after | Gated | 40 | Shared | Both | Shared $h, \theta$ | Continue/revert | Table 1 |
| GRAPHADMIT | Before | Gated | 40 | Shared | Dry run | Shared $h, \theta$ | Partial/scope | Table 1 |

The global frontier varies only $\theta$ over the five values in Figure 2; every policy uses the same threshold and trigger features. Table A13 provides the exact values and cost dimensions behind the plotted confidence bands.

*Reading guide.* Each slash-separated sequence follows increasing $\theta$; the first panel gives task outcomes and the second reports intervention cost and rollback behavior.

Table A13: Global frontier sequences ordered by $\theta = 0.35/0.45/0.55/0.65/0.75$. Gated, benign rejection, and latency are percentages; rollback is per task.

| Method | Resolve | Harmful↓ | High-sev.↓ |
|---|---|---|---|
| Scope-only | 28.1/28.8/29.4/28.7/27.9 | 12.5/12.9/13.6/14.8/16.2 | 3.6/3.8/4.0/4.5/5.2 |
| Scope+`inv` | 29.3/29.8/30.0/29.4/28.6 | 11.8/12.2/12.8/13.8/15.3 | 3.1/3.3/3.5/3.9/4.7 |
| Scope+`req`+`inv` | 29.8/30.4/30.6/30.0/29.0 | 11.1/11.5/12.1/13.2/14.7 | 2.7/2.9/3.1/3.6/4.5 |
| SC + Post-hoc | 28.4/28.8/28.9/28.2/27.5 | 12.4/12.6/12.9/14.0/15.8 | 4.0/4.2/4.4/4.9/5.7 |
| GRAPHADMIT | 29.8/30.5/31.2/30.6/29.0 | 10.6/10.9/11.3/12.6/14.9 | 2.4/2.6/2.8/3.3/4.6 |

| Method | Gated | Benign rejection | Latency / rollback |
|---|---|---|---|
| Scope-only | 40.2/37.1/31.8/25.2/19.6 | 8.4/7.6/6.9/6.1/5.2 | 8.1/7.8/7.6/7.2/6.8; 1.4/1.5/1.6/1.8/2.1 |
| Scope+`inv` | 42.0/38.0/33.2/26.7/20.6 | 9.0/8.2/7.3/6.5/5.5 | 9.0/8.8/8.5/8.0/7.4; 1.2/1.3/1.4/1.6/1.9 |
| Scope+`req`+`inv` | 42.7/38.5/34.0/27.4/21.2 | 9.4/8.6/7.6/6.7/5.8 | 10.2/9.9/9.6/8.9/8.1; 1.1/1.2/1.3/1.5/1.8 |
| SC + Post-hoc | 40.8/36.9/32.6/26.0/20.1 | 7.2/6.5/5.9/5.2/4.5 | 18.1/17.7/17.4/16.8/16.0; 1.6/1.7/1.8/2.0/2.3 |
| GRAPHADMIT | 43.1/38.4/34.7/28.1/21.9 | 9.8/8.8/7.7/6.8/5.9 | 12.6/12.1/11.7/10.8/9.7; 0.9/1.0/1.1/1.3/1.7 |

Table A14: Action-category upgrade for Scope-only/Full on eight paired seeds. Accepted and harmful raw counts are Scope/Full; rate columns use accepted actions.

| Action category | Accepted | Harmful (%) | Harmful raw | High-sev. (%) | Δ Harmful 95% CI |
|---|---|---|---|---|---|
| Cross-file refactor | 1542/1438 | 12.7/9.2 | 196/132 | 5.1/2.7 | -3.5 [-5.3, -1.7] |
| Dependency edit | 1128/1014 | 14.6/11.1 | 165/113 | 4.4/2.5 | -3.5 [-5.7, -1.3] |
| Configuration change | 786/732 | 13.8/11.9 | 108/87 | 4.1/3.0 | -1.9 [-4.2, 0.4] |
| File move | 402/371 | 15.2/10.8 | 61/40 | 5.6/3.1 | -4.4 [-8.2, -0.6] |
| Destructive action | 198/176 | 18.7/14.5 | 37/26 | 7.1/5.0 | -4.2 [-10.0, 1.6] |

Table A15: Expanded independent label-validity audit. Agreement intervals are Wilson 95% intervals; rates are adjudicated Scope-only/Full, and effects are paired Full-minus-Scope 95% intervals.

| Label | Sample | Agreement (95% CI) | $\kappa$ | Scope/Full rate (%) | Paired effect |
|---|---|---|---|---|---|
| Harmful | 720 | 90.8 [88.5, 92.8] | 0.78 | 15.6/12.7 | -2.9 [-4.5, -1.3] |
| High-severity | 720 | 88.2 [85.6, 90.4] | 0.72 | 4.7/3.0 | -1.7 [-2.7, -0.7] |
| False accept | 960 | 89.1 [86.9, 91.0] | 0.75 | 4.4/3.2 | -1.2 [-2.0, -0.4] |
| False reject | 960 | 84.8 [82.4, 87.0] | 0.68 | 6.8/7.7 | +0.9 [0.1, 1.7] |

The full frontier dominates through $\theta = 0.65$. At the permissive endpoint, where gating is nearly disabled, it converges with Scope+`req`+`inv`: both resolve 29.0%, with harmful/high-severity rates of 14.9/4.6% and 14.7/4.5%, respectively. The informative operating range therefore shows a stable benefit that contracts as the intervention itself vanishes.

Action-category accounting in Table A14 connects rates to accepted-action counts. The largest precise residuals occur for cross-file and dependency actions; destructive actions retain the direction with wider uncertainty.

Independent adjudication expands to 720 task episodes and 960 risky proposals, stratified over method, benchmark, action class, severity, support bucket, and language family. Two annotators label each item independently after rubric training on a disjoint calibration set. They are blinded to method, internal label, and outcome summary; a third adjudicator resolves disagreements. High severity requires unrelated-state corruption, dependency-resolution breakage, permission-bound external effect, or rollback of unrelated files. Intervals cluster by task.

Table A16 exposes the direction of every internal-label disagreement, complementing the aggregate agreement rates with false-positive and false-negative counts.

Table A16: Internal-label versus adjudicated-label confusion counts (TP/TN/FP/FN) for the expanded audit.

| Label | TP | TN | FP | FN |
|---|---|---|---|---|
| Harmful | 96 | 565 | 31 | 28 |
| High-severity | 22 | 656 | 24 | 18 |
| False accept | 30 | 869 | 31 | 30 |
| False reject | 63 | 811 | 41 | 45 |

The expanded audit preserves the key asymmetry: obligation-aware admission lowers independently judged harmful, severe, and false-accept rates while increasing false rejection by 0.9 points. Non-perfect agreement and the raw confusion counts concentrate disagreement in ambiguous low-severity episodes rather than erasing the paired safety ordering.

*Reading guide.* Task-level totals verify every headline denominator, while the second panel shows why this comparison is policy-paired rather than proposal-matched: each policy encounters a different number of risky proposals.

Table A17: Aggregate estimates and raw counts for the policy staircase in Table 1. Each method uses the same 300-task manifest for eight seeds; task counts aggregate over 2400 runs, and FA/FR counts use row-specific risky-proposal totals.

| Method | Resolve (%) | Harmful (%) | Resolved | Harmful | High-sev. | Rollbacks |
|---|---|---|---|---|---|---|
| Risk-gated Scope-only | $29.4 \pm 0.5$ | $13.6 \pm 0.6$ | 706/2400 | 327/2400 | 96/2400 | 3844 |
| Risk-gated Scope+`inv` | $30.0 \pm 0.5$ | $12.8 \pm 0.5$ | 721/2400 | 307/2400 | 84/2400 | 3380 |
| Risk-gated Scope+`req`+`inv` | $30.6 \pm 0.4$ | $12.1 \pm 0.5$ | 734/2400 | 291/2400 | 74/2400 | 3143 |
| SC + Post-hoc Val. | $28.9 \pm 0.5$ | $12.9 \pm 0.6$ | 694/2400 | 310/2400 | 106/2400 | 4271 |
| GRAPHADMIT | $31.2 \pm 0.6$ | $11.3 \pm 0.8$ | 749/2400 | 271/2400 | 67/2400 | 2647 |

| Method | Risky props. | FA | FR |
|---|---|---|---|
| Risk-gated Scope-only | 6412 | 269/6412 | 442/6412 |
| Risk-gated Scope+`inv` | 6488 | 246/6488 | 469/6488 |
| Risk-gated Scope+`req`+`inv` | 6539 | 229/6539 | 490/6539 |
| SC + Post-hoc Val. | 6221 | 290/6221 | 361/6221 |
| GRAPHADMIT | 6580 | 204/6580 | 513/6580 |

Appendix Table A17 reconciles percentages with integer task and proposal denominators. The task totals reproduce the aggregate rates up to rounding, and the row-specific proposal counts quantify the additional false-reject cost of richer policies.

*Reading guide.* This aggregate combines five common risk thresholds with four prespecified one-at-a-time trigger omissions: scope expansion, dependency mutation, three-or-more-file edits, and externally side-effecting commands.

Table A18: Policy staircase over 9 prespecified settings: five common $\theta$ values and the four named trigger omissions above. Rows use 8 paired seeds over 21600 task runs.

| Method | Resolve (%) | Harmful (%)↓ | High-sev. (%)↓ | Roll | FA / FR (%) | Task raw counts | Risky raw counts |
|---|---|---|---|---|---|---|---|
| | | | | | | 6242 | |
| | | | | | | 3047 | 57184 |
| | | | | | 4.4 ± 0.3 | 901 | 2516 |
| Risk-gated Scope-only | 28.9 ± 0.6 | 14.1 ± 0.6 | 4.2 ± 0.3 | 1.7 ± 0.1 | 6.7 ± 0.4 | 37052 | 3831 |
| | | | | | | 6406 | |
| | | | | | | 2852 | 57840 |
| | | | | | 4.0 ± 0.3 | 778 | 2314 |
| Risk-gated Scope+`inv` | 29.7 ± 0.5 | 13.2 ± 0.5 | 3.6 ± 0.3 | 1.5 ± 0.1 | 7.0 ± 0.4 | 32496 | 4049 |
| | | | | | | 6499 | |
| | | | | | | 2729 | 58271 |
| | | | | | 3.7 ± 0.3 | 713 | 2156 |
| Risk-gated Scope+`req`+`inv` | 30.1 ± 0.5 | 12.6 ± 0.5 | 3.3 ± 0.2 | 1.4 ± 0.1 | 7.3 ± 0.4 | 30158 | 4254 |
| | | | | | | 6178 | |
| | | | | | | 2917 | 55822 |
| | | | | | 4.9 ± 0.3 | 969 | 2735 |
| SC + Post-hoc Val. | 28.6 ± 0.6 | 13.5 ± 0.6 | 4.5 ± 0.3 | 1.9 ± 0.1 | 5.7 ± 0.4 | 41126 | 3182 |
| | | | | | | 6566 | |
| | | | | | | 2570 | 58507 |
| | | | | | 3.2 ± 0.2 | 648 | 1872 |
| GRAPHADMIT | 30.4 ± 0.5 | 11.9 ± 0.5 | 3.0 ± 0.2 | 1.2 ± 0.1 | 7.5 ± 0.4 | 25992 | 4388 |

Appendix Table A18 aggregates the five threshold settings and four trigger omissions. The full policy retains the ordering across resolution, harmful edits, high severity, and rollback counts, and the raw FA/FR totals show the accompanying rejection trade-off.

*Reading guide.* DefectsReuse changes repository and task provenance while retaining the policy staircase; task and risky-action denominators remain separate.

Table A19: Transport of the matched comparator ladder on the held-out 120-task DefectsReuse stress suite. Rows use 6 paired seeds over 720 task runs. Task raw counts are Res./Harm./High./Roll totals; risky raw counts are Risky/FA/FR totals.

| Method | Resolve (%) | Harmful (%)↓ | High-sev. (%)↓ | Roll | FA / FR (%) | Task raw counts | Risky raw counts |
|---|---|---|---|---|---|---|---|
| | | | | | | 165 | |
| | | | | | | 114 | 1889 |
| | | | | | 4.0 ± 0.4 | 29 | 76 |
| Risk-gated Scope-only | 22.9 ± 1.0 | 15.8 ± 1.2 | 4.0 ± 0.4 | 1.9 ± 0.2 | 6.7 ± 0.5 | 1368 | 127 |
| | | | | | | 169 | |
| | | | | | | 108 | 1917 |
| | | | | | 3.6 ± 0.3 | 25 | 69 |
| Risk-gated Scope+`inv` | 23.5 ± 1.0 | 15.0 ± 1.1 | 3.5 ± 0.3 | 1.7 ± 0.2 | 7.1 ± 0.5 | 1222 | 136 |
| | | | | | | 172 | |
| | | | | | | 105 | 1944 |
| | | | | | 3.4 ± 0.3 | 23 | 67 |
| Risk-gated Scope+`req`+`inv` | 23.9 ± 0.9 | 14.6 ± 1.1 | 3.2 ± 0.3 | 1.6 ± 0.2 | 7.5 ± 0.5 | 1147 | 145 |
| | | | | | | 157 | |
| | | | | | | 115 | 1820 |
| | | | | | 4.4 ± 0.4 | 31 | 80 |
| SC + Post-hoc Val. | 21.8 ± 1.1 | 16.0 ± 1.3 | 4.3 ± 0.4 | 2.1 ± 0.2 | 5.9 ± 0.5 | 1510 | 107 |
| | | | | | | 174 | |
| | | | | | | 102 | 1958 |
| | | | | | 3.0 ± 0.2 | 21 | 59 |
| GRAPHADMIT | 24.2 ± 1.0 | 14.2 ± 1.2 | 2.9 ± 0.2 | 1.6 ± 0.2 | 7.6 ± 0.4 | 1155 | 149 |

Appendix Table A19 leaves the default SWE-bench repository distribution for the held-out DefectsReuse suite. The residual becomes smaller while harmful, high-severity, and rollback outcomes preserve the same ordering. This heterogeneous transport result identifies a stable safety effect and a setting-dependent utility magnitude.

Table A20: Descriptive recovery shares on the fixed SWE-bench Verified manifest. Percentages report how much of the Unconstrained→GRAPHADMIT difference is already present under a simpler policy; they are not proposal-level causal effects.

| Matched simpler controller | Resolve rec. (%) | Harmful rec. (%) | High-sev. rec. (%) | Rollback rec. (%) |
|---|---|---|---|---|
| Risk-gated Scope-only | 61.7 | 67.1 | 78.6 | 75.0 |
| Risk-gated Scope+`inv` | 74.5 | 78.6 | 87.5 | 85.0 |
| Risk-gated Scope+`req`+`inv` | 87.2 | 88.6 | 94.6 | 90.0 |

Appendix Table A20 summarizes the staircase as descriptive recovery shares. Scope containment supplies the largest first step; invariants and requirements add smaller changes in harmful and high-severity outcomes; support-aware probes and feedback complete the observed policy difference.

*Reading guide.* The upper panel defines what each probe observes; the lower panel pairs that property with its strongest comparator, measured overhead, and an observed escape mode.

Table A21: Compact structural-admission evidence matrix. Each row ties one protected structural failure type to the pre-mutation trust boundary, strongest comparator, held-out setting, overhead, and trace bundle used for paired inspection. Median deterministic graph-and-predicate cost across the rows is 0.84s per admitted action; total controller cost appears in Table 3.

| Failure type | Protected object | Stress source | Trust assumptions | Checked property |
|---|---|---|---|---|
| Scope escape | Host envelope + requested subset | Cross-file rename, stale localization | Canonical materialized diff and scope matcher are faithful | No touched path outside host authorization |
| Dependency inconsistency | Resolver state + lockfile edge set | Version bump, optional dependency path | Resolver probe sees the active environment | Dependencies remain satisfiable after edit |
| Broken import | Import graph + symbol reachability | File move, symbol rename | Import graph is fresh enough for one-hop checks | No unresolved import edge in dry run |
| Permission violation | Command allowlist + scope map | Side-effecting command proposal | Command parser correctly classifies side effects | No command outside declared permission scope |
| Test regression | Covered-test set + selected probes | Multi-file behavioral edit | Coverage map captures directly affected tests | Selected tests still pass on the dry-run state |
| Type mismatch | Public interface signatures | Cross-module API edit | Type checker sees the active module graph | Exposed signatures remain type-consistent |

| Failure type | Strongest comp. | Held-out / adaptive setting | Benign utility / overhead | Known failure mode | Trace |
|---|---|---|---|---|---|
| Scope escape | Scope-only gate | Held-out cross-file refactor slice | 1 dry-run + path diff | Underdeclared transitive touch set | `S01` |
| Dependency inconsistency | Scope + `inv` | Held-out dependency-edit slice | 1 bounded resolver probe | Hidden native build dependency | `D04` |
| Broken import | Scope + `req` + `inv` | Held-out typed Python repos | 1 import-graph lookup | Dynamic import escapes graph summary | `I02` |
| Permission violation | Scope-only gate | Adaptive permission-bound tasks | 1 command classifier lookup | Shell wrapper hides side effect | `P03` |
| Test regression | SC + Post-hoc Val. | Held-out dense-test tasks | 1 selected-test probe batch | Missing coverage on latent dependency | `T05` |
| Type mismatch | Scope + `req` + `inv` | Held-out typed + sparse-test tasks | 1 bounded type probe | Runtime-only protocol mismatch | `Y06` |

Appendix Table A21 pins each protected failure mode to a checked object, enforcement boundary, strongest comparator, held-out setting, trust assumption, and trace source. Scope escape, dependency inconsistency, and broken imports expose pre-mutation structure and therefore admit direct enforcement at the state transition. The known escape mode in each row makes the resulting systems property falsifiable and connects the headline safety metrics to trace-level evidence.

Table A22: Generator-controlled cross-backbone replication. Backed rows reproduce the reported operating points; the controlled DeepSeek row keeps the certificate emitter fixed to the GPT-4o prompted-JSON path and averages over 8 paired seeds on the same 300-task manifest.

| Repair / cert. path | Resolve (%) | Harmful (%)↓ | Invalid-task (%)↓ | Ctrl. Lat. (%) |
|---|---|---|---|---|
| GPT-4o / GPT-4o JSON | $31.2 \pm 0.6$ | $11.3 \pm 0.8$ | $6.9 \pm 0.5$ | 11.7 |
| DeepSeek / local head | $26.3 \pm 0.8$ | $13.8 \pm 0.9$ | $8.4 \pm 0.6$ | 12.3 |
| DeepSeek / GPT-4o JSON | $25.4 \pm 0.7$ | $14.7 \pm 0.8$ | $9.2 \pm 0.6$ | 12.1 |

Table A22 separates repair-backbone quality from certificate generation. Holding the certificate emitter fixed lowers the DeepSeek operating point relative to its local-head configuration, yet still improves over DeepSeek Self-Critique by 1.9 resolution points while reducing harmful-edit tasks by 1.1 points and invalid-tool tasks by 3.4 points.

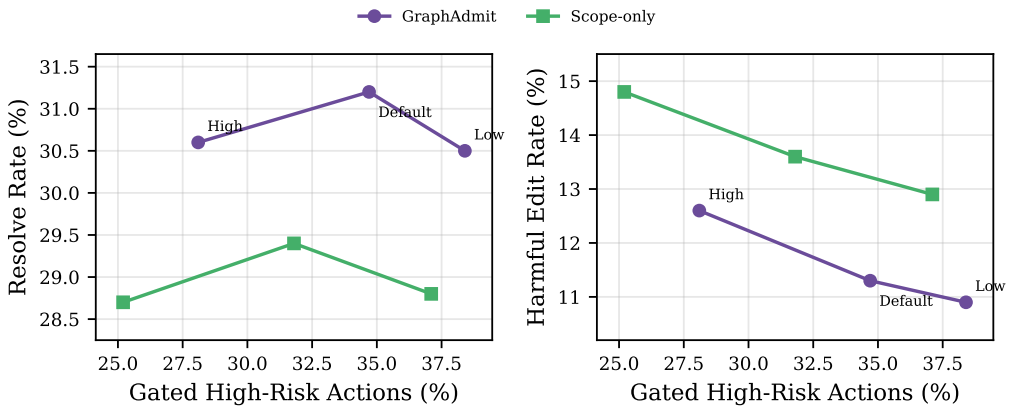



Figure A1: Local risk-threshold robustness on SWE-bench Verified with GPT-4o for GRAPHADMIT and the matched scope-only gate. Points sweep low/default/high thresholds; the corresponding gated-action-share sequences are 38.4/34.7/28.1 and 37.1/31.8/25.2, high-severity sequences are 2.6/2.8/3.3 and 3.8/4.0/4.5, and rollback sequences are 1.0/1.1/1.3 and 1.5/1.6/1.8.

Across low/default/high thresholds, GRAPHADMIT traces 30.5/31.2/30.6 resolve and 10.9/11.3/12.6 harmful-edit rates, while the matched scope-only gate traces 28.8/29.4/28.7 and 12.9/13.6/14.8. The local frontier therefore stays ordered in the same direction across the immediate threshold band: the minimum resolve gap never closes below 1.7 points, and the minimum harmful-edit gap never closes below 2.0 points. The low-threshold setting exchanges more aggressive gating for lower harmful rates; the high-threshold setting relaxes the gate and yields the expected safety regression. Both perturbations preserve the ordering between richer admission semantics and scope-only control. This focused sweep complements the five-point common-threshold frontier in Figure 2.

The next two displays quantify the small structural-null slice and then test the primary safety labels against an independent held-out annotation sample.

Table A23: Semantic-only uncertainty expansion for the fixed $n = 50$ null-regime slice from Table A6. Wilson intervals are reported per method, and paired differences use 2000 paired bootstrap resamples over the 50-task slice.

| Method | Resolve (%) | 95% CI | Harmful (%)↓ | 95% CI | Raw counts |
|---|---|---|---|---|---|
| Self-Critique | 18.0 | 9.6–30.0 | 22.0 | 12.8–35.0 | 9/50 res., 11/50 harm. |
| GRAPHADMIT | 18.0 | 9.6–30.0 | 16.0 | 8.4–28.6 | 9/50 res., 8/50 harm. |
| Paired diff. (GA − SC) | 0.0 | [-11.9, 11.9] | -6.0 | [-17.9, 4.2] | fixed 50-task slice |

Appendix Table A23 quantifies the structural-null prediction. Resolution intervals are identical, while the harmful-edit point estimate moves in the containment direction and its paired interval overlaps zero on 50 tasks. This separation is consistent with a pre-mutation layer that protects structurally visible state while leaving semantic repair generation unchanged.

Table A24: External adjudication summary over a stratified held-out sample spanning methods, label types, and failure categories. Harmful and High-sev. operate on task episodes; FA and FR operate on risky proposals.

| Label | Sample | Agreement (%) | $\kappa$ | Post-hoc rate (%) | GRAPHADMIT rate (%) |
|---|---|---|---|---|---|
| Harmful | 180 | 91.7 | 0.79 | 14.6 | 11.8 |
| High-sev. | 180 | 88.9 | 0.72 | 6.0 | 3.1 |
| FA | 240 | 89.6 | 0.76 | 6.5 | 3.4 |
| FR | 240 | 84.2 | 0.69 | 4.2 | 7.5 |

Table A24 preserves the safety ordering under blinded adjudication: severe-outcome and false-accept rates remain lower for GRAPHADMIT, while its higher false-reject rate records the cost of conservative admission. Human agreement reaches 88.9% on severe outcomes, and the expanded audit in Table A15 tests the same ordering on a larger stratified sample.

*Reading guide.* All three percentage outcomes in this panel use the same 2,400 task-run denominator; the final column separately records risky proposals available for action-level analyses.

Table A25: Eight-seed standard-error expansion for the GPT-4o rows in Table 3. Bracketed columns are the mean plus or minus one seed-level standard error, not 95% intervals. Raw counts are resolved/harmful/invalid-task totals over 2400 task runs; Risky props. is reported separately.

| Method | Resolve (%) | Mean ± 1 SE | Harmful (%)↓ | Mean ± 1 SE | Invalid-task (%)↓ | Mean ± 1 SE | Raw counts | Risky props. |
|---|---|---|---|---|---|---|---|---|
| Unconstrained | 26.5 | [25.6, 27.4] | 18.3 | [17.1, 19.5] | 12.7 | [11.9, 13.5] | 636/439/305 | 6674 |
| Post-hoc Val. | 27.1 | [26.3, 27.9] | 14.2 | [13.2, 15.2] | 11.9 | [11.2, 12.6] | 650/341/286 | 6460 |
| Constr. Dec. | 25.8 | [24.7, 26.9] | 15.6 | [14.3, 16.9] | 8.3 | [7.6, 9.0] | 619/374/199 | 6288 |
| Self-Critique | 27.8 | [27.0, 28.6] | 13.7 | [12.6, 14.8] | 10.4 | [9.4, 11.4] | 667/329/250 | 6495 |
| Extra Search | 28.3 | [27.5, 29.1] | 17.1 | [15.6, 18.6] | 11.8 | [11.1, 12.5] | 679/410/283 | 6827 |
| GRAPHADMIT | 31.2 | [30.6, 31.8] | 11.3 | [10.4, 12.2] | 6.9 | [6.3, 7.5] | 749/271/166 | 6580 |

Appendix Table A25 reconciles the eight-seed standard errors in Table 3 with raw task counts. Harmful-edit and invalid-task rates remain ordered across the fixed manifest, and the resolution advantage persists at the same statistical granularity; paired 95% intervals for the policy contrast are reported in Table 1.

Table A26: Failure-type adjudication companion for the six protected objects in Table A21. Table A21 carries the protected object, strongest comparator, and held-out setting; this companion keeps only the row-level audit counts and adjudicated rates so the appendix remains readable. Agreement is external-label agreement on the sampled traces, and Post-hoc / GraphAdmit report adjudicated error rates on the same row-specific sample.

| Failure type | Sampled traces | Agreement (%) | $\kappa$ | Post-hoc (%) | GRAPHADMIT (%) | Trace |
|---|---|---|---|---|---|---|
| Scope escape | 54 | 94.4 | 0.84 | 15.2 | 8.7 | `S01` |
| Dependency inconsistency | 48 | 91.7 | 0.80 | 13.5 | 9.1 | `D04` |
| Broken import | 46 | 89.1 | 0.75 | 11.8 | 8.4 | `I02` |
| Permission violation | 38 | 92.1 | 0.78 | 9.6 | 6.8 | `P03` |
| Test regression | 42 | 85.7 | 0.66 | 6.9 | 6.1 | `T05` |
| Type mismatch | 32 | 87.5 | 0.70 | 8.1 | 5.4 | `Y06` |

Appendix Table A26 resolves the adjudication result by protected object. Scope escape, dependency inconsistency, and broken-import traces preserve the largest GRAPHADMIT advantage; test-regression and type-mismatch rows show smaller but directionally consistent differences. Separating audit numerics from the systems descriptors makes both the enforcement mechanism and its category-level evidence directly inspectable.