When Can the Runtime Take Over? A Decision-Level Audit of LLM Agent Control

Anonymous authors
Paper under double-blind review

Abstract

LLM agents divide control between models and software runtimes, but runtime executability alone does not justify replacing a model-mediated decision. Continued model mediation can add inference work, whereas premature runtime takeover can violate unresolved behavioral obligations. We introduce Control Substitution Audit (CSA), a decision-level protocol for evaluating whether a specified runtime policy can replace an attributed model contribution under fixed boundary-visible information and a behavioral contract. CSA connects attribution of the reference model's contribution to direct runtime intervention and executed contract checking, while reporting policy coverage and downstream outcomes separately. In six selected edit conditions executed in Writer, accepted perturbation replays support model-to-commit attribution, and all six matched direct runtime substitutions satisfy their exact contracts. On a separate controlled set, broader deterministic support increases direct-action coverage from 4/22 to 15/22 and reduces recorded proposal-model tokens by 47.1%, but introduces one direct contract violation; execution completion remains 21/24 in both policy arms. These results characterize takeover as policy- and condition-relative: reduced model work or completed execution alone does not establish valid substitution.

1 Introduction

Modern LLM agents combine model reasoning with deterministic runtime control (Wu et al., 2024b; Kim et al., 2024). During execution, the runtime accumulates state, prior model outputs and tool results. This information may already constrain an action that the agent still asks a model to propose. The question concerns an existing model contribution at a specific decision boundary, such as selecting a route or constructing an edit. We study *decision-level control substitution*: whether a specified runtime policy can replace the model's contribution at that boundary, using only information available there while preserving the same behavioral obligation.

Existing work organizes and externalizes agent control: StateFlow, LLMCompiler and SwiftSage structure state transitions, plan execution and controller selection (Wu et al., 2024b; Kim et al., 2024; Lin et al., 2023); TraceCompiler compiles recurring patterns through attributable dependencies, retains unresolved model decisions and validates compiled execution (El Yadouni & Li, 2026). These mechanisms alone do not establish whether a particular existing model contribution can be transferred to a specified runtime policy under unchanged information and obligations. This evidence gap matters in both directions: continued mediation of an already resolved decision adds proposal calls, token and inference work, and a longer control path; premature takeover can commit an incorrect action or omit unresolved semantic work, violating the obligation. Our focus is the evidence needed to justify that transfer.

In Writer, the AI Writer workflow in LazyMind (LazyAGI, 2026), a model proposes edits and the runtime applies them. For "Delete exactly the selected paragraph," localization and the literal request fix the edit: the old string is the selected text and the new string is empty, allowing the literal policy to ACT. For "Rewrite the selected paragraph to make it more concise," the same target leaves replacement text unresolved, so this policy must DELEGATE that work to the model. Figure 1 illustrates the distinction: *runtime executability is not decision sufficiency*.

A takeover claim requires evidence about the reference model's contribution, the replacement policy's behavior and its executed consequences; a model call or an applicable edit alone is insufficient.

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.

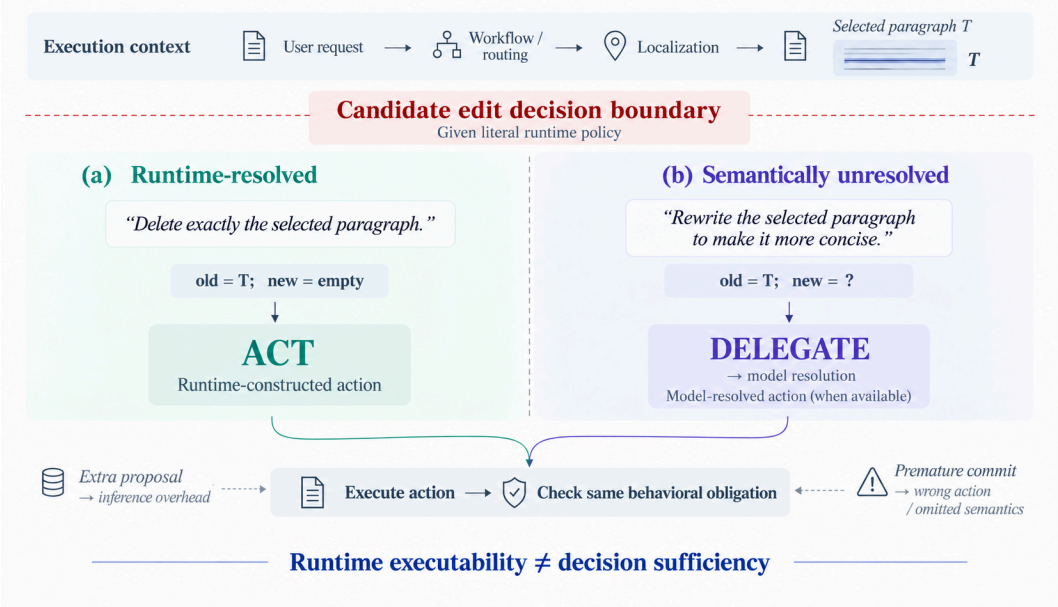

Figure 1: **The task at an edit decision boundary.** After routing and localization expose selected paragraph T, a given literal runtime policy constructs an exact deletion ($\texttt{old} = T$, $\texttt{new} =$ empty) but delegates a concise rewrite whose replacement remains unresolved. Delegation continues only when model resolution produces an action. Both runtime-constructed and model-resolved actions require execution and checking the same behavioral obligation. The annotations illustrate conceptual risks of extra model mediation and premature direct commitment, not measured outcomes.

We therefore introduce *Control Substitution Audit* (CSA), a manual evidence protocol with three roles. *Attribute* traces accepted model-produced information to the committed decision. *Substitute* tests whether the given policy acts or delegates under fixed observations and checks executed consequences against the same behavioral contract. *Bound* determines which conclusions these local outcomes support. Here, "when" refers to the declared boundary, observations, policy and behavioral conditions supporting takeover; CSA evaluates these choices rather than searching for an optimal takeover point.

In six selected edit conditions executed in Writer, accepted perturbation replays support model-to-commit attribution, and all six matched direct substitutions pass their exact contracts. On a separate controlled set, broader deterministic support increases direct coverage from 4/22 to 15/22, with direct contract pass 4/4 versus 14/15. Recorded proposal requests fall from 19 to 8 and tokens from 50,544 to 26,737 (47.1%), while one direct contract violation occurs and execution completion remains 21/24 in both arms. Two independently initialized blinded LLM auditors agree on all 18 disclosed packets; the packet–key mismatch limits the interpretation. Separately, three matched Writer workflow pairs show 21 to 12 mean observed model-module spans with generation retained, a system-level observation rather than isolated causal evidence. Our contributions are:

- We formulate decision-level control substitution for existing agent decisions, distinguishing observed model mediation from policy-relative replaceability under fixed boundary-visible information and behavioral obligations.
- We introduce CSA, connecting model-to-commit attribution, fixed-information runtime intervention and executed contract checking to scoped interpretations of direct action, delegation and their observed outcomes.
- We connect attribution in selected Writer edit conditions to matched runtime substitutions, using a controlled policy comparison and blinded audit evidence to distinguish coverage, contract validity, proposal work, downstream completion and inter-session agreement.

2 Related Work

2.1 Externalizing and Compiling Agent Control

Agent architectures organize reasoning, interaction and collaboration (Karpas et al., 2022; Yao et al., 2023; Xu et al., 2023; Wu et al., 2024a; Hong et al., 2024), while learned tool use and action interfaces structure external actions (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024; Yang et al., 2024). StateFlow organizes state-driven workflows with rule- or model-governed transitions (Wu et al., 2024b); LLMCompiler separates planning, task dispatch and parallel execution (Kim et al., 2024); SwiftSage switches heuristically between a behavior-cloned small model and an LLM planner (Lin et al., 2023). TraceCompiler compiles recurring intents using attributable producer–consumer dependencies, retains unresolved decisions as residual LLM nodes, and validates execution on withheld instances (El Yadouni & Li, 2026). AWO packages recurring tool-call sequences into deterministic meta-tools (Abuzakuk et al., 2026). Related approaches program LM pipelines (Khattab et al., 2024; Beurer-Kellner et al., 2023), execute LM-generated programs (Gao et al., 2023), or generate reusable tools (Cai et al., 2024). These mechanisms organize, externalize or reuse workflow control; a takeover claim about an existing decision additionally requires evidence about the particular contribution being replaced.

2.2 Intervention, Contracts, and Delegation

Behavioral obligations extend beyond a valid tool call. Agent Behavioral Contracts supplies runtime constraints and recovery (Bhardwaj, 2026); COVENANT checks workflow requirements before committing controller state (Wang et al., 2026b); runtime architecture patterns separate proposal, verification and commitment (Srinivasan, 2026). Runtime guards enforce specified constraints (Xiang et al., 2025; Wang et al., 2026a); ToolEmu tests agent risks through emulated tool execution (Ruan et al., 2024). Interventions probe control beyond observed execution: Zhang et al. (2026) execute alternative actions from identical trajectory prefixes to assess control value. Selective prediction separates accepted-decision performance from coverage: SelectiveNet jointly learns prediction and rejection under a coverage constraint (Geifman & El-Yaniv, 2019), while learning to defer jointly considers a predictor and a downstream expert (Mozannar & Sontag, 2020). LLM routing and cascades select models under quality–cost tradeoffs (Ding et al., 2024; Chen et al., 2024; Ong et al., 2025). Within these lines, CSA evaluates a supplied policy rather than compiling a workflow or learning a rejector. Its question is whether that policy replaces an attributed model contribution under fixed observations and behavioral obligations. Direct validity, policy coverage and downstream completion are therefore reported separately.

3 Method

3.1 Decision-Level Control Substitution

Returning to the edit boundary in Figure 1, the available target and literal request resolve deletion for the given policy, whereas concise rewriting leaves replacement content unresolved. We formalize this distinction at a decision boundary.

A decision boundary is an interface at which an agent selects, triggers or parameterizes an action, such as a route, edit or recovery step. One model call can contribute to several decisions. At a candidate takeover boundary, we fix what the runtime can see ($\mathcal{O}_d$), what direct actions it can take (A_d), and what behavior those actions must satisfy (C_d):

$$d = (\mathcal{O}_d, A_d, C_d), \qquad o_d : \mathcal{X}_d \to \mathcal{O}_d. \tag{1}$$

A condition $x \in \mathcal{X}_d$ fixes the request and pre-event state, including relevant artifact versions, the execution prefix and any declared perturbation. The observation map o_d exposes the allowed state, artifacts, constraints, tool outcomes and errors. A_d contains the direct actions, and C_d specifies their behavioral obligations: admissible transitions, required outputs, preservation requirements and permitted recovery. An edit proposal belongs to this action space because its parameters determine an executable change, even when constructing those parameters requires semantic work.

The given runtime policy is

$$\rho_d : \mathcal{O}_d \to A_d \cup \{\bot\}, \qquad \bot = \textsc{Delegate}. \tag{2}$$

Delegation returns the unresolved decision to a path permitted by C_d, without an unsupported commit. Whether it satisfies a local obligation depends on that contract; it does not by itself deliver a required downstream artifact. We evaluate the supplied policy rather than search for one over all possible runtime implementations.

The reference model action is evidence about the original control path, not the ground truth for the task. A replacement may choose a different action while satisfying the same obligation. Likewise, an existing model-produced plan or localization result may be part of $o_d(x)$ if it precedes the substituted decision. In that case the claim concerns the later decision, not an entirely model-free process. Expected outputs used for scoring, future outcomes, hidden reasoning and undeclared information are excluded from policy input. A restriction on replacement model calls is a treatment constraint; both paths retain the same task obligations.

3.2 Substitution Criterion

Reference attribution is recorded per condition as

$$M(d, x) \in \{\textsc{Observed}, \textsc{RuntimeOnly}, \textsc{Uncertain}\}, \tag{3}$$

where OBSERVED means that accepted model-produced information determines the committed control effect; the other labels indicate an existing runtime mechanism or insufficient attribution evidence. Let $\mathrm{Sat}_{C_d}(x, a) = 1$ denote an observed contract pass after executing action a under condition x. Under the fixed observation map, policy and contract, CSA awards scoped direct-substitution credit when the following recorded requirements hold:

$$M(d, x) = \textsc{Observed}, \qquad \rho_d(o_d(x)) = a \in A_d, \qquad \mathrm{Sat}_{C_d}(x, a) = 1. \tag{4}$$

Thus the original decision must be model-mediated, the runtime must act directly using the same visible information, and its executed action must satisfy the same behavioral contract. Delegation or missing evidence receives no direct-substitution credit.

CSA operationalizes these three requirements. We first establish reference attribution, then test the specified policy under fixed information and behavioral obligations, and finally bound the conclusion supported by the resulting evidence. Appendix Figure 2 shows the sequence, with recording fields in Appendix A.3.

3.3 Attribute the Original Decision

The first condition asks whether there was a model decision to replace: call presence is not control contribution. Attribution starts from the committed control effect and traces back through validation to the proposal. The auditor identifies which model-produced fields actually selected, triggered or parameterized the change in path, state, artifact, schedule, recovery or termination. Proposal, validation and commitment must refer to the same decision instance and pre-state. Source inspection identifies the responsible handlers and their transformations; execution records establish which dependency was realized. Deterministic normalization or transformation can preserve the model dependency: attribution requires a traceable action-determining dependency, not literal equality between proposal and committed fields. If validation rewrites a proposal, only contributions on which the committed effect still depends support model attribution; an entirely overridden proposal receives no such credit.

The attribution labels in Equation 3 depend on this evidence chain. OBSERVED means that an accepted model contribution determines the control-relevant committed effect. RUNTIMEONLY requires evidence that runtime state and logic determine the action without that dependency. Missing or conflicting links yield UNCERTAIN. Proposal, verification and commit authority are recorded separately: a runtime can commit a model-selected action, while a model proposal can be entirely overridden. Neither a nearby model call nor optimized source alone establishes reference mediation, and a label for one x is not inherited by unseen conditions.

3.4 Test Runtime Substitution

Fixed-information intervention. For a declared evaluation, the auditor fixes the observation map, policy, behavioral contract, test conditions and scoring rule. At the specified interface, ρ_d receives

only the pre-event observations $o_d(x)$; future information, expected outputs, hidden reasoning and undeclared information are excluded. It replaces the model contribution while the condition and surrounding mechanisms remain as declared. Local paired tests share the pre-event input; complete-workflow comparisons match workload and configuration but may reach different internal states. These designs support different levels of interpretation.

Direct action or delegation. A direct runtime action (ACT) selects or constructs the audited action without calling a model at that boundary. Invoking the designated model for that decision, including through a tool or fallback, is DELEGATE. In the running example, the literal policy constructs a deletion from the selected text and literal request; a concise rewrite leaves replacement text unresolved and is delegated. Model outputs produced before the event may remain declared observations, and generation at other boundaries may continue. Delegation records this policy's behavior, not proof that every runtime policy lacks sufficient information.

Executed contract checking. Direct actions are executed before their state or artifact consequences are checked against C_d. This later evidence is available to the auditor and checker, not to the policy selecting the action. For literal deletion, the check requires removal of exactly the selected text and preservation of the rest. Delegation is checked for the permitted path and any required recovery; subsequent task completion is recorded separately.

The contract states the obligations; the checker or oracle is the declared procedure assessing them at the stated endpoint. Passing an applicability guard does not establish output correctness, and missing evidence cannot certify unchecked obligations. In Equation 4, $\mathrm{Sat}_{C_d}(x, a) = 1$ denotes an observed pass; 0 denotes an observed violation. These labels concern one recorded execution, not a guarantee over repetitions. Repeated attempts require separate records; absent result evidence remains unscored. Each record identifies the direct or delegated behavior, outcome evidence and assessment endpoint.

Candidate and condition selection are retained in the evidence record. Historical pilots, development conditions and frozen independent conditions remain distinct. Fixing a policy during a test does not retrospectively make selection prospective or blind to earlier outcomes. Earlier pilot labels remain historical policy hypotheses in the supplementary records.

3.5 BOUND THE CLAIM

Reference attribution and observed behavior jointly determine the permissible conclusion (Table 1). Equation 4 gives direct-substitution credit only to condition-matched OBSERVED mediation followed by a direct action and an executed contract pass. A failed alternative rejects that policy on the tested condition; it does not exclude other policies using the same information.

Let D_d^{reach} be the evaluated conditions that reach the boundary, D_d^{act} those on which ρ_d acts directly, and D_d^{score} those direct actions with sufficient result evidence. Thus

$$D_d^{\mathrm{score}} \subseteq D_d^{\mathrm{act}} \subseteq D_d^{\mathrm{reach}}. \tag{5}$$

We report direct-action coverage and scored direct-contract pass rate:

$$\mathrm{Coverage}_d = \frac{|D_d^{\mathrm{act}}|}{|D_d^{\mathrm{reach}}|}, \qquad \mathrm{Success}_d = \frac{\sum_{x \in D_d^{\mathrm{score}}} \mathrm{Sat}_{C_d}(x, \rho_d(o_d(x)))}{|D_d^{\mathrm{score}}|}. \tag{6}$$

The number of unscored direct actions is disclosed alongside these measures. The scored direct-contract pass rate ($\mathrm{Success}_d$) is N/A when D_d^{score} is empty, including when no direct action occurs; coverage is N/A if no condition reaches the boundary. Coverage measures how often the policy acts directly; $\mathrm{Success}_d$ is its scored direct-contract pass rate, not task success. Calling a passing action a model substitution additionally requires the condition-matched attribution above.

Task completion, output adequacy and system cost are reported separately. The combined policy and fallback may complete tasks outside the direct-action set, while failures before the audited boundary remain in the relevant task denominator. Equal-sized denominators therefore need not contain the same conditions.

Table 1: Condition-level CSA conclusions from attribution, policy behavior and contract evidence.

Reference evidence and observed behavior	Supported conclusion
OBSERVED, direct action, contract pass	Scoped direct substitution
OBSERVED, compliant delegation	Valid delegation; no direct substitution
RUNTIMEONLY, contract pass	Existing runtime mechanism validated
Direct action with a contract violation	Tested policy violates the contract on this condition
Insufficient attribution or result evidence	No direct-substitution conclusion

Example. In the paired paragraph deletion, R1's accepted proposal removes the selected text and two extra newlines; R2 directly removes only the selected text and passes the exact preservation contract. The reference is model-mediated even though its output is incorrect. Appendix A.3 traces the full observation–proposal–commit–checker chain and distinguishes the two continuation boundaries.

4 EXPERIMENTAL SETUP

We evaluate reference attribution (RQ1), direct substitution (RQ2), policy-relative coverage and realized model work (RQ3), and agreement between blinded CSA audit sessions (RQ4). Writer, LazyMind's AI Writer workflow (LazyAGI, 2026), is the primary stateful document-agent case study, evaluated using the frozen source snapshots in Appendix A.1. Two Presentation routing fixtures instantiate the same audit logic at another workflow boundary (Appendix A.4). A matched Writer workflow study supplies system-level context. Appendix Table 3 maps the core questions to their evidence units and readouts.

System and boundary. Writer separates persistent Artifacts from model Context and exposes Workspace operations through `WriterCommand`; the runtime manages progression, termination and bounded concurrency. Editing isolates proposal construction from deterministic application. R1 uses the native model proposer; R2 uses a fixed literal policy with the same proposer as fallback. Both receive the same source, request, selection and real localization/planning result, with common snapshot/uniqueness guards and native application. A separate exact-output checker tests the requested edit, including whitespace, punctuation and non-target preservation. The endpoint is applied local Markdown, excluding Core persistence and cloud synchronization.

RQ1–RQ2: reference attribution and paired direct edits. The original development set has 18 conditions from three related document groups; the independent set has eight structurally different synthetic documents frozen before testing. Each condition has one execution per arm with alternating order and a fixed policy. Six development conditions receive direct R2 proposals. For each matched R1 instance, Experiment A reapplies the recorded proposal to reproduce its commit, then changes only its `new_string` to a unique sentinel. The perturbed proposal must pass the frozen guard and change the native commit exactly at the predicted region. Guard rejection is non-probative; infrastructure failure is separate. This replay makes no model call and does not rerun the six original R2 edits. Archived entry, runtime-mechanism, restricted-controller and Presentation probes remain separate in Appendix A.4.

RQ3: frozen policies and proposal work. Experiment C compares Literal v1 with Normalized v2. The latter normalizes request text before recognizing additional deletion and supplied-replacement forms; normalization precedes literal extraction and can alter the supplied replacement text. Both policies were fixed before the new controlled set of 24 conditions across 14 document groups was created; neither sees expected outputs. They share the observation boundary, contract, guards and application code. Two conditions stop at the common guard, leaving 22 proposal-reaching conditions. Direct coverage and scored direct-contract pass use these separate denominators.

After direct outcomes were sealed, a frozen completion protocol executed every delegated arm independently, without cross-arm output reuse or cache. It retained all direct outcomes, including

violations, and reused the frozen observations without rerunning localization/planning. The Writer proposer requests `deepseek-v4-flash` (response identifier `deepseek-flash`), temperature zero, a 4,096-token output limit, one transport attempt per operation and at most two proposal attempts. Recorded input plus output tokens and HTTP requests include retries and exclude shared upstream work. Applied execution completion is reported separately from contract pass and unscored outputs. Experiment B reaccounts the earlier paired proposal ledgers with the same decision-local scope.

RQ4: blinded interpretation. Two independently initialized blinded LLM auditor sessions receive only the CSA manual and differently shuffled versions of 18 controlled evidence packets. Each returns 18 schema-valid records of attribution, behavior, contract result and conclusion before key unsealing. Blinding is procedural: workers share a host filesystem and are instructed to access only their packet/manual directories. We compare all four fields between sessions and against the key; model identities and decoding settings were not recorded. The earlier executable protocol ablation is a separate controlled sanity check (Appendix A.2).

System-level observation and request-aware assessment. E5 uses one Chinese request for a six-section article of approximately 3,000–3,500 characters, in three fresh-session pairs. A uses model-mediated entry and continuation; B uses the explicit Writer binding and advances the singleton ready frontier (the sole ready next step). Both retain prepare, outline and draft under the same archived model configuration and declared workflow contract. Elapsed time includes conversation creation and synchronization; observed Writer module spans include generation and are not provider requests. The later request-aware assessment gives each of the six frozen anonymous drafts two stateless `gpt-5.6-sol` ratings with the complete original request and rubric, without arm, cost or trace metadata. Mechanical constraints are checked separately. The inferential unit is the draft, not the rating (Appendix A.8).

5 RESULTS

5.1 RQ1: ACCEPTED MODEL PROPOSALS DETERMINE THE SIX REFERENCE COMMITS

Source and execution records link each selected R1 proposal through validation to its saved edit. All six original commits are reproduced by deterministic reapplication. In 6/6 isolated replays, the perturbed proposal is accepted and changes the committed output exactly at the predicted region; none is non-probative or an infrastructure failure. This provides counterfactual support for model-to-commit dependency on these six selected edit conditions executed in Writer.

5.2 RQ2: SIX MATCHED DIRECT EDITS PASS THE EXACT CONTRACT

All six original condition-matched R2 direct edits pass the full-output contract and non-target preservation check with zero proposal-model requests. Together with RQ1 attribution, they establish six scoped direct substitutions; all direct actions are scored. In the worked paragraph deletion (Appendix A.3), R1 deletes two extra newlines while R2 preserves them. Reference mediation and replacement correctness are therefore separate: the behavioral obligation, not reproduction of the model's output, defines success.

5.3 RQ3: BROADER COVERAGE REDUCES PROPOSAL WORK BUT CAN VIOLATE THE CONTRACT

The original narrow policy covers 6/12 proposal-reaching development conditions, with 6/6 direct passes, but delegates all five reached independent conditions: coverage 0/5 and scored direct-contract pass rate N/A. Unsupported request syntax can cause delegation even when the required literal content is available.

The frozen controlled comparison makes this policy dependence explicit (Table 2). On the same 22 reached conditions, v1 acts directly on four and v2 on fifteen. Direct contract pass is 4/4 versus 14/15, with no unscored direct actions. In the negated deletion condition (C08), v2 overmatches the

Table 2: Controlled policy comparison. Direct-action coverage uses 22 reached conditions; direct contract pass uses scored direct actions (none unscored). Recorded proposal work includes fallback retries and excludes shared upstream work. Execution completion uses all 24 conditions, including unscored outputs and v2's direct contract violation.

Policy	Direct-action	Direct contract	Recorded proposal work		Execution
	coverage	pass	Requests	Tokens	complete
Literal v1	4/22	4/4	19	50,544	21/24
Normalized v2	15/22	14/15	8	26,737	21/24

instruction and deletes text that the exact contract requires unchanged. C08 is counted as a direct contract violation, with no fallback attempt.

Completing the delegated paths uses 19 versus 8 proposal-model requests and 50,544 versus 26,737 recorded tokens: 11 fewer requests and 23,807 fewer tokens, a 47.1% reduction relative to v1. Broader direct support reduces recorded proposal-model work on this controlled set without improving combined execution completion, which is 21/24 in both arms. Those completed executions comprise 15 contract passes and six unscored outputs for v1, versus 14 passes, six unscored outputs and the one direct violation for v2.

Both policies delegate the multi-operation condition (C24) and exhaust the frozen two-attempt proposal budget at the output-length limit. Fallback application completes on 17/18 v1 delegations and 6/7 v2 delegations; each arm has one retry. The two common-guard rejections remain outside the proposal-reaching denominator and inside the 24-condition execution denominator. Appendix A.1 groups these outcomes by condition family and reports the earlier paired-edit proposal-work ledger.

5.4 RQ4: Agreement Between Two Blinded Audit Sessions

The two independently initialized blinded LLM auditors agree on all 18 disclosed controlled packets for each of the four fields (Cohen's $\kappa = 1.0$). Each agrees with the sealed key on 15/18 attribution and conclusion labels, and 18/18 behavior and contract labels (Appendix Table 6). The three shared disagreement packets (DS01–DS03) disclose reference paths that copy declared observations or emit a fixed runtime literal without using the model proposal, supporting RuntimeOnly; proposal perturbation also leaves the commits unchanged. The key instead assigns Observed/direct substitution from undisclosed source-family metadata.

5.5 System-Level Observation: Retained Generation and Bounded Adequacy

Across three matched Writer pairs, mean observed module spans decrease from 21 to 12 while internal generation remains nine in both arms. All six runs complete prepare, outline and write_document and reach workflow completion; workspace outputs report successful saving of `draft_document` with six major sections. Descriptive mean elapsed time is 107.0 s versus 84.2 s, with a mean within-pair relative reduction of 21.53%; B's drafts are also shorter (Appendix Table 7). These measurements characterize complete workflow paths, with internal states allowed to differ between arms.

Request-aware output adequacy. The six drafts receive twelve repeated request-aware ratings, all PASS for overall content adequacy (Appendix Table 11). Redundancy/truncation receives seven PASS and five BORDERLINE ratings, averaging 1.583 on the 0–2 scale. Separate programmatic checks find six major sections and the complete-body proxy in 6/6 drafts, but only 2/6 meet the strict 3,000–3,500 CJK-character band (5/6 meet the declared approximate band).

6 Discussion and Limitations

Policy-relative takeover. The proposal-perturbation replays and matched edits connect an attributed reference contribution to successful direct substitution. The controlled policy comparison then shows why the answer must name the policy: fixed observations and obligations admit different

direct coverage, and broader support introduces a contract violation. Realized proposal work also depends on fallback and retries. Equal execution completion includes different scored outcomes; valid substitution combines condition-matched attribution with a passing direct action.

Audit interpretation and evidence alignment. Two independently initialized blinded LLM audit sessions agree on all four fields on the constructed disclosed-evidence suite. Their shared departures from the sealed key expose sensitivity to what evidence is disclosed and how the key is justified. The study measures agreement between these two sessions; natural-system accuracy and human-auditor reliability remain unmeasured. The executable protocol ablation is a sanity check; comparisons with ordinary audit practice or compilation systems on common natural conditions remain untested.

Selection and policy limits. The six attribution and direct-substitution cases are selected development conditions executed in Writer. The coverage set is controlled, and the two tested policies leave optimality and the competitive risk–coverage frontier unresolved. Writer and its proposal-model family provide the primary evidence. Presentation routing fixtures supply a second workflow instantiation at local endpoints. Broader natural workloads, additional models and workflows, and prospective candidate selection are needed to assess prevalence and transfer.

Downstream limits. E5 covers one workload and three pairs. Request-aware model ratings support bounded content adequacy, with redundancy and strict-length limitations; repeated ratings do not add independent drafts or establish semantic equivalence. Semantic editing outputs remain unscored. Output length co-varies with workflow control, so causal end-to-end latency remains unresolved. Recorded proposal usage includes fallback retries and excludes shared upstream work; billing savings require independent expenditure accounting.

Evidence across endpoints. The local intervention and complete-workflow study address different claims. At the edit boundary, sharing a frozen source, selection and upstream result permits a condition-matched comparison of proposal construction and application. In a complete workflow, different controllers may produce different intermediate states and draft lengths even when the declared workload and obligations are shared. The workflow observations therefore describe the behavior of the tested systems, while the local checks identify where a specified direct action satisfies its obligation. Keeping these endpoints separate also clarifies the role of retained generation: fewer control-related model spans can coexist with substantial model-mediated writing. The observed reduction in work is informative about the evaluated policy, but its interpretation depends on which decisions were replaced, which operations remained model-mediated and which outputs were actually checked.

7 Conclusion

CSA evaluates whether a specified runtime policy can take over an attributed model contribution under fixed observations and obligations. It makes the audited object a decision under a declared condition, connecting the reference model's attributed contribution to an executed test of a specified runtime policy. In six selected edit conditions executed in Writer, accepted proposal perturbations support model-to-commit dependency, and the matched direct runtime edits satisfy their exact contracts. On the controlled set, broader support increases direct coverage and reduces recorded proposal work, while introducing a direct violation and leaving execution completion unchanged.

Two independently initialized blinded audit sessions agree on their interpretations of the disclosed packets. Their disagreement with the retained key illustrates the importance of matching a label to the evidence made available for that decision. The agreement result concerns these sessions and packets, with the underlying auditor-model identities unresolved.

Together, these results characterize runtime takeover as policy- and condition-relative. Reduced model work or completed execution alone does not establish valid substitution: a takeover claim requires condition-matched attribution and a passing direct action under the declared information boundary and behavioral contract. Policy coverage and downstream behavior therefore remain separate outcomes rather than consequences of local substitution validity.

Reproducibility Statement

The submission supplement contains sanitized policy specifications, controlled-condition records, audit materials, derived per-condition outcomes, request-usage records and checksum manifests, with an offline verifier for file integrity, record consistency and selected aggregate arithmetic. Request-aware E5 materials include the request, rubric, derived ratings and prompt records with draft bodies removed. Private internal traces, proprietary source artifacts and complete E5 drafts are excluded. Appendix A.1 describes provenance and release limits; Appendix A.2 reports unresolved auditor-model metadata. Appendix A.3 gives the recording schema, and Appendix A.8 specifies repeated-rating units. Editable figure sources are retained.

AI Use Statement

Language-model assistance was used to organize research materials, restructure and edit the manuscript, check claim–evidence consistency, and inspect the compiled document. Language models also served as the evaluated agent components, as the fresh-context audit raters (Section 5.4), and as the blinded adequacy judges described in Section 4. The authors are responsible for reviewing all AI-assisted work and for the methods, evidence boundaries, and claims. Hidden model reasoning was not used as experimental evidence.

References

Sami Abuzakuk, Anne-Marie Kermarrec, Rishi Sharma, Rasmus Moorits Veski, and Martijn de Vos. Optimizing agentic workflows using meta-tools. *arXiv preprint arXiv:2601.22037*, 2026. doi: 10.48550/arXiv.2601.22037. URL https://arxiv.org/abs/2601.22037.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting Is Programming: A Query Language for Large Language Models. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1946–1969, 2023. doi: 10.1145/3591300. URL https://doi.org/10.1145/3591300.

Varun Pratap Bhardwaj. Agent behavioral contracts: Formal specification and runtime enforcement for reliable autonomous AI agents. *arXiv preprint arXiv:2602.22302*, 2026. doi: 10.48550/arXiv.2602.22302. URL https://arxiv.org/abs/2602.22302.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*, pp. 54067–54089, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/ed91353f700d113e5d848c7e04a858b0-Abstract-Conference.html.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=cSimKw5p6R.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Awadallah. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *The Twelfth International Conference on Learning Representations*, pp. 41348–41366, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b47d93c99fa22ac0b377578af0a1f63a-Abstract-Conference.html.

Salma El Yadouni and Guanyi Li. TraceCompiler: Skill-guided mining and compilation of LLM agent traces into mostly deterministic workflows. *arXiv preprint arXiv:2608.02680*, 2026. doi: 10.48550/arXiv.2608.02680. URL https://arxiv.org/abs/2608.02680.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2151–2159. PMLR, 2019. URL https://proceedings.mlr.press/v97/geifman19a.html.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022. doi: 10.48550/arXiv.2205.00445. URL https://arxiv.org/abs/2205.00445.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *The Twelfth International Conference on Learning Representations*, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f1cf02ce09757f57c3b93c0db83181e0-Abstract-Conference.html.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24370–24391. PMLR, 2024. URL https://proceedings.mlr.press/v235/kim24y.html.

LazyAGI. LazyMind. Public software repository, 2026. URL https://github.com/LazyAGI/LazyMind. Accessed September 26, 2026.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. SwiftSage: A generative agent with fast and slow thinking for complex interactive tasks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 23813–23825, 2023. doi: 10.52202/075280-1034. URL https://doi.org/10.52202/075280-1034.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7076–7087. PMLR, 2020. URL https://proceedings.mlr.press/v119/mozannar20b.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs with Preference Data. In *The Thirteenth International Conference on Learning Representations*, pp. 34433–34448, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565, 2024. doi: 10.52202/079017-4020. URL https://doi.org/10.52202/079017-4020.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In

The Twelfth International Conference on Learning Representations, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, pp. 27031–27098, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/7274ed909a312d4d869cc328ad1c5f04-Abstract-Conference.html.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997. URL https://doi.org/10.52202/075280-2997.

Vasundra Srinivasan. A methodology for selecting and composing runtime architecture patterns for production LLM agents. *arXiv preprint arXiv:2605.20173*, 2026. doi: 10.48550/arXiv.2605.20173. URL https://arxiv.org/abs/2605.20173.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*, pp. 2938–2950. ACM, 2026a. doi: 10.1145/3744916.3764546. URL https://doi.org/10.1145/3744916.3764546.

Jincheng Wang, Min Zheng, and Tao Wei. COVENANT: Natural-language workflow compilation for aligned agent execution. *arXiv preprint arXiv:2607.25400*, 2026b. doi: 10.48550/arXiv.2607.25400. URL https://arxiv.org/abs/2607.25400.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling*, 2024a. URL https://openreview.net/forum?id=BAakY1hNKS.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. StateFlow: Enhancing LLM task-solving through state-driven workflows. In *Proceedings of the First Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=3nTbuygoop.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM agents via knowledge-enabled reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68316–68342. PMLR, 2025. URL https://proceedings.mlr.press/v267/xiang25a.html.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023. doi: 10.48550/arXiv.2305.18323. URL https://arxiv.org/abs/2305.18323.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik 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. doi: 10.52202/079017-1601. URL https://doi.org/10.52202/079017-1601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Chubin Zhang, Zhenglin Wan, Xingrui Yu, Jingxuan Wu, Qi Wen, Pengfei Zhou, Wangbo Zhao, and Ivor Tsang. Calibration is not control: Why LLM-agent oversight needs intervention. *arXiv preprint arXiv:2606.21399*, 2026. doi: 10.48550/arXiv.2606.21399. URL https://arxiv.org/abs/2606.21399.

A Additional Experimental Details

Table 3: Core evidence chain. RQ1 and RQ2 use the same six edit conditions; controlled policy tests and complete workflows have separate endpoints.

Question	Evidence unit	Intervention / comparison	Main readout
RQ1 Attribution	6 selected Writer edit conditions	Perturb accepted proposal field	6/6 commit-sensitive
RQ2 Substitution	Same 6 edits	Model proposal vs. runtime direct action	6/6 exact direct pass
RQ3 Coverage & work	24 controlled cases	Literal v1 vs. Normalized v2	Coverage: 4/22→15/22; tokens: 50.5k→26.7k
RQ4 Agreement	18 packets, 2 blinded LLM audits	Interpret disclosed evidence	18/18 inter-auditor agreement
System context	3 Writer workflow pairs	Model-led vs. selective runtime control	Mean spans: 21→12; generation: 9→9

A.1 Experimental Provenance and Freezing Protocol

Policies, conditions and source snapshots were frozen before their execution. Policies were fixed before controlled-condition creation, and direct outcomes were sealed before delegated completion. Sanitized policy and condition records, derived outcomes and checksum manifests are supplied in the submission supplement; private source snapshots and exact repository identities are withheld.

Shared attribution and substitution trials. The six reference edits comprise three exact replacements and three structural deletions. Each replay retains the source and observation, original and perturbed proposals, application artifacts and classification. It first reproduces the original commit, then perturbs only the proposed replacement text. All six accepted perturbations change the predicted region. Their matched R2 outcomes are the same six original direct trials, with exact contract passes and zero proposal requests; replay adds counterfactual attribution evidence.

Decision-local work ledger. Table 4 reports the earlier proposal requests and recorded tokens. The ledgers include retries and unscored semantic application and exclude shared localization/planning. The independent set's two denominators of five have only four conditions in common: exact-task count includes one guard-rejected task, whereas proposal-reaching count includes a semantic rewrite.

Table 4: Recorded proposal work in the earlier paired editing study, separate from the controlled policy comparison. R1/R2 are the model-proposal and runtime-with-fallback arms.

Scope	R1 requests	R2 requests	R1 tokens	R2 tokens
Six direct development conditions	6	0	14,063	0
All development reached (12)	12	6	26,479	13,207
Old independent reached (5)	5	6	17,276	25,380

Policy freeze and delegated completion. Literal v1 uses the original literal recognizer. Normalized v2 applies Unicode NFKC, quotation-mark replacement and whitespace collapse to request text before recognition and replacement-literal extraction, so a supplied literal can change. It recognizes additional deletion and supplied-replacement forms and delegates when its recognizers produce no proposal. Source and selected text are not normalized; the exact-output checker compares the applied output directly with the frozen expected string. Both receive the same frozen source, request/selection and plan; expected outputs are available only to the checker. Table 5 groups the controlled conditions; the supplementary ledger preserves every condition and outcome.

After direct outcomes were sealed, independent proposal calls evaluated the 18 v1 and seven v2 delegated arms under the same policies, conditions and checker, without cross-arm output reuse. The 27 HTTP attempts comprise 25 first attempts and two retries, all with recorded usage. In each arm, C24 exhausts two length-limited responses before validation/application. Six completed outputs per policy remain contract-unscored. C08 is counted as a v2 direct contract violation, with no fallback attempt.

Table 5: The 24 controlled conditions grouped for presentation. PASS/FAIL denote scored contract outcomes; the six unscored semantic/underspecified cases complete application, while C24 produces no applicable proposal. Two guard rejections leave 22 reached conditions. The supplementary ledger provides all condition IDs.

Condition family	n	Literal v1	Normalized v2	Main observation
Canonical literal edits	4	4 direct PASS	4 direct PASS	Shared direct support
Expanded literal / structural forms	10	10 delegated PASS	10 direct PASS	Broader support removes proposal work
Negated deletion (C08)	1	Delegated PASS	Direct FAIL	Negation overmatch violates the contract
Semantic / underspecified	6	6 delegated, unscored	6 delegated, unscored	Completion without contract scoring
Multi-operation request (C24)	1	Delegated; length limit	Delegated; length limit	No completion after two attempts per arm
Guard rejection	2	2 rejected	2 rejected	Excluded from direct-coverage denominator

Executable provenance. The recorded environment has a pre-existing dirty worktree and a LazyLLM submodule differing from the root gitlink; the frozen live-run source snapshots govern replay. The environment records distinguish Python 3.10.12 in the stage-zero shell from Python 3.11.16 in delegated completion, without establishing a common interpreter for historical runs. Replay uses isolated artifacts and makes no proposal-model calls. Sanitized runtime settings, policy specifications, conditions and derived request/outcome ledgers are supplied; private source identities and machine records are withheld.

A.2 Blinded CSA Interpretation and Controlled Sanity Check

Two independently initialized blinded LLM auditor sessions receive only the manual and separately shuffled packet files, without shared conversation history or the sealed answer key. Repository access is prohibited by instruction; both workers share the host filesystem. Both result files are checked for 18 unique IDs and valid field enums before key unsealing. The manual requires separate attribution, behavior, contract and conclusion labels and allows insufficient-evidence judgments. The underlying model identifiers, versions and decoding settings were not recorded in the retained audit artifacts. The result measures agreement between the two sessions with model identity unresolved; whether the sessions used the same model is unknown. Table 6 reports every field.

Table 6: Blinded interpretation: exact agreement (Cohen's κ), by field. Key agreement is reported with the DS01–DS03 evidence-alignment defect, not as unqualified accuracy.

Field	Auditor–auditor	Auditor 1–key	Auditor 2–key
Reference attribution	18/18 (1.000)	15/18 (0.739)	15/18 (0.739)
Intervention behavior	18/18 (1.000)	18/18 (1.000)	18/18 (1.000)
Contract result	18/18 (1.000)	18/18 (1.000)	18/18 (1.000)
Supported conclusion	18/18 (1.000)	15/18 (0.786)	15/18 (0.786)

In DS01 and DS03, the disclosed reference path copies a declared pre-event observation; in DS02, it emits a fixed runtime literal. These paths do not read the model proposal, supporting the auditors' RuntimeOnly/runtime-mechanism labels; proposal perturbation also leaves the commits unchanged. The retained key instead assigns Observed/direct substitution from undisclosed source-family metadata.

The earlier executable 18-condition suite compares the full protocol with model-call-presence, local-replacement-only and completion-only rules. It is a constructed protocol sanity check, not a fresh annotation study or natural-system benchmark.

A.3 Audit Procedure and Candidate Selection

The audit separates reference attribution, direct substitution validity, policy coverage and downstream task outcomes. Figure 2 shows the sequence. A passing runtime mechanism without observed reference mediation is not credited as direct substitution.

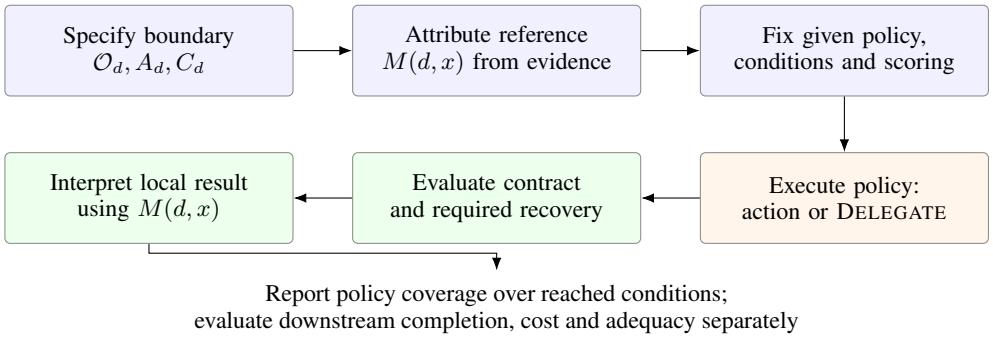

Figure 2: Audit sequence. Reference mediation and the frozen contract determine the interpretation of an action or delegation outcome. Completion, cost, and draft adequacy are evaluated separately.

Decision-level recording template. For each condition, the record identifies the decision boundary, request and pre-event state, observation map, allowed actions, contract and scoring rule, the specified policy and its version, and the reference and replacement evidence locations. The authority tuple $\Gamma_d = (\gamma_p, \gamma_v, \gamma_c)$ records proposal, verification and commit roles, each labeled MODEL, RUNTIME, SHARED or UNKNOWN. It supplements d rather than establishing mediation by itself. The attribution record links the proposal's action-determining fields, their acceptance or transformation, and the committed effect under the same pre-state, yielding $M(d, x)$. The outcome record retains direct action or delegation, executed result, contract and recovery checks, missing evidence, and the interpretation. Aggregate records disclose the reached, direct and scored-direct sets and unscored direct actions, with downstream outcomes and costs separately indexed.

Two continuation boundaries. E3 continuation projection and E5 upper-level continuation/inspection control concern different decisions, despite sharing the word "continuation." The compact boundary comparison below fixes the distinction; Section 4 gives their study designs.

Boundary	Observation and decision	Evidence and supported scope
E3: continuation projection	Committed route and completion/material facts → expose the continuation frontier; before commitment, preserve both candidates.	Existing runtime projection: activate the committed target and prune its sibling, or leave the choice unresolved. RUNTIMEONLY mechanism validation.
E5: upper-level continuation/inspection	Workflow state and tool outcomes → inspect progress and issue the next continuation; B advances the singleton ready frontier.	Workflow-level control comparison: mean continuation/inspection spans 8→0. Not a condition-matched substitution test for each internal decision.

Worked example: a paired paragraph deletion. Consider the recorded development condition `NEW-MD-structural_delete-03` from the paired editing study (Section 4). *Observation:* both arms receive the same source Markdown, request to delete exactly the selected text, selection, and fixed real localization/planning result. Write the source as $D = P\|T\|Q$, where T is the selected paragraph text and the surrounding newline separators belong to P and Q. The contract requires the

full output $P\|Q$, preserving every character outside T; the frozen expected document is reserved for scoring.

Reference contribution → validation → committed effect: R1's native model proposal sets the old string to T followed by two newline characters and the new string to empty. Validation accepts the proposal and native application saves the corresponding deletion, including those two newlines. The model's replacement fields therefore determine the committed local Markdown effect, supporting OBSERVED attribution even though that output violates the exact preservation obligation.

Runtime action/delegation → executed output → checker: R2 recognizes the literal deletion and passes the shared snapshot and uniqueness guards. It acts directly with old string T and an empty new string, making no proposal-model call and no delegation on this condition. Native application saves $P\|Q$. The separate full-document exact-output oracle and non-target preservation check pass for R2; R1's saved output fails because it deletes the two extra newlines. For this policy and condition, R2 passes the exact local Markdown contract, supporting a direct substitution.

Historical candidate records. Earlier pilot labels and the candidate inventory, including the reasons for excluding four candidates, are retained in the supplementary records as historical policy hypotheses.

A.4 SUPPORTING LOCAL CONTROL TESTS

Writer decisions. The explicit-entry pilot (P1) replaces an accepted model-proposed start with runtime routing from an explicit Writer binding: routing calls fall from one to zero, and exactly one prepared session is created. Revision validation (E2) uses no model calls: a valid write creates one revision and artifact; a stale write returns HTTP 409 with no additional revision. Continuation projection (E3) activates one committed target and prunes its sibling, while preserving both candidates before route commitment. E2 and E3 validate existing runtime mechanisms. The no-generation control (N1), without a generator or previous draft, abstains and produces no saved draft against a requirement of at least 3,000 characters; it contributes no scored direct action.

P1 is an archived entry-boundary pilot. Its records share a workflow tree hash but have different revision identifiers; replacement-input equivalence is archive-declared because the raw replacement request payload is unavailable. Three infrastructure/harness failures precede its successful prepared-session result.

Presentation routes. Two synthetic fixtures instantiate the attribution–substitution logic at a second workflow's routing boundary. Observations, actions and contracts precede reference outcomes; both reference calls requested `deepseek-v4-flash` at temperature zero, and the recorded response metadata identifies `deepseek-flash`. With disabled-background markers and a self-contained request, PPT-P1 replaces one reference routing call with a direct runtime projection to `build_outline`, passing the action contract with zero routing calls. For a request requiring current facts and sources, PPT-P2's reference projects `collect_materials`; the runtime delegates without committing a skip, passing the no-skip delegation contract. These endpoints cover direct routing and delegation, with no completed-slide, quality or presentation-latency measurement.

A.5 MATCHED LONG-FORM WRITING: PER-RUN MEASUREMENTS

The matched study (E5) uses one Chinese request about a sustainable personal learning system and three pairs of runs. A uses model-mediated entry and continuation; B uses the explicit Writer binding and advances the singleton ready frontier. The frozen protocol declares a shared workload, runtime, workflow revision/tree and task contract. The archived model configuration specifies `deepseek-v4-flash`; no returned model identifier or pinned backend snapshot was retained. Each arm starts a new conversation/session, in order A1/B1, B2/A2, A3/B3. B waits for two identical authoritative projections after transitions and polls final completion once per second. Runner wall time includes conversation creation and this synchronization. Earlier transport and state-version failures are outside the final clean pair set and remain in the companion records.

The frozen contract requires completion of prepare, outline and write_document, persistence of `draft_document`, approximately 3,000–3,500 Chinese characters and approximately six major

sections, binding to the intended workflow revision/tree, and zero KB, Web, image-search or image-generation operations. Retained traces and frozen-runner outputs support three-stage execution and workflow completion in all six runs; workspace outputs report successful draft saving with six major sections. A's recorded routing and revision/tree match the protocol; B's execution follows the frozen runner. No named KB, Web, image-search, or image-generation operation appears in the retained traces.

Table 8 reports every run's time, model-span count, draft size and archived contract label. E5 provenance is partially reconstructed: the original clause-level PASS evaluator, historical length-tolerance implementation, B's per-run revision/tree binding and complete observability of forbidden operations remain unverified. The six archived PASS labels are retained with that scope. Means and pairwise reductions use unrounded records. A1's original runner output is unavailable, so its wall time retains the archived final-summary source.

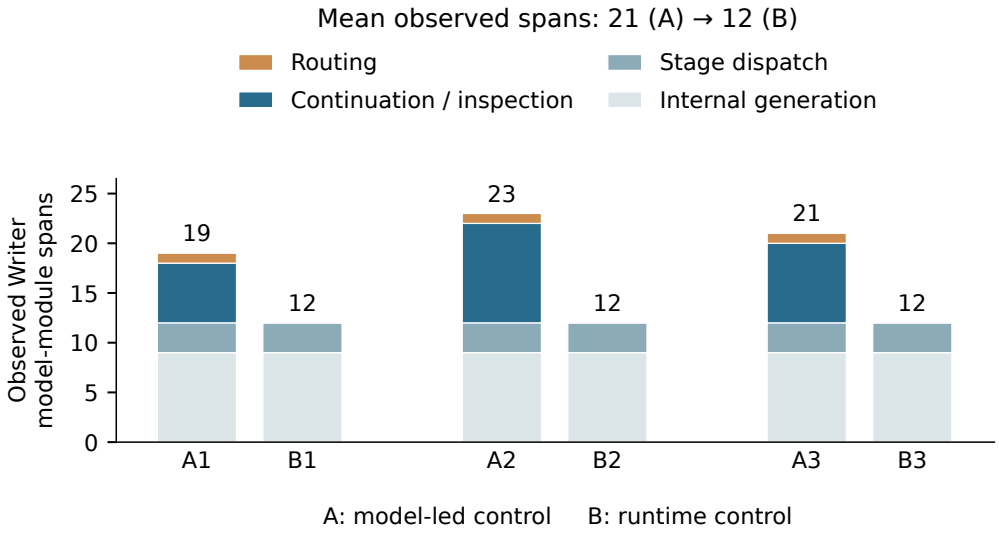

Figure 3: Model activity in all three matched long-form writing pairs. Selective control retains dispatch and internal generation while removing routing and continuation/inspection from the observed Writer spans. Post-completion memory is outside this endpoint.

Table 7: Matched long-form writing under the archived `deepseek-v4-flash` configuration: three pairs, one prompt, fresh sessions. Means are over three runs per arm; the label row counts archived PASS labels. The wall-time value -21.53% is the *mean paired relative change*, $\frac{100\%}{3}\sum_{i=1}^{3}(t_{B_i}-t_{A_i})/t_{A_i}$, not the relative change between arm means; -22.8 s is the mean paired absolute change. Generation is retained in both arms.

Metric	A: model control	B: selective control	Mean change
Wall time (s)	107.0	84.2	-22.8 (-21.53%)
Observed Writer module spans	21	12	-9
Draft characters	4,341	3,651	-690
Major sections	6	6	0
Archived contract label	3/3	3/3	–

Model activity and accounting scope. Observed model-module spans cover Writer execution, including internal generation (Figure 3). The 99 Writer spans comprise 63 in A and 36 in B; all have recorded input and output usage. Three post-completion memory-maintenance calls are accounted for separately in the supplementary ledger and remain part of session expenditure. Roles are assigned from tool proposals, ancestry, and input/output evidence.

Table 8: All six matched long-form writing runs (E5). A uses model-mediated control; B uses explicit entry and runtime continuation. Wall clocks and draft sizes retain the archived runner/artifact source; A1's original runner output is unavailable. Workspace outputs report six major sections. PASS entries are archived labels; the original clause-level evaluator has not been recovered.

Run	Wall time (s)	Writer model spans	Draft characters	Archived contract label
A1	109.360	19	4,317	PASS
B1	90.814	12	3,497	PASS
A2	108.974	23	3,819	PASS
B2	92.691	12	3,700	PASS
A3	102.705	21	4,888	PASS
B3	69.132	12	3,757	PASS

Table 9: Usage and cumulative model time for the six matched E5 runs. All 99 observed Writer model spans have input/output usage; the independent provider-request denominator remains unknown. Summed span durations include overlap and are not additive components of wall time.

Run	Input tokens	Output tokens	Summed span time (s)
A1	104,879	29,060	144.816
B1	43,607	22,707	121.007
A2	138,194	25,881	145.779
B2	42,491	27,493	136.619
A3	145,157	25,044	143.369
B3	37,224	24,027	110.849

Model-module spans do not establish an independent provider-request denominator or complete billing coverage: hidden retries and requests absent from exported spans remain unresolved. Summed span durations in Table 9 include overlap and therefore cannot partition runner wall time.

A.6 Historical Motivation

Historical Writer profiles motivated examining model-mediated control. They span four code revisions and five creation, outline-expansion and editing tasks, so they are not used as causal CSA evidence. Detailed historical profiles, task definitions, failure/retry records and editable historical figures remain local and are excluded from the submission supplement.

A.7 Paired Local Editing

The comparison replaces construction of an edit proposal after a shared real localization and planning step. R1 is *Model proposal*; R2 is *Runtime proposal with model fallback*. The runtime policy attempts a narrow literal construction and otherwise invokes the same model proposer as R1. Both arms receive the same Markdown source, request, selection and frozen upstream result, use common guards, and share deterministic application (Figure 4). The endpoint is applied local Markdown and a confirmed output file, excluding Core persistence and cloud write-back.

Literal construction and applicability. For an authorized literal deletion, write $D = P\|T\|Q$, with separators assigned by the contract. The runtime constructor proposes replacing T with the empty string; deterministic application must produce $D' = P\|Q$ while preserving P and Q exactly. Source, request, the shared plan and available selection are policy inputs; frozen expected outputs are used only for scoring. Supported structural deletion and supplied exact replacement forms permit direct construction. Other syntax or a semantic rewrite without supplied replacement content can cause delegation. Snapshot and uniqueness checks establish applicability, not whether the proposed text expresses the intended semantic change.

Condition-matched attribution for direct edits. For each of the six development conditions where R2 acts directly, the R1 record contains a real model-produced old/new-text proposal, accepted by the common validator and executed by the shared application function. Proposal fields agree with the application input, and the applied output agrees with the committed file record. Both arms use the same pre-proposal source and upstream result for that condition. This links the reference model contribution to its committed effect; it does not require the reference output to satisfy the exact-output contract. All six R2 outputs satisfy their frozen full-output checks. No direct action in this comparison lacks result scoring.

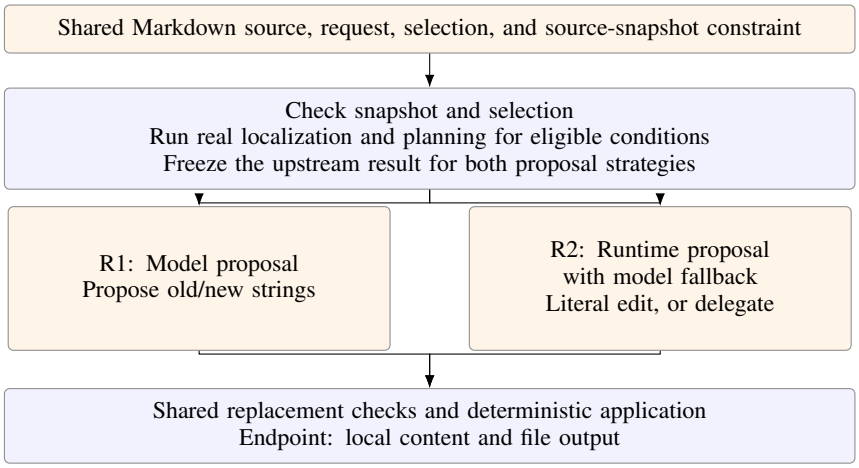

Figure 4: The paired editing comparison holds real localization and planning fixed before changing proposal construction. The runtime policy acts on supported literal forms and otherwise uses the same model proposer as R1. Both paths share mechanical applicability checks and deterministic application. Expected post-edit outputs are available only to the checker.

Conditions and model settings. The development set contains 18 conditions from three related template document groups: nine exact edits, three semantic rewrites and six expected rejections. The independent set contains eight frozen, structurally different synthetic documents with one natural request each: five exact edits, one semantic rewrite and two expected rejections. Development exact tasks comprise three structural deletions, three fragment deletions and three exact replacements. Independent exact tasks test deletion under repeated headings and nested lists, a code-fence/heading update, a heading-free fragment deletion, and a disjoint table/prose update. The policy and cases remain fixed throughout the comparison. Each condition has one run per arm, with alternating arm order; the two sets are reported separately.

The requested model is `deepseek-v4-flash`; responses identify `deepseek-flash`. Actual requests use temperature zero, a 4,096-token output limit and no streaming. Localization, planning and each arm's proposal operation independently allow at most two attempts. All selected conditions were run, and request costs include failed attempts. Localization and planning are performed once per eligible condition and their outputs shared; their requests are reported separately from proposal requests.

Contracts and denominators. Exact edits require character-for-character equality to the frozen full output, including punctuation and whitespace. An unexpected guard rejection remains an exact-task failure; expected rejections are scored separately. Non-target preservation checks cover the declared edit envelope's exterior. The full-output check additionally constrains unchanged text inside a multi-target envelope. Semantic application completion is recorded without a quality score. Table 10 reports the outcomes and denominators.

Markdown targeting uses heading paths and occurrence, the document root, or caller selection. The shared harness checks snapshots and selection/replacement uniqueness. A narrow request recognizer can delegate because its syntax is unsupported even when the information needed for an edit is present.

Table 10: Live paired editing outcomes. R1 uses model proposals; R2 uses a frozen runtime policy with the same model as fallback. Each condition has one run per arm. Semantic applications are unscored for quality. Request counts exclude shared localization/planning and include retries.

Set	Measure	R1	R2
Development	Exact-output contracts passed	6/9	6/9
18 conditions	Correct expected rejections	6/6	6/6
3 document groups	Semantic applications completed	3/3	3/3
	Model-proposal HTTP attempts	12	6
	Proposal retry attempts	0	0
	Direct runtime proposals / reached	–	6/12
Independent	Exact-output contracts passed	4/5	3/5
8 conditions	Correct expected rejections	2/2	2/2
8 documents	Semantic applications completed	1/1	1/1
	Model-proposal HTTP attempts	5	6
	Proposal retry attempts	0	1
	Direct runtime proposals / reached	–	0/5

The independent exact set contains four proposal-reaching conditions and one unexpected guard rejection. Its proposal-reaching set contains those four exact edits and one semantic rewrite. Shared upstream requests are 24 for development and 10 for the independent set.

Development outcomes. Both arms pass six of nine exact-output contracts and correctly reject all six expected rejection conditions. All three semantic rewrites per arm reach application, with semantic quality unscored. R2 directly constructs six of the twelve proposals and falls back for six, reducing proposal-model requests from twelve to six. Its six direct proposals pass their exact contracts. Equal aggregate pass counts do not mean identical outcomes: R1's paragraph deletion removes two extra newlines on one condition that R2 passes, while a different fragment deletion passes in R1 and fails in R2's model fallback. Shared planning errors also remain in the measured outputs.

Independent-document outcomes. R1 passes four of five exact contracts and R2 passes three. The shared guard unexpectedly rejects the code-fence/heading condition before proposal; this failure remains in both five-condition exact denominators. The five conditions that reach proposal are four exact edits plus the semantic rewrite, so the proposal-coverage and exact-output denominators have equal sizes but different members. R2 directly constructs none of these five proposals and falls back on all five. Both arms correctly reject the two expected-rejection conditions and complete the semantic application without a quality score.

R2's additional exact failure occurs on nested-list deletion: both its initial model response and its retry end at the output-length limit without an applicable proposal. R1 succeeds on its first response to the same request payload. These responses were successful HTTP transports; the failure is model completion under the fixed output limit. First-attempt results among the four exact conditions reaching proposal are 4/4 for R1 and 3/4 for R2; the failed retry does not change the final outcome or remove the guard-rejected fifth exact condition.

Cost scope. Development uses 24 shared localization/planning requests, 12 R1 proposal requests and 6 R2 proposal requests. The independent set uses 10 shared requests, 5 R1 proposal requests and 6 R2 requests including the unsuccessful retry. Across both sets, all 63 locally confirmed HTTP attempts have usage records. These counts include semantic proposals, fallbacks and retries; they are observed sends rather than an independent gateway or billing denominator. The runtime policy saves proposal requests on the development set but none on the independent set. Shared upstream work is excluded from these arm-specific proposal counts. The combined experiment-process clock includes shared work and both arms and is not an independent pipeline latency for either strategy.

Earlier offline checks. Before the live comparison, one mocked-upstream traversal of the 18 development conditions checked nine runtime literal outputs, three delegation branches, and six common rejections per arm. Its model branches were not executed. These earlier checks estab-

lish implementation behavior under supplied plans; they are not additional live repetitions. Their condition-level table remains in the supplementary records.

A.8 Request-Aware Blinded Draft Assessment

The later request-aware evaluation uses six frozen anonymous E5 drafts, each assessed with the original request. Each receives two stateless `gpt-5.6-sol` calls with low reasoning effort; temperature and seed are unspecified, with seed unsupported. Each prompt contains only the draft ID, complete original request, subjective rubric and draft. Arm, pair, latency, trace, cost and prior scores are withheld; arm decoding occurs after outputs are immutable. The retained key identifies arms, but the exact mapping from blinded drafts to individual runs and persisted artifacts has not been recovered. These are repeated ratings by one evaluator model, not twelve independent drafts or human judgments. Two initial pre-response infrastructure failures are retained separately from the twelve valid ratings.

Table 11: Request-aware E5 ratings: six drafts, two repeated ratings per draft. Counts are ratings out of twelve; the inferential unit remains the draft. Mechanical request constraints are checked separately.

Dimension	PASS	BORDERLINE	FAIL	Mean / 2
Topic/goal coverage	12	0	0	2.000
Actionability	12	0	0	2.000
Coherence/progression	12	0	0	2.000
Section content	12	0	0	2.000
Redundancy/truncation	7	5	0	1.583
Overall content adequacy	12	0	0	2.000

The six subjective dimensions are topic/goal coverage, actionability, coherence/progression, section content, redundancy/truncation and overall content adequacy, scored FAIL=0, BORDERLINE=1 and PASS=2. The rubric explicitly excludes character counts, section counts and forbidden tool execution. The submission supplement contains the frozen request, rubric, derived ratings and per-draft checks, with draft bodies removed from the supplied prompt records; complete drafts and prompts are withheld.

Post-hoc programmatic checks separately count CJK characters, major sections and developed paragraphs. The complete-body proxy requires at least twelve developed paragraphs and is not a semantic completeness proof. The strict length band is 3,000–3,500 CJK characters; the declared approximate band is 2,700–3,850. Exactly six major sections and the body proxy pass on all six drafts, approximate length on five, and strict length on two. These CJK counts differ from the original E5 total-character metric in Table 7; neither these checks nor the later adequacy ratings define the original PASS labels. Draft text alone cannot verify whether forbidden tools were called.

Companion data. The self-contained submission supplement provides derived per-condition editing outcomes, request usage, audit materials, E5 run-level summaries and recorded evaluation settings. Its manifest lists every supplied file. Private internal traces, proprietary source artifacts and complete E5 draft text are excluded; the package supports checking disclosed records and reproducing selected aggregate arithmetic, not native Writer replay or rerating of withheld drafts.