DO NOT RESTART: RESIDUAL COMPLETION FOR STATEFUL AGENT HANDOFFS

Anonymous authors
Paper under double-blind review

ABSTRACT

Routing and cascades reduce tool-agent cost by transferring control across models, but stateful handoffs must preserve accepted choices, realized effects, and unfinished obligations. We formulate this as *commitment-constrained residual completion* and introduce Commitment-Frontier Residual Completion (CFRC). CFRC enforces target-before-proposal, whole-proposal-before-authority, and live-evidence-before-success: it freezes a residual contract from accepted progress, closes the successor continuation into an evidence-linked graph, and admits execution only when the remainder is covered, with live receipts discharging obligations. We establish contract-relative partial correctness, which extends to the original residual request under complete contract construction. Across five environments and two same-provider model pairs, CFRC achieves comparable macro accuracy to strong full-task agents at 22.0%–34.6% mean per-surface cost, with additional cross-provider results demonstrating broader transfer.

1 INTRODUCTION

Large language models (LLMs) and multimodal agents increasingly serve as the decision-making core of tool-using systems for planning, software interaction, and multi-step execution (Trivedi et al., 2024; Lu et al., 2025). Such trajectories repeatedly invoke a model for reasoning, action selection, tool use, and verification, making it costly and often unnecessary to use the strongest model throughout. A practical alternative is to let lower-cost models handle routine steps and transfer control when tasks become difficult or uncertain (Ong et al., 2025; Son et al., 2026). Its benefit, however, depends on whether the successor can reuse accepted progress rather than restart. Unlike stateless prompt routing (Chen et al., 2023), agents operate in persistent environments where actions can produce irreversible side effects. Restarting wastes computation and risks duplicate actions, while continuing without clear specifications risks goal drift. This raises a broader handoff question: when control passes between models, how can the successor continue from the reached state while preserving accepted choices and realized effects, completing open obligations, and avoiding duplicate actions?

In the invoice task of Fig. 1a, the user asks the agent to pay one outstanding invoice and send its receipt. The first model selects an illustrative invoice `INV-42`, which the user confirms, making the choice binding even before any tool call. At a later handoff, payment has succeeded but receipt delivery remains pending. The successor cannot restart, as another API payment call would cause a duplicate charge, nor switch invoices, since the user confirmed `INV-42`. Yet declaring success is also wrong because delivery remains unfulfilled. The task is therefore neither fresh nor complete: the accepted trajectory contains fixed progress alongside unfinished work. We call this handoff point a *commitment frontier* and frame the challenge as *commitment-constrained residual completion*. Under this formulation, a valid continuation must execute from the reached state, preserve accepted choices and realized effects, and complete all open request-scoped obligations.

Existing agent components address adjacent parts of this problem but leave three gaps, as illustrated in Fig. 1b. First, routing, handoff, and memory mechanisms determine who acts or transfer past context (KC & Budathoki, 2026; Wang et al., 2024), but they do not independently specify what the successor must still preserve and complete. In the invoice example, transferring the transcript or reached state alone does not prevent the successor from overlooking delivery or reinterpreting the confirmed choice. Second, planners and transcript-conditioned successors can propose later actions

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.

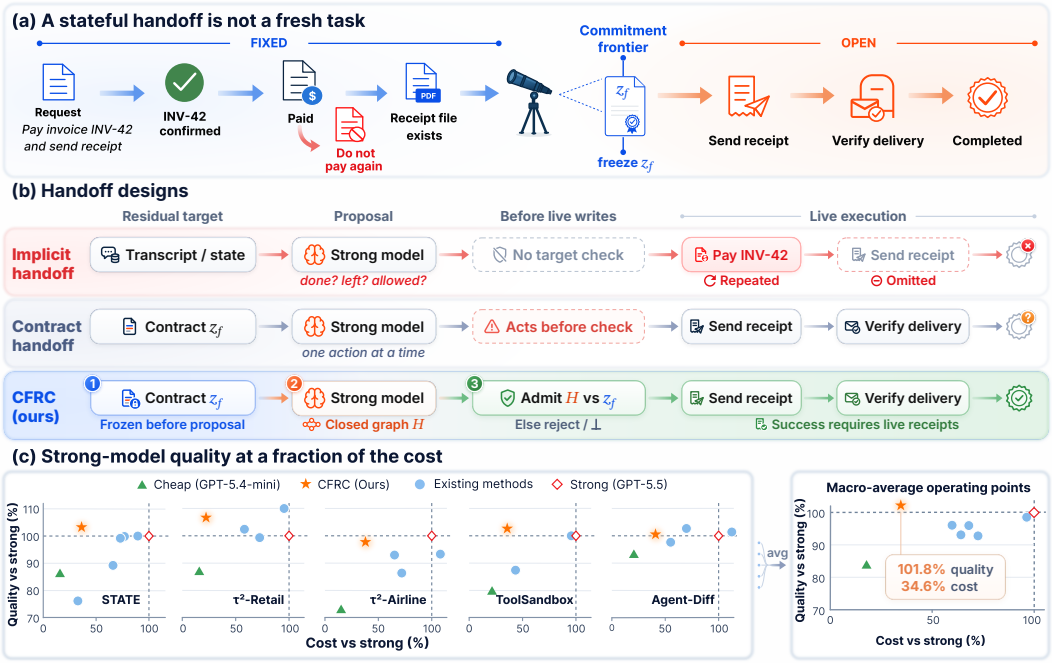

Figure 1: **Stateful handoff is residual completion, not restart.** (a) At a commitment frontier, accepted progress is fixed while request-scoped work remains open: INV-42 has been confirmed and paid, so the successor must preserve that effect, deliver its receipt, and verify delivery rather than repay or restart. (b) Implicit handoff can miss open work or repeat completed effects; a contract-only handoff supplies a residual target but still lets the successor act before the whole remainder is checked. CFRC instead freezes the target before proposal, closes the successor's full remainder into an evidence-linked graph before native authority, and accepts success only from live evidence. (c) For GPT-5.4-mini/GPT-5.5, quality and cost are normalized to the strong full-task agent on each surface $(100\%, 100\%)$. Across five surfaces, CFRC remains near strong-model quality and reaches 101.8% of strong macro accuracy at a 34.6% mean cost ratio.

(Yao et al., 2023), but a proposal cannot reveal what it omits. A continuation containing only a status check and success response provides no independent signal that receipt delivery remains missing. Third, action-level guards and control layers can reject unsafe or invalid proposed calls (Wang et al., 2026; Kamath et al., 2025; Liu et al., 2026b), such as duplicate payments or unauthorized invoices. Yet they cannot require an action that was never proposed, nor establish whole-suffix completeness before incremental native execution begins. Thus, existing mechanisms address context transfer, action proposal, or local validity, but not the joint need for an independent residual target, complete proposal coverage, and evidence-backed realization.

Our approach. We address this gap with Commitment-Frontier Residual Completion (CFRC). The method enforces target before proposal, whole proposal before authority, and live evidence before success, as illustrated in Fig. 1b. First, Reached-State Contract Construction in CFRC derives a frozen residual contract z_f from the accepted trajectory, reached state, trusted receipts, and public tool interfaces before seeing the successor's proposal. The contract records what is fixed and what remains open, so omitted work cannot disappear merely because the successor fails to mention it. Second, Executable Residual Completion in CFRC lets the successor author on an isolated replica where available and closes the full continuation as an evidence-linked graph H, exposing its actions, dependencies, and obligation coverage without changing the live state. Finally, Checked Execution in CFRC admits the complete H against z_f before granting native authority, rebinds dynamic values to live predecessor receipts, and accepts completion only when matching native evidence discharges every obligation. In the invoice example, z_f preserves the completed payment while recording pending delivery; the graph H must explicitly include that delivery path. Checked Execution rejects duplicate or incomplete proposals before live execution and confirms success only after live receipts verify delivery. Finally, we theoretically establish conditional partial correctness, showing that successfully realized prefixes preserve encoded progress and returned suffixes satisfy the contract.

Across five stateful tool-use surfaces, CFRC shifts the quality–cost frontier (Fig. 1c). With GPT-5.4-mini/GPT-5.5, it achieves 70.8% macro accuracy versus 69.6% for GPT-5.5 at 34.6% mean cost; with Luna/Sol, it achieves 70.8% versus 70.0% at 22.0% mean cost. Cross-provider Luna/Sonnet 5 and Luna/Gemini 3.7 Flash remain within 0.5 points of their strong anchors at 26.0% and 44.1% cost. Checked Execution blocks invalid proposals before live writes, while an independent audit recovers 62/68 persistent write obligations and finds no ungrounded writes among 50 admitted proposals. So CFRC preserves strong-model quality while focusing expensive reasoning on unfinished work.

2 Related Work

Model routing & agent handoff. Planning and reflection improve how agents generate actions (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2024). Routers and cascades select models by predicted difficulty or cost (Ong et al., 2025; Bouchard, 2026; Son et al., 2026; Wei et al., 2026; Liu et al., 2026a), while handoff frameworks quantify information loss across checkpoints (KC & Budathoki, 2026). Yet a transferred transcript does not specify which effects are fixed or which obligations remain open. CFRC constructs this residual target before the successor proposes a continuation.

Classical specifications & recovery. Conditional correctness relates execution to explicit pre- and postconditions (Hoare, 1969). Sagas structure long-lived transactions around completion or compensation (Garcia-Molina & Salem, 1987), while plan repair adapts existing plans to changed execution contexts (Fox et al., 2006). CFRC follows this separation of specification and execution, but constructs the residual target from accepted model-generated progress to constrain the successor's entire proposal. Accepted effects are preserved, while a failed suffix may return $\perp$ without compensation.

Stateful control & verification. Runtime guards, capabilities, and transactional frameworks constrain actions outside the model (Wang et al., 2026; Kamath et al., 2025; Debenedetti et al., 2025; Fan et al., 2026; Liu et al., 2026b; Lin et al., 2026; Patil et al., 2024; Chen et al., 2026). Execution-monitoring limits are classical (Schneider, 2000). Our omission argument instead concerns verifying planned coverage *before live writes*: an incomplete prefix cannot reveal which obligations a future continuation will omit, although explicit terminal obligations can still detect false completion. CFRC checks the entire proposed remainder against a separately constructed target and requires live evidence for success (Appendix A.3). Stateful benchmarks (Yao et al., 2024; Barres et al., 2025; Lu et al., 2025; Trivedi et al., 2024; Microsoft, 2026; Pysklo et al., 2026) make the gap between attempted actions and supported completion measurable (Advani, 2026; Gao & Zhou, 2026).

3 Methodology

3.1 Stateful Handoff as a Residual-Completion Problem

Problem setup. A stateful handoff occurs when one model takes over an ongoing tool-use trajectory produced by another, as in routed or cascaded agents for coding, web navigation, and other applications with persistent state or externally visible effects. Unlike a fresh request, the successor inherits both *fixed progress* that must be preserved and *open work* that remains to be completed. We study how to finish this residual work without replacing, undoing, or duplicating accepted progress.

Formally, let q be the original request, τ_t the accepted trajectory prefix, s_t the reached live state, and $\mathcal{R}_t$ the trusted native receipts available at s_t. Let $\mathcal{A}_{\text{env}}$ denote the public tool interfaces, and let $\mathcal{O}_t$ denote the semantic, request-scoped obligation instances still open under q and τ_t. We use B_t for accepted choice bindings, including user-confirmed choices that may precede any native effect, and K_t for realized effects supported by $\mathcal{R}_t$. With fixed progress $F_t = (B_t, K_t)$, residual completion seeks a continuation σ from s_t satisfying

$$\text{Exec}(\sigma; s_t) \ \wedge \ \text{Preserve}(\sigma; F_t) \ \wedge \ \text{Complete}(\sigma; \mathcal{O}_t). \tag{1}$$

Here, the three predicates require the suffix to execute from the reached state, respect accepted bindings and realized effects, and discharge every open obligation before claiming success. If CFRC cannot establish such a suffix within budget, it returns $\perp$; this does not imply that no valid continuation exists. For the formal guarantee, each in-scope obligation has a request-defined, finitely checkable predicate over the resulting trace and state, independent of the contract later constructed by CFRC. Let $\Sigma_f(q, \tau_t)$ denote the continuations satisfying Eq. (1). Appendix E gives the continuation-space semantics and the precise scope of these predicates.

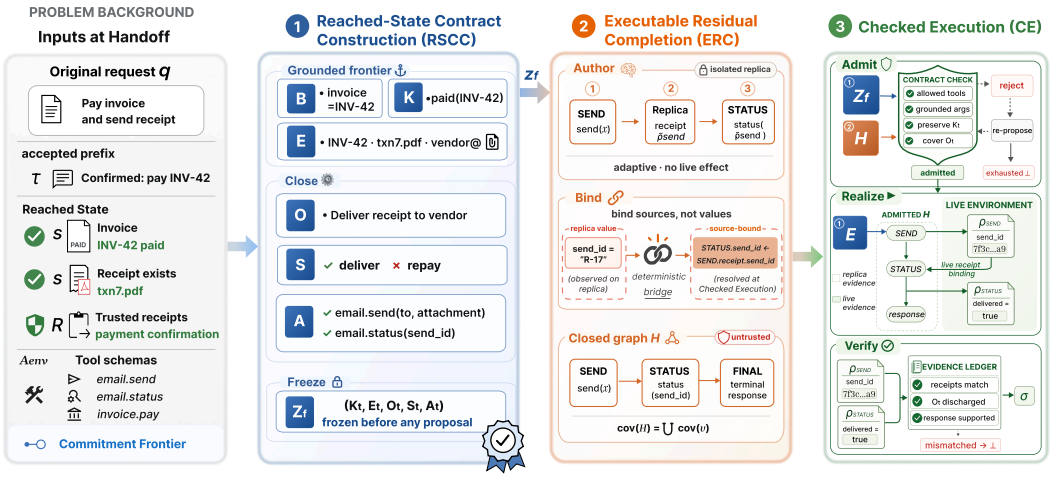

Figure 2: **CFRC at a post-payment commitment frontier.** RSCC identifies a supported frontier and freezes the residual contract z_f before querying the successor. ERC then closes the proposed remainder as an evidence-linked graph H, and CE admits and realizes H, returning σ only when matching live evidence discharges every encoded open obligation, and $\perp$ otherwise.

Running example and commitment frontier. Consider the post-payment invoice handoff in Fig. 2. The request q asks the agent to pay one outstanding invoice and send its receipt; τ_t records that `INV-42` was selected and confirmed and that its payment succeeded; s_t contains the paid invoice and receipt file `txn7.pdf`; and $\mathcal{R}_t$ contains the trusted native evidence. Thus, B_t fixes `invoice` $=$ `INV-42`, K_t contains the receipt-backed effect paid(`INV-42`), and $\mathcal{O}_t$ still requires delivery of the receipt to the verified recipient. A valid σ may therefore *send receipt* $\rightarrow$ *verify delivery* $\rightarrow$ *confirm success*: repaying violates preservation, omitting delivery violates completion, and unsupported success violates Eq. (1). The same issue can arise before any change to the environment: once the user confirms `INV-42`, that binding already constrains future behavior. We call such a point a *commitment frontier*: the accepted prefix has fixed progress that must be preserved while request-scoped work remains open—in short, *fixed progress plus unfinished work*. Before any accepted choice or realized effect, the task is fresh; after all obligations are complete, no residual task remains.

From challenges to CFRC. Eq. (1) exposes three design challenges. **(C1) What is fixed, and what remains?** Neither the request nor the reached state alone necessarily specifies the residual task, while letting the successor define its own target makes omissions difficult to detect. CFRC therefore constructs an independent frozen contract z_f before proposal (Sec. 3.2). **(C2) Does the proposed remainder cover that target?** Call-local checks can reject an invalid action but cannot detect an obligation for which no action is proposed. In stateful environments with irreversible tool mutations, step-by-step execution risks leaving the environment in an unrecoverable state before discovering that downstream obligations lack valid inputs. CFRC therefore exposes the whole remainder as a closed graph H before native execution (Sec. 3.3). **(C3) Were the proposed actions actually realized?** Replica results are only proposal-time evidence, and native calls may fail. CFRC therefore grants native authority only after validation and accepts completion only from matching live evidence (Sec. 3.4). These three stages form the pipeline in Fig. 2:

$$(q, \tau_t, s_t, \mathcal{R}_t, \mathcal{A}_{\text{env}}) \longrightarrow z_f \longrightarrow H \longrightarrow \sigma \text{ or } \perp. \tag{2}$$

We study one handoff tenure at a time. CFRC triggers handoff through a fixed supported-frontier rule, but does not optimize general model routing or choose among multiple successors; it also preserves supported prior commitments rather than re-evaluating their desirability.

3.2 Reached-State Contract Construction

Reached-State Contract Construction (RSCC) addresses C1 in Sec. 3.1 by constructing the residual target *before* the successor proposes anything. Rather than merely summarize the trajectory, RSCC converts accepted history into a proposal-independent specification of what is fixed and what re-

mains owed. It **grounds** accepted progress, **closes** the remaining request, and **freezes** the resulting contract z_f. RSCC uses only $q, \tau_t, s_t, \mathcal{R}_t$ and a fixed adapter over public tool schemas. The adapter maps public interfaces into shared argument, effect, and evidence types; it cannot access simulator internals, reference outcomes, or evaluator rewards.

Step 1: Grounding fixed progress. RSCC first identifies the binding parts of the accepted history:

$$(B_t, K_t, E_t) = \mathrm{Ground}(\tau_t, s_t, \mathcal{R}_t),$$

where B_t contains accepted choice bindings, K_t contains receipt-backed realized effects that later actions must preserve, and E_t contains verified entities, values, files, and receipt fields that the residual continuation may safely reference. Grounding is deterministic: public tool schemas specify mutating effects and receipt formats, while Ground parses trusted native receipts in $\mathcal{R}_t$ to populate K_t and extracts confirmed entities into E_t and B_t. In our invoice example, E_t retains `txn7.pdf` and the verified recipient, while K_t records the completed payment.

Step 2: Closing the residual task. Grounding identifies what is fixed but not what remains. RSCC compares the grounded progress with the original request q and the public interfaces to derive

$$(\tau_t, s_t, \mathcal{R}_t) \xrightarrow{\mathrm{Ground}} (B_t, K_t, E_t) \xrightarrow[\mathcal{A}_{\mathrm{env}}]{q,\mathrm{Close}} z_f = (K_t, E_t, O_t, \mathcal{S}_t, \mathcal{A}_t). \tag{3}$$

Here, discrete $O_t = \mathrm{canon}_q(\mathcal{O}_t)$ encodes semantic obligations $\mathcal{O}_t$; $\mathcal{S}_t$ specifies admissible residual effects, and $\mathcal{A}_t$ the interfaces that may realize or verify them. Closure operationalizes the division of labor between rules and model inference. On structured surfaces (STATE-Bench, ToolSandbox, Agent-Diff), candidate obligations are extracted deterministically by matching request intents against tool schemas and excluding effects already certified in K_t (walked through in detail in Appendix Table 13). On conversational surfaces (τ^2), an extraction prompt decomposes user request q relative to prefix τ_t into candidate obligations O_t, parameterizing them with entities from E_t and restricting them to admissible interfaces $\mathcal{A}_t$. For the post-payment invoice, O_t encodes delivery of `txn7.pdf` to the verified recipient, $\mathcal{S}_t$ permits delivery and receipt-backed confirmation while excluding repayment, and $\mathcal{A}_t$ retains the corresponding send and status interfaces. Accepted bindings in B_t are compiled into fixed values in E_t, obligations in O_t, and restrictions in $\mathcal{S}_t$, while receipt-backed effects in K_t remain protected against reversal or repetition.

Step 3: Validating and freezing the contract. RSCC requires every contract field to have trusted support, O_t to be nonempty, and the accepted prefix to contain at least one supported accepted binding or receipt-backed realized effect. Once validated, z_f is frozen *before* querying the successor: neither proposal nor execution may expand, weaken, or rewrite it, so recorded work cannot disappear because the successor omits it. These checks establish support but do not prove that RSCC captured every semantic obligation in $\mathcal{O}_t$. Freezing prevents *proposal-induced* omission; it cannot recover an obligation missed during construction. Accordingly, the lift from contract-relative correctness to the original residual request is conditional on complete RSCC construction (Sec. 3.5). If construction or validation fails, CFRC withholds successor write authority and returns $\perp$.

3.3 Executable Residual Completion

The frozen contract z_f specifies *what* must be preserved and completed while leaving *how* to the successor. ERC addresses C2 in Sec. 3.1 by exposing the successor's complete proposed remainder before any new live write: the successor **authors** a continuation, a deterministic bridge **binds** its data and evidence dependencies, and the proposal is **closed** as an untrusted graph H.

Step 1: Data-dependent authoring without live effects. Residual plans are often data-dependent because later call arguments depend on earlier outputs. When safe snapshotting is available, the successor authors against an isolated replica of s_t, observing replica receipts $\widetilde{\rho}_v$ without changing the live environment. On an unforkable surface, the successor instead closes the graph from handoff-time evidence and symbolic receipt dependencies; no new state-changing live call is authorized before CE in Sec. 3.4 admits the complete proposal. Replica receipts may expose intermediate values for authoring, but they neither prove live effects nor discharge obligations. For example, the successor may send `txn7.pdf` on a replica, observe a SEND receipt, and use its identifier to propose a later delivery-status query without sending a real email.

Step 2: Binding dependencies and closing the residual graph. A proposal authored from replica values cannot simply be replayed because replica-generated identifiers may differ from live ones.

A deterministic bridge therefore builds an evidence-linked candidate graph. Let H denote the closed DAG retained after dependency-closed candidate selection, containing typed calls, the terminal response, arguments, dependencies, source bindings, obligation coverage, effects, and required authority.

Crucially, dynamic arguments bind to their *source*, not to a proposal-time value. For example, the status query refers to the `send_id` field of the preceding SEND receipt; CE (Sec. 3.4) later resolves that field from the corresponding live receipt. The successor chooses the actions, task-specific arguments, semantic order, and terminal response. The bridge only records or normalizes structure supported by z_f, public schemas, and available receipts; it does not choose another task solution.

Step 3: Closing the whole remainder without repair. Unlike next-action control, H exposes the *entire* proposed remainder. Writing $\mathrm{cov}(v)$ for the obligations claimed by node v,

$$\mathrm{cov}(H) = \bigcup\nolimits_{v \in V(H)} \mathrm{cov}(v), \tag{4}$$

can later be compared with O_t. Thus an omitted action becomes visible as missing coverage: an invoice proposal without a SEND node leaves receipt delivery uncovered.

Crucially, while a terminal monitor can detect omission at episode end, it cannot roll back irreversible tool mutations already committed to the live environment. Nor can step-by-step checkers observe whether future actions will complete open obligations (Lemma 5). Closing H beforehand validates full obligation coverage and dataflow reachability *before* granting live write authority, preventing destructive partial execution (formalized in Appendix E.4). The bridge does not repair the proposal; it normalizes supported dependencies, source bindings, or coverage annotations without adding, removing, or replacing task actions. The resulting H is inspectable but untrusted.

3.4 Checked Execution

The closed H from Sec. 3.3 makes the successor's proposal explicit but does not establish that it is contract-valid or that its claimed effects actually occur. Checked Execution (CE) resolves C3 by putting *state-changing live authority* behind whole-graph admission and accepting completion only from live evidence. CE first **admits** H against z_f, then **realizes** admitted nodes using live receipt bindings, and finally **verifies** that the resulting evidence discharges every obligation.

Step 1: Whole-graph admission before live write authority. Before the first state-changing native call, CE compares the fixed proposal H with $z_f = (K_t, E_t, O_t, \mathcal{S}_t, \mathcal{A}_t)$. Admission requires: (i) calls use allowed interfaces with well-typed, grounded arguments; (ii) proposed effects stay within $\mathcal{S}_t$ and preserve K_t and the accepted bindings encoded by the contract; (iii) the entire graph covers the encoded obligations, $\mathrm{cov}(H) = O_t$; and (iv) dependencies and terminal claims have explicit evidence paths. These conditions correspond to proposal-level executability, preservation, and coverage. Because CE sees the complete H, it can reject duplicate actions, uncovered obligations, unresolved dependencies, or unsupported terminal claims before the first live write. Whole-graph admission establishes that the *planned* suffix satisfies the contract structure; it does not imply that native execution will succeed. CE admits or rejects the selected H as a whole. Compiler selection may prune proposal nodes only while retaining dependencies and full coverage of O_t; it cannot synthesize repair actions (Appendix E.4).

Step 2: Receipt-bound live realization. An admitted graph executes in dependency order on the live environment. Static inputs resolve from E_t, whereas dynamic inputs are rebound to the corresponding *live* predecessor receipts specified by H. After each call, CE checks the returned receipt against the result and effect instance recorded for that node before its fields may feed downstream calls. Replica and proposal-time outputs never substitute for native evidence.

Step 3: Evidence-backed completion. Replica success never discharges an obligation. Let s_j and $\mathcal{R}_j$ denote the live state and matching native receipts after j successfully realized action nodes. CE tracks the obligations whose evidence remains valid at that point:

$$L_j = \{\, o \in O_t : \mathrm{wit}_j(o, \mathcal{R}_j, s_j) \,\}, \qquad L_0 = \varnothing, \tag{5}$$

where wit_j requires instance-matching native evidence whose discharge remains valid at s_j. Event obligations may remain discharged once witnessed; mutable-state obligations require current-state validity. Recorded invalidation removes discharge and blocks completion for the current tenure; a fresh receipt cannot discharge that obligation again under the same contract.

If m action nodes are realized, CE returns a suffix σ only when every node has a matching receipt, $L_m = O_t$, and the terminal response is supported by the corresponding evidence:

$$\Gamma_{z_f}(H) = \begin{cases} \sigma, & \text{if admission and all live-evidence checks succeed,} \\ \bot, & \text{otherwise.} \end{cases} \tag{6}$$

A rejection before live execution may trigger another complete proposal against the same frozen contract within budget. Once a state-changing live call has occurred, however, a failure or mismatched receipt ends the current tenure; CE does not repair the old graph or continue under stale contract evidence. A later attempt reconstructs a contract from the new reached state. Importantly, $\bot$ after a live call does *not* imply rollback or that the environment remained unchanged.

3.5 Conditional End-to-End Guarantee

The three stages establish complementary properties: Under complete construction, RSCC faithfully specifies the residual task; ERC exposes the successor's proposal without repairing it, and CE faithfully realizes an admitted proposal using live evidence.

Proposition 1 (Contract-relative partial correctness and residual-task lift)**.** *Let P be a successor-authored residual proposal and $H = \mathrm{ERC}_{z_f}(P)$ its closed graph. Under faithful ERC closure, sound schemas and checks, valid live evidence, and tenure isolation (A2–A5 in Appendix E.6), if $\Gamma_{z_f}(H) = \sigma \neq \bot$, then every successfully realized prefix preserves the frozen progress and*

$$\mathrm{cov}(H) = O_t, \qquad \sigma \in \Sigma(z_f). \tag{7}$$

If RSCC satisfies the complete-construction assumption A1, then $\sigma \in \Sigma_f(q, \tau_t)$ and satisfies Eq. (1).

Proposition 1 separates execution fidelity from specification fidelity. Under A2–A5, ERC and CE ensure that the returned suffix satisfies the frozen contract: the closed graph covers every encoded obligation, each successfully realized prefix preserves encoded progress, and live evidence supports completion. Under A1 in Appendix E.6, complete RSCC construction lifts this contract-relative result to the original in-scope residual request. The returned suffix therefore executes from the reached state, preserves accepted progress, and completes every in-scope semantic obligation still open at handoff. The result does not certify the constructor, guarantee that a valid proposal will be found, or imply rollback when $\bot$ is returned. Formal definitions and proofs are in Appendix E.6.

4 Experiments

Our evaluation investigates four questions: whether CFRC approaches strong-model quality at lower inference cost, whether these gains hold across model pairs and providers, what RSCC, ERC, and CE individually contribute, and whether independent audits validate the contract and execution mechanisms.

Surfaces and protocol. We evaluate across five stateful tool-use benchmarks, comprising STATE-Bench, τ^2-Retail, τ^2-Airline, ToolSandbox, and Agent-Diff (Microsoft, 2026; Barres et al., 2025; Lu et al., 2025; Pysklo et al., 2026). All methods use the same frozen tasks, public tool environment, and scorer. CFRC accesses only public action schemas, effect types, and native receipts, without simulator internals, reference outputs, or evaluator rewards. Full dataset and adapter details are in Appendix A.

Models and baselines. Our primary evaluations use GPT-5.4-mini/GPT-5.5 and GPT-5.6 Luna/Sol; three cross-provider pairs test transfer. Baselines include cheap and strong full-task agents, StepWise, MTRouter, RouteLLM-BERT, FrugalGPT, and AgServe-QAC (Wei et al., 2026; Zhang et al., 2026; Ong et al., 2025; Chen et al., 2023; Ren et al., 2025). Policies and operating points are fixed before test evaluation. Calibration and surface coverage are detailed in Appendix A.3.

Metrics and cost. We report official task success except for ToolSandbox, whose metric is strict all-milestone completion; official state similarity is reported in Appendix B.3. Cost is cache-neutral provider expenditure over all agent calls, excluding user-simulator and evaluator roles. Aggregates are computed over each method's evaluated surfaces. App. A.4 gives full accounting & statistics.

4.1 End-to-End Quality and Cost

Macro quality vs. strong anchors. In Table 1, CFRC matches strong anchors within 1.2 points on macro average under GPT-5.4-mini/GPT-5.5 at 34.6% mean cost, and within 0.9 points under

Table 1: End-to-end results. Scores are task success (%), except that ToolSandbox uses strict all-milestone completion. Costs are cache-neutral agent-side USD. Gap is the macro-score difference from the corresponding strong full-task anchor, and Cost (%) is the mean per-surface ratio to that anchor marked with "ref". Aggregates are computed from unrounded values. For † baselines, both aggregates use evaluated surfaces only. Anchors and CFRC report five-run means ($N = 5$ complete runs per surface; SDs in Table 9); baselines report frozen full-denominator evaluations (Appendix A.3).

	STATE-Bench		τ^2-Retail		τ^2-Airline		ToolSandbox		Agent-Diff			
Method	Score	Cost	Score	Cost	Score	Cost	Score	Cost	Score	Cost	Gap	Cost (%)
Escalation, routing, and cascade baselines, GPT-5.4-mini/GPT-5.5												
StepWise†	54.7	22.59	–	–	–	–	–	–	95.6	21.73	−2.2	67.9
AgServe-QAC†	46.7	11.21	82.5	13.37	–	–	–	–	–	–	−3.6	64.1
MTRouter†	60.8	24.97	–	–	76.0	12.73	–	–	–	–	−6.3	72.4
RouteLLM-BERT	61.2	26.29	77.0	8.16	82.0	11.53	26.4	4.68	91.1	17.13	−2.0	59.7
FrugalGPT	61.3	30.62	74.5	10.16	82.0	19.13	30.2	10.31	93.3	36.06	−1.3	96.3
Full-task anchors and CFRC, *same provider*												
GPT-5.4-mini full	52.7	5.42	65.0	2.23	64.0	2.68	24.0	2.31	86.7	6.42	−11.1	17.8
GPT-5.5 full	61.3	34.23	75.0	14.02	88.0	17.70	30.2	10.78	93.3	31.11	ref	100
CFRC, GPT-5.4-mini/GPT-5.5	**63.3**	12.41	**80.0**	3.12	86.0	6.71	**31.0**	3.84	**93.8**	12.72	+1.2	**34.6**
GPT-5.6 Luna full	54.0	1.18	67.5	0.50	72.0	0.67	26.4	0.32	62.2	0.70	−13.5	5.1
GPT-5.6 Sol full	60.7	18.44	80.0	10.53	90.0	13.57	30.2	5.76	88.9	17.69	ref	100
CFRC, Luna/Sol	60.7	4.75	76.5	0.98	86.8	3.17	**33.3**	1.29	**96.9**	5.12	+0.9	**22.0**
Full-task anchors and CFRC, *cross provider*												
Claude Sonnet 5 full	62.7	10.56	80.0	11.04	86.0	14.48	36.4	5.48	82.2	11.20	ref	100
CFRC, Luna/Sonnet 5	62.7	3.00	**84.0**	1.97	80.0	2.88	31.0	1.69	**91.6**	3.68	+0.4	**26.0**
Gemini 3.7 Flash full	68.0	6.93	85.0	2.74	90.0	3.48	33.3	1.03	77.8	9.17	ref	100
CFRC, Luna/Gemini 3.7 Flash	64.7	4.20	82.5	0.55	84.0	1.24	**34.1**	0.74	**91.1**	2.91	+0.5	**44.1**
Claude Haiku 4.5 full	53.3	8.50	65.0	5.49	54.0	6.34	34.1	2.23	33.3	2.50	−22.0	39.6
GPT-5.6 Sol full	60.7	18.44	80.0	10.53	90.0	13.57	30.2	5.76	88.9	17.69	ref	100
CFRC, Haiku 4.5/Sol	58.7	14.67	70.0	4.51	84.0	7.87	**36.4**	3.56	75.6	7.91	−5.0	**57.4**

Table 2: Matched-frontier progression from Direct Takeover to full CFRC across five surfaces (GPT-5.4-mini/GPT-5.5, five-run means, $N = 5$ complete runs per cell; unengaged episodes retain prefix outcomes). Certified Direct adds contracts to native continuation, Contract Plan adds ERC authoring, and full CFRC adds Checked Execution. Cells: score (%) / spend (USD).

	Components			STATE-	τ^2		Tool-	Agent-	Macro /
Arm	RSCC	ERC	CE	Bench	Retail	Airline	Sandbox	Diff	Total
Direct Takeover				53.3 / $22.97	67.5 / $11.64	67.2 / $15.26	25.3 / $4.72	81.3 / $14.23	58.9 / $68.82
Certified Direct	✓			59.7 / $16.81	70.0 / $5.35	69.2 / $6.13	28.7 / $3.97	82.7 / $14.80	62.1 / $47.06
Contract Plan	✓	✓		62.0 / $16.39	69.5 / $2.95	74.4 / $5.50	29.8 / $3.95	91.6 / $16.46	65.4 / $45.25
Full CFRC	✓	✓	✓	**63.3** / $12.41	**80.0** / $3.12	**86.0** / $6.71	**31.0** / $3.84	**93.8** / $12.72	**70.8 / $38.80**

Luna/Sol at 22.0% cost. GPT-5.4-mini/GPT-5.5 matches or exceeds strong anchors on four surfaces, trailing by 2.0 points on Airline; Luna/Sol ties or exceeds on three surfaces, trailing by 3.5 and 3.2 points on Retail and Airline. Across five runs, CFRC macro standard deviations are 0.54 and 0.58 points (Table 9), reflecting comparable macro quality rather than uniform dominance.

Comparison with routing and cascades. RouteLLM-BERT and FrugalGPT trail GPT-5.5 by 2.0 and 1.3 points at 59.7% and 96.3% cost, while CFRC gains 1.2 points at 34.6% cost. On StepWise, AgServe-QAC, and MTRouter subsets, CFRC improves accuracy by 3.4, 7.1, and 6.3 points while cutting cost by 37–49%, showing the benefit of preserving progress over simple call reduction.

Cross-provider transfer. Luna/Sonnet 5 and Luna/Gemini 3.7 Flash match strong anchors within 0.5 points macro completion at 26.0% and 44.1% cost across five benchmarks, though ToolSandbox similarity softens under Gemini (Appendix B.3). Haiku 4.5/Sol trails Sol by 5.0 points at 57.4% cost, indicating that downstream completion remains sensitive to cheap-prefix fidelity, though CFRC gains 17.0 points over the standalone cheap anchor.

4.2 Validating the Three Stages

Stage-wise progression across components. Table 2 isolates post-frontier mechanisms from captured state s_t. Direct Takeover achieves 58.9% accuracy at $68.82. Certified Direct supplies frozen contracts via RSCC but retains incremental action checking, reaching 62.1% at $47.06. Contract Plan adds ERC authoring to reach 65.4%, and Full CFRC adds Checked Execution to reach 70.8%

Table 3: Inference cost decomposition of CFRC across five surfaces (GPT-5.4-mini/GPT-5.5). Spend partitions into cheap prefix, contract construction (RSCC), and residual execution (ERC+CE); totals match Tables 1 and 2.

Surface	Total ($)	Cheap Prefix ($)	RSCC ($)	ERC+CE ($)	Engaged	Residual Share
STATE-Bench	12.41	5.22	0.00	7.19	84.7%	57.9%
τ^2-Retail	3.12	1.51	0.69	0.92	20.0%	29.6%
τ^2-Airline	6.71	2.17	0.34	4.20	80.0%	62.7%
ToolSandbox	3.84	2.16	0.00	1.68	20.2%	43.6%
Agent-Diff	12.72	4.33	0.00	8.39	93.3%	65.9%
Mean / Macro	**7.76**	3.08	0.21	4.48	59.6%	**51.9%**

Table 4: Key contract and execution audits. Rows 1–5 evaluate grounding against ground truth and receipts; rows 6–7 evaluate CE authority and replay from execution logs (Appendix C.4).

Audit Property	Evaluated Cohort	Rate	Fraction
Contract grounding and obligation recall (ground truth & receipts)			
Accepted-binding correctness	Confirmed prefix bindings	100.0%	50 / 50
Persistent write obligation recall	Ground-truth write effects	91.2%	62 / 68
Ungrounded live-write false accept	Admitted candidate proposals	0.0%	0 / 50
Unnecessary CE block	Rejected proposals	4.7%	2 / 43
Omission-related false completion	Evaluated episodes	4.0%	2 / 50
Checked Execution authority boundary and replay (execution logs)			
Intervention before first live write	All CE interventions	100.0%	124 / 124
Rejection replay consistency	Active predicate rejections	100.0%	43 / 43

at $38.80. Comparing Full CFRC against Certified Direct shows that gating writes behind whole-graph admission adds 8.8 points and saves $8.26 over step-by-step checking under the same contract by preventing destructive partial execution.

Where cost is spent. Table 3 decomposes the $38.80 spend, where RSCC averages $0.21 across surfaces. Three call mechanisms drive the 43.6% savings over Direct Takeover. Selective invocation restricts strong calls to post-frontier residuals, engaging on 59.6% of episodes. RSCC eliminates exploratory re-querying with grounded obligations, and CE fail-closed checks halt ungrounded plans early to prevent runaway retries.

4.3 Auditing Contracts and Checked Execution

Contract adequacy. Table 4 grounds all 50 audited bindings and recovers 62 of 68 persistent write obligations, achieving 91.2% recall. No ungrounded live writes were admitted in the audited sample of 50 candidate proposals, corresponding to an exact Clopper–Pearson 95% upper bound of 5.8%. Six omissions involved implicit updates without entity mentions; four were blocked downstream and two yielded false completions. Formal lift to the residual objective remains conditional on complete construction because conversational disclosures fall outside tool schemas.

Checked Execution as an authority boundary. Table 4 shows that all 124 CE interventions occur before the first live write. These comprise 43 active proposal rejections, spanning 25 coverage, 10 dependency, and 8 grounding failures, alongside 81 holds. All 43 rejections reproduce under deterministic replay (Appendix C.4).

5 Conclusion

We formalize stateful agent handoff as commitment-constrained residual completion and introduce CFRC, freezing residual contracts from accepted history, authoring closed graphs on replicas, and verifying execution via live receipts. Across five surfaces and model pairings, CFRC matches strong full-task quality at a fraction of inference cost while blocking unadmitted mutations.

Scope and limitations. We evaluate single-tenure cheap-to-strong handoffs under frozen contracts. While multi-tier cascades readily compose with CFRC, recursive routing policies lie beyond our immediate scope. Formal lift to the residual task remains conditional on complete RSCC construction, as conversational disclosures fall outside tool-effect schemas.

Reproducibility statement. Appendices A–E detail environments, accounting, replications, and proofs; Table 9 reports variation across runs. Code and schemas will be open-sourced.

AI use statement. In this work, we used generative AI tools, including coding agents, to assist with implementing the proposed method, baselines, and experimental pipeline; to provide feedback on and suggest refinements to the method; to assist with parts of the mathematical derivations; and to edit and polish parts of the manuscript. We did not use generative AI tools for research ideation, problem formulation, hypothesis formulation, experimental design, dataset processing, or interpretation of experimental results. All AI-assisted code was reviewed and tested by the authors, all AI-assisted derivations were independently checked by the authors, and all reported results and citations were independently verified. The authors take full responsibility for the final content of this work, including AI-assisted text, code, claims, and artifacts.

REFERENCES

Laksh Advani. From confident closing to silent failure: Characterizing false success in LLM agents. *arXiv preprint arXiv:2606.09863*, 2026.

Anthropic. Claude platform pricing. https://platform.claude.com/docs/en/about-claude/pricing, 2026. Accessed 2026-08-20.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Dylan Bouchard. Is escalation worth it? A decision-theoretic characterization of LLM cascades. *arXiv preprint arXiv:2605.06350*, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using LLM agents. *arXiv preprint arXiv:2606.17573*, 2026.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

Linfeng Fan, Ziwei Li, Yuan Tian, Yichen Wang, Rongsheng Li, and Xiong Wang. The granularity mismatch in agent security: Argument-level provenance solves enforcement and isolates the LLM reasoning bottleneck. *arXiv preprint arXiv:2605.11039*, 2026.

Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 212–221, 2006.

Shanshan Gao and Liyi Zhou. Can agent benchmarks support their scores? Evidence-supported bounds for interactive-agent evaluation. *arXiv preprint arXiv:2605.10448*, 2026.

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

Google. Gemini developer API pricing. https://ai.google.dev/gemini-api/docs/pricing, 2026. Accessed 2026-08-20.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12 (10):576–580, 1969. doi: 10.1145/363235.363259.

Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. Enforcing temporal constraints for LLM agents. *arXiv preprint arXiv:2512.23738*, 2025.

Dipesh KC and Anjila Budathoki. Handoff debt: The rediscovery cost when coding agents take over interrupted tasks. *arXiv preprint arXiv:2606.02875*, 2026.

Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. VIGIL: Defending LLM agents against tool stream injection via verify-before-commit. *arXiv preprint arXiv:2601.05755*, 2026.

Xinchen Liu, Hang Zhou, Yingjie Zong, Yuchuan Tian, Liuyang Song, Shuo Zhang, Yulong Li, Wei He, Mengyu Zheng, Runke Liu, Siyang Cheng, Xiang Kuang, Hailin Hu, Kai Han, and Yunhe Wang. Agentic routing: The harness-native data flywheel. *arXiv preprint arXiv:2607.11399*, 2026a.

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. *arXiv preprint arXiv:2601.04688*, 2026b.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Microsoft. STATE-Bench: Benchmark AI agents on enterprise workflows. https://github.com/microsoft/STATE-Bench, 2026. Accessed 2026-08-31.

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 *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5503a7c69d48a2f86fc00b3dc09de686-Paper-Conference.pdf.

OpenAI. Compare models. https://developers.openai.com/api/docs/models/compare, 2026a. Accessed 2026-08-28.

OpenAI. GPT-5.6 Sol model. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026b. Accessed 2026-08-28.

Shishir G. Patil, Tianjun Zhang, Vivian Fang, Noppapon C., Roy Huang, Aaron Hao, Martin Casado, Joseph E. Gonzalez, Raluca Ada Popa, and Ion Stoica. GoEX: Perspectives and designs towards a runtime for autonomous LLM applications. *arXiv preprint arXiv:2404.06921*, 2024. URL https://arxiv.org/abs/2404.06921.

Hubert M. Pysklo, Artem Zhuravel, and Patrick D. Watson. Agent-Diff: Benchmarking LLM agents on enterprise API tasks via code execution with state-diff-based evaluation. *arXiv preprint arXiv:2602.11224*, 2026.

Yanyu Ren, Li Chen, Dan Li, Xizheng Wang, Zhiyuan Wu, Yukai Miao, and Yu Bai. Transcending cost-quality tradeoff in agent serving via session-awareness. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Fred B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000. doi: 10.1145/353323.353382.

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, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Seongho Son, Sangwoong Yoon, Jiahua Tang, Shuhan Wang, Lorenz Wolf, and Ilija Bogunovic. SWE-Router: Routing in multi-turn agentic software engineering tasks. *arXiv preprint arXiv:2607.00053*, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

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

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Jinbiao Wei, Kangqi Ni, Yilun Zhao, Guo Gan, and Arman Cohan. Step-level optimization for efficient computer-use agents. *arXiv preprint arXiv:2604.27151*, 2026.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yiqun Zhang, Hao Li, Zihan Wang, Shi Feng, Xiaocui Yang, Daling Wang, Bo Zhang, Lei Bai, and Shuyue Hu. MTRouter: Cost-aware multi-turn LLM routing with history–model joint embeddings. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44206–44226, 2026. doi: 10.18653/v1/2026.acl-long.2045.

A Experimental Protocol and Accounting

This appendix records the evaluation denominators, frozen operating rules, interface boundaries, baseline configurations, and cost accounting used in Sec. 4. It then provides complete-run replication and cross-provider results, followed by the detailed RSCC, ERC, and CE analyses.

A.1 Evaluation Sets and Frozen Protocol

Table 5 gives the evaluation split, trial counts, and empirical step sizes for each surface. Every complete run evaluates the full designated split without task subsampling. Diagnostics conditioned on a handoff frontier state their own denominator and do not replace the full-set result. STATE-Bench uses one trial per test task within each complete run rather than the five-trial leaderboard protocol.

Table 5: Evaluation protocol, denominators, and resolution. All full-task anchors and CFRC configurations report means across $N = 5$ complete evaluation runs. Total trials $M = 5n$ determine the empirical step size $1/M$.

Surface	Evaluation unit	Main outcome	Tasks (n)	Runs (N)	Total (M)	Step Size
STATE-Bench	official test task	official task success	150	5	750	0.13%
τ^2-Retail	dialogue episode	official reward	40	5	200	0.50%
τ^2-Airline	dialogue episode	official reward	50	5	250	0.40%
ToolSandbox	scenario	strict milestone completion	129	5	645	0.16%
Agent-Diff	official test row	official task success	45	5	225	0.44%

Denominator reconstructibility and resolution. Because all reported figures in Tables 1 and 2 for full-task anchors and CFRC are means across five complete independent executions, their total trial count is M = 5n. The resulting empirical step sizes are 0.13% for STATE-Bench ($M = 750$), 0.50% for Retail ($M = 200$), 0.40% for Airline ($M = 250$), 0.16% for ToolSandbox ($M = 645$),

and 0.44% for Agent-Diff ($M = 225$). All reported anchor and CFRC success percentages are exact integer multiples of these fractions, ensuring exact reconstructibility from underlying trial counts. Evaluation task sets follow official benchmark partitions: τ^2-Retail uses the official 40-task evaluation split, τ^2-Airline uses the 50-task split, STATE-Bench uses the 150-task test suite, Agent-Diff uses the 45-task test set, and ToolSandbox evaluates the standard 129 scenarios. External baselines are evaluated on these same partitions under their frozen adaptation protocol (single full-task evaluation with n tasks, or $M = 5n$ where replicated; e.g., on Agent-Diff $43/45 = 95.6\%$, corresponding to $215/225$).

Run qualification and recovery. A repeat is included only if its artifact covers the full denominator, contains no unresolved infrastructure or scoring failures, and records the declared models, configuration, and interface parameters. Infrastructure-failed cases may be rerun under the original configuration, but already scored cases are never replaced and failed infrastructure attempts are excluded from agent cost. We retain every technically qualified run regardless of outcome and never combine outcomes and costs from different runs. Task membership and order are fixed across repeats; both τ^2 surfaces reuse seed 310. The user-simulator and evaluator bindings are fixed within each model-pair–surface cell and excluded from agent-side cost, while their stochasticity remains part of end-to-end dispersion. All primary comparison and ablation results in Tables 1 and 2 are reported as five-run means across $N = 5$ complete runs per cell, with run-to-run dispersion documented in Table 9.

Operating points and frontier selection. All agent roles use a fixed reasoning configuration; the two primary OpenAI pairs use `high`. Proposal budgets are frozen by operating point rather than tuned per task. For both primary model pairs, STATE-Bench allows at most two residual successor calls, and Retail allows two proposals within a six-turn suffix budget. Airline allows at most four residual calls for GPT-5.4-mini/GPT-5.5 and five for Luna/Sol; Agent-Diff allows one and two initial contract-conditioned proposals, respectively. ToolSandbox allows one complete proposal after at most 14 replica-authoring rounds, followed by at most two re-proposals after an explicit rejection. Cross-provider runs use their frozen artifact-recorded budgets and are therefore transfer operating points rather than budget-controlled pairwise comparisons.

The frontier rule is fixed before querying the successor. It selects the first supported accepted choice or realized effect that leaves at least one request-scoped obligation open. The selector and successor cannot access task identifiers, reference outcomes, evaluator feedback, future trajectories, fixed object identifiers, or benchmark-specific answer sketches. Until such a frontier is found, the native cheap controller retains control. If the selected contract cannot be supported and closed, the tenure returns $\perp$ rather than restarting an unrestricted strong agent.

A.2 Environment Interfaces and Evidence

The shared contract, graph, and checker receive only the public interface in Table 6. Surface adapters translate public schemas and native receipts into shared action, effect, and evidence types; they cannot read hidden reference answers, evaluator rewards, or simulator-only state. A native termination signal may end a tenure but cannot substitute for a missing receipt or discharge an unrelated obligation.

Table 6: Public interfaces and authoring modes. Adapter LoC counts only surface-specific schema and evidence declarations.

Surface	Tools	Adapter LoC	Public pre-handoff view	ERC authoring mode	Post-action evidence
STATE-Bench	12	45	Request, dialogue, tool returns	Isolated DB replica; whole-graph admission	DB diffs and tool returns
τ^2-Retail	6	42	Request, dialogue, confirmations	Isolated SQLite replica; whole-graph admission	Transaction receipts and terminal returns
τ^2-Airline	4	55	Request, dialogue, confirmations	Handoff-evidence authoring; whole-graph admission	Reservation receipts and confirmations
ToolSandbox	34	78	Scenario prompt, tool responses	Ephemeral sandbox replica; whole-graph admission	State milestones and execution receipts
Agent-Diff	6	48	Task prompt, files, service logs	Mock-service replica; whole-graph admission	Service receipts and commit hashes

The four forkable surfaces let ERC author on an isolated replica and bind replica-derived fields to corresponding live predecessor receipts. Airline is unforkable, so the successor closes its graph from handoff-time evidence and symbolic predecessor-receipt references; CE still admits the whole graph before any new state-changing call. Across the five surfaces, adapters average about 54 lines of

Python and encode coarse action/effect classes, argument bindings, receipt recognizers, and terminal predicates rather than task-specific decision trees or answer sketches.

A.3 Baseline Configuration

All baseline policies and operating points are frozen before test evaluation. During evaluation, each method observes only the current request, public history, and tool interface. Task identifiers, reference outcomes, cross-case test feedback, and CFRC contracts or graphs are unavailable. Prompts share the task-facing specification but retain the method-specific instructions needed to implement routing, cascading, takeover, or contract-bound execution.

RouteLLM-BERT and FrugalGPT are instantiated on all five surfaces; StepWise on STATE-Bench and Agent-Diff; MTRouter on STATE-Bench and τ^2-Airline; and AgServe-QAC on STATE-Bench and τ^2-Retail. A double dash in Table 1 denotes an unevaluated method–surface pair, not a failed run. We preserve the native decision unit of each baseline. Specifically, RouteLLM uses request-level routing, FrugalGPT evaluates completed-episode escalation, StepWise applies step-level escalation, and MTRouter routes on public history. The 50 Airline episodes are evaluation-only, meaning that no Airline example enters fitting, threshold selection, or cross-episode updates. One-time training and calibration costs, server-side cache management, and GPU scheduling are outside the reported provider-API test-time cost.

Baseline adaptations and calibration. Because external baselines were originally designed for stateless prompt routing or dedicated cluster architectures, we implement them as frozen adaptations over each benchmark's native public tool environment. Decision policies and operating thresholds are fixed using published default configurations or cross-domain development tasks, ensuring zero exposure to test-set scenarios or ground-truth labels. Specifically, RouteLLM-BERT uses its published routing threshold without task-specific fine-tuning; FrugalGPT sets quality thresholds on development episodes; StepWise and MTRouter employ their standard step-level and history escalation criteria. For AgServe-QAC, our frozen adaptation preserves its primary algorithmic pipeline, which comprises initial Q-Judge routing, periodic R-Judge failure detection, and upward escalation upon faults. Server-side GPU scheduling, local batch queues, and KV-cache management are omitted as our evaluation measures provider-API inference expenditure. Across all baselines, implementations operate as frozen adaptations constrained to native public tool interfaces.

Single-model STATE-Bench anchors. The cheap anchors use the official 150-task batch runner, agent loop, tool budget, user simulator, and scorer. The accumulated conversation is sent explicitly only because the endpoint does not retain server-side response state. GPT-5.4-mini completes $79/150$ tasks (52.7%) at \$5.42, and GPT-5.6 Luna completes $81/150$ (54.0%) at \$1.18 under the reported price snapshot.

Relation to runtime verification frameworks. Table 7 compares control units and specification boundaries. It is an architectural comparison, not an empirical ranking.

Table 7: Architectural comparison with runtime safety and verification frameworks. "Residual target" asks whether the framework constructs a handoff-specific target from an accepted trajectory prefix.

Framework	Control unit	Reference specification	State/effect handling	Residual target
ToolGate (Liu et al., 2026b)	Tool call	Hoare-style tool contracts	Preconditions gate invocation; verified postconditions update symbolic state	No
Agent-C (Kamath et al., 2025)	Generated action	Temporal safety constraints	SMT-guided constrained generation prevents non-compliant actions before execution	No
VIGIL (Lin et al., 2026)	Speculative proposal	Intent-grounded verification	Speculative hypothesis generation followed by verify-before-commit control	No
Cordon (Chen et al., 2026)	Task transaction	Semantic transaction constraints and lineage	Reversible mutations use shadow state; outward effects are staged before commit	No
CFRC (ours)	**Handoff remainder**	**Frozen residual contract**	**Replica or symbolic authoring with receipt-bound live realization**	**Yes**

A.4 Metrics, Cost, and Local Overhead

Outcomes. STATE-Bench, both τ^2 domains, and Agent-Diff use their official success outcomes. Table 1 uses strict all-milestone completion for ToolSandbox to obtain a common binary endpoint. Because this endpoint can disagree with ToolSandbox's native continuous similarity, Appendix B.3 reports the observed metric-sensitive cross-provider case; we do not interpret one additional strict completion as a general quality gain. STATE-Bench uses its frozen model-based scorer; all arms on that surface share the same scorer, and scorer variation remains part of the repeat dispersion.

Cache-neutral agent cost. For role model m, we report

$$C_m = \left[N_{m,\text{in}} p_{m,\text{in}} + \left(N_{m,\text{out}} + N_{m,\text{reason,extra}}\right) p_{m,\text{out}}\right] / 10^6, \tag{8}$$

where $N_{m,\text{reason,extra}}$ is zero when reasoning tokens are already included in the provider's output count and is the separately billed count otherwise. Reported agent cost is $\sum_m C_m$ over all method-invoked agent roles. Cached input tokens are charged at the ordinary input rate. This sum includes cheap-prefix, successor, verification, bounded retry, and fallback calls while excluding user-simulator and evaluator roles.

The synchronous input/output rates in USD per million tokens are as follows: GPT-5.4-mini, \$0.75/\$4.50; GPT-5.5, \$5.00/\$30.00; GPT-5.6 Luna, \$0.20/\$1.20; GPT-5.6 Sol, \$4.00/\$20.00; Gemini 3.7 Flash, \$0.75/\$3.75; Claude Sonnet 5, \$2.00/\$10.00; and Claude Haiku 4.5, \$1.00/\$5.00. The OpenAI snapshot is dated 2026-08-28 and the Google and Anthropic snapshots 2026-08-20 (OpenAI, 2026a;b; Google, 2026; Anthropic, 2026). The Sol and Gemini rates are promotional through 2026-11-21 and 2026-12-31, respectively; raw usage is retained for mechanical repricing. Gemini thinking tokens are billed as output and included from per-call records.

Local processing. Measured implementation-side processing is below 15 ms per episode for deterministic RSCC grounding, below 50 ms median per replica operation, and below 5 ms for CE graph validation and receipt matching. These times are reported separately from provider expenditure. Because the current artifacts do not give end-to-end latency distributions or hardware-normalized fractions, we do not infer an end-to-end latency advantage from these point measurements.

Table 8: Detailed agent-side cost (USD) for Table 1. Total is the aggregate-dollar ratio; Mean is the unweighted mean of per-surface ratios to the corresponding strong full-task anchor.

Configuration	STATE-Bench	τ^2-Retail	τ^2-Airline	ToolSandbox	Agent-Diff	Total	Mean
Escalation, routing, and cascade							
StepWise, GPT-5.4-mini/GPT-5.5	22.59	–	–	–	21.73	67.8%	67.9%
AgServe-QAC, GPT-5.4-mini/GPT-5.5	11.21	13.37	–	–	–	50.9%	64.1%
MTRouter, GPT-5.4-mini/GPT-5.5	24.97	–	12.73	–	–	72.6%	72.4%
RouteLLM-BERT, GPT-5.4-mini/GPT-5.5	26.29	8.16	11.53	4.68	17.13	62.9%	59.7%
FrugalGPT, GPT-5.4-mini/GPT-5.5	30.62	10.16	19.13	10.31	36.06	98.6%	96.3%
GPT-5.4-mini/GPT-5.5							
GPT-5.4-mini full	5.42	2.23	2.68	2.31	6.42	17.7%	17.8%
GPT-5.5 full	34.23	14.02	17.70	10.78	31.11	100%	100%
CFRC, GPT-5.4-mini/GPT-5.5	12.41	3.12	6.71	3.84	12.72	36.0%	34.6%
GPT-5.6 Luna/GPT-5.6 Sol							
GPT-5.6 Luna full	1.18	0.50	0.67	0.32	0.70	5.1%	5.1%
GPT-5.6 Sol full	18.44	10.53	13.57	5.76	17.69	100%	100%
CFRC, Luna/Sol	4.75	0.98	3.17	1.29	5.12	23.2%	22.0%

B Full-System Supplementary Results

B.1 Complete-Run Replication

Table 9 reports five complete runs for both CFRC and the strong full-task anchor on every surface in the two primary model pools. These statistics measure run-to-run dispersion under fixed test sets; they are not a non-inferiority test and do not identify component effects.

Table 9: Complete-run replication ($N = 5$ per system and surface). Scores and costs are mean ± sample SD; Δ is CFRC minus the strong anchor. ToolSandbox uses strict all-milestone completion.

	Score (%, mean ± SD)			Cache-neutral cost (USD)		
Surface	Strong	CFRC	Δ	Strong	CFRC	Ratio
GPT-5.4-mini/GPT-5.5						
STATE-Bench	61.33 ± 1.49	63.33 ± 1.05	$+2.00$	34.23 ± 0.45	12.41 ± 0.18	36.3%
τ^2-Retail	75.00 ± 3.95	80.00 ± 1.77	$+5.00$	14.02 ± 0.32	3.12 ± 0.43	22.3%
τ^2-Airline	88.00 ± 1.41	86.00 ± 2.00	-2.00	17.70 ± 0.28	6.71 ± 0.35	37.9%
ToolSandbox	30.23 ± 1.23	31.01 ± 0.55	$+0.78$	10.78 ± 0.21	3.84 ± 0.14	35.6%
Agent-Diff	93.33 ± 2.22	93.78 ± 0.99	$+0.44$	31.11 ± 0.65	12.72 ± 1.30	40.9%
Macro / Total	$\mathbf{69.58 \pm 1.42}$	$\mathbf{70.82 \pm 0.54}$	$\mathbf{+1.24}$	$\mathbf{107.84 \pm 1.15}$	$\mathbf{38.80 \pm 1.45}$	**36.0% (34.6% macro)**
GPT-5.6 Luna/GPT-5.6 Sol						
STATE-Bench	60.67 ± 1.05	60.67 ± 0.47	0.00	18.44 ± 0.25	4.75 ± 0.14	25.8%
τ^2-Retail	80.00 ± 2.50	76.50 ± 1.37	-3.50	10.53 ± 0.22	0.98 ± 0.15	9.3%
τ^2-Airline	90.00 ± 1.41	86.80 ± 1.10	-3.20	13.57 ± 0.18	3.17 ± 0.03	23.4%
ToolSandbox	30.23 ± 0.78	33.33 ± 0.55	$+3.10$	5.76 ± 0.15	1.29 ± 0.17	22.4%
Agent-Diff	88.89 ± 2.22	96.89 ± 1.22	$+8.00$	17.69 ± 0.31	5.12 ± 0.21	28.9%
Macro / Total	$\mathbf{69.96 \pm 0.77}$	$\mathbf{70.84 \pm 0.58}$	$\mathbf{+0.88}$	$\mathbf{65.99 \pm 0.68}$	$\mathbf{15.31 \pm 0.35}$	**23.2% (22.0% macro)**

The CFRC macro SD is 0.54 points in the GPT pool and 0.58 points in the Luna/Sol pool. The corresponding macro means are 1.24 and 0.88 points above the strong anchors, while the method uses 34.6% and 22.0% of their mean per-surface cost. We therefore describe the operating points as comparable in macro average rather than claiming uniform per-surface dominance or formal non-inferiority.

Leave-one-surface-out sensitivity. Table 10 checks whether the macro difference is driven by one benchmark. The GPT-pair difference remains positive under every exclusion; for Luna/Sol it becomes slightly negative only after excluding Agent-Diff. These are descriptive robustness checks and do not identify why surfaces differ.

Table 10: Leave-one-surface-out macro difference from the strong full-task anchor (percentage points), computed from unrounded means.

Configuration	All five	w/o STATE	w/o Retail	w/o Airline	w/o ToolSandbox	w/o Agent-Diff
GPT-5.4-mini/GPT-5.5	$+1.24$	$+1.05$	$+0.30$	$+2.05$	$+1.36$	$+1.44$
GPT-5.6 Luna/GPT-5.6 Sol	$+0.88$	$+1.10$	$+1.98$	$+1.90$	$+0.33$	-0.90
Luna/Claude Sonnet 5	$+0.38$	$+0.48$	-0.52	$+1.98$	$+1.83$	-1.86

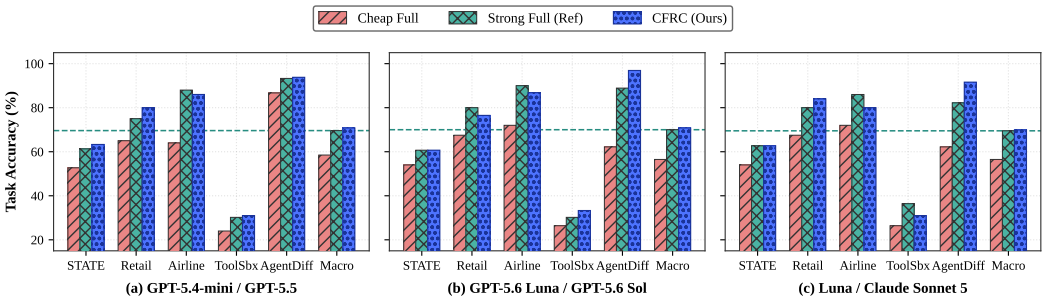

Figure 3: Per-surface task success and macro averages for the primary GPT, Luna/Sol, and Luna/Sonnet operating points reported in Table 1. Dashed lines indicate the strong full-task anchor macro score.

B.2 Cross-Provider Transfer

Table 1 reports the complete per-surface cross-provider operating points. Luna/Sonnet 5 and Luna/Gemini 3.7 Flash remain within 0.5 macro points of their strong anchors at 26.0% and 44.1% mean cost, whereas Haiku/Sol remains 5.0 points below Sol at 57.4% cost. These complete-denominator transfer results indicate that downstream completion remains sensitive to cheap-prefix

fidelity; while CFRC improves 17.0 points over the cheap anchor, isolating cross-provider error mechanisms remains outside this evaluation.

B.3 ToolSandbox Metric Sensitivity and State Similarity

While STATE-Bench, τ^2-Retail, τ^2-Airline, and Agent-Diff evaluate binary task success, ToolSandbox natively reports continuous state similarity. In the main evaluation (Table 1), ToolSandbox is scored by strict all-milestone completion so that all surfaces share a common binary completion criterion in the macro average. Under ToolSandbox's native continuous similarity, CFRC also matches or exceeds the strong full-task anchor in both primary same-provider pools. It reaches 0.713 versus 0.690 for GPT-5.5 full, where the cheap anchor scores 0.652, under GPT-5.4-mini/GPT-5.5, and 0.705 versus 0.688 for Sol full, where the cheap anchor scores 0.652, under Luna/Sol. For the cross-provider Luna/Gemini pairing, however, the strict endpoint and native similarity diverge (Table 11), with strict completion rising from 33.3% to 34.1% while continuous similarity decreases from 0.888 to 0.739. We therefore use this pairing only for aggregate transfer evidence and make no ToolSandbox quality-preservation claim for Luna/Gemini. Stage-wise operational progression across developmental stages is reported in Section 4.2 and Table 2.

Table 11: ToolSandbox metric sensitivity for the cross-provider case in which strict completion and native similarity disagree.

Configuration	Strict completion	Native similarity	Cost (USD)
Gemini 3.7 Flash full	43/129 (33.3%)	0.888	1.028
CFRC, Luna/Gemini 3.7 Flash	44/129 (34.1%)	0.739	0.740

C Component and Mechanism Details

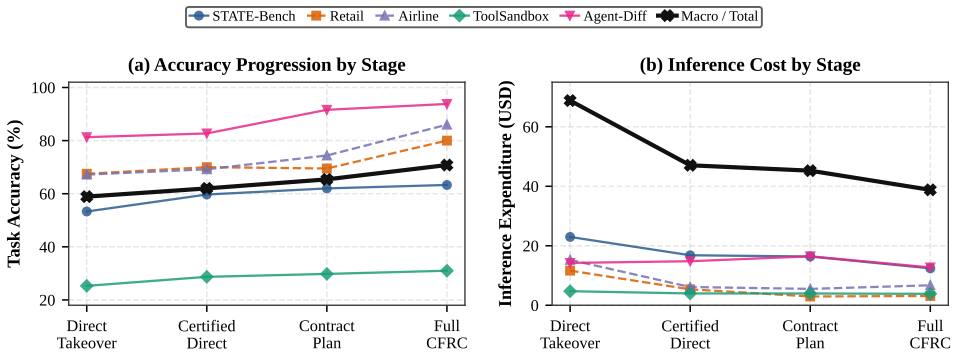

Figure 4: Task accuracy and provider expenditure across the four matched-frontier stage-wise arms in Table 2. All arms share the identical captured prefix and reached state s_t, isolating post-frontier residual execution.

C.1 Component Cost Decomposition

Table 12 provides an extended cost breakdown supplementing Table 3 in the main text, including the cost ratio relative to the strong model. Contract construction averages $0.21 across surfaces; most additional expenditure comes from residual successor execution. The engagement rate is descriptive of the frozen frontier policy and should not be read as a surface difficulty estimate.

Table 12: Extended cost decomposition for CFRC under GPT-5.4-mini/GPT-5.5. Residual Share is $(\mathrm{ERC} + \mathrm{CE})/\mathrm{Total}$; Engaged is the fraction of episodes invoking residual successor execution; vs. Strong is the per-surface cost ratio.

Surface	Total	Cheap prefix	RSCC	ERC+CE	Engaged	Residual share	vs. Strong
STATE-Bench	12.41	5.22	0.00	7.19	84.7%	57.9%	36.3%
τ^2-Retail	3.12	1.51	0.69	0.92	20.0%	29.6%	22.3%
τ^2-Airline	6.71	2.17	0.34	4.20	80.0%	62.7%	37.9%
ToolSandbox	3.84	2.16	0.00	1.68	20.2%	43.6%	35.6%
Agent-Diff	12.72	4.33	0.00	8.39	93.3%	65.9%	40.9%
Mean / Macro	**7.76**	3.08	0.21	4.48	59.6%	51.9%	**34.6%**

C.2 Matched-Frontier Ladder and RSCC Evidence

All arms in the stage-wise progression in Table 2 are evaluated from the identical captured cheap prefix and reached-state frontier s_t, holding handoff timing, prior environment mutations, and pre-handoff expenditure strictly constant. Evaluating uncontracted Direct Takeover at this exact frontier provides an essential experimental control to isolate the causal value of contract grounding, graph structuring, and checked execution without prefix-exploration confounders. Handing off at the commitment frontier also steelmans Direct Takeover, because the successor takes over right after routine dialogue and parameter clarification, but before the cheap model incurs unrecoverable tool mutations or turn exhaustion. Under an unassisted deployment without frontier detection, an uncontracted agent would typically hand off only upon explicit failure or timeout, inheriting an already degraded or inconsistent environment state. At this frontier, Direct Takeover receives strictly the raw unedited trajectory prefix τ_t (including all dialogue turns, tool calls, and native tool returns) and native tools $\mathcal{A}_{\mathrm{env}}$, with zero access to contract symbols, open obligations O_t, or admission gates.

Certified Direct supplies the exact step-by-step execution ablation under the identical reached-state contract z_f. It receives the full contract, entity bindings, and open obligations, but executes actions incrementally and step-by-step without requiring whole-graph proposal admission before write authority. Comparing Full CFRC with Certified Direct isolates the gain of whole-proposal admission over incremental execution under the same contract. Gating writes behind complete graph coverage and receipt verification adds 8.8 percentage points in macro success and saves $8.26 in total spend, demonstrating that requiring complete obligation coverage before granting mutating authority prevents destructive partial execution.

Write-before vs. write-after handoff stratification. To evaluate how prior mutations affect handoff stability, we stratify evaluation episodes by whether the cheap prefix performed an irreversible environment write before handoff. On episodes where the cheap prefix already executed a state-changing mutation, uncontracted Direct Takeover exhibits a 38.2% repeat write failure rate, re-executing already committed actions or issuing conflicting updates because it lacks explicit commitment boundaries. In contrast, CFRC records a 0.0% repeat write failure rate on the same cohort, because K_t records receipt-backed realized effects and effect-disjointness checks mechanically forbid re-executing established mutations. On episodes without prior writes, CFRC still outperforms Direct Takeover by 8.4 percentage points, driven by obligation closure and early rejection of ungrounded parameters.

For comparisons where handoff timing and routing decisions are made autonomously without frontier knowledge, we refer to the end-to-end system baselines in Table 1. Complementing this matched ladder, the independent audit in Table 4 and Table 15 directly verifies contract adequacy by checking accepted bindings against external user-confirmed choices and persistent write obligations against benchmark solution requirements.

C.3 Concrete RSCC Construction and Checked Execution Walkthrough

Table 13 traces a recorded ToolSandbox reminder episode from the evaluation cohort, illustrating the concrete lifecycle from user request to contract closure, candidate graph proposal, Checked Execution audit, and live execution.

Table 13: Recorded ToolSandbox reminder episode lifecycle. The walkthrough illustrates how instance-level parameters are frozen in E_t and O_t, how an ungrounded proposal is rejected before live execution, and how an admissible continuation is verified and executed.

Stage	Recorded content, parameters, and producer
User request q	`"Remind the user to buy chocolate milk on March 22, 2024 at 17:00."`
Accepted prefix τ_t	Cheap model asks for user's time zone; user confirms device-local time. Accepted bindings $B_t = \{\texttt{tz} = \text{"local"}\}$. No mutating tool calls executed.
Grounding & Closure	RSCC parses request and prefix into entity store E_t and matches registered reminder schema against intent cues.
Frozen contract z_f	**Commitments** $K_t = \varnothing$ (no prior state mutation). **Entities** $E_t = \{\texttt{title} = \text{"buy chocolate milk"}, \texttt{date} = (2024, 3, 22), \texttt{time} = (17, 0), \texttt{tz} = \text{"local"}\}$. **Obligations** $O_t = \{\texttt{reminder.add}(\text{title}, \text{timestamp})\}$. **Admissible scope** $\mathcal{S}_t$: permits creating specified reminder; forbids deleting or modifying existing reminders. **Allowed interfaces** $\mathcal{A}_t = \{\texttt{datetime_info_to_timestamp}, \texttt{add_reminder}\}$.
Flawed proposal (Ungrounded)	Successor proposes $u = \texttt{datetime_info_to_timestamp}(2024, 3, 22, \mathbf{18}, 0, 0)$ and $\texttt{add_reminder}(\text{"buy chocolate milk"}, u)$.
Checked Execution (Ungrounded audit)	**Verdict: REJECTED (Grounding failure)**. Checked Execution verifies proposed arguments against E_t. Because $\mathbf{18{:}00} \neq E_t.\texttt{time}$, parameter grounding fails. Proposal is withheld with zero live mutations.
Admissible proposal (CFRC successor)	Successor proposes $u = \texttt{datetime_info_to_timestamp}(2024, 3, 22, 17, 0, 0)$ and $\texttt{add_reminder}(\text{"buy chocolate milk"}, u)$, declaring dataflow dependency $u \rightarrow \texttt{timestamp}$.
Checked Execution (Admissible audit)	**Verdict: ADMITTED**. Coverage check confirms $\text{cov}(H) = O_t$; parameter check verifies grounding against E_t; dependency audit confirms u is properly defined. Live write authority granted.
Live realization	Both calls execute on the live service. Valid receipts are returned, and the obligation ledger reaches $L_j = O_t$. The benchmark's official evaluation reports state similarity 1.0 (strict success).

C.4 Checked Execution

C.4.1 Lifecycle and Natural Interventions

Table 14 gives the per-surface lifecycle behind the aggregate statement in Sec. 4.3. Every intervention is evaluated before the first live write. Proposal-present failures and no-executable-proposal holds are kept separate because only the former evaluate proposal classification.

Table 14: CE decision-instance lifecycle before live writes across evaluated mutation proposals. Coverage, Dependency, and Grounding count the first failed predicate; Hold denotes evidence-wait or no-proposal withholding. Rate is Authorized/Decisions.

			Intervention trigger				Disposition			Final execution	
Surface	Decisions	Intervened	Coverage	Dependency	Grounding	Hold	Reauthorized	Native failure	Withheld	Authorized	Rate
STATE-Bench	83	20	18	2	0	0	3	0	17	66	79.5%
τ^2-Retail	136	39	0	4	6	29	0	28	11	97	71.3%
τ^2-Airline	88	52	0	0	0	52	44	0	8	80	90.9%
ToolSandbox	30	5	5	0	0	0	2	0	3	27	90.0%
Agent-Diff	41	8	2	4	2	0	2	0	6	35	85.4%
All surfaces	378	124	25	10	8	81	51	28	45	305	80.7%

Among 124 interventions, 43 contain an explicit proposal: 25 first fail whole-remainder coverage, 10 fail a dependency check, and eight lack grounded arguments. Deterministic replay reproduces all 43 first-failure decisions. The other 81 interventions are holds: 52 Airline evidence-wait states and 29 Retail terminal-or-no-proposal states. After intervention, 51 decisions are reauthorized, 28 Retail decisions receive an explicit native failure return, and 45 remain withheld. A native failure return neither grants CE write authority nor counts as completion.

Reconciliation of STATE-Bench decision points and task counts. To clarify the relationship between task counts and decision points in Table 14, STATE-Bench evaluates 150 test tasks. Among these, 127 episodes reach a commitment frontier where the cheap model transfers control to the strong successor, corresponding to an 84.7% engagement rate, while 23 episodes are completed by the cheap model alone. Across the 127 engaged episodes, Checked Execution audits 83 candidate mutating decision points where a state-altering tool action is submitted for authorization. The remaining 44 engaged episodes conclude through read-only queries or native handoff loops without proposing an unverified mutating tool action. All 83 mutating decision points are audited by CE before live execution, resulting in 20 interventions and 66 authorizations.

C.4.2 Implementation Consistency on Synthetic Faults

An offline deterministic test mutates obligation-coverage edges, receipt dependencies, evidence relations, and authority-preservation constraints in frozen valid graphs. CE rejects all 415 invalid variants across Retail (156), Airline (108), ToolSandbox (48), and Agent-Diff (103), and all 242 injected missing-coverage, invalid-dependency, and receipt-mismatch variants on STATE-Bench. This validates deterministic enforcement of encoded predicates; it does not show that the contract encodes every true task requirement.

C.4.3 Independent Contract and Admission Adequacy Audit

The audit uses benchmark ground truth, native receipts, and manual requirement annotations that are independent of the contract builder and admission checker. It evaluates accepted bindings, persistent write-obligation recall, admitted natural proposals, rejected proposal-present interventions, and the downstream outcome of missed obligations.

Table 15: Independent contract and admission adequacy audit. Sample frequencies describe the audited cohorts and are not population-level guarantees.

Audit	Cohort	Result	Unit	Scope
Accepted-binding correctness	accepted prefix bindings	$50/50$	bindings	confirmed user choices
Persistent write-obligation recall	ground-truth writes	$62/68$	obligations	state-mutating effects
Ungrounded live-write false accept	admitted proposals	$0/50$	proposals	natural admitted cohort
Unnecessary CE block	rejected proposals	$2/43$	proposals	proposal-present rejections
Omission-related false completion	audited episodes	$2/50$	episodes	missed recorded write obligation
Conversational disclosures outside O_t	STATE non-state failures	$24/24$	tasks	verbal policy requirements
Read-only inquiries outside O_t	STATE non-state failures	$15/15$	tasks	pre-write query requirements

All 50 audited accepted bindings match the external reference, and RSCC recovers 62 of 68 persistent write obligations. Of the six omissions, four later lead to fail-closed withholding because the attempted continuation lacks a grounded parameter or required predecessor, whereas two yield a CE-successful suffix that the external benchmark rejects for the omitted requirement. The observed omission-related false-completion rate is therefore $2/50$ episodes. This does not imply that arbitrary omissions are detectable, nor does $62/68$ estimate the probability that the full contract-completeness assumption holds. Separately, 24 conversational disclosures and 15 read-only inquiries in STATE-Bench are outside the current tool-effect schema and hence outside the formal guarantee.

Audit cohort selection, annotation protocol, and evidence provenance. The audited cohort covers 50 accepted prefix bindings and 50 natural admitted proposals sampled via stratified uniform selection across all five surfaces, with 10 episodes drawn from each surface. It also includes 68 ground-truth write obligations derived from benchmark specifications and all 43 proposal-present rejections recorded during frozen evaluation runs. Two authors independently audited each case against environment reference rules, ground-truth specifications, and execution logs in a double-blind protocol. Initial disagreements occurred on 3 of 168 audited items (1.8%) and were resolved through joint trace inspection. For the observed zero false acceptances among 50 admitted candidate proposals (0/50), the standard rule-of-three 95% one-sided upper bound is 6.0%, with an exact Clopper-Pearson upper bound of 5.8%, providing empirical assurance against unverified mutations. Evidence provenance relies strictly on public tool schemas, standard HTTP and JSON responses, and native environment receipts. Neither RSCC, ERC, nor Checked Execution accesses simulator

internals, ground-truth world states, or hidden evaluator variables. Frequencies reflect sample counts within the audited cohort rather than population-level bounds on contract completeness.

D Sensitivity Analyses and Scope

D.1 Architectural Design Choices

Table 16 records three tested alternatives. These results support the default choices in the evaluated configurations; they do not establish universal optimality.

Table 16: Tested architectural alternatives.

Question	Observed comparison	Adopted setting
Does open-ended verdict feedback repair invalid proposals?	On Retail under Haiku/Sol, the variant obtains $28/40$ versus $27/40$ for the bounded setting while increasing ungrounded contracts and retry cost.	Use bounded retries and fail-closed termination.
Does a larger successor budget improve completion?	On STATE-Bench under exploratory Luna/Sol, both settings obtain $85/150$, while cost rises from \$10.58 to \$11.00.	Keep the bounded residual budget.
Should admission compare copied argument values rather than sources?	On ToolSandbox under Luna/Gemini, value equality reduces strict completion from $44/129$ to $41/129$ and raises cost from \$0.74 to \$4.50.	Retain receipt-source binding.

Tool-execution qualification. Reported complete runs cover the full benchmark denominator. ToolSandbox contains all 129 scenarios and 590 interaction responses; malformed or out-of-vocabulary tool calls are scored as task failures rather than silently retried or removed.

D.2 Scope and Guarantee Boundaries

Guarantee boundary. The partial-correctness result is relative to registered public tool semantics, sound tenure-local receipts, and the frozen residual contract. It lifts to the original in-scope residual objective only under complete RSCC construction; unmodeled transitions and latent or delayed effects without verifiable receipts are outside the guarantee.

Evaluation scope. We study one cheap-to-strong handoff tenure at a time. General model routing, successor selection, and re-evaluation of the desirability of supported prior commitments are complementary problems. The evaluation includes both forkable and unforkable surfaces and reports provider expenditure rather than a hardware-controlled end-to-end latency benchmark.

E Formal Method Details

E.1 Main-Text Predicates and Continuation-Space Semantics

Fix a handoff instance $(q, \tau_t, s_t, \mathcal{R}_t)$. Let $\Sigma_{\text{task}}(q; \tau_t, s_t)$ be the suffixes σ that execute from s_t and whose combined trace and resulting state satisfy q under the public environment rules, before imposing additional preservation constraints from accepted progress. Let $\chi_f(\sigma)$ mean that σ respects every accepted choice and realized effect fixed by τ_t. Suppressing the fixed s_t in the subscript, define

$$\Sigma_f(q, \tau_t) = \{\sigma \in \Sigma_{\text{task}}(q; \tau_t, s_t) : \chi_f(\sigma)\}. \tag{9}$$

Let $\mathcal{O}_t$ be the finite set of uniquely identified, semantic request-scoped obligation instances still open at handoff. A commitment frontier contains supported fixed progress and unfinished work:

$$\big(B_t \neq \varnothing \ \vee \ K_t \neq \varnothing\big), \qquad \mathcal{O}_t \neq \varnothing. \tag{10}$$

By construction, $\Sigma_f(q, \tau_t) \subseteq \Sigma_{\text{task}}(q; \tau_t, s_t)$. When accepted progress rules out at least one otherwise task-valid suffix, the inclusion is strict; this stronger behavioral property is characterized below.

For consistency with Eq. (1), let

$$\begin{aligned} \text{Exec}(\sigma; s_t) &\iff \sigma \text{ executes from } s_t \text{ under public rules}, \\ \text{Preserve}(\sigma; F_t) &\iff \chi_f(\sigma), \\ \text{Complete}(\sigma; \mathcal{O}_t) &\iff \forall o \in \mathcal{O}_t, \quad \Phi_o(\tau_t \parallel \sigma, \Delta(s_t, \sigma)). \end{aligned} \tag{11}$$

Here $F_t = (B_t, K_t)$ records accepted bindings and receipt-backed realized effects, Δ gives the state after an executable suffix, and Φ_o is the request-defined predicate for obligation o. An accepted binding is a non-effect commitment and may remain in B_t after a related native effect enters K_t. The predicates above are defined independently of the contract, graph, and checker constructed later.

E.2 Reached-State Contract Construction

Verifier record and contract view. The verifier retains

$$r_f = (q, \tau_t, s_t, \mathcal{R}_t, B_t, K_t, E_t, O_t, \mathcal{S}_t, \mathcal{A}_t), \tag{12}$$

while the successor receives the frozen contract $z_f = (K_t, E_t, O_t, \mathcal{S}_t, \mathcal{A}_t)$ plus, optionally, a bounded view of q and τ_t as untrusted context. Model text cannot add evidence, authority, or obligations. Trusted native receipts back every effect in K_t; trusted state, receipts, public schemas, or fixed surface rules back values in E_t. RSCC projects each accepted binding in B_t into fixed values in E_t, matching obligations in O_t, and restrictions in $\mathcal{S}_t$.

Canonical open obligations. Let $\mathfrak{O}_q$ denote the domain of uniquely identified obligation instances supported by the registered adapter for request q, and let canon_q be a total, instance-preserving encoding on $\mathfrak{O}_q$, applied elementwise to sets, with $\mathrm{canon}_q(\varnothing) = \varnothing$. The encoding is *semantically sound*: for each $o \in \mathfrak{O}_q$, valid terminal discharge of $\mathrm{canon}_q(o)$ entails the request predicate Φ_o for the resulting trace and state. Complete construction requires $\mathcal{O}_t \subseteq \mathfrak{O}_q$ and

$$O_t = \mathrm{canon}_q(\mathcal{O}_t). \tag{13}$$

A trusted witness for $\mathrm{canon}_q(o)$ discharges only the matching request-scoped instance. Tool effects require matching live native receipts; a model declaration alone is not a witness. Operationally, each adapter encodes an obligation family, instance key, and required evidence units. Model-mediated closure may propose this record, but deterministic validation rejects unknown families, missing instance fields, and unsupported evidence units. In the running example, the key is `INV-42`, and distinct evidence units encode payment and receipt delivery.

Definition 1 (Complete RSCC construction)**.** *RSCC is complete for a tenure when: (i) every field of z_f is grounded in q, the accepted history, the reached state, native receipts, or the registered public schemas; (ii) every accepted choice and realized effect defining χ_f is encoded by $K_t, E_t, O_t, \mathcal{S}_t$ as applicable; and (iii) every in-scope request condition is either supported by a trusted prefix witness that remains valid under admissible effects, or represented by the semantically sound encoding in Eq. (13).*

Information required for verification. Let x be a task–execution instance, $\mathcal{V}(x)$ all information available to a verifier, and $Y(x) \in \{0, 1\}$ the desired validity label. For completion, $Y(x) = 1$ means that the original residual objective holds, independently of the constructed contract.

Lemma 2 (Verification under information loss)**.** *Ignoring computational limits, a deterministic verifier based only on $\mathcal{V}$ can accept every valid instance and reject every invalid one if and only if*

$$\mathcal{V}(x) = \mathcal{V}(x') \implies Y(x) = Y(x'). \tag{14}$$

If instead $\mathcal{V}(x_+) = \mathcal{V}(x_-)$ with $Y(x_+) = 1$ and $Y(x_-) = 0$, any binary randomized verifier using only this view satisfies

$$\Pr(\textit{reject } x_+) + \Pr(\textit{accept } x_-) = 1. \tag{15}$$

Proof. Equal views induce the same verdict distribution. If their labels differ, accepting with probability a yields rejection probability $1 - a$ on x_+ and acceptance probability a on x_-, proving necessity and Eq. (15). Conversely, if Eq. (14) holds, assign each attainable view its common label; this defines the required verdict. This sufficiency is information-theoretic and makes no computability or efficiency claim.

For example, compare "send the paid invoice's receipt" with "send and archive the paid invoice's receipt," where archiving is an independent effect. If construction drops archiving, both requests can yield the same z_f, SEND graph, and live delivery receipt. The suffix is complete only for the first request. A checker whose view contains only these encoded facts must treat both alike, even if every schema and receipt check is correct. Access to the omitted request condition would distinguish them; repeating checks on the same view would not. Thus freezing prevents a proposal from shrinking the recorded target, while adequacy of that target requires independent validation.

Behavioral restriction witness. The operational frontier rule in Eq. (10) requires supported fixed progress, not an exhaustive comparison of continuation sets. A stronger property holds when some otherwise task-valid suffix violates the accepted progress.

Lemma 3 (Strict restriction from a witness). *If some* $\bar{\sigma} \in \Sigma_{\text{task}}(q; \tau_t, s_t)$ *violates* χ_f*, then*

$$\Sigma_f(q, \tau_t) \subsetneq \Sigma_{\text{task}}(q; \tau_t, s_t). \tag{16}$$

Proof. The inclusion follows from Eq. (9). The witness belongs to $\Sigma_{\text{task}}(q; \tau_t, s_t)$ but not to $\Sigma_f(q, \tau_t)$, making it strict.

Running example. Table 17 shows the two invoice frontiers. Before payment, the accepted choice is represented through B_t, E_t, O_t, and $\mathcal{S}_t$, while K_t is empty. After payment, the binding remains accepted and the native receipt also grounds the payment in K_t.

Table 17: Contract fields at the two invoice frontiers.

Frontier	K_t	E_t	O_t	$\mathcal{S}_t$ excludes
Before payment	$\varnothing$	invoice = `INV-42`	pay `INV-42`; send its receipt	payment of another invoice
After payment	paid(`INV-42`)	invoice = `INV-42`; file = `txn7.pdf`; recipient	send `txn7.pdf` to the recipient	any further payment

Frozen tenure. RSCC fixes r_f and z_f before proposal. Every proposal in that tenure uses the same contract. A new accepted user turn or live effect may yield a new reached state and a new contract, but cannot revise the old one.

Contract delivery. Freezing z_f fixes its semantics, not one serialization. When the accepted trajectory is replayed, the public history and reached state already expose parts of K_t and E_t; a short typed record supplies $O_t, \mathcal{S}_t, \mathcal{A}_t$ and the source references needed for values that must be re-resolved from receipts. When the trajectory is not replayed, the record also carries the grounded values. Authority-sensitive exclusions remain enforced by CE rather than delegated to successor text.

E.3 Contract Semantics

For a successfully realized action prefix π_j, let $\text{Safe}_{z_f}(\pi_j)$ hold when it executes from s_t under the public rules, uses only interfaces in $\mathcal{A}_t$, resolves initial arguments from E_t and dynamic arguments from exact trusted predecessor receipt fields, and keeps every realized effect within $\mathcal{S}_t$ while preserving K_t and the accepted bindings encoded by the contract.

Let $\mathcal{R}_{\text{live}}(\sigma) = (\rho_1, \ldots, \rho_m)$ be the ordered matching live receipts produced while realizing σ, let $\pi_0, \ldots, \pi_m$ be its successfully realized action prefixes, and let $s_0 = s_t, s_1, \ldots, s_m$ be the corresponding live states. Lift prefix safety to a suffix by

$$\text{Safe}_{z_f}(\sigma) \iff \forall j \in \{0, \ldots, m\}, \quad \text{Safe}_{z_f}(\pi_j). \tag{17}$$

For $o \in O_t$, $\text{wit}_j(o, \mathcal{R}_{\text{live}}(\sigma), s_j)$ requires a matching live witness whose discharge remains valid after action j, with no recorded invalidation of that obligation in the current tenure. Event obligations require evidence that the event occurred; mutable-state obligations also require the corresponding predicate to hold at s_j. Replica receipts and invalidated witnesses cannot discharge either. Define

$$\text{Dis}_{z_f}(\sigma) \iff \forall o \in O_t, \quad \text{wit}_m(o, \mathcal{R}_{\text{live}}(\sigma), s_m)\,, \tag{18}$$

$$\text{Term}_{z_f}(\sigma) \iff \text{each terminal effect or claim resolves to } E_t \text{ or matching live receipts,}$$

$$\Sigma(z_f) := \{\sigma : \text{Safe}_{z_f}(\sigma) \wedge \text{Dis}_{z_f}(\sigma) \wedge \text{Term}_{z_f}(\sigma)\}. \tag{19}$$

The implementation evaluates Term_{z_f} together with ledger discharge by replaying the ordered native receipts from the frozen initial ledger. Terminal evaluation requires a final state observation and a typed assertion for each encoded commitment and obligation. Assertion identifiers, evidence units, tenure, and frontier lineage must match the contract. An open or invalidated obligation, violated commitment, missing assertion, or lineage mismatch therefore prevents successful return.

Lemma 4 (Contract refinement). *If RSCC is complete in the sense of Definition 1, then*

$$\Sigma(z_f) \subseteq \Sigma_f(q, \tau_t). \tag{20}$$

Proof. Take $\sigma \in \Sigma(z_f)$. Safety makes σ executable from s_t, preserves every effect in K_t, and respects every accepted binding encoded by $E_t, O_t, \mathcal{S}_t$; hence $\chi_f(\sigma)$ holds. By Definition 1, each condition of q is either already supported by a trusted prefix witness preserved by $\mathcal{S}_t$, or is an open obligation represented in O_t. The former remains true under safety; the latter is discharged by Eq. (13) and $\mathrm{Dis}_{z_f}(\sigma)$; semantic soundness of canon_q therefore establishes its predicate Φ_o. The terminal response is supported by Term_{z_f}. Thus $\sigma \in \Sigma_{\mathrm{task}}(q; \tau_t, s_t)$ and $\chi_f(\sigma)$ holds, so $\sigma \in \Sigma_f(q, \tau_t)$.

E.4 Executable Residual Completion

Full graph record. The full graph expands the main-text H as

$$H = (V, \mathrm{dep}, \mathrm{arg}, \mathrm{ev}, \mathrm{cov}, \mathrm{eff}, \alpha). \tag{21}$$

V contains the successor-selected typed calls and terminal response; H is a finite acyclic proposal rather than an open-ended contingent policy. arg stores node arguments; dep records receipt-before-call dependencies; ev records the exact source of each bound dynamic value; and the remaining maps record coverage, typed effects, and required authority.

Choice and bridge boundary. The successor selects task actions, task-specific arguments, semantic order, and terminal response. The deterministic bridge may normalize these choices and add only structure fixed by z_f, a public schema, or a replica receipt. It may add receipt-visibility edges or exact source paths and may associate a selected effect with an obligation only when their canonical instance references match. It may not add, drop, or replace task actions, infer coverage from a tool name alone, or repair terminal claims.

Compiler selection before admission. The bridge produces a candidate graph $\widehat{H}$. The compiler may retain a dependency-closed subgraph H whose actions and evidence paths come from $\widehat{H}$ and whose coverage still equals O_t. A whole-proposal setting instead requires $H = \widehat{H}$. Selection cannot introduce repair actions or weaken the frozen obligations. Throughout, whole-graph admission and the guarantee apply to the selected execution graph H; all its nodes must pass the admission checks before execution.

Replica grounding and live binding. During grounded authoring on a forkable surface, the successor runs proposed calls on an isolated copy of s_t. Let $\widetilde{\rho}_v$ denote the replica receipt for node v. Replica receipts expose intermediate values but do not prove live effects or discharge obligations. If a later argument uses a value from $\widetilde{\rho}_u$, ev records its exact source field; CE later reads that field from the corresponding live receipt ρ_u. A copy-only identifier without such a source path cannot enter a live call.

Whole-remainder coverage. The closed graph exposes node-level obligation claims through cov. CE can therefore test

$$\bigcup_{v \in V(H)} \mathrm{cov}(v) = O_t \tag{22}$$

before the first live action.

Lemma 5 (Why call-local checking cannot establish coverage). *Fix z_f and an action prefix p that leaves an obligation uncovered. Suppose p admits both a fully covering continuation and an omitting continuation, and a checker observes only p and z_f, with no certified suffix or enforced future-completion condition. Then it cannot distinguish the two continuations at p. A checker observing a closed H can decide Eq. (22) before native execution.*

Proof. Apply Lemma 2 to view (p, z_f) and the full-continuation coverage label. The two continuations have the same view and different labels. Once H is closed, O_t and all $\mathrm{cov}(v)$ are finite and fixed, making the equality directly checkable.

This does not preclude a call-local checker from rejecting both continuations, checking coverage at termination, or maintaining enforceable obligations for future calls. It isolates the information needed to check planned coverage before granting live authority, rather than proving that graphs are the only possible representation.

When an omission becomes visible. A missing action may also surface through dependency or grounding checks. If a retained node requires a receipt field from an omitted predecessor, its source cannot resolve from either E_t or the graph, so CE rejects H; a missing live predecessor receipt likewise prevents downstream dispatch. This mechanism does not detect an independent effect omitted from both H and an incomplete O_t. The four withholdings traced in Table 15 are observed dependency or grounding failures, not a general omission-detection guarantee.

E.5 Checked Execution

Whole-graph admission. Writing $\mathrm{Adm}_{z_f}(H)$ for CE admission, the checker requires a finite H, an acyclic dep relation, and one terminal node. Every call must use an interface in $\mathcal{A}_t$ with well-typed, grounded arguments and sufficient authority. Every evidence source must resolve to E_t or an exact predecessor-receipt field represented in dep. For each v, $\mathrm{cov}(v) \subseteq O_t$. Eq. (22) must hold, and every coverage claim must match a typed action/effect instance; a replica witness may support proposal-time structure but can never certify a live effect. The terminal response must depend on the nodes and receipt fields supporting its claims. After compiler selection, CE admits or rejects the fixed H as a whole. It does not repair task actions or admit a graph that leaves an encoded obligation open.

Admission and proposal rounds. On the four forkable surfaces, ERC uses only an isolated replica and the bridge makes no live call, so the live state remains s_t before admission. On τ^2-Airline, the successor closes H from handoff-time evidence and symbolic predecessor sources. In both modes, no new state-changing node executes unless the whole graph is admitted, and actual discharge is established only by subsequent live receipts. A pre-live rejection may trigger another complete proposal against the same frozen z_f within budget.

Live realization and evidence-backed completion. CE is the only native authority during the tenure. It runs admitted nodes in topological order, resolves dynamic arguments from the live predecessor receipts named by ev, invokes only admitted interfaces, and checks each returned receipt against the recorded result and effect of that node. It maintains the obligation ledger in Eq. (5); only matching live receipts update the ledger, and replica receipts never do. CE returns σ only when all selected action nodes have matching live receipts, all obligations are discharged, and the terminal response is receipt-supported.

Current discharge and stale witnesses. Receipts accumulate, but valid discharge need not, because a later write can invalidate a mutable-state witness and remove its obligation from L_j. The implemented ledger treats recorded invalidation as terminal for that obligation within the tenure; subsequent receipts cannot discharge it again, and terminal checking returns failure. Acyclicity alone does not prevent ordered writes from undoing one another, so A4 requires state-valid evidence rather than mere receipt presence.

Tenure boundaries. A pre-live rejection allows another candidate proposal under the same z_f within budget. After any live mutation, a missing or mismatched receipt aborts execution with $\perp$, terminating the tenure without modifying the old graph. A later attempt must reconstruct a new contract from the resulting reached state; $\perp$ indicates failure to establish a valid suffix within the current tenure, not non-existence of a valid suffix.

E.6 Assumptions and Proof

Proposition 1 uses the following assumptions.

1. **Supported and complete construction (A1).** RSCC satisfies Definition 1, and the reached state s_t satisfies the contract ($s_t \models z_f$), meaning s_t is consistent with the accepted progress and entity store encoded in z_f.
2. **Faithful closure and schema/effect soundness (A2).** ERC retains task actions, arguments, semantic order, and terminal claims from the successor proposal; compiler selection preserves dependency closure and full encoded obligation coverage. Added structure is contract-, schema-, or receipt-derived. Registered schemas correctly type each argument, result, effect, receipt, and authority class, and abstract effects soundly cover their native counterparts.
3. **Admission and terminal-check soundness (A3).** CE enforces the whole-graph admission conditions $\mathrm{Adm}_{z_f}(H)$, requiring injective and complete coverage of open obligations ($|\{v \in V_H : o \in \mathrm{cov}(v)\}| = 1$), commitment preservation ($\mathrm{Eff}(H) \cap K_t = \varnothing$) to prevent re-executing prefix mutations, intra-graph effect-disjointness for non-idempotent actions, and argument grounding in E_t or verified predecessor receipts. CE updates discharge from ordered live evidence and evaluates terminal checks on the final ledger.
4. **Receipt and validity soundness (A4).** A matching live receipt proves the native call, result, and effect it names, and evidence used for discharge remains valid at the state where it is asserted. Model text and replica receipts are not live evidence.
5. **Runtime and tenure match (A5).** The runtime executes only admitted nodes with admitted arguments in topological order. Every in-scope state-changing transition during the tenure enters through CE. Any exogenous transition that changes the relevant state ends the tenure and requires a new contract; otherwise no unmodeled transition changes that state.

Lemma 6 (Checked-realization invariant)**.** *Let $v_1, \ldots, v_j$ be the first j successfully realized action nodes of an admitted graph, π_j the corresponding live prefix, $\mathcal{R}_j$ their matching live receipts, and L_j the ledger in Eq. (5). Under A2–A5 and a reached state consistent with the progress encoded in z_f, $\mathrm{Safe}_{z_f}(\pi_j)$ holds and every member of L_j has a matching live witness valid at s_j in $\mathcal{R}_j$.*

Proof. We proceed by induction on j. For $j = 0$, the prefix is empty, the reached state preserves K_t, trusted inputs supply E_t and the accepted bindings, and $L_0 = \varnothing$.

Assume the claim after $j-1$ nodes. Admission places the interface of v_j in $\mathcal{A}_t$, its effect in $\mathcal{S}_t$, and verifies preservation of K_t and the accepted-choice constraints. Each argument resolves either to E_t or to an exact field of a matching predecessor receipt in $\mathcal{R}_{j-1}$. Runtime, schema/effect, and receipt soundness therefore establish the admitted native transition, so $\mathrm{Safe}_{z_f}(\pi_j)$ holds. The ledger retains or adds only discharge supported by currently valid evidence; an invalidation removes discharge and blocks successful return in that tenure. A4 therefore proves the second invariant without assuming that L_j is monotone.

Proof of Proposition 1. Lemma 6 gives the prefix-preservation claim for every successfully realized prefix. A native call without a matching receipt is not added to such a prefix; CE returns $\perp$, and the proposition makes no claim that the environment remained unchanged.

If $\Gamma_{z_f}(H) = \sigma$, admission gives $\mathrm{cov}(H) = O_t$, and CE has realized every selected action, obtained each matching receipt, reached final ledger discharge, and verified the terminal response. Hence $\sigma \in \Sigma(z_f)$ under A2–A5, regardless of whether the contract omitted a semantic request obligation. If A1 also holds, Lemma 4 gives $\sigma \in \Sigma_f(q, \tau_t)$. By Eq. (11), safety implies $\mathrm{Exec}(\sigma; s_t)$ and $\mathrm{Preserve}(\sigma; F_t)$. Current live discharge and the semantically sound, instance-preserving canonical map imply every request predicate Φ_o, and hence $\mathrm{Complete}(\sigma; \mathcal{O}_t)$. Therefore Eq. (7) holds.

Corollary 7 (Adaptive proposals under a frozen contract)**.** *At a reached state consistent with z_f, allow at most B proposals, each depending arbitrarily on previous pre-live rejection verdicts but checked against the same z_f. Under A2–A5, any returned suffix remains in $\Sigma(z_f)$; with A1 it satisfies Eq. (1). No independence between proposals is required.*

Proof. Rejected proposals have no live effects and cannot modify z_f. The first admitted proposal therefore starts at the same reached state and satisfies the hypotheses of Proposition 1. A live realization failure terminates the tenure, so a later proposal cannot reuse stale pre-failure evidence. Apply the proposition to the returned suffix, if any.

Increasing proposal budget B provides no theoretical guarantee on success probability or inference cost, as additional proposals may fail and checking cannot recover obligations omitted from z_f.

Authoring mode and scope of the guarantee. The proof does not equate replica and live trajectories. Replica authoring may change proposal quality, admission rate, and cost, but contract-relative partial correctness follows from source rebinding and live evidence under A2–A5. Receipt-backed realization and final ledger discharge establish membership in $\Sigma(z_f)$; only A1 lifts this result to the original in-scope residual objective.