ACCEPT, CHALLENGE, OR AMEND? RECEIVER-SIDE INFORMATION HANDLING IN MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In multi-agent systems, specifying who communicates with whom does not determine how a receiving agent should handle incoming information. We formulate receiver-side information handling as a separately configurable system design dimension and study three representative policies: ACCEPT, in which the receiver uses the sender's information as the basis for subsequent actions; CHALLENGE, in which the receiver returns local evidence and requests sender-side revision; and LOCAL AMEND, in which the receiver locally adjusts its subsequent actions within its own responsibility boundary. Using Qwen3-Max and GPT-5.6-Luna, we conduct paired end-to-end evaluations across ALFWorld, AppWorld, ToolSandbox, and Collab-Overcooked, together with controlled handoff experiments in ALFWorld to investigate the underlying mechanisms. Our results show that different fixed handling policies lead to distinct execution trajectories, realized intervention behavior, and computational costs, while no single fixed policy consistently dominates across the tested environments and model backbones. Controlled experiments further show that when receivers possess diagnostic local evidence, challenging or locally amending erroneous handoffs substantially improves immediate repair; in contrast, interventions without such evidence can incur additional cost without improving final task performance. Motivated by these findings, we introduce an evidence-gated adaptive receiver-side handling strategy that dynamically selects among ACCEPT, CHALLENGE, and LOCAL AMEND at each information handoff based on local conflict evidence, repair reliability, and responsibility boundaries. Across the eight model–environment combinations, the adaptive strategy attains or ties the highest primary task-performance point estimate in seven cases and remains close to the best fixed policy in the remaining case. While not always minimizing token usage, it exhibits more consistent relative performance across environments and model backbones and generally avoids the highest token costs associated with fixed policies. Overall, our findings establish receiver-side information handling as an important system design dimension that should be explicitly configured and evaluated in multi-agent systems. Our code and reproduction instructions are available in an anonymous repository.[1]

1 INTRODUCTION

LLM-based multi-agent systems coordinate specialized roles, tools, and communication, and recent work shows that communication topology and message selection affect both performance and cost (Wu et al., 2024; Hong et al., 2024; Zhang et al., 2025a; Leong et al., 2025). Yet specifying who communicates with whom does not determine what authority an incoming message should have once it reaches a receiver. Figure 1 illustrates the resulting choice: use the message as the basis for action, challenge it for sender revision, or locally amend execution.

Consider a planner asking an executor to retrieve an object from a location that conflicts with the executor's current evidence. If the executor can identify a reliable correction within its own scope,

[1] https://anonymous.4open.science/r/receiver-side-information-handling-70DC

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.

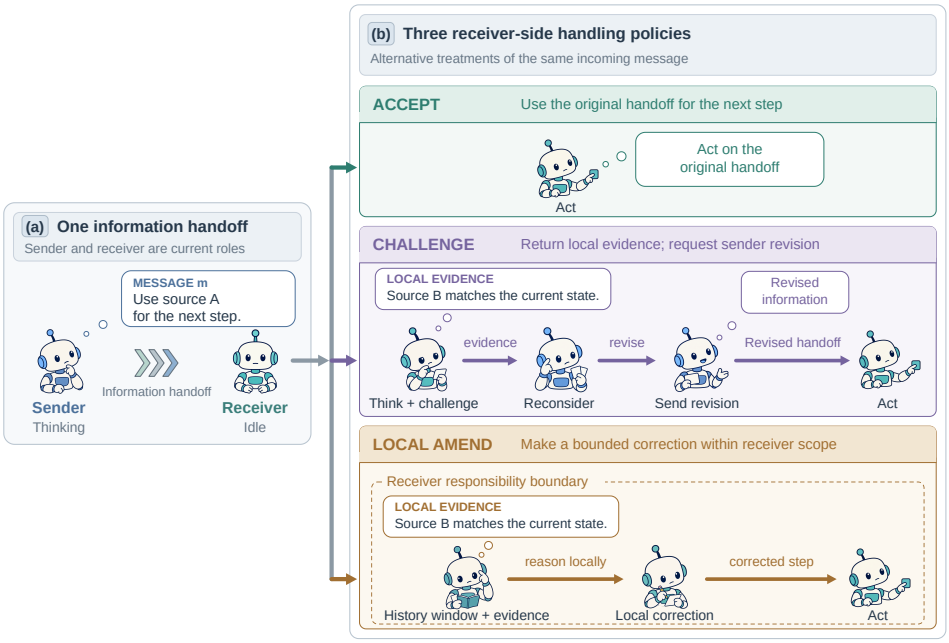

Figure 1: Three alternative treatments of the same information handoff. ACCEPT uses the incoming message, CHALLENGE returns local evidence for sender revision, and LOCAL AMEND makes a bounded correction within the receiver's responsibility.

local amendment can avoid an unnecessary round trip; if it detects the conflict but lacks repair information, sender revision is more appropriate; without concrete conflict evidence, intervention may add friction. We therefore formulate *receiver-side information handling* as a separately configurable design dimension at each information handoff. ACCEPT uses the sender's information as the default basis for execution; CHALLENGE returns local evidence and requests sender-side revision; LOCAL AMEND makes an evidence-supported change only within the receiver's responsibility boundary.

We evaluate these policies with Qwen3-Max and GPT-5.6-Luna across ALFWorld, AppWorld, ToolSandbox, and Collab-Overcooked (Shridhar et al., 2021; Trivedi et al., 2024; Lu et al., 2025; Sun et al., 2025). Paired end-to-end experiments measure task outcomes and token cost, while controlled ALFWorld handoffs isolate conflict detectability, evidence ownership, and receiver visibility. Fixed policies produce different outcomes and costs with no consistent winner. Controlled results show that conflict detection alone does not imply reliable local repair: sender revision is useful when repair information remains upstream, whereas local amendment becomes effective when relevant evidence is available to the receiver. Matched-revision controls further show that diagnostic feedback contributes beyond merely granting another model call. These findings motivate an evidence-gated inline strategy that chooses among the three handling modes during receiver generation.

Our contributions are threefold: (1) we formalize receiver-side handling as an explicit per-handoff design dimension that separates acceptance, sender revision, and bounded local repair; (2) across two backbones and four environments, we connect configured handling to end-to-end outcomes and resource use, while controlled checkpoints identify how evidence ownership changes the useful repair path; and (3) we instantiate the resulting evidence-gated principle inline and evaluate its task-level behavior, route selection, and matched controls. Together, the results show that multi-agent design should specify not only how information is exchanged, but also how receivers decide what to do with it.

2 Related Work

Multi-agent organization and communication. LLM-based multi-agent systems structure collaboration through roles, workflows, and communication. AutoGen, MetaGPT, CAMEL, AgentVerse, and ChatDev make such organization programmable across conversational, role-based, and task-specialized settings (Wu et al., 2024; Hong et al., 2024; Li et al., 2023; Chen et al., 2024; Qian et al., 2024). A complementary line optimizes or analyzes who communicates with whom: AgentPrune, AMAS, GoAgent, MacNet, G-Designer, and recent propagation analyses study communication sparsity, topology, and information flow (Zhang et al., 2025a; Leong et al., 2025; Chen et al., 2026; Qian et al., 2025; Zhang et al., 2025b; Shen et al., 2025); see also the survey of Guo et al. (2024). Our focus is orthogonal: after a task-relevant handoff arrives, what operational authority should it have at the receiver?

Feedback, correction, and evidence use. Debate and evaluator ensembles show that multi-agent feedback can improve reasoning or assessment, but correction is not uniformly beneficial and can exhibit sycophancy or disagreement collapse (Du et al., 2024; Chan et al., 2024; Yao et al., 2025; Sinha, 2026). Self-refinement and tool-grounded critique likewise emphasize selective use of diagnostic evidence (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024; Huang et al., 2024); evidence-driven integration also appears in multi-agent reinforcement learning (Sun et al., 2024). Failure-analysis and belief-revision work studies inter-agent misalignment and revision of agent-held information (Cemri et al., 2025; Moore et al., 2026; Ye et al., 2026). Following interactive-agent evaluation such as ReAct (Yao et al., 2023), we intervene at executable handoffs by separating *whether* the receiver intervenes from *where* repair occurs, and measure immediate repair separately from realized handling, final task outcome, and cost.

3 Receiver-Side Information Handling: Formulation and Study Framework

We formalize a handoff as a local decision problem at the receiver. The purpose is not to posit a universally optimal policy, but to make explicit which information and costs can change the preferred repair path and to derive testable contrasts for the experiments.

3.1 Handoffs and Executable Handling Policies

At handoff t, sender s provides receiver r with a task-relevant message. The receiver conditions on its visible information and responsibility boundary:

$$h_t = (m_{s\to r,t}, I_t^r, B_t^r). \tag{1}$$

Here I_t^r contains receiver-visible observations, history, and tool feedback, while B_t^r limits which decisions the receiver may legitimately alter. Sender and receiver are exchange-specific roles, covering both planner–executor handoffs and bidirectional peer communication.

We study $\mathcal{P} = \{A, C, M\}$: ACCEPT ($A$) keeps the message as the default basis for execution; CHALLENGE (C) returns concrete local evidence and requests one sender revision; LOCAL AMEND (M) changes only the receiver's own next executable decision when an in-scope correction is supported. Routine parameter filling is not counted as amendment. NATIVE is a separate reference condition preserving the original system. Changing only this handling protocol leaves the task, roles, and environment interface fixed; induced revision calls and tokens are part of its system-level cost. Appendix F specifies environment interfaces. Selected execution excerpts illustrating these handling paths appear in Appendix G.

3.2 Evidence State and Cost-Aware Repair Choice

The receiver-side distinction can be summarized by an evidence state

$$e_t = (c_t, \ell_t, u_t), \tag{2}$$

where c_t indicates a concrete receiver-visible conflict, ℓ_t indicates reliable evidence for an in-scope local repair, and u_t indicates that disambiguating repair information is available upstream to the

sender. Operationally, C1 corresponds to $c_t = 1, \ell_t = 1$; C2 to $c_t = 1, \ell_t = 0, u_t = 1$; and C3 to $c_t = 0, \ell_t = 0$ despite a latent environment-record error. These variables describe information conditions rather than outcome-derived policy labels.

For controlled handoffs with a replay-validated repair set $\mathcal{R}_t$, define $R_t(p) = \mathbf{1}[a_{t+1}(p) \in \mathcal{R}_t]$ and handling cost $C_t(p)$ (e.g., calls or tokens). A cost-aware decision lens is

$$\pi^\star(e) = \arg\max_{p \in \mathcal{P}_B} \Big\{ \Pr(R_t(p) = 1 \mid e_t = e) - \lambda\, \mathbb{E}[C_t(p) \mid e_t = e] \Big\}, \tag{3}$$

where $\mathcal{P}_B$ contains only responsibility-boundary-respecting choices and $\lambda \geq 0$ expresses the analysis trade-off rather than a fitted training objective. Thus a fixed policy can dominate only if the same choice maximizes this quantity across relevant evidence states. The formulation predicts instead that local repair becomes more attractive when ℓ_t is available, sender-side revision becomes more attractive when repair evidence remains upstream, and unsupported intervention should not be rewarded merely for occurring. Section 5 instantiates this principle with an inline evidence-gated policy; Equation 3 is the study lens, not an additional router.

3.3 Controlled Conditions and Evaluation Estimands

We construct replay-validated ALFWorld checkpoints under C1 (receiver-local repairable), C2 (sender-revision appropriate), and C3 (locally undetectable), pairing each Fault message with a state-consistent Clean message. Assignment uses information visibility rather than policy outcomes. Environment replay verifies recoverability; staged human review first judges receiver-visible conflict and repairability, then inspects sender-only history to determine evidence ownership. Appendix A gives the construction and review workflow.

The controlled estimand isolates the handoff mechanism,

$$\tau^{\mathrm{imm}}_{p,q}(e) = \mathbb{E}[R_t(p) - R_t(q) \mid e_t = e], \qquad \tau^{\mathrm{e2e}}_{p,q} = \mathbb{E}[Y_i(p) - Y_i(q)], \tag{4}$$

where $Y_i(p)$ is the complete-task endpoint under policy p. The first restores the same checkpoint and message across branches; the second starts from the same task and initial state but permits trajectories to diverge. Their separation is deliberate: τ^{imm} tests the proposed evidence–repair mechanism, whereas τ^{e2e} measures the total system consequence. Appendix A.3 specifies repair-set construction and scoring.

4 Empirical Characterization and Mechanism Analysis

4.1 Experimental Setup and Measurement

Environments and backbones. We evaluate planner–executor systems in ALFWorld (Shridhar et al., 2021), AppWorld (Trivedi et al., 2024), and ToolSandbox (Lu et al., 2025). Collab-Overcooked (Sun et al., 2025) provides a two-agent peer-collaboration setting. Using Qwen3-Max and GPT-5.6-Luna separately, we evaluate 50 ALFWorld tasks, 30 AppWorld tasks, 30 ToolSandbox scenarios, and all 30 Collab-Overcooked recipes. The three fixed policies and NATIVE yield 1,120 completed end-to-end trajectories; adaptive results appear in Section 5.

Matched interfaces. Within each environment–backbone stratum, policies share tasks, initial states, tools, output limits, and environment budgets, with temperature configured as zero. Planner–executor interfaces combine longer planner histories with executor-side execution information; Collab-Overcooked retains native role asymmetries. In ToolSandbox, the backbone also drives the user simulator, so cross-model comparisons concern the whole simulated system. Collab-Overcooked NATIVE preserves its original communication rhythm and serves as an end-to-end reference without matching model-call counts. Appendix A.1 specifies task selection and strata; Appendix B.1 details visibility, decoding, budgets, and analysis-specific procedures.

Endpoints and inference. We retain native environment metrics and stratify analyses by backbone. ToolSandbox's combined similarity measures minefield-constrained partial completion, not success probability; strict success requires a score of 1.0. Mechanism analyses pair tasks or checkpoints, using exact McNemar tests for binary outcomes, Holm adjustment within specified comparison families, and paired confidence intervals. Branches, intervention events, and model calls are not independent samples; these analyses do not estimate uncertainty from repeated API sampling.

Table 1: Fixed-policy end-to-end task outcomes.

Environment / metric	Model	NATIVE	ACCEPT	CHALLENGE	LOCAL AMEND
ALFWorld Success (%) ↑ · n = 50	Qwen	62.0	50.0	62.0	**72.0**
	Luna	78.0	76.0	82.0	**86.0**
AppWorld Success (%) ↑ · n = 30	Qwen	33.3	23.3	**43.3**	**43.3**
	Luna	50.0	43.3	43.3	**53.3**
ToolSandbox Similarity ↑ · n = 30	Qwen	0.642	0.698	**0.831**	0.808
	Luna	0.745	0.702	**0.755**	0.714
Collab-Overcooked Success (%) ↑ · n = 30	Qwen	76.7	**80.0**	**80.0**	76.7
	Luna	80.0	73.3	**90.0**	80.0

Success is a percentage; ToolSandbox uses native similarity. Bold marks the row maximum, not significance. Strict completion and tokens are in Appendix B.2.

Table 2: ALFWorld immediate repair on C1/C2 Fault handoffs.

Condition	Model	ACCEPT	CHALLENGE	LOCAL AMEND
C1 · receiver-local n = 25 per model	Qwen	23/25 (92%)	23/25 (92%)	**25/25 (100%)**
	Luna	17/25 (68%)	24/25 (96%)	**25/25 (100%)**
C2 · sender-informed n = 25 per model	Qwen	6/25 (24%)	**24/25 (96%)**	7/25 (28%)
	Luna	1/25 (4%)	**24/25 (96%)**	4/25 (16%)

Entries are repairs / n on Fault handoffs. Repair requires the first executed action to belong to the frozen replay-validated set. Clean and C3 controls appear in Table B3.

4.2 Handling Choices Change End-to-End Behavior and Cost

Table 1 establishes that assigning a receiver-side handling rule changes end-to-end outcomes. On ALFWorld, LOCAL AMEND has the highest success point estimate for both backbones (72.0% Qwen; 86.0% Luna), compared with 50.0% and 76.0% under ACCEPT. The ordering is not universal: CHALLENGE and LOCAL AMEND tie for Qwen in AppWorld, CHALLENGE has the highest ToolSandbox similarity for both backbones, and Luna CHALLENGE reaches 90.0% in Collab-Overcooked.

Handling also changes computational cost, and performance and cost do not move monotonically together. In ALFWorld, LOCAL AMEND improves the success point estimate over ACCEPT while using 22.2%/17.4% fewer tokens for Qwen/Luna. In Luna Collab-Overcooked, CHALLENGE raises success from 73.3% to 90.0% while using 10.1% fewer tokens than ACCEPT; in ToolSandbox, its higher similarity instead costs 10.2%/12.8% more tokens. Receiver-side handling therefore affects both task behavior and resource use, but no fixed rule dominates both dimensions. The complete token table is reported in Appendix B.2; task-paired outcomes and realized handling are examined in Appendices D and E.

4.3 Evidence Ownership Conditions the Value of Repair Paths

We compare three evidence-ownership conditions, holding each checkpoint fixed across its message and policy branches. Table 2 summarizes the ALFWorld panel of 75 reviewed checkpoints (25 each in C1, C2, and C3), evaluated with Clean/Fault messages and all three policies. C1 makes a correction locally identifiable to the receiver; C2 places the disambiguating repair evidence only in the sender's longer history; C3 lacks locally diagnostic evidence. Category assignment uses information visibility and replay checks rather than policy outcomes; construction details are in Appendix B.3.

The contrast follows evidence ownership. In C2, where repair information remains upstream, CHALLENGE repairs 24/25 handoffs for both backbones, versus 6/25 and 1/25 under ACCEPT and 7/25 and 4/25 under LOCAL AMEND (Qwen/Luna). In C1, where repair evidence is already local, LOCAL AMEND repairs 25/25 for both backbones while requiring one initial call and about 1.39k tokens, compared with three calls and about 3.23–3.27k tokens for CHALLENGE. Thus, the useful repair path depends on where reliable evidence resides, not on a globally superior policy.

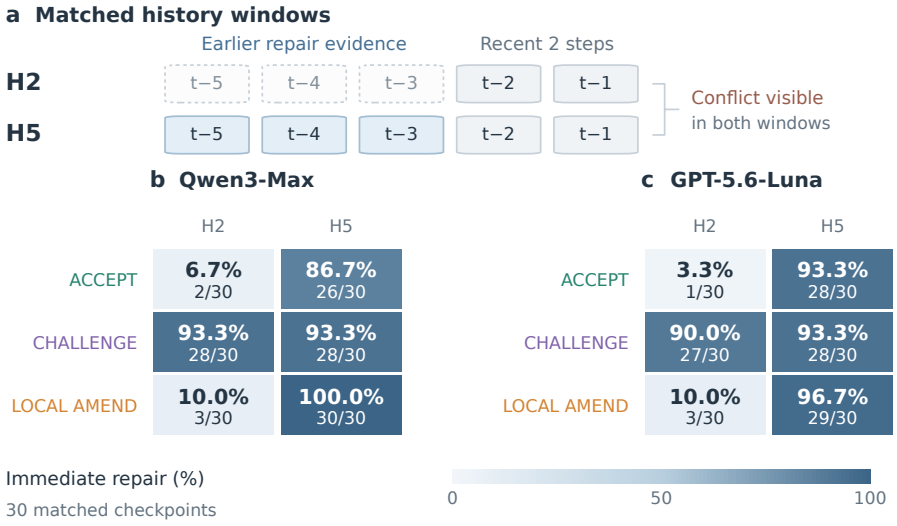

Figure 2: Evidence-window crossover on matched Fault handoffs. (a) H2 and H5 expose the same recent conflict, while H5 additionally reveals earlier repair evidence. (b,c) Immediate repair for Qwen3-Max and GPT-5.6-Luna; each cell reports the rate and count over 30 matched checkpoints.

C3 and Clean controls show why this is an evidence-conditioned effect rather than a generic benefit of intervention. ACCEPT executes the reference action on all 75 Clean handoffs for each backbone, while intervention policies occasionally depart unnecessarily (Qwen: 70/75 CHALLENGE, 68/75 LOCAL AMEND; Luna: 75/75 and 74/75). A small ToolSandbox C2 replication is directionally consistent—CHALLENGE repairs 3/12 versus 0/12 for either alternative under both backbones—but is statistically inconclusive. Its construction and replication details are reported in Appendix B.4.

4.4 Moving Repair Evidence to the Receiver Changes Policy Effects

To change evidence availability within the same state, we reuse 30 reviewed ALFWorld checkpoints and vary only the receiver's initial history window. H2 exposes the conflict but hides earlier source-location evidence; H5 reveals that evidence, while the sender remains at ten steps. All branches then use the same NATIVE Planner-10/Executor-5 continuation, yielding 360 branches per backbone.

Figure 2 shows that the effect changes sharply when the receiver gains repair evidence. LOCAL AMEND rises from 3/30 to 30/30 for Qwen and from 3/30 to 29/30 for Luna, whereas CHALLENGE remains near 90–93%. The pre-specified LOCAL AMEND–CHALLENGE interaction is +90.0 percentage points for Qwen (95% CI [76.7, 100.0]; Holm $p = 8.94 \times 10^{-8}$) and +83.3 for Luna ([70.0, 96.7]; $p = 2.38 \times 10^{-7}$). Moving evidence to the receiver therefore changes the relative value of local versus sender-side repair.

$$\Delta_{\text{window}} = (R_{M,H5} - R_{M,H2}) - (R_{C,H5} - R_{C,H2}) \tag{5}$$

M denotes LOCAL AMEND; C denotes CHALLENGE. The interaction is measured on Fault handoffs.

At H5, LOCAL AMEND requires one initial call and about 1.40k tokens, versus three calls and 3.25–3.33k tokens for CHALLENGE; their immediate-repair point estimates are not statistically distinguishable. ACCEPT also improves at H5, and final success under the common continuation changes little. The controlled conclusion is therefore an information–handling interaction at the handoff: access to relevant evidence changes which intervention route is efficient. Because H5 adds relevant history rather than context length alone, we do not interpret this as a pure window-size effect. Supplementary recovery, final-success, cost, and Clean-control results are in Appendix B.5.

4.5 Diagnostic Feedback Contributes Beyond a Revision Opportunity

A final control asks whether CHALLENGE helps merely because it grants the sender another call. On 60 paired ALFWorld checkpoints, both arms receive exactly one revision opportunity. Concrete receiver evidence improves immediate repair over a generic reconsideration request (Qwen 42/60 vs. 35/60; Luna 52/60 vs. 36/60), and diagnostic evidence improves over a truthful but weak fact (45/60 vs. 30/60; 54/60 vs. 31/60); both contrasts in the second study survive Holm correction. Final task success changes little, so the strongest supported effect is better immediate repair and revision quality rather than a generic extra-computation benefit. Error taxonomy, human review, prompt matching, full statistics, and final-success endpoints are reported in Appendix B.6.

4.6 Synthesis: From Handling Rules to Evidence-Conditioned Design

Together, the results characterize receiver-side information handling as a distinct system design dimension. Fixed rules change end-to-end outcomes and cost; matched handoffs show that the useful repair route depends on evidence ownership; moving evidence within the same checkpoint changes policy effects; and matched-call controls show that diagnostic content matters beyond an extra revision opportunity. This motivates an evidence-conditioned principle: preserve message-guided execution without concrete conflict, allow bounded local repair when the receiver has sufficient evidence, and request sender revision when the needed information remains upstream. Section 5 evaluates an adaptive implementation of this principle.

5 Evidence-Gated Adaptive Receiver-Side Handling

Section 4 shows that receiver-side handling changes system behavior, yet no fixed policy dominates across environments and backbones. This motivates evidence-gated adaptation: intervene only when the receiver sees a concrete conflict, and assign repair according to whether it has reliable in-scope evidence. We evaluate the resulting inline policy at three levels—complete tasks, controlled route selection, and intervention-matched controls—to test both system-level utility and the mechanism behind the choice.

5.1 Evidence-gated choice within the receiver's generation

At handoff h_t, the receiver ACCEPTs when no concrete local conflict is visible; LOCAL AMENDs when a conflict is visible and a reliable correction is supported within its own action scope; otherwise it CHALLENGEs for sender-side revision. Uncertainty alone is not conflict, and detecting an error does not imply knowing a safe repair.

$$\pi_{\mathrm{EG}}(h_t) = \begin{cases} \text{ACCEPT}, & c_t = 0, \\ \text{LOCAL AMEND}, & c_t = 1 \text{ and } \ell_t = 1, \\ \text{CHALLENGE}, & \text{otherwise.} \end{cases} \tag{6}$$

Here c_t denotes visible conflict and ℓ_t denotes a reliable in-scope repair. Both are judgments produced inside the receiver's ordinary generation—not external labels, calibrated probabilities, or additional model calls.

inline_adaptive performs this choice inside the receiver's ordinary generation. ACCEPT and LOCAL AMEND directly yield an executable decision; CHALLENGE triggers the Section 3 revision exchange, whose calls and tokens are counted. The policy therefore changes only how incoming information is handled, not the user goal, tool permissions, or role boundaries. Figure 3 summarizes this one-generation routing logic; implementation and evaluation boundaries are detailed in Appendix C.1.

5.2 End-to-end performance and token cost

We evaluate adaptive handling on the same four environment panels and two backbones as Section 4. Within each environment–backbone stratum, Table 3 compares it with the highest-scoring fixed/ref-

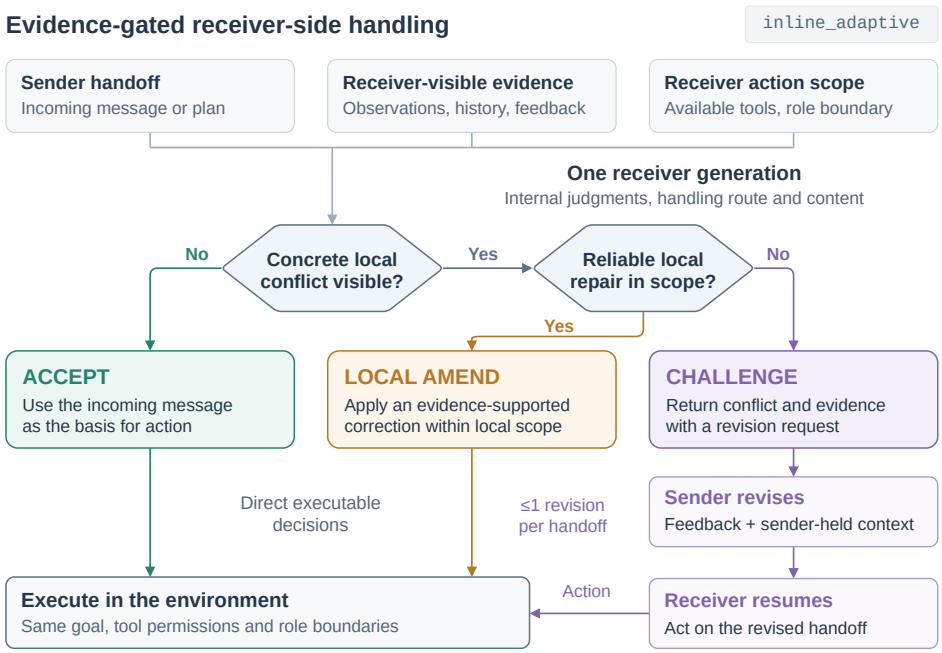

Figure 3: Evidence-gated handling within one receiver generation. Internal conflict and repair judgments determine the route and its content. ACCEPT and LOCAL AMEND yield executable decisions; CHALLENGE adds at most one sender-revision exchange before receiver execution.

Table 3: Adaptive performance and cost against the best fixed/reference condition.

Model	Fixed rule	Fixed score	Adaptive score	Fixed tokens (k)	Adaptive tokens (k)	Token change
ALFWorld · Success (%) · n = 50						
Qwen	M	72.0	78.0	50.31	65.70	+30.6%
Luna	M	86.0	86.0	37.02	41.24	+11.4%
AppWorld · Success (%) · n = 30						
Qwen	M	43.3	46.7	203.71	219.66	+7.8%
Luna	M	53.3	53.3	392.87	294.53	-25.0%
ToolSandbox · Similarity · n = 30						
Qwen	C	0.8311	0.8579	21.66	20.55	-5.1%
Luna	C	0.7547	0.7526	21.48	22.97	+7.0%
Collab-Overcooked · Success (%) · n = 30						
Qwen	A	80.0	80.0	197.07	217.15	+10.2%
Luna	C	90.0	93.3	210.53	262.45	+24.7%

A = ACCEPT, C = CHALLENGE, M = LOCAL AMEND; NATIVE is eligible. The comparator maximizes score and breaks ties by lower token use. Token change = 100 × (adaptive/fixed - 1); similarity is not a success percentage. Values are point estimates.

erence condition, breaking score ties by lower mean token use. This is a retrospective aggregate comparator, not a per-task oracle; native environment metrics are retained.

Across the eight environment–backbone combinations, adaptive handling matches or exceeds the best fixed/reference point estimate in seven: four higher, three tied, and one lower. Representative gains include Qwen ALFWorld (78.0% vs. 72.0%), Qwen AppWorld (46.7% vs. 43.3%), and Luna Collab-Overcooked (93.3% vs. 90.0%). Qwen ToolSandbox also improves similarity from 0.8311 to 0.8579 while using 5.1% fewer tokens; Luna AppWorld maintains 53.3% success while using 25.0% fewer tokens. Other gains sometimes require more computation, and Luna ToolSandbox is slightly below its reference (0.7526 vs. 0.7547). Thus, the adaptive policy is competitive overall, but

the result is a context-dependent performance–cost trade-off rather than universal dominance. Full five-condition results are reported in Appendix C.2.

5.3 Controlled route selection

To verify that the deployed policy does not collapse to one route, we evaluate a Qwen-only controlled set of 30 checkpoints paired with Clean/Fault messages (60 one-generation inputs). inline_adaptive matches the expected route on 56/60 inputs (93.3%), selects all three routes, and produces 60/60 protocol-valid outputs. All ten expected LOCAL AMEND cases are correct; the four errors occur at the CHALLENGE boundary. This test establishes route-selection fidelity, not downstream repair success. The full protocol, class counts, and confusion matrix are deferred to Appendix C.3.

5.4 Evidence-gated handling under matched intervention frequencies

On the evidence-regime and evidence-window development panels, we compare inline_adaptive with frequency-matched random handling, using one frozen random schedule per backbone. Route and model-call counts are matched within each backbone and panel. Across both backbones and panels, inline_adaptive raises immediate reference-action accuracy by 10.8–22.0 percentage points (all reported 95% checkpoint-bootstrap intervals above zero) and lowers Clean-message intervention by 28.0–60.0 points. These results favor the complete evidence-conditioned handling protocol at matched frequencies. They do not isolate intervention placement, because route instructions and feedback content also differ. Token use is 5.8% higher on Qwen and 10.1% higher on Luna. Appendices C.4–C.5 detail construction, matching, costs, and uncertainty.

Together, these results make receiver-side handling an explicit per-handoff system decision rather than a rule fixed for an entire run. Controlled routing shows that ACCEPT, CHALLENGE, and LOCAL AMEND can all be realized; end-to-end evaluation shows that adaptive selection remains competitive across environments; and matched controls favor evidence-conditioned handling at equal route frequencies and model-call counts. The contribution is therefore the evidence-gated design principle and the explicit configurability of receiver-side handling, not a claim that adaptation is always more accurate, cheaper, or algorithmically complex.

6 Discussion and Scope

Receiver-side information handling is a system-level interface choice: local evidence determines whether intervention is justified, while evidence ownership conditions where repair should occur. End-to-end and controlled results jointly support this view: fixed handling rules change outcomes and cost, moving diagnostic evidence changes relative policy effects, and matched controls show that feedback content matters beyond an extra revision opportunity. The inline evidence-gated policy is one implementation of this principle rather than the central contribution. The evaluation spans two backbones and four environments, but controlled panels intentionally concentrate diagnostic handoffs and should not be interpreted as prevalence estimates. Broader model families, communication topologies, and deployment settings remain to be tested. Within the evaluated settings, receiver-side handling materially changes behavior, outcomes, and computation and should therefore be specified explicitly rather than treated as an incidental prompting detail.

7 Conclusion

We identify receiver-side information handling as a distinct design dimension in multi-agent systems. Across multiple environments and model backbones, different handling policies alter execution trajectories, task outcomes, and computational costs, while no fixed policy consistently dominates. Controlled experiments further show that effective handling depends on the receiver's available evidence and repair responsibility. Motivated by these findings, we implement an evidence-gated adaptive strategy that selects among ACCEPT, CHALLENGE, and LOCAL AMEND during execution. Overall, our results suggest that multi-agent coordination should explicitly model not only what information is communicated, but also how the receiver decides to act on it.

Reproducibility Statement

The anonymous repository provides code and reproduction instructions. Sections 3–5 specify the handling treatments and evaluation endpoints. Appendices A–C give controlled constructions, inference details, and complete fixed/adaptive results; Appendices D–G provide task-paired analyses, process audits, the interface/policy reference, and execution excerpts. The source bundle includes the figures, tabular summaries, prompt reference, and selected trajectory excerpts used in this manuscript.

Generative AI Use Statement

Generative AI was used to assist with manuscript polishing and organization, English-language refinement, development of figure concepts and layouts, and implementation of research ideas. It was not used to directly generate the figures included in the paper. The experimental agents studied in this work are themselves language models. The authors remain fully responsible for the final manuscript, experimental claims, and all submitted materials.

Ethics Statement

The experiments use benchmark environments and simulated interfaces. Responsibility boundaries are part of the handling policy; benchmark results do not establish safe autonomous deployment in consequential real-world systems. The selected excerpts omit authentication secrets and unnecessary account details.

References

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A. Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why Do Multi-Agent LLM Systems Fail? In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4082. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b1041e52d3be19f0a9bc491657488e4a-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. In *The Twelfth International Conference on Learning Representations*, pp. 9079–9093, 2024. URL https://openreview.net/forum?id=FQepisCUWu.

Hongjiang Chen, Xin Zheng, Yixin Liu, Pengfei Jiao, Shiyuan Li, Huan Liu, Zhidong Zhao, Ziqi Xu, Ibrahim Khalil, and Shirui Pan. GoAgent: Group-of-Agents Communication Topology Generation for LLM-based Multi-Agent Systems. arXiv preprint arXiv:2603.19677, 2026. URL https://arxiv.org/abs/2603.19677.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/578e65cdee35d00c708d4c64bce32971-Abstract-Conference.html.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *The*

Twelfth International Conference on Learning Representations, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large Language Model Based Multi-agents: A Survey of Progress and Challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8048–8057, 2024. doi: 10.24963/ijcai.2024/890. URL https://www.ijcai.org/proceedings/2024/890. Survey Track.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations*, pp. 23247–23275, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large Language Models Cannot Self-Correct Reasoning Yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. AMAS: Adaptively Determining Communication Topology for LLM-based Multi-agent System. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2061–2070. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-industry.144. URL https://aclanthology.org/2025.emnlp-industry.144/.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008. Curran Associates, Inc., 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

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/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Keith Moore, Jun Kim, David Lyu, Jeffrey Heo, and Ehsan Adeli. Ask WhAI: Probing Belief Formation in Role-Primed LLM Agents. In *Proceedings of AI for Accelerated Research Symposium*, volume 3 of *EPiC Series in Technology*, pp. 121–137. EasyChair, 2026. doi: 10.29007/qcnq. URL https://easychair.org/publications/paper/BTN7.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling Large Language Model-based Multi-Agent Collaboration. In *The Thirteenth International Conference on Learning Representations*, pp. 41488–41505, 2025. URL https://openreview.net/forum?id=K3n5jPkrU6.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the Information Propagation Effects of Communication Topologies in LLM-based Multi-Agent Systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12347–12361. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.623. URL https://aclanthology.org/2025.emnlp-main.623/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *The Ninth International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Debu Sinha. SycoBench-600: Measuring Sycophancy and Correction Selectivity in LLM Assistants. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 35278–35284. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1759. URL https://aclanthology.org/2026.findings-acl.1759/.

Chuxiong Sun, Zehua Zang, Jiabao Li, Jiangmeng Li, Xiao Xu, Rui Wang, and Changwen Zheng. T2MAC: Targeted and Trusted Multi-Agent Communication through Selective Engagement and Evidence-Driven Integration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (13):15154–15163, 2024. doi: 10.1609/aaai.v38i13.29438. URL https://ojs.aaai.org/index.php/AAAI/article/view/29438.

Haochen Sun, Shuwen Zhang, Lujie Niu, Lei Ren, Hao Xu, Hao Fu, Fangkun Zhao, Caixia Yuan, and Xiaojie Wang. Collab-Overcooked: Benchmarking and Evaluating Large Language Models as Collaborative Agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4922–4951. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.249. URL https://aclanthology.org/2025.emnlp-main.249/.

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/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Binwei Yao, Chao Shang, Wanyu Du, Jianfeng He, Ruixue Lian, Yi Zhang, Hang Su, Sandesh Swamy, and Yanjun Qi. Peacemaker or Troublemaker: How Sycophancy Shapes Multi-Agent Debate. arXiv preprint arXiv:2509.23055, 2025. URL https://arxiv.org/abs/2509.23055.

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

Jingwei Ye, Zhi Wang, Xin Li, Cong Gao, Chenbin Su, Jieshuai Yang, Jianfei Tang, and Ge Chu. MACAA: Belief-Revision Multi-Agent Reasoning for Code Authorship Verification. arXiv preprint arXiv:2605.09421, 2026. URL https://arxiv.org/abs/2605.09421.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the Crap: An Economical Communication Pipeline for LLM-based Multi-Agent Systems. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bbc461518c59a2a8d64e70e2c38c4a0e-Abstract-Conference.html.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-Designer: Architecting Multi-agent Communication Topologies via Graph Neural Networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76678–76692, 2025b. URL https://proceedings.mlr.press/v267/zhang25cu.html.

Appendix roadmap. Appendix A describes task sampling, checkpoint construction, and the frozen immediate-repair criterion; Appendix B reports fixed-policy protocols and supporting mechanism results; Appendix C gives complete adaptive outcomes and matched controls; Appendix D provides task-paired visualizations; Appendix E documents realized handling; Appendix F consolidates execution interfaces and policy contracts; and Appendix G presents boxed execution excerpts.

A Dataset Sampling and Controlled-Handoff Construction

We use complementary sampling designs for complete-task evaluation and controlled mechanism tests. The end-to-end panels cover fixed benchmark task strata, whereas the checkpoint panels deliberately vary conflict visibility and repair-evidence ownership. The latter are diagnostic panels, not estimates of the natural prevalence of handoff errors.

A.1 Frozen End-to-End Task Panels

All policies and both backbones use the same task identifiers within each environment. Selection is specified by task metadata and frozen sampling orders, independently of the compared policy outcomes. Table A1 gives the allocation; strata are interpreted within each environment rather than pooled into a single cross-benchmark success rate.

Table A1: Composition of the fixed end-to-end evaluation panels.

Environment	Tasks	Prespecified allocation
ALFWorld	50	Simple pick-and-place: 9; look-at-in-light: 9; clean, cool, heat, and pick-two: 8 each.
AppWorld	30	15 test-normal and 15 test-challenge tasks; distinct task generators within each split.
ToolSandbox	30	Insufficient information: 6; state dependency: 4; canonicalization: 11; standard: 9.
Collab-Overcooked	30	All 30 recipes in the evaluation suite.

ALFWorld. Tasks come from the valid-unseen split. A seeded sampler (20260904) allocates eight tasks to each of the six task families and assigns the two remaining slots to two families, producing the counts above. Within a family, it visits distinct goal/floorplan groups before selecting another trial from a previously used group. This balances task-family coverage while limiting repeated goal configurations; it does not reproduce the benchmark's natural family frequencies.

AppWorld. A frozen parent ordering uses seed 20260908 for test-normal and 20260909 for test-challenge. Task generators are shuffled, and one task variant is selected per generator. After excluding five development tasks, the formal panel takes the next 15 eligible, previously uninspected tasks in each split. The 15:15 allocation gives the normal and challenge settings equal coverage, without choosing tasks on the basis of success under any handling policy.

ToolSandbox. The candidate set contains 129 base scenarios. Excluding 34 scenarios that depend on network tools leaves 95 eligible scenarios. Scenarios are assigned to mutually exclusive buckets in the priority order insufficient information, state dependency, canonicalization, and standard, then ordered using seed 20260908. The 30-scenario panel takes the prescribed count from each frozen bucket after excluding the five calibration scenarios. This keeps formal evaluation disjoint from calibration while covering several tool-interaction demands; it targets the offline-tool subset rather than all ToolSandbox capabilities.

Collab-Overcooked and paired execution. All 30 recipes are retained, with their native resource ownership and asymmetric role knowledge. The 140 task instances across environments yield 1,120 fixed/reference trajectories (140 tasks $\times$ four conditions $\times$ two backbones), plus 280 adaptive trajectories. The independent paired unit remains a task within an environment; trajectories, calls, and interventions are repeated observations rather than additional tasks.

A.2 Evidence-Ownership Construction

The ALFWorld evidence-ownership panel contains 75 distinct task checkpoints, with 25 in each of C1, C2, and C3. Selection uses checkpoint state, information visibility, and replay checks rather than policy success. Unmodified prefixes contribute 22/3/25 checkpoints across C1/C2/C3; controlled detours contribute 3/22/0. Both sources use constructed Fault messages. Condition balance therefore supports matched diagnostic contrasts, not prevalence estimation.

Each checkpoint fixes its task, action prefix, state identifier, observation, admissible actions, and remaining budget. Clean and Fault counterparts differ only in the incoming message. Under Planner-10/Executor-5, sender-only evidence must lie in the sixth-to-tenth most recent completed steps and must not already be revealed by the task or receiver view. Detours move evidence across this visibility boundary while preserving task-relevant object facts. A task contributes at most one selected checkpoint.

Review proceeds in two stages. The receiver-only stage examines the task, Fault message, recent receiver history, current observation, and admissible actions, recording the evidence supporting conflict detection and local repairability. The sender-context stage then checks whether the additional history supplies the missing disambiguating fact. Ambiguous ownership, unrecoverable replay, answer leakage, or an unverified repair prevents inclusion. Repeated replay checks validate recovery of the same state and do not increase the sample size.

A.3 Frozen Acceptable Repair Sets and Immediate Scoring

For each eligible checkpoint i, the construction record fixes the replay prefix, environment state, incoming Clean/Fault messages, receiver-visible context, and acceptable first-action set $\mathcal{R}_i$ before the compared policy outcomes are inspected. The set is an operational reference for the constructed handoff, not an exhaustive catalogue of every valid long-horizon strategy.

1. **Recover the checkpoint.** Replay the recorded prefix and check the resulting observation, admissible-action interface, and task-relevant state. An unrecoverable or unstable checkpoint is excluded.
2. **Identify candidate repairs from the construction.** Use the state-consistent counterpart and the diagnosed error to specify an executable step that addresses the conflict within the receiver's action scope. The information-condition label is established from the information available to each role, not from whether a tested policy succeeds.
3. **Validate membership by replay.** Check that each retained candidate is executable at the restored checkpoint and provides an acceptable repair under the study's construction checks. The feedback-content panel additionally replays repair-plus-expert continuations and verifies acceptable aliases twice: 51 checkpoints have one accepted action and nine have two to five. These alias counts apply to that 60-checkpoint panel, not to every experiment.
4. **Freeze before comparison.** Keep the resulting set and evaluation labels outside model inputs. Do not add an action to the reference set after observing a policy's output merely to change its score. Replay validation establishes the recorded criterion, not uniqueness among all possible repairs.

The immediate endpoint is

$$Y_i^{\text{repair}} = \mathbf{1}\left[a_i^{\text{first}} \in \mathcal{R}_i\right], \tag{7}$$

where a_i^{first} is the first environment action actually executed after handling, including any allowed sender revision. Emitting a challenge or revised plan alone is not a repair. An executable action outside the frozen set is a nonmatch under this operational measure, even if an alternative strategy could eventually work. C3 Fault handoffs do not have a locally justified immediate repair reference: the endpoint is undefined and is not counted as zero. Recovery windows, semantic review, and final task success are reported separately. ToolSandbox operational matching and its post-hoc semantic sensitivity check likewise remain distinct.

Table A2: Error families in the 60-task feedback-content panel. Each contributes ten tasks.

Family	Constructed conflict
Wrong source location	The handoff names a source inconsistent with the available object-location evidence.
Closed-container prerequisite	The requested retrieval omits opening the currently closed container.
Stale duplicate pickup	The message requests an object that the receiver already holds.
Stale negative search	The message repeats a search despite recent negative evidence.
Omitted required transformation	Placement is requested before the required heating, cooling, or cleaning.
Simulator-affordance mismatch	A proposed real-world procedure conflicts with the simulator's available operation.

A.4 Evidence-Window Construction

The H2/H5 crossover retains the same checkpoint, incoming message, task, current observation, executable actions, model, and output budget. The treatment changes the receiver's recent-history field. In the construction procedure, a source-observation checkpoint is followed by a detour to another location and a look action, placing the source evidence in the third-to-fifth most recent steps. The short window must expose a concrete conflict without uniquely identifying the repair; the longer window must add the missing repair evidence. Checkpoints with a uniquely determined H2 repair, mismatched object identity, unstable state replay, or a repair that cannot be followed to completion within the prescribed action budget are excluded. The formal panel contains 30 task checkpoints: 11 simple pick-and-place, ten look-at-in-light, and nine clean-then-place tasks. These task-family counts result from the visibility and replay eligibility requirements, rather than a target benchmark-frequency distribution. Only the controlled handoff uses the assigned window and handling rule; subsequent continuation uses the common Native Planner-10/Executor-5 interface.

A.5 Feedback-Content Construction

The evidence-versus-reconsideration study contains 60 distinct ALFWorld tasks, with ten handoffs from each of six error families (Table A2). Candidates are derived from replayable expert prefixes. Expert rollouts on 134 valid-unseen tasks yielded 506 candidate handoffs from 125 completed tasks; nine tasks did not complete within the 150-step candidate-generation budget. The six candidate-family counts were 123, 11, 125, 109, 63, and 75, respectively, in the order shown in Table A2. Diagnostic evidence states a receiver-visible conflict rather than providing the reference action verbatim. Candidate repairs and eligible aliases are checked by environment replay and subsequent expert completion. Inclusion and branch order are fixed without reading policy outcomes; the two feedback branches start from the same checkpoint and each permits one sender revision. The error-family breakdown describes the designed panel, not the natural frequency of these errors.

Selection uses seed 20260918 and first enforces ten tasks per error family. Among eligible candidates, it prioritizes avoiding previously used evidence-window tasks, broadening goal-group coverage, and balancing task families, with a fixed hash order resolving remaining ties. Fourteen attempted candidates are rejected before freezing because checkpoint recovery or repair-plus-continuation replay is unstable. The selected panel contains 60 distinct tasks across 44 goal groups: 14 clean, 12 heat, 11 cool, ten simple pick-and-place, eight look-at-in-light, and five pick-two tasks. Three tasks overlap an earlier evidence-window panel because of the limited closed-container candidate pool. Branch order alternates within error families, giving 30 evidence-first and 30 reconsideration-first pairs.

The relevance control reuses the 60 frozen checkpoints. Both branches retain the same challenge warning and one revision opportunity; only the receiver-local fact changes. The control fact states that the current admissible-action list includes `help`, which is true at these checkpoints and is not a member of their prespecified repair sets. The full revision inputs are matched by whitespace-delimited word count, not by a guarantee of identical provider-token counts. The control consequently tests informative conflict feedback against a true but nondiagnostic fact; it does not equate their semantic salience.

Table B1: End-to-end interfaces and budgets.

Environment	Panel	Planner / executor history	Episode budget	Output limit per call
ALFWorld	50 valid-unseen tasks	10 / 5 completed steps	50 environment steps	2,048 / 4,096 tokens
AppWorld	15 normal + 15 challenge	6 / 3 attempt records	50 attempts	1,024 / 2,048 tokens
ToolSandbox	30 scenarios	4 / 2 complete cycles	30 messages	1,024 / 2,048 tokens
Collab-Overcooked	All 30 recipes	Native peer histories	80 environment steps	2,048 tokens per agent

Planner/executor limits are listed in that order. ToolSandbox additionally limits the user simulator to 1,024 output tokens per call. Temperature is configured as zero; configuration seeds are 0 for ALFWorld and Collab-Overcooked, and 20260908 for AppWorld and ToolSandbox. These are configurations, not repeated-seed evaluations.

A.6 ToolSandbox Replication

The ToolSandbox replication applies the same information-ownership criterion using native tool interfaces. A read-only lookup exposes a repair-relevant parameter; a filler cycle moves it outside the receiver's two-cycle window, and a state-preserving failed call makes the conflict visible. The sender's four-cycle window retains the parameter. The 12 checkpoints belong to nine construction families; family-clustered intervals account for this shared construction. The frozen operational action-match endpoint and the subsequent semantic sensitivity analysis remain separate (Appendix B.4).

A.7 Freezing, Overlap, and Interpretation

Task identifiers, replay prefixes, incoming-message variants, evaluation labels, and operational repair references are fixed for the compared branches. Evaluation labels and reference answers are withheld from model inputs. Clean messages provide negative controls for unnecessary intervention, while Fault messages test repair under the specified information condition. Constructed eligibility and error-family balance make the mechanism observable; they do not establish benchmark-wide error prevalence.

The evidence-ownership and H2/H5 panels share 21 underlying task identifiers and are analyzed separately. The two feedback studies deliberately reuse their 60 checkpoints but execute separate paired trajectories. Counts from different panels, repeated conditions, or repeated replay checks are therefore not added to claim a larger independent sample. The adaptive controlled checks reuse these information conditions as described in Appendices C.3 and C.4.

B Fixed-Policy Protocols and Supplementary Results

B.1 Experimental Protocol and Measurement

The end-to-end panels use Qwen3-Max and GPT-5.6-Luna separately, with recorded API identifiers `qwen3-max-2026-01-23` and `gpt-5.6-luna`. Each task is evaluated under NATIVE, ACCEPT, CHALLENGE, and LOCAL AMEND. The total is $(50 + 30 + 30 + 30) \times 4 \times 2 = 1{,}120$ completed trajectories, excluding adaptive evaluations and controlled handoff branches. Table B1 summarizes the interfaces and budgets.

ALFWorld. The planner retains up to ten completed action–observation steps and the executor five. The executor also receives the current admissible-action list. **AppWorld.** Six/three recent attempt records contain code outputs or harness feedback. Both roles receive history-derived indices of variable mentions, API attempts, attempted side effects, and cached schema identifiers. The executor additionally receives retrieved API candidates and cached documentation. The indices do not independently verify live state or successful side effects. Invalid protocol attempts consume the attempt budget even when no code is executed.

ToolSandbox. Public user dialogue is retained for both roles; the recent tool history is windowed by complete interaction cycles, preserving associated calls and results. Only the executor receives the

Table B2: Initial handling cost and C2 downstream recovery.

Model	Policy	C1 repair /25	C1 initial tokens / calls	C2 repair /25	C2 recovery@3 /25	C2 recovery@10 /25	C2 final success /25
Qwen	ACCEPT	23	1,221 / 1	6	21	22	20
Qwen	CHALLENGE	23	3,228 / 3	24	24	25	22
Qwen	LOCAL AMEND	25	1,389 / 1	7	20	22	22
Luna	ACCEPT	17	1,265 / 1	1	21	22	20
Luna	CHALLENGE	24	3,266 / 3	24	24	24	22
Luna	LOCAL AMEND	25	1,390 / 1	4	21	23	20

Token means in the C1 column concern the first handoff. C2 recovery is based on a prespecified reference action. These denominators do not count model calls as independent samples.

current native tool-schema view. Policies start from copies of the same scenario snapshot. **Collab-Overcooked.** Native resource isolation and asymmetric task knowledge are retained without imposing planner–executor history windows.

Matched output and environment limits do not imply equal realized call counts or tokens. ToolSandbox uses the same backbone for its agents and user simulator, so a backbone change affects the whole simulated system. Collab-Overcooked NATIVE retains its original communication rhythm and is not a call-matched control. Native task metrics are retained; ToolSandbox similarity is minefield-constrained partial completion, while strict full success requires a score of 1.0.

Mechanism analyses pair tasks or checkpoints within each backbone. Binary comparisons use exact McNemar tests; confidence intervals preserve pairing, and Holm adjustment is applied within each stated comparison family. Window interactions use paired sign-flip tests. ToolSandbox replication intervals are clustered by construction family. The analysis-specific families and resampling settings are given below. Branches, events, and model calls are not independent samples, and repeated API-sampling uncertainty is not estimated.

B.2 Complete End-to-End Costs and Strict Completion

The complete five-condition cost and strict-success results are consolidated in Appendix C.2, Tables C2 and C3. They retain the same fixed-policy values used in the main text; aggregate task scores and token means do not supply paired uncertainty or token dispersion.

B.3 ALFWorld Evidence-Ownership Panel

Operational conditions and repair-set construction are defined in Section 3.2 and Appendix A.3. Figure B1 shows the paired repair contrasts; Tables B2 and B3 report handling cost, recovery, and controls.

The panel contains 75 unique checkpoints, with 25 in each evidence condition. Clean/Fault messages and three persistent handling policies produce 450 branches per backbone. Fifty checkpoints require no added detour; 25 use controlled detours that preserve task-relevant world state. Message errors are constructed in both strata. Assignment uses information visibility and replay validation, not observed policy success.

For C1/C2 Fault handoffs, a repair requires the first executed action after handling to belong to the frozen, replay-validated acceptable set. A challenge message or sender revision alone is insufficient. Recovery@k records whether the frozen clean-reference action is first reached within k post-handoff actions; it does not imply complete state restoration or final success. C3 immediate reference repair is undefined and is not scored as zero.

B.4 ToolSandbox C2 Replication

Twelve checkpoints from nine construction families use native ToolSandbox snapshots and tools. A controlled three-cycle prefix performs a successful read-only lookup, a filler interaction, and

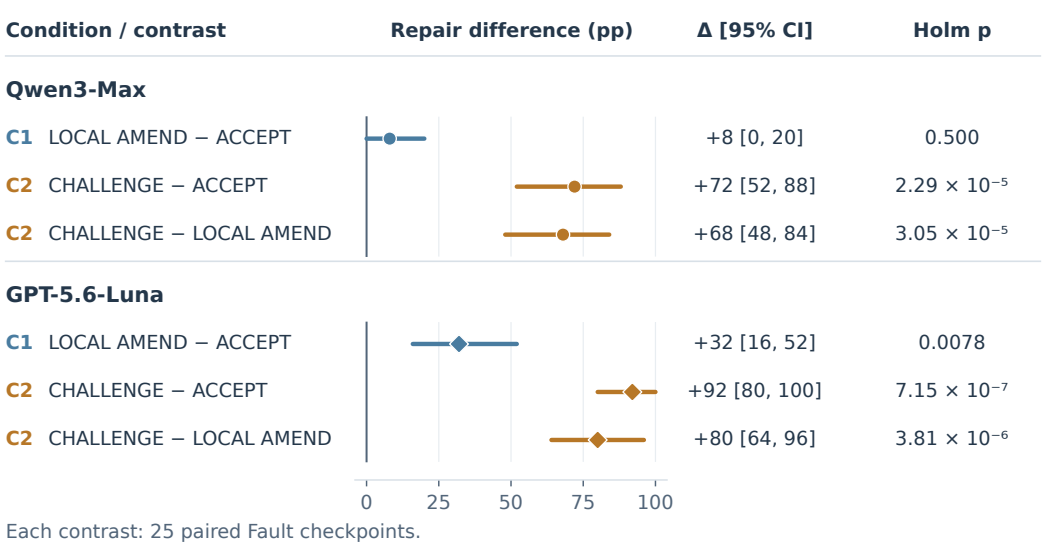

Figure B1: Principal paired immediate-repair contrasts (25 checkpoints per condition). Points show repair-rate differences in percentage points; horizontal lines show 95% paired nonparametric bootstrap intervals from 50,000 checkpoint resamples. P-values are two-sided exact McNemar tests with Holm adjustment across the three named contrasts within each backbone; this grouping was implemented after results and is exploratory rather than preregistered. Circles denote Qwen and diamonds Luna; color distinguishes C1 from C2 contrasts.

Table B3: Clean and C3 controls.

Model	Policy	All Clean reference /75	C3 Clean success /25	C3 Fault success /25	C3 Fault recovery@10 /25
Qwen	ACCEPT	75	18	11	7
Qwen	CHALLENGE	70	18	12	6
Qwen	LOCAL AMEND	68	19	13	8
Luna	ACCEPT	75	20	12	4
Luna	CHALLENGE	75	22	14	6
Luna	LOCAL AMEND	74	20	16	8

The Clean column measures first-action agreement, not merely whether an intervention marker was emitted. C3 final outcomes are secondary endpoints.

a state-preserving failed call. Under the 4/2-cycle sender/receiver windows, the repair parameter remains visible to the sender while both roles see the conflict. Clean/Fault × three policies yields 72 trajectories per backbone. The assigned rule remains active during subsequent replanning; later trajectories are not fixed.

Provider settings follow each backbone's deployment configuration, so replication comparisons are interpreted within backbone. The panel tests a controlled information asymmetry rather than its natural prevalence. Table B4 separates operational repair from reviewed repair.

B.5 Evidence-Window Construction and Supplementary Outcomes

The crossover uses 30 unique valid-unseen checkpoints: 11 simple pick-and-place, 10 look-at-in-light, and nine clean-then-place tasks. Controlled detours move the agent away from the correct source without changing task-relevant object facts. The correct-source evidence lies in the third-to-fifth most recent records. H2 reveals the conflict but not the source; H5 reveals it. Screening first assesses H2 and then the newly exposed records, followed by replay validation. This is a selected mechanism panel, not a random sample of the benchmark distribution.

Table B4: Immediate repair in the ToolSandbox replication.

Model	Policy	Operational repair	Reviewed repair
Qwen	ACCEPT	0/12	0/12
Qwen	CHALLENGE	3/12	2/12
Qwen	LOCAL AMEND	0/12	0/12
Luna	ACCEPT	0/12	0/12
Luna	CHALLENGE	3/12	3/12
Luna	LOCAL AMEND	0/12	0/12

Each row contains 12 checkpoints. The operational endpoint is the preregistered first-action repair measure; the reviewed column is a post-hoc semantic sensitivity check. The replication is small and is used as directional evidence for the handoff-level mechanism rather than as an end-to-end policy ranking.

Table B5: Fault recovery, final success, and first-handoff versus full-trajectory cost.

Model	Window	Policy	Initial repair	Recovery @3	Recovery @5	Final success	Initial calls	Initial tokens	Full logical tokens
Qwen	H2	ACCEPT	2/30	26/30	27/30	24/30	1.0	1,094	25,519
Qwen	H2	CHALLENGE	28/30	28/30	28/30	25/30	3.0	2,941	23,115
Qwen	H2	LOCAL AMEND	3/30	27/30	27/30	24/30	1.0	1,252	25,805
Qwen	H5	ACCEPT	26/30	29/30	29/30	24/30	1.0	1,222	24,534
Qwen	H5	CHALLENGE	28/30	28/30	28/30	25/30	3.0	3,245	23,938
Qwen	H5	LOCAL AMEND	30/30	30/30	30/30	24/30	1.0	1,399	24,002
Luna	H2	ACCEPT	1/30	27/30	28/30	24/30	1.0	1,090	24,842
Luna	H2	CHALLENGE	27/30	28/30	28/30	25/30	3.0	3,045	23,906
Luna	H2	LOCAL AMEND	3/30	24/30	26/30	24/30	1.0	1,305	27,775
Luna	H5	ACCEPT	28/30	28/30	29/30	23/30	1.0	1,232	26,667
Luna	H5	CHALLENGE	28/30	28/30	28/30	25/30	3.0	3,331	24,987
Luna	H5	LOCAL AMEND	29/30	29/30	30/30	24/30	1.0	1,397	27,204

All rates use 30 checkpoints. Logical tokens include the logical cost of cache-served responses. Initial handling has no cache hits. The main-text interaction appears in Figure 2.

Only the history field of the initial receiver prompt changes; the sender retains ten steps. Clean/Fault $\times$ H2/H5 $\times$ three policies gives 360 branches per backbone. After the controlled first action, every branch uses the same NATIVE Planner-10/Executor-5 continuation. H5 adds repair-relevant facts together with history, so the intervention is an information-access change rather than a pure context-length treatment. Table B5 reports Fault outcomes and costs; Table B6 gives the Clean controls.

The prespecified primary interaction is $(R_{M,H5} - R_{M,H2}) - (R_{C,H5} - R_{C,H2})$, on Fault handoffs. Here M is LOCAL AMEND and C is CHALLENGE. Intervals use 50,000 paired checkpoint bootstrap replicates (seed 20260919). Binary contrasts use exact McNemar tests; interactions use exact paired sign flips. Holm adjustment is separate within each backbone and named family: six primary contrasts, five negative-control contrasts, and three secondary final-success contrasts. The plan was specified before formal outcomes, without external registration.

All Fault H5–H2 final-success comparisons have Holm $p = 1.0$. At H5, LOCAL AMEND avoids the extra initial exchange, but its immediate-repair advantage over CHALLENGE is not statistically resolved. The evidence supports an information–handling interaction and a local communication advantage, not a demonstrated gain in final success.

B.6 Feedback Controls, Review, and Complete Endpoints

The six error families and their construction are listed in Appendix A.5, Table A2.

Table B6: Clean negative controls in the window crossover.

Model	Window	Policy	Initial reference match	Explicit intervention	Final success
Qwen	H2	ACCEPT	30/30	0/30	24/30
Qwen	H2	CHALLENGE	27/30	3/30	24/30
Qwen	H2	LOCAL AMEND	27/30	1/30	24/30
Qwen	H5	ACCEPT	30/30	0/30	24/30
Qwen	H5	CHALLENGE	28/30	2/30	24/30
Qwen	H5	LOCAL AMEND	28/30	2/30	24/30
Luna	H2	ACCEPT	30/30	0/30	24/30
Luna	H2	CHALLENGE	30/30	0/30	24/30
Luna	H2	LOCAL AMEND	29/30	1/30	24/30
Luna	H5	ACCEPT	30/30	0/30	22/30
Luna	H5	CHALLENGE	28/30	2/30	24/30
Luna	H5	LOCAL AMEND	28/30	2/30	24/30

Explicit intervention and first-action error are not interchangeable. Clean ACCEPT is at ceiling, limiting what this control can exclude.

Table B7: Immediate repair under matched revision opportunities.

Model	Treatment	Control	Δ (pp)	95% CI	Holm p
I · Concrete evidence vs. reconsideration					
Qwen	42/60	35/60	+11.7	[-3.3, 26.7]	0.1892
Luna	52/60	36/60	+26.7	[13.3, 40.0]	0.001711
II · Diagnostic evidence vs. weak fact					
Qwen	45/60	30/60	+25.0	[13.3, 36.7]	2.747×10^{-4}
Luna	54/60	31/60	+38.3	[25.0, 51.7]	3.099×10^{-6}

Each study uses 60 paired checkpoints per model and one sender-revision call per arm. Δ is treatment minus control in percentage points (pp); intervals use paired bootstrap. Holm adjustment is across the two models within each study.

Table B8: Final task success in both feedback studies.

Study	Model	Treatment /60	Control /60	Δ (pp)	95% paired CI
I	Qwen	49	46	+5.0	[-5.0, 15.0]
I	Luna	57	58	-1.7	[-8.3, 5.0]
II	Qwen	51	48	+5.0	[-3.3, 13.3]
II	Luna	57	57	0.0	[-5.0, 5.0]

Study I: evidence versus reconsideration. Study II: diagnostic evidence versus weak fact. Every interval includes zero; these are secondary endpoints.

Two paired studies reuse the same 60 ALFWorld checkpoints, spanning six fault types with ten checkpoints each. They execute separate trajectories and must not be pooled into a single three-arm experiment. Each arm gives the sender exactly one revision call, followed by the same NATIVE execution protocol. Study I compares concrete receiver evidence with a generic reconsideration request. Study II matches the challenge structure and error reminder, replacing only the receiver-local fact with diagnostic evidence or a truthful weak fact. Study II uses structurally matched control feedback; the diagnostic condition is not token-exactly matched, so the contrast is interpreted as a comparison of diagnostic information packages under the same revision opportunity. Tables B7 and B8 separate immediate repair from final task success.

B.6.1 Human Semantic Review

Review of the 60 stimuli separates conflict validity, receiver visibility, actionability, and interface ambiguity. All 60 contain a supported problem and locally visible relevant evidence; actionability is rated YES for 49, PARTIAL for two, and NO for nine. Fifteen carry semantic or interface

Table B9: Human semantic review of 240 sender revisions in Study I.

Model	Study I arm	Complete / partial / no repair	Exact executable revision /60	New error /60	Executor consistency /60
Qwen	Evidence	30 / 30 / 0	31	30	52
Qwen	Reconsideration	2 / 43 / 15	9	45	49
Luna	Evidence	30 / 30 / 0	31	30	59
Luna	Reconsideration	0 / 43 / 17	4	46	49

Each model × arm cell contains 60 revisions. Complete/partial/no repair are mutually exclusive judgments of repair to the original handoff. Exact executability, new error, and executor consistency are separate dimensions and can overlap with repair labels. Executor consistency means that the first subsequent action is compatible with the revision's broad intent; it does not require matching the clean-reference action. Independent double coding and inter-rater agreement are not established.

qualifications, including ten simulator-affordance cases. Replay establishes an executable recovery route, not human-level uniqueness of the repair. Table B9 reports the separate content review of sender revisions.

Review uses the sender's task, private history, original handoff, feedback, and revision, together with the receiver's visible context and next action. Labels are not inferred from final task success. Under the strict actionability rule, the plan must specify one exact currently admissible action; broad multi-step prose may be understandable but fails that interface criterion. A consistent executor action can result from executor-side concretization or correction, so it is not proof of a fully repaired sender message.

Each study uses two-sided exact McNemar tests for immediate repair and a two-comparison Holm family across the two backbones. Paired percentile bootstrap uses 100,000 task resamples, with seed 20260918 for Study I and 20260919 for Study II. Final success, recovery windows, fault-type strata, and token outcomes are secondary or exploratory. Time-to-repair and time-to-success are censored at the 50-step budget, assigning 51 to unreached endpoints rather than restricting analysis to successful branches.

C Adaptive Handling Protocols and Supplementary Results

C.1 Implementation and Evaluation Boundaries

Deployed inline policy. Route selection is integrated into the receiver's ordinary generation. ACCEPT produces an action, LOCAL AMEND produces a supported amendment and an action, and CHALLENGE produces feedback for sender revision. A challenge adds the revision exchange and subsequent receiver execution, all included in measured cost. There is no standalone routing-model call. In ALFWorld, the structured response is parsed deterministically; format validity is distinct from action validity and task success.

The matched-frequency comparison uses a frozen single-handoff harness: checkpoint restoration, route selection, at most one sender revision, and one post-handoff action. It matches realized route and model-call counts within each backbone and panel, while token use and handling content remain unmatched. This action-execution endpoint is separate from the generation-only route check in Appendix C.3.

C.2 Complete End-to-End Performance and Cost

Tables C1–C3 report all five conditions on the task panels used in Table 3. Within each environment–backbone stratum, the comparator maximizes the native task score among NATIVE, ACCEPT, CHALLENGE, and LOCAL AMEND; ties are broken by lower mean token use. This retrospective aggregate choice is not a per-task oracle. Token change is 100 × (adaptive tokens / comparator tokens - 1).

Figure C1 presents each condition as a mean task score and mean token cost. Higher scores and lower costs are preferred within an environment. Axes are matched across backbones within each

Table C1: Native task outcomes under all five conditions.

Environment	Model	n	Metric	NATIVE	ACCEPT	CHALLENGE	LOCAL AMEND	ADAPTIVE
ALFWorld	Qwen	50	Success (%)	62.0	50.0	62.0	72.0	78.0
ALFWorld	Luna	50	Success (%)	78.0	76.0	82.0	86.0	86.0
AppWorld	Qwen	30	Success (%)	33.3	23.3	43.3	43.3	46.7
AppWorld	Luna	30	Success (%)	50.0	43.3	43.3	53.3	53.3
ToolSandbox	Qwen	30	Similarity	0.6425	0.6977	0.8311	0.8077	0.8579
ToolSandbox	Luna	30	Similarity	0.7445	0.7022	0.7547	0.7138	0.7526
Collab-Overcooked	Qwen	30	Success (%)	76.7	80.0	80.0	76.7	80.0
Collab-Overcooked	Luna	30	Success (%)	80.0	73.3	90.0	80.0	93.3

ALFWorld uses 50 tasks; each other environment uses 30. ToolSandbox similarity is a partial-completion score, not a success percentage. Values are point estimates without inferential significance marks.

Table C2: Mean tokens per task (thousands; three decimal places), including measured revision exchanges.

Environment	Model	NATIVE	ACCEPT	CHALLENGE	LOCAL AMEND	ADAPTIVE
ALFWorld	Qwen	51.598	64.696	75.945	50.308	65.697
ALFWorld	Luna	40.402	44.794	54.078	37.022	41.239
AppWorld	Qwen	198.430	246.253	214.954	203.713	219.660
AppWorld	Luna	243.892	278.997	222.395	392.874	294.531
ToolSandbox	Qwen	17.083	19.654	21.659	22.057	20.548
ToolSandbox	Luna	20.687	19.039	21.477	22.319	22.970
Collab-Overcooked	Qwen	322.318	197.067	207.872	204.006	217.150
Collab-Overcooked	Luna	662.868	234.101	210.535	233.548	262.450

Matching task panels and environment budgets does not imply matched realized calls or tokens. These condition means are descriptive point estimates; they do not establish statistical equivalence between policies.

Table C3: ToolSandbox strict full success (score = 1.0), alongside the primary similarity endpoint.

Model	NATIVE	ACCEPT	CHALLENGE	LOCAL AMEND	ADAPTIVE
Qwen	2/30	4/30	5/30	4/30	3/30
Luna	4/30	3/30	5/30	2/30	5/30

For Qwen, ADAPTIVE improves mean similarity over CHALLENGE (0.8579 versus 0.8311) while strict full success is 3/30 versus 5/30. The similarity improvement must not be described as a higher full-success rate. Luna has 5/30 strict successes under both conditions despite its slightly lower adaptive similarity.

environment; ToolSandbox retains native similarity rather than a success percentage. The points summarize condition means, without uncertainty intervals or significance marks.

C.3 Controlled Check of the Deployed Inline Policy

The route-only check uses `qwen3-max-2026-01-23` and the ALFWorld `inline_adaptive` prompt builder used by the deployed policy. Thirty checkpoints are each paired with Clean and Fault messages, producing 60 inputs. The checkpoint classes contain ten cases each: locally repairable conflict (C1), conflict requiring sender-held repair information (C2), and no receiver-visible diagnostic conflict (C3). Clean inputs expect ACCEPT; Fault inputs expect LOCAL AMEND, CHALLENGE, and ACCEPT for C1, C2, and C3 respectively. Thus the expected class sizes are 40/10/10 for ACCEPT/CHALLENGE/LOCAL AMEND.

Each input receives one executor generation at temperature 0, configured seed 0, and a 4,096-token output limit. The receiver receives the task, recent visible action–observation history (up to five steps),

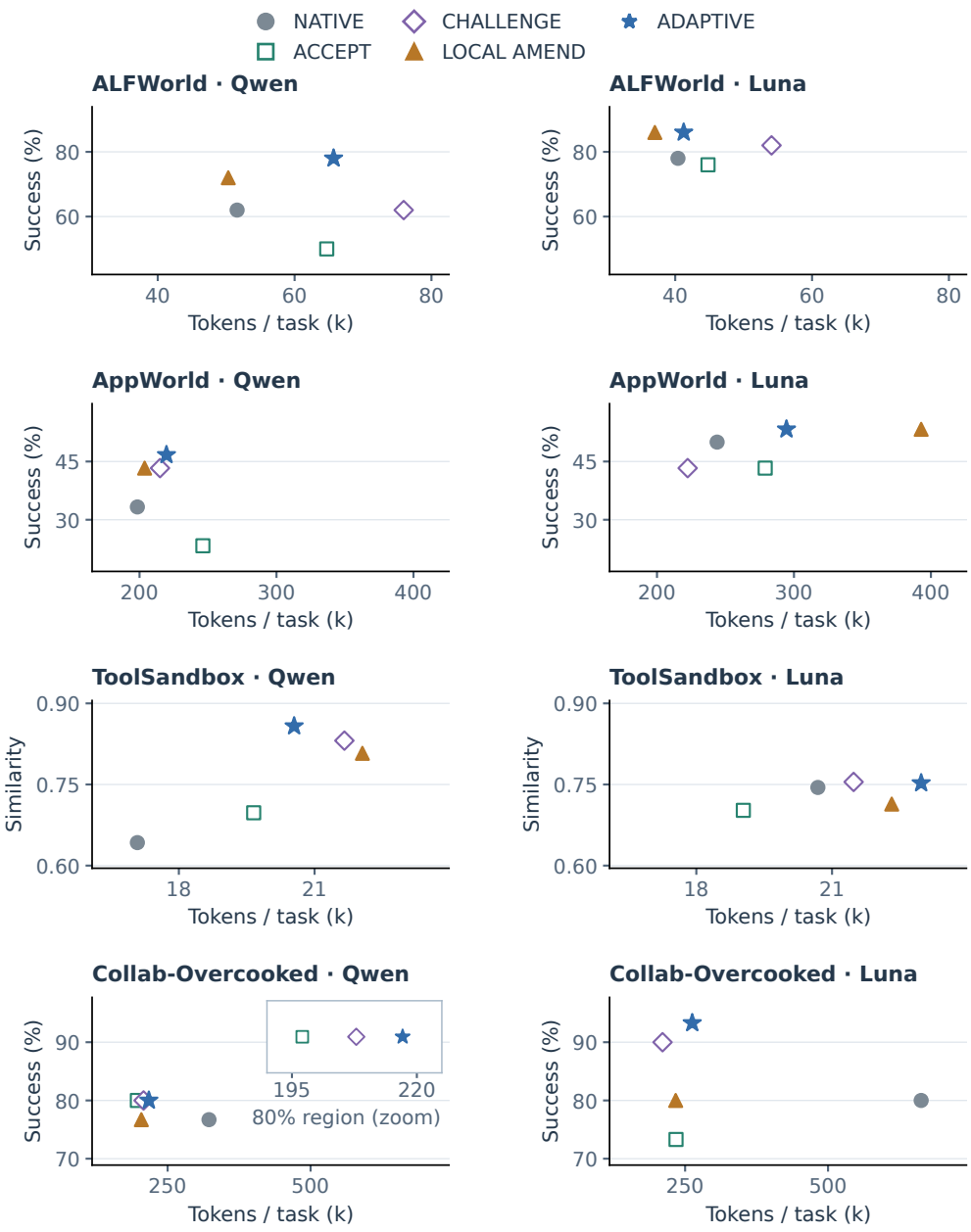

Figure C1: Task performance and token cost for all five conditions. Each row is one environment; columns distinguish Qwen and Luna. All 40 condition means are retained. The Qwen Collab-Overcooked inset enlarges the three 80% points at their original coordinates.

current observation, admissible actions, and incoming message. Evaluation labels and reference actions are not supplied. There is no planner call, environment action, or trajectory continuation. C2 includes controlled state-preserving navigation/look detours to separate recent receiver context from earlier sender information; it is not a natural-frequency sample of error types.

The selected counts are 42 ACCEPT, six CHALLENGE, and 12 LOCAL AMEND. All 60 outputs are protocol-valid and no run failed. The four disagreements all concern expected CHALLENGE: two inputs are accepted and two are locally amended. The ten expected amendments are all selected correctly. Total usage is 87,265 input tokens plus 5,270 output tokens, or 92,535 tokens over 60

Table C4: Expected versus selected routes for the deployed inline policy.

Expected route	Inputs	ACCEPT	CHALLENGE	LOCAL AMEND	Recall (%)
ACCEPT	40	40	0	0	100.0
CHALLENGE	10	2	6	2	60.0
LOCAL AMEND	10	0	0	10	100.0

Rows are expected routes; columns are selected routes. Overall agreement is 56/60 (93.3%), macro-F1 is 87.8%, and balanced accuracy is 86.7%. The descriptive always-ACCEPT label baseline agrees on 40/60 inputs (66.7%); it is not an executed task baseline.

Table C5: Panel composition and endpoint eligibility per method and backbone.

Panel / condition	Checkpoints	Inputs	Expected route	Reference-action inputs
Regime: all Clean	75	75	ACCEPT	75
Regime: C1 Fault	25	25	LOCAL AMEND	25
Regime: C2 Fault	25	25	CHALLENGE	25
Regime: C3 Fault	25	25	ACCEPT	0
Window: Clean H2/H5	30	60	ACCEPT	60
Window: Fault H2	30	30	CHALLENGE	30
Window: Fault H5	30	30	LOCAL AMEND	30

Rows reuse checkpoints. Regime/Window contribute 150/120 routing inputs, 125/120 reference-action inputs, and 75/60 Clean inputs. The 25 C3 Fault cases have no defined single-step reference action and are excluded only from action accuracy, not routing accuracy.

receiver calls. These data establish non-degenerate route selection on the controlled set. They do not show that the emitted amendment would repair the state or that a later sender revision would succeed. Table C4 gives the route confusion matrix.

C.4 Controlled Panels and Frequency-Matched Random Handling

The evidence-regime panel contains 75 ALFWorld checkpoints, balanced across C1/C2/C3 (25 each), with Clean/Fault messages at each checkpoint (150 inputs). The evidence-window panel contains 30 checkpoints crossed with Clean/Fault and receiver histories H2/H5 (120 inputs). H2 and H5 retain the most recent two and five completed action–observation steps. The sender retains up to ten steps. The panels share 21 underlying task IDs, so they are analyzed separately rather than described as 105 independent tasks. Table C5 makes the endpoint-specific denominators explicit.

Every branch restores the frozen checkpoint and checks the restored state digest, observation, and admissible actions. A receiver generation selects a route and handles the message. CHALLENGE permits one sender revision followed by one receiver generation; the branch then executes exactly one environment action and stops. ACCEPT and LOCAL AMEND use one model call; a completed CHALLENGE exchange uses three. Both backbones are configured with temperature 0 and seed 0, executor output limit 4,096, and sender output limit 2,048. Sender-only history is supplied during revision, not during the receiver's initial choice.

Receiver decision instruction

ACCEPT when no concrete Receiver-visible evidence contradicts the incoming plan. Uncertainty, an unvisited search location, or a different preference is not a conflict. LOCAL_AMEND when Receiver-visible evidence establishes a concrete conflict and also determines one reliable correction that is currently executable within the Receiver's own step-level action scope. CHALLENGE when Receiver-visible evidence establishes a concrete conflict, but the visible evidence does not determine one reliable local correction. State the conflict and evidence so the Sender can revise using its own context.

The allowed outputs contain an action, an amendment plus action, or a challenge. The receiver is not given the expected route, C1/C2/C3 labels, reference actions, or prior policy outcomes.

Table C6: Exact route-count and call-count matching for inline_adaptive and random handling.

Model	Panel	Inputs	ACCEPT	CHALLENGE	LOCAL AMEND	Calls / condition
Qwen	Regime	150	93	29	28	208
Qwen	Window	120	59	53	8	226
Luna	Regime	150	99	21	30	192
Luna	Window	120	57	27	36	174

Each row applies to both conditions. Calls equal the input count plus two additional calls per CHALLENGE, yielding 434 calls per Qwen condition and 366 per Luna condition across the panels. Route counts and model-call counts are matched; token use and handling content are not.

Table C7: inline_adaptive and frequency-matched random handling on the controlled development inputs.

Model	Method	Route (%)	Action (%)	Clean intervention (%)	A / C / M counts
Qwen	inline_adaptive	81.9	93.5	4.4	152/82/36
Qwen	Matched random	48.5	80.4	38.5	152/82/36
Luna	inline_adaptive	95.6	92.7	3.0	156/48/66
Luna	Matched random	40.0	72.7	48.1	156/48/66

Each row uses 270 routing inputs, 245 eligible reference-action inputs, and 135 Clean inputs. A/C/M denote ACCEPT/CHALLENGE/LOCAL AMEND.

Table C8: Recorded cost of inline_adaptive and frequency-matched random branches.

Model	Method	Calls	Input tokens	Output tokens	Total tokens	Tokens/input
Qwen	inline_adaptive	434	304,550	36,895	341,445	1264.6
Qwen	Matched random	434	274,178	48,687	322,865	1195.8
Luna	inline_adaptive	366	258,149	45,331	303,480	1124.0
Luna	Matched random	366	229,156	46,577	275,733	1021.2

One input is one controlled handoff branch, not one complete ALFWorld task. Despite identical model-call counts, inline_adaptive uses 5.8% more tokens than matched random on Qwen and 10.1% more on Luna.

The random control matches inline_adaptive's realized ACCEPT/CHALLENGE/LOCAL AMEND counts separately within each backbone and panel. A frozen schedule orders unit IDs by SHA-256(model | seed | panel | unit_id), with schedule seed 20260919, and assigns a route multiset with the matched counts. Assignment does not read Clean/Fault status, history window, expected route, evidence class, or reference action. Matching is performed at model $\times$ panel level, not separately within those evaluation strata. Table C6 reports the matched counts.

C.5 Matched-Control Results, Cost Accounting, and Uncertainty

Expected-route agreement is the fraction of initial realized routes matching the evaluation label. Reference-action accuracy is the fraction of eligible inputs whose executed action matches a specified reference action after any permitted revision. Clean intervention is the fraction of Clean inputs receiving CHALLENGE or LOCAL AMEND; it measures intervention incidence, not observed downstream harm. Table C7 summarizes 270 inputs per method and backbone for description only; inferential comparisons remain separate by panel. Table C8 reports the associated token accounting.

For the primary paired contrast, eligible inline_adaptive-minus-random differences are first averaged within each checkpoint, then averaged equally across checkpoints. The 95% intervals resample those checkpoint means 100,000 times, keeping the paired measurements together, and use the 2.5th/97.5th percentiles. Regime and Window resample 75 and 30 checkpoints respectively. These are pointwise intervals, not multiplicity-adjusted simultaneous intervals. Because C3 Fault has no reference action, the checkpoint-weighted Regime action contrast need not equal subtraction of pooled input-level action percentages. Table C9 reports these checkpoint-weighted effects.

Table C9: inline_adaptive minus frequency-matched random: checkpoint-weighted paired effects (percentage points).

Panel	Model	Checkpoints	Route Δ [95% CI]	Action Δ [95% CI]	Clean intervention Δ [95% CI]
Regime	Qwen	75	+39.3 [+28.7, +49.3]	+15.3 [+6.7, +24.0]	-28.0 [-40.0, -16.0]
Regime	Luna	75	+48.7 [+41.3, +56.0]	+22.0 [+15.3, +28.7]	-33.3 [-44.0, -22.7]
Window	Qwen	30	+25.8 [+17.5, +35.0]	+10.8 [+4.2, +17.5]	-41.7 [-55.0, -28.3]
Window	Luna	30	+64.2 [+53.3, +75.0]	+15.0 [+9.2, +20.0]	-60.0 [-73.3, -45.0]

Route/action differences favor inline_adaptive in all four comparisons; Clean intervention differences are negative. Route frequencies and call counts are matched, but route instructions and feedback content also differ, so these are protocol-level effects. Intervals describe the fixed development panels and one frozen random schedule per backbone, not uncertainty across random schedules, repeated API samples, or new task distributions.

Simple-rule comparison and scope. The original development comparison also includes a visible-evidence rule. It attains 100% expected-route agreement in both backbones, with reference-action accuracy of 98.8% for Qwen and 95.5% for Luna, using 380 calls and approximately 274.1k/274.4k tokens. The rule was revised after five development errors, so these are resubstitution results, not held-out generalization. This competitive baseline supports the evidence-gating principle without establishing a need for a complex controller. The matched-random contrast also changes route instructions and feedback content; it does not isolate intervention location while holding all prompt content or token use fixed.

D Experiment 1: Task-Level Paired Analysis

The end-to-end means in Section 4.2 summarize how many tasks succeed but not whether the same tasks succeed under different receiver-side policies. A *rescue* is a task that fails under ACCEPT and succeeds under the target policy; a *harm* is the reverse, and the outcome-switch rate is $(\text{rescue} + \text{harm})/n$. Figure D1 visualizes task-level paired outcomes. Equal aggregate success can still conceal different successful task sets; these descriptive pairs do not replace the controlled-handoff mechanism experiments.

ToolSandbox uses a continuous similarity score, so we preserve paired scene-level changes rather than binarizing them. Figure D2 jointly shows the mean paired difference, its 95% interval, and the numbers of improved/tied/decreased scenarios. Qwen shows larger positive shifts than Luna, while the Luna LOCAL AMEND interval crosses zero; this supports the paper's environment- and backbone-dependent interpretation rather than a universal policy advantage.

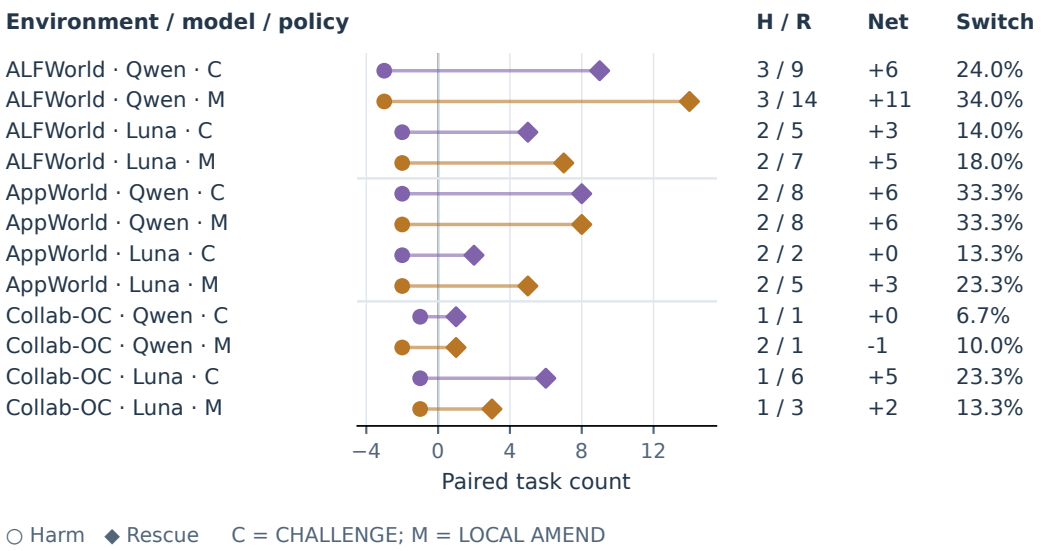

Figure D1: Paired task-level outcome switching relative to ACCEPT in the three binary-outcome environments. Circles mark harms (H) and diamonds rescues (R); separate columns report their counts, net change, and outcome-switch rate. C/M denote CHALLENGE/LOCAL AMEND; Collab-OC denotes Collab-Overcooked.

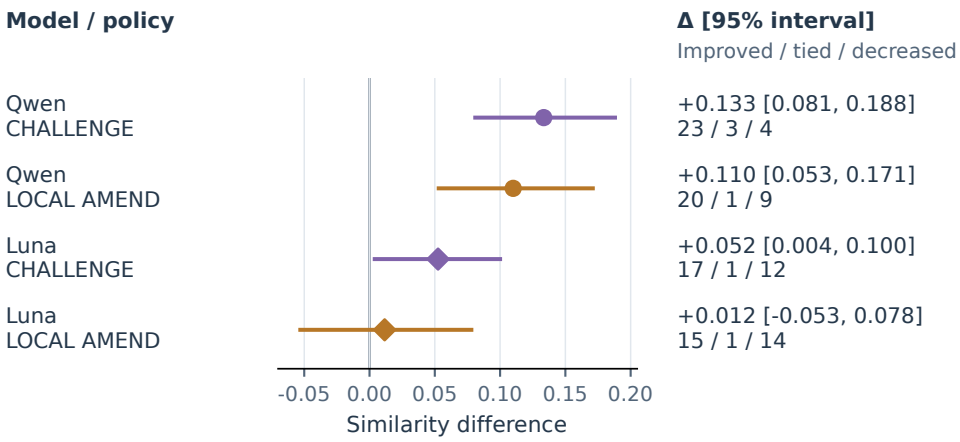

Figure D2: ToolSandbox scene-level paired similarity changes relative to ACCEPT. Points are mean paired differences and horizontal lines are 95% intervals; the separate text column gives each mean, interval endpoints, and improved/tied/decreased counts out of 30. Circles denote Qwen and diamonds Luna; color distinguishes CHALLENGE from LOCAL AMEND. Similarity is continuous, not a success probability.

E EXPERIMENT 1: REALIZED HANDLING PROCESS AUDIT

A separate human audit examines whether the configured receiver-side conditions are accompanied by observable substantive handling in actual trajectories.

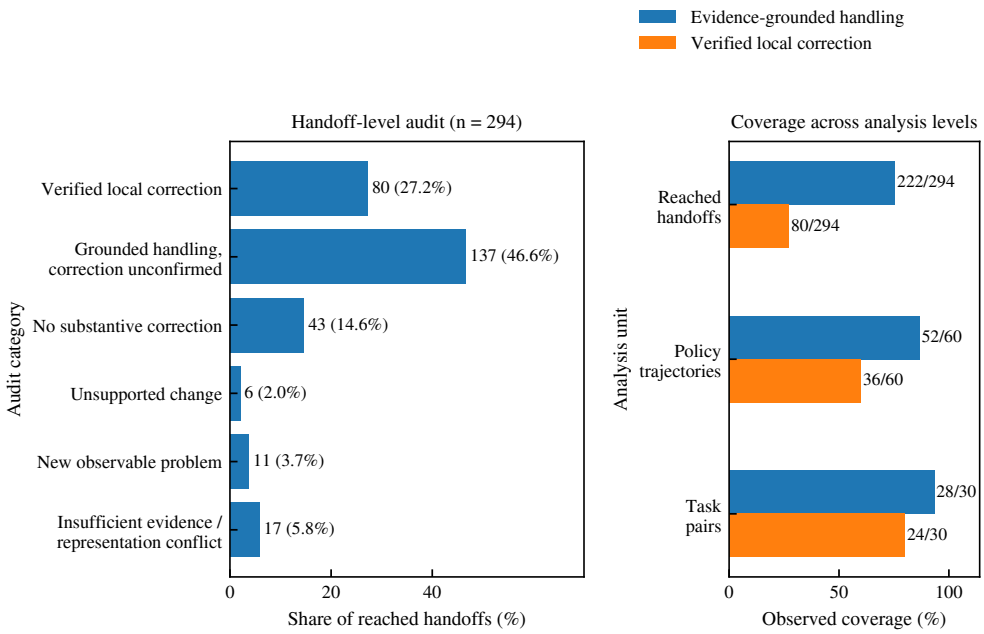

Figure E1: Process-level audit of realized receiver-side handling in Experiment 1. Left: mutually exclusive handoff-level audit categories across 294 reached handoffs. Right: coverage of evidence-grounded handling and verified local correction across reached handoffs, policy trajectories, and task pairs. Evidence-grounded handling totals 222 = 80 + 137 + 5; the last five events are new-problem cases with confirmed starting evidence. These counts describe observed process behavior rather than a trigger-conditioned adherence rate.

Sampling and coverage. Each task pair contains CHALLENGE and LOCAL AMEND trajectories for the same task and backbone. Within each environment–backbone group, tasks are stratified by terminal outcomes into discordant, both-successful, and both-failed pairs. Deterministic hash ordering selects up to two discordant pairs and one pair from each other nonempty stratum. This yields four pairs per group except ToolSandbox–Luna, which has no discordant pairs and contributes two, for 30 pairs overall. The event sample includes all marked challenge/amendment events from the selected trajectories and at most one deterministically selected unmarked handoff per trajectory. Thus, the audit is an outcome-stratified process sample, not an exhaustive census of all handoffs or an estimate of population intervention prevalence.

Event classifications and derived coverage. The archive contains 295 handoff-related records, of which 294 reach the receiver. Among reached handoffs, 80 are verified local corrections, 137 are evidence-grounded substantive handling events whose correction is not confirmed, 43 contain no substantive correction, six are unsupported changes or challenges, 11 introduce a newly observable problem, and 17 have insufficient evidence or a representation conflict. Five of the 11 newly observable problems have confirmed starting evidence, so the derived evidence-grounded total is 222/294 (75.5%). These quantities are process-level observations: because the denominator is all reached handoffs rather than all opportunities that should have triggered intervention, 75.5% is not interpreted as a policy-adherence rate.

At coarser levels, 52/60 policy trajectories and 28/30 task pairs contain at least one evidence-grounded substantive handling event; verified local correction appears in 36/60 trajectories and 24/30 task pairs.

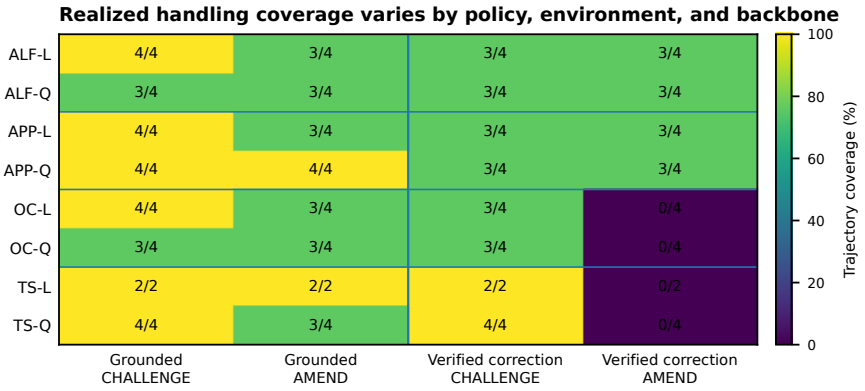

Figure E2: Coverage of evidence-grounded handling and verified local correction in the audit sample. ALF, APP, OC, and TS denote ALFWorld, AppWorld, Collab-Overcooked, and ToolSandbox; Q/L denote Qwen/Luna. CHAL and AMEND denote CHALLENGE and LOCAL AMEND. Cells show trajectories with at least one event over the sampled trajectories in that policy group. The heatmap is descriptive; group sizes are small and each task contributes a paired policy comparison.

A task pair contains two policy trajectories, so the 60 trajectories are not treated as 60 independent tasks. Figure E1 summarizes these levels; Figure E2 shows their environment–backbone breakdown.

The audit complements, rather than replaces, the paired task outcomes and controlled-handoff experiments. It shows that receiver-side handling appears as realized intervention behavior in recorded trajectories, while stronger claims about strict protocol compliance or causal mediation would require a trigger-conditioned opportunity denominator and event-to-outcome linkage.

F Environment Interfaces and Policy Reference

F.1 Shared policy contracts

The shared definitions below apply across environments; Table F1 specifies how the observable interface and executable response differ. A policy assignment defines the receiver's contract. It is distinct from the route realized on any one handoff: a conditional policy may accept a conflict-free message without changing its assignment. Figure F1 summarizes the three handling paths.

NATIVE. Retain the environment-specific reference interaction without the additional fixed handling constraint. In Collab-Overcooked this also retains the original communication rhythm, so it is not a call-matched control.

ACCEPT. Use the incoming message as the basis for the next executable response. Mapping a stated goal to valid tool syntax does not by itself constitute a substantive amendment.

CHALLENGE. When a concrete conflict warrants intervention, return the relevant receiver evidence and request sender revision before executing the resulting response. An emitted challenge tag alone does not establish that the revision was received or executed.

LOCAL AMEND. Use receiver-visible evidence to modify the next action within the receiver's responsibility boundary. The change must be distinguished from routine parameter completion or repeating a correction already present in the incoming message.

inline_adaptive. Select ACCEPT, CHALLENGE, or LOCAL AMEND within the receiver's generation using visible conflict and the availability of a reliable in-scope repair. A selected CHALLENGE still incurs the sender-revision exchange.

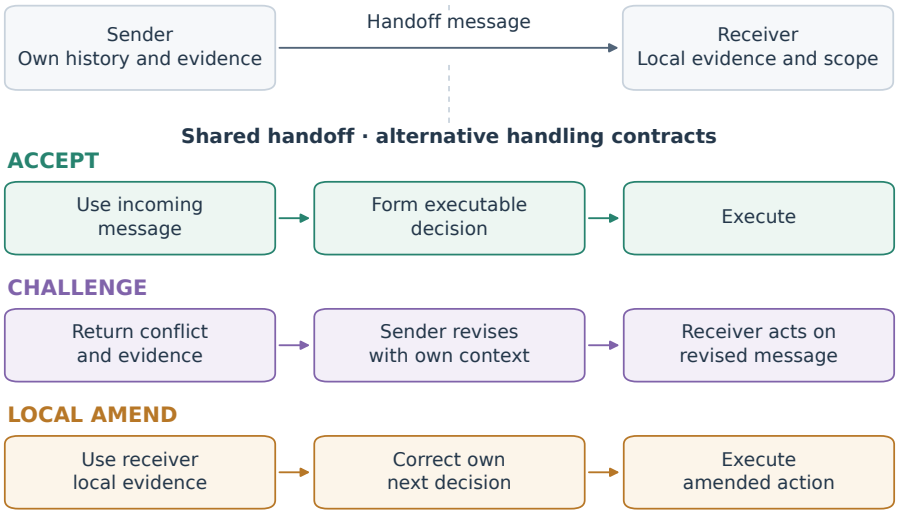

Figure F1: Receiver-side handling at a shared handoff. ACCEPT continues message-guided execution, CHALLENGE returns local evidence for sender-side revision, and LOCAL AMEND changes the receiver's own executable decision. The sender/receiver information boundary is separate from the communication connection.

Table F1: Environment-specific realization of the shared handling contracts. Interfaces and budgets follow Section 4.1.

Environment	Evidence interface	Executable response and boundary
ALFWorld	Planner/Executor recent history: 10/5 completed steps; current observation and receiver admissible actions.	Exact simulator action; 50 environment steps. A local repair changes the receiver's next action, not the task goal.
AppWorld	Planner/Executor raw windows: 6/3 code–output records, plus shared history-derived indices and cached schemas.	Executable application code or finalization; 50 attempts. Successful code execution is not equivalent to task success.
ToolSandbox	Planner/Executor history: 4/2 cycles; public dialogue retained; executor receives native tool schemas.	Native tool call or user reply; 30-message budget. A truthful capability clarification is distinct from completing the requested operation.
Collab-Overcooked	Peer messages, scene and action histories, role-specific resources and knowledge.	Role-valid action sequence; 80 steps. Plan submission is distinguished from the subsequent successful environment action.

F.2 Prompt Reference and Reproduction

Reproduction uses the frozen task manifests, environment-specific prompt templates, and checkpoint replay prefixes. The contracts above summarize the shared role boundaries; each environment instantiates them using its own observation and execution interfaces. The complete end-to-end results are consolidated in Appendix C.2 rather than repeated here. Code and reproduction instructions are available at `https://anonymous.4open.science/r/receiver-side-information-handling-70DC`.

G SELECTED EXECUTION EXCERPTS

These cases illustrate the handling paths in recorded trajectories. They are not a random sample, a policy ranking, or a causal comparison. Each block separates the incoming message, the receiver's handling, and recorded execution feedback. Quoted excerpts are exact substrings; surrounding summaries are paraphrases. Authentication values and unnecessary account details are omitted. Successful cases are accompanied by a local-correction case with final task failure.

TR01: ALFWorld: challenge followed by sender revision

Setting: ALFWorld; qwen; challenge; step 3.
The planner proposes an action absent from the current admissible-action list. The receiver challenges its executability, and the sender returns a revised plan.

> **Incoming:** <plan>examine mug 1</plan>
>
> **Challenge:** The plan "examine mug 1" cannot be executed because "examine mug 1" is not in the current admissible actions list.
>
> **Sender revision:** <plan>take mug 1 from desk 1</plan>
>
> **Executed action:** take mug 1 from desk 1
>
> **Environment feedback:** You pick up the mug 1 from the desk 1.

Observed outcome and interpretation: The recorded sequence links a challenge to a sender revision and an executed replacement. This trajectory succeeds. It does not establish that the same task would have failed without the challenge.

TR02: ALFWorld: bounded search amendment

Setting: ALFWorld; luna; local_amend; step 5.
The current desk observation does not list a pencil, and the requested lamp examination is unavailable. The receiver changes the search location without requesting sender revision.

> **Incoming:** <plan>Examine desklamp 1 to locate or reveal the pencil underneath it.</plan>
>
> **Amendment:** Replace examining desklamp 1 with searching desk 2, since examining the lamp is unavailable and desk 1 shows no pencil.
>
> **Executed action:** go to desk 2
>
> **Environment feedback:** You arrive at desk 2. On the desk 2, you see a alarmclock 1, a bowl 3, and a pencil 1.

Observed outcome and interpretation: The new observation contains the target pencil, and the trajectory later succeeds. This documents a useful search redirection; it does not show that the receiver could uniquely infer the correct desk beforehand.

TR03: ALFWorld: inline adaptation selects local amendment

Setting: ALFWorld; luna; inline_adaptive; step 5.
The task requires cleaning a cloth. The receiver chooses a directly available cleaning operation instead of merely placing the held cloth in the sink.

> **Incoming:** <plan>Put cloth 2 in sinkbasin 1.</plan>
>
> **Selected route:** local_amend
>
> **Executed action:** clean cloth 2 with sinkbasin 1
>
> **Environment feedback:** You clean the cloth 2 using the sinkbasin 1.

Observed outcome and interpretation: The simulator confirms cleaning, and the full trajectory succeeds. The evidence supports an executed local transformation, rather than inferring cleaning from an action tag.

TR04: AppWorld: acceptance with execution verification

Setting: AppWorld; luna; accept; steps 8–9.
The received plan identifies song 112 for playback. The executor consults the API contract, performs the operation, and checks the current song before finalizing.

> **Incoming excerpt:** call spotify.play_music with song_id=112
>
> **Execution excerpt:** play_response = apis.spotify.play_music(**play_args)
>
> **Returned feedback:** Song (Silent Tears) is now playing.
>
> **Verification excerpt:** "is_playing": true

Observed outcome and interpretation: The trajectory succeeds. This illustrates ordinary plan-following with implementation work and result checking; those steps are not automatically evidence of an intervention.

TR05: ToolSandbox: challenge of an unavailable capability

Setting: ToolSandbox; qwen; challenge; routing turn 3.
The plan requires a current-time tool that is absent from the exposed interface. The challenge is followed by a revision and a dispatched response explaining the limitation.

> **Incoming:** <plan>Use the get_current_time tool to obtain the current timestamp, then calculate the start and end timestamps for yesterday to use with the search_reminder tool.</plan>
>
> **Challenge excerpt:** The plan requires a 'get_current_time' tool to obtain the current timestamp, but this tool is not available in the provided toolset.
>
> **Sender revision:** <plan>Ask the user to provide the date they consider "yesterday" in YYYY-MM-DD format, since neither the system nor the available tools can determine the current date to calculate it automatically.</plan>
>
> **Dispatched response:** <user_response>Since I don't have access to the current date and you're unsure of it too, I can't determine what "yesterday" refers to. If you can recall any details about the reminder (like its content or approximate time), I can try searching with that information instead.</user_response>

Observed outcome and interpretation: The record supports a capability clarification in the conversation. It does not show that the original reminder query was successfully executed.

TR06: ToolSandbox: local amendment reports a capability limit

Setting: ToolSandbox; luna; local_amend; routing turn 2.
The planner requests deletion, but the receiver has no deletion capability. It replaces the unexecutable instruction with a grounded limitation statement.

> **Incoming:** <plan>Invoke the contact-deletion tool for person_id 9e137f06-916a-5310-8174-cf0b7e9f7054, then verify the operation result before reporting completion.</plan>
>
> **Amendment:** The requested contact-deletion capability is not available in the exposed tools, so I can't remove Fredrik Thordendal from your contacts.
>
> **User-response excerpt:** I can't delete contacts because that action isn't available here.

Observed outcome and interpretation: This is a conversational correction, not evidence that the contact was deleted. Subsequent acceptance of a limitation-aware plan is consistent with the assigned conditional policy.

TR07: Collab-Overcooked: local repair respects resource ownership

Setting: Collab-Overcooked; qwen; local_amend; authority_events[18]; successful action timestamps 38 and 40.
The chef requests that the assistant put mashed cauliflower into the chef-side oven. The assistant instead retrieves it from its own blender and places it on the shared counter.

> **Incoming excerpt:** Please pick up the mashed cauliflower from blender0 and put it into oven0.
>
> **Receiver plan excerpt:** pickup(mashed_cauliflower, blender0); place_obj_on_counter(); request('pickup(mashed_cauliflower, counter)')
>
> **Successful history excerpt:** {'timestamp': 38, 'action': 'pickup(mashed_cauliflower, blender0)'}, {'timestamp': 40, 'action': 'place_obj_on_counter()'}

Observed outcome and interpretation: The successful-action history confirms both local steps; this trajectory ultimately succeeds. A plan-commit flag alone would not have established these effects.

TR08: Collab-Overcooked: inline challenge enables coordination

Setting: Collab-Overcooked; qwen; inline_adaptive; authority_events[1]; successful action timestamp 5.
The assistant cannot access the chef-side pot. It selects CHALLENGE, receives a revised request to put the ingredient on the counter, and then carries out that request.

> **Incoming excerpt:** Please put the sweet potato into pot0 so I can start cooking it.
>
> **Selected route:** CHALLENGE
>
> **Sender revision excerpt:** Please place the sweet potato on the counter so I can pick it up and cook it in pot0.
>
> **Successful history excerpt:** {'timestamp': 5, 'action': 'place_obj_on_counter()'}

Observed outcome and interpretation: The record contains a sender revision, post-revision acceptance, and subsequent successful action. The full trajectory succeeds; the excerpt does not identify a counterfactual success gain.

TR09: Boundary case: local correction without final task success

Setting: AppWorld; luna; inline_adaptive; AppWorld step 16.
An AppWorld inline-adaptive run corrects a recipient-name mismatch using the observed contact record, while preserving the intended address operation.

> **Amendment excerpt:** Use the observed sister name "Cheryl Martinez" rather than "Cheryl Smith"
>
> **Returned feedback:** Added address to your account.

Observed outcome and interpretation: The API confirms address creation, but the trajectory ends without overall task success. This favorable local effect is retained with its negative final outcome to distinguish intervention execution from end-to-end utility.