

# Lost in the Handoff: Contract-Aware Compliance Checking for Multi-Agent LLM Systems

Yang He<sup>1</sup> Jinlin Wang<sup>1</sup> Yulong Ji<sup>2\*</sup> Jianwei Zhang<sup>3</sup>

<sup>1</sup>National Key Laboratory of Fundamental Science on  
Synthetic Vision, Sichuan University

<sup>2</sup>School of Aeronautics and Astronautics, Sichuan University

<sup>3</sup>School of Computer Science, Sichuan University  
Chengdu 610065, China  
jyl@scu.edu.cn

## Abstract

When LLM agents exchange messages in a pipeline, abstractive compression discards task constraints. Routing tags get absorbed into prose, privacy fields get paraphrased, hard instructions get softened. Even exhaustive reminder-based prompting retains only 53% of required routing markers, as measured by RTD: constraint tokens are low-salience relative to task content and are dropped at generation step boundaries even when restated in full. We propose **C2-FAIC** (Contract-Claim Failure-Aware Inter-agent Compliance), which guards the handoff moment directly: it maintains task requirements in a contract ledger outside agent context and checks each candidate message before it enters the shared trace. On 916 task-metric slices across AgentCollabBench, C2-FAIC is strongest on tracer retention and cross-context leakage control, while also reducing instruction decay and claim propagation relative to no intervention. Critic agents remain stronger on pure IDR and CPR process scores, showing that contract enforcement and generic critique address different failure structures. Multi-agent reliability requires controlling what crosses communication boundaries, not only evaluating what the pipeline ultimately produces.

## 1 Introduction

Multi-agent LLM collaboration depends on agents faithfully relaying task constraints across the conversation. In practice, they do not. As requirements travel from agent to agent, each message generation step compresses the context: routing tags are absorbed into prose; privacy constraints get paraphrased into shared context; hard instructions are treated as suggestions rather than requirements. The receiving agent builds on what it received without knowing what was lost.

The extent of this loss is striking. Even when every agent receives all task constraints at every turn, nearly half of required routing markers still fail to survive handoffs under the strongest reminder strategy. [Cemri et al. \(2025\)](#) find that context mismanagement and instruction drift account for the majority of multi-agent LLM failures. [Xie et al. \(2026\)](#) show how a single dropped constraint cascades into shared belief across a whole collaboration trace. The loss is structural: each handoff compresses context anew, and constraint reminders are discarded with it.

This loss produces no visible signal. A pipeline that drops a routing tag, paraphrases a privacy field into shared context, and forwards an unsupported claim can still produce a fluent, coherent final answer. Outcome-based evaluation is blind to these process failures ([Mazumder et al., 2026](#); [Zhu et al., 2025](#)). The final answer looks correct. The requirements were lost at the handoff.

Existing controls address parts of this problem, including programmable and agent-specific runtime guardrails ([Rebedea et al., 2023](#); [Xiang et al., 2024](#); [Wang et al., 2025](#)). Reminder prompting cannot prevent later compression from discarding constraints. Verifiers and critics can revise candidate or final messages ([Cobbe et al., 2021](#); [Madaan et al., 2023](#); [Shinn et al., 2023](#); [Gou et al., 2024](#)), while tracing tools explain failures ([Zhang et al., 2025](#); [Wang, 2026](#); [Ma et al., 2025](#)) and topology-aware routing changes communication paths ([Leong et al., 2025](#)). The open question is narrower: whether a controller combining persistent contract state and typed interventions improves on a stateless boundary guard across multi-hop handoffs.

C2-FAIC addresses the handoff moment directly. It maintains a contract ledger of task requirements outside agent context, checks each candidate message before it commits, and applies a targeted repair when a contract would be violated. This design also reveals an important structural distinction. Some

\*Corresponding author.

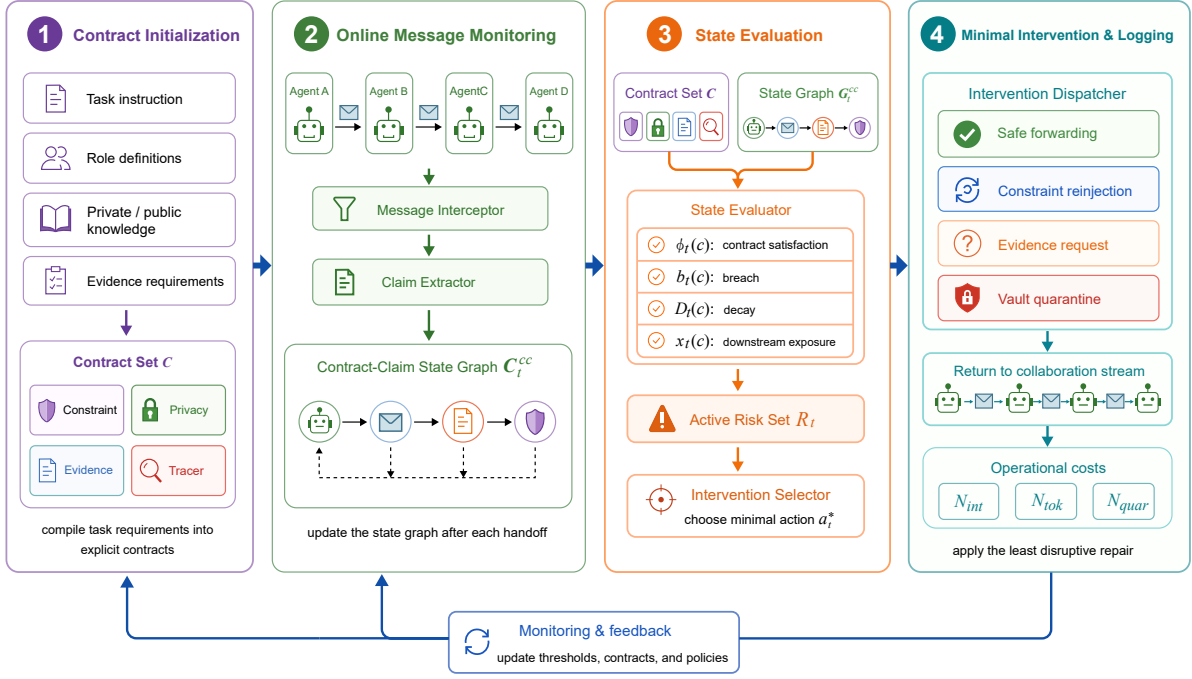


Figure 1: C2-FAIC contract-aware closed-loop governance. Task requirements are compiled into constraint, privacy, evidence, and tracer contracts before collaboration starts. At each handoff boundary, the contract-claim graph is updated and active risks trigger minimal interventions. The controller logs intervention costs and provides feedback to dynamically update system policies and thresholds.

failures reduce to token presence or absence (routing tags, private fields) and respond well to contract enforcement. Others involve diffuse generation behavior (instruction decay) and require critic-style correction. The two strategies address different failure categories and are not interchangeable.

Our contributions:

1. We compare C2-FAIC with a Stateless Contract Guard under identical public contracts and checker settings.
2. We introduce a *checkability taxonomy* (Section 3.5) that classifies process failures as formally checkable, semantically checkable, or behavioral, guiding the choice of local enforcement strategy.
3. We evaluate process reliability, runtime cost, and descriptive final-output quality on the full AgentCollabBench manifest (900 unique task files, 916 task-metric slices), explicitly identifying where stateful enforcement helps and where self-critique or stateless enforcement remains preferable.

## 2 Related Work

**Multi-agent failure as a process-level phenomenon.** Multi-agent LLM failures often arise during collaboration even when the final answer

obscures their origin. Cemri et al. (2025) taxonomize why multi-agent systems fail, finding that breakdowns commonly involve inter-agent communication errors, context mismanagement, and instruction drift. Xie et al. (2026) model how a single early error cascades through a collaboration trace, reinforcing itself as shared team belief. Jia et al. (2026) demonstrate through fault-injection experiments that reliability degrades systematically as the number of agents and interaction rounds increase. Together, these studies establish the collaboration trace as a useful unit of analysis and motivate controls against error propagation. AgentCollabBench (Mazumder et al., 2026) operationalizes four such failures as benchmark metrics and provides our evaluation setting. C2-FAIC focuses specifically on persistent, contract-specific checks applied when each candidate message is admitted to the next agent’s context.

**Prompting, critic, and verifier controls.** Prompting-based approaches inject requirements into agent context as reminders or system instructions (Liu et al., 2023). Reminders can help with shallow instruction forgetting. Our results nevertheless show that reminder-based prompting does not prevent handoff constraint loss across the four studied failure types. Verifier and

judge-style methods (Cobbe et al., 2021; Zheng et al., 2023) and critic/self-correction approaches (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024) evaluate or revise generated candidates before release or at finalization. Such methods suit behavioral failures such as instruction decay, which are diffuse across generated text. C2-FAIC differs by maintaining a contract-specific risk graph across handoffs and checking every candidate message before shared-trace admission.

### Topology, tracing, and context management.

Topology-aware systems change the communication structure to route around known failure patterns (Leong et al., 2025), but topology alone cannot prevent a privacy-sensitive field from being paraphrased into a forwarded message. Tracing and debugging tools such as GraphTracer, AgentTrace, and DoVer (Zhang et al., 2025; Wang, 2026; Ma et al., 2025) locate failures after the shared trace already contains the problematic message. SagaLLM (Chang and Geng, 2025) applies transaction-style context management in LLM planning pipelines. It validates outputs before transactional commit to preserve planning coherence, whereas C2-FAIC tracks individual task contracts across inter-agent handoffs.

### Structured runtime enforcement for LLM agents.

Runtime enforcement is already an active line of agent research. NeMo Guardrails provides model-independent programmable rails for conversational applications (Rebedea et al., 2023); GuardAgent translates natural-language guard requests into executable checks over agent inputs and outputs (Xiang et al., 2024); and AgentSpec supplies declarative trigger–predicate–enforcement rules for agent actions (Wang et al., 2025). Task Shield checks whether instructions and tool calls remain aligned with the user’s task (Jia et al., 2025), while LlamaFirewall combines runtime scanners for prompt injection, reasoning alignment, and insecure code (Chennabasappa et al., 2025). These systems establish that external checking and pre-commit intervention are not themselves novel. C2-FAIC instead studies persistent, per-obligation state across language-level inter-agent handoffs, together with failure-typed revision and fallback; Section 5.2 tests this narrower claim against a matched stateless guard.

### Runtime verification and Design by Contract.

Runtime monitoring frameworks such as JavaMOP

and RV-Monitor synthesize monitors from formal property specifications and observe program execution traces at runtime (Jin et al., 2012; Luo et al., 2014). Their use of an external monitor resembles C2-FAIC’s contract-claim graph. Design by Contract likewise represents obligations as checkable preconditions and postconditions outside the component under test (Meyer, 1992). C2-FAIC adopts this separation of obligations from the monitored component, but the monitored objects are generated messages rather than deterministic program states. Natural language obligations may require semantic reasoning, so they cannot always be reduced to LTL, regular expressions, or pattern matching. This distinction motivates the semantically checkable category in §3.5.

**The handoff boundary gap.** Prior systems diagnose, revise, validate, or govern generated messages and actions, so C2-FAIC does not claim external runtime enforcement as a new primitive. To our knowledge, the specific combination of persistent contract-specific state, admission checks on every candidate inter-agent message, and failure-typed recovery has not been isolated empirically. C2-FAIC targets and directly tests this narrower boundary-control question.

## 3 C2-FAIC

### 3.1 The Handoff Boundary

A *handoff* occurs when a candidate message  $m_{t+1}^{\text{cand}}$  would be appended to the committed trace  $H_t = (m_1, \dots, m_t)$  and exposed to the next agent. C2-FAIC checks this candidate and either commits, revises, or replaces it without changing agent roles, topology, or scheduling. The controller wraps the existing collaboration pipeline and is transparent to sender and receiver when a message passes. Only a revision action asks the scheduled sender for a repaired message.

### 3.2 Runtime Contracts

C2-FAIC compiles task requirements into *runtime contracts*: checkable obligations over routing tags, private fields, evidence, and downstream constraints. Examples include preserving a routing tag until the writer, keeping a named field inside its originating role, requiring support before forwarding a claim, and retaining a hard instruction in downstream messages.

**Definition 1** (Operational contract monitor). Each

| Contract type                 | Process failure    | Metric | Risk state |
|-------------------------------|--------------------|--------|------------|
| $C^{\text{con}}$ (constraint) | Instruction decay  | IDR↓   | MISSING    |
| $C^{\text{evi}}$ (evidence)   | False-claim spread | CPR↓   | VIOLATED   |
| $C^{\text{priv}}$ (privacy)   | Context leakage    | CLC↓   | VIOLATED   |
| $C^{\text{tr}}$ (tracer)      | Tracer loss        | RTD↑   | MISSING    |

Table 1: Contract types and corresponding benchmark failures. Here, missing denotes an unmet instruction-preservation obligation or an absent tracer, whereas violated denotes an unsupported claim or a privacy breach.

natural-language contract  $c$  is compiled into a monitor

$$\mathcal{M}_c = (\alpha_c, \psi_c, \theta_c, \delta_c, \prec_c). \quad (1)$$

Here  $\alpha_c$  is an activation predicate over the current trace and role scope;  $\psi_c$  extracts satisfaction and breach evidence from a candidate handoff;  $\theta_c$  is its satisfaction threshold;  $\delta_c$  updates persistent contract state; and  $\prec_c$  is a fixed priority order over admissible interventions.

Each monitor has type  $\tau(c) \in \{\text{con}, \text{priv}, \text{evi}, \text{tr}\}$  and role scope  $\rho(c)$  (Table 1). Contracts are instantiated before collaboration and stored outside agent context; Section 4 specifies the information available to the inducer. External storage lets one controller supervise different team topologies without relying on each agent to remember the contracts in its prompt.

### 3.3 Contract-Claim Graph and Online Checking

C2-FAIC maintains the contract-claim graph

$$\begin{aligned} \mathcal{G}_t^{\text{cc}} &= (V_t, E_t), \\ V_t &= A \cup M_t \cup Q_t \cup C, \\ E_t &= E_t^{\text{msg}} \cup E_t^{\text{claim}} \cup E_t^{\text{sup}} \cup E_t^{\text{risk}}, \end{aligned} \quad (2)$$

where  $A$ ,  $M_t$ ,  $Q_t$ , and  $C$  denote agents, committed messages, extracted claims, and contracts; the edge sets record message flow, claim extraction, support, and risk. The graph is persistent monitoring state, not a learned graph model.

For each candidate, monitor  $\mathcal{M}_c$  produces

$$(q_{t+1}^c, h_{t+1}^c) = \psi_c(\mathcal{G}_t^{\text{cc}}, H_t, m_{t+1}^{\text{cand}}), \quad (3)$$

where  $q_{t+1}^c \in [0, 1]$  is satisfaction evidence and  $h_{t+1}^c \in \{0, 1\}$  marks a hard breach. Write  $a_{t+1}^c = \alpha_c(H_t)$ ,  $p_{t+1}^c = \mathbf{1}[h_{t+1}^c = 0 \wedge q_{t+1}^c \geq \theta_c]$ , and  $o_{t+1}^c = (a_{t+1}^c, p_{t+1}^c, h_{t+1}^c)$ . Let  $\mathcal{S}^- = \{\text{MISSING}, \text{VIOLATED}\}$  denote the noncompliant states.

**Definition 2** (Contract state). Let  $z_t^c \in \{\text{COVERED}, \text{MISSING}, \text{VIOLATED}, \text{RECOVERED}\}$  be the state stored before the candidate arrives. The candidate-conditioned state is

$$\tilde{z}_{t+1}^c = \begin{cases} \delta_c(z_t^c, o_{t+1}^c) \\ z_t^c, & a_{t+1}^c = 0, \\ \text{VIOLATED}, & h_{t+1}^c = 1, \\ \text{RECOVERED}, & p_{t+1}^c = 1, z_t^c \in \mathcal{S}^-, \\ \text{COVERED}, & p_{t+1}^c = 1, \\ \text{MISSING}, & \text{otherwise.} \end{cases} \quad (4)$$

The RECOVERED state records that a previously missing or violated obligation has been restored, rather than merely labeling the final message after the fact.

The candidate risk set is induced by the same transition:

$$\tilde{\mathcal{R}}_{t+1} = \{c \in C : a_{t+1}^c = 1, \tilde{z}_{t+1}^c \in \mathcal{S}^-\}. \quad (5)$$

The candidate is committed iff  $\tilde{\mathcal{R}}_{t+1} = \emptyset$ ; otherwise the selector intervenes.

### 3.4 Local Intervention at the Handoff Boundary

Let  $\hat{m} = m_{t+1}^{\text{cand}}$  and let  $T_a$  transform it under action  $a \in \mathcal{A}_t$ . Define residual risk  $r_{t+1}(a) = |\mathcal{R}_{t+1}(T_a(\hat{m}))|$  and the feasible set  $\mathcal{F}_{t+1} = \{a \in \mathcal{A}_t : r_{t+1}(a) = 0\}$ . The selector chooses the least disruptive feasible action:

$$a_{t+1}^* = \begin{cases} \min_{\prec} \mathcal{F}_{t+1}, & \mathcal{F}_{t+1} \neq \emptyset, \\ \text{FALLBACK}, & \text{otherwise.} \end{cases} \quad (6)$$

The fixed action order covers safe forwarding, reinjection, evidence request, quarantine/redaction, revision, and safe fallback. Exact-anchor loss triggers reinjection, privacy violations trigger redaction or quarantine, and unsupported claims trigger an evidence-oriented repair. Revision asks the sender to preserve the threatened contract; fallback substitutes a minimal safe message when revision is unavailable or remains risky. The order is fixed by contract type rather than tuned against validation scores, and every check uses only the current trace. After committing  $m_{t+1} = T_{a_{t+1}^*}(m_{t+1}^{\text{cand}})$ , the controller stores the updated state and cost; persistent state can therefore distinguish identical messages reached through different histories.

### 3.5 Checkability Taxonomy of Process Failures

C2-FAIC classifies failures by how violations appear in candidate messages:

**Formally checkable.** Discrete objects such as routing tags or named fields permit exact matching; RTD and CLC fall here and usually admit a direct local action.

**Semantically checkable.** CPR requires matching extracted claims against evidence and therefore semantic scoring, but the threatened object is still identifiable in the message.

**Behavioral.** IDR is diffuse across the response and cannot be reduced to one token or claim, making whole-response critic-style correction more suitable.

Checkability guides local enforcement, but does not guarantee gains from persistent state.

## 4 Experimental Setup

**Benchmark, data preparation, and process metrics.** AgentCollabBench (Mazumder et al., 2026) records each team’s ordered message trace and scores one process failure per *task-metric slice*. Its 900 task files span data engineering, DevOps, and software engineering and yield 916 slices: RTD 232, IDR 230, CPR 226, and CLC 228. The 16 extra slices arise because some task files participate in two metric conditions. For model generalization, we select 200 hard tasks (50 per metric, domain-balanced), first taking poor no-control slices (RTD below 0.5 or nonzero IDR/CPR/CLC) and then filling cells by no-control difficulty. We compare No control and C2-FAIC with gpt-4o-mini, gpt-5.4-mini, and deepseek-v4-flash, recording calls, tokens, revisions, fallbacks, and unresolved risks. Benchmark metrics are trace-level diagnostics, not run-time feedback (Table 1); Table 2 reports the full-manifest comparison analyzed in Section 5.

**Compared methods.** We compare against six generic baselines: no intervention, three reminder-style prompting controls (static, periodic, always-on), a post-hoc verifier, and self-critique prompting. All baselines are our implementations in an AgentCollabBench-compatible runner; Table 3 cites the prior work motivating each family. Three ablations isolate C2-FAIC components:

w/o revision disables targeted message rewriting; w/o fallback disables safe substitution; w/o contract check disables the monitoring layer entirely and serves as the internal baseline. The direct structured comparison uses a Stateless Contract Guard with the same public contracts, checker, model, prompt configuration, endpoint, and task slices as C2-FAIC, but without persistent state, revision, recovery, or typed fallback. This comparison evaluates the full controller under matched public-context conditions; it does not isolate the effect of persistence from differences in intervention policy.

**Public-context contract induction.** The generic-baseline and ablation runs induce contracts from harness-visible task and role text without querying metric judgments. The matched structured comparison is stricter: induction uses only the public task description and root prompt and excludes identifiers, metric labels, injected failures, ground truth, rubrics, and private prompts. Its frozen bundle contains 1,177 contracts across 1,144 phases (259 empty); all 916 method pairs share identical contract hashes.

**Metrics and model protocol.** We report official process metrics, calls, tokens, and intervention counts. OQS grades terminal outputs against task rubrics on a 0–100 scale; C2-FAIC outputs are reconstructed from persisted leaf outputs (or the final cyclic message), with all 916 rows passing the reconstruction check. OQS intervals use 10,000 task-clustered bootstrap replicates and cover task sampling, not judge or endpoint variance; matched process effects use 10,000 unique-task-clustered replicates with Holm-adjusted two-sided *p*-values. Main runs use gpt-4o-mini; official IDR/CPR scoring uses deepseek-v4-flash, while internal semantic checks use the backbone. The controller receives neither hidden answers nor official metric judgments.

## 5 Results and Analysis

Results are organized by whether a failure leaves an inspectable trace object or manifests as diffuse behavioral drift; we report process reliability, final-output quality, and controller cost separately.

### 5.1 Main Process Reliability

Table 2 places C2-FAIC against all baselines on the full 900-task, 916-slice manifest. C2-FAIC reaches

| Family        | Method                  | RTD $\uparrow$ |                   | IDR $\downarrow$ |                   | CPR $\downarrow$ |                   | CLC $\downarrow$ |                   |
|---------------|-------------------------|----------------|-------------------|------------------|-------------------|------------------|-------------------|------------------|-------------------|
|               |                         | Score          | Calls/Tok.(k)     | Score            | Calls/Tok.(k)     | Score            | Calls/Tok.(k)     | Score            | Calls/Tok.(k)     |
| No control    | No intervention         | 0.3367         | 20.0/36.8         | 0.0455           | 19.9/28.0         | 0.1285           | 19.9/26.2         | 0.0198           | 39.9/64.5         |
| Prompting     | Static reminder         | 0.4045         | 20.0/30.8         | 0.0155           | 19.9/24.0         | 0.0791           | 19.9/21.6         | 0.0215           | 39.9/51.6         |
| Prompting     | Periodic reinjection    | 0.4767         | 20.0/30.7         | 0.0140           | 19.9/23.9         | 0.0896           | 19.9/21.1         | 0.0203           | 39.9/48.8         |
| Prompting     | Always reminder         | 0.5348         | 20.0/30.7         | 0.0208           | 19.9/23.8         | 0.0729           | 19.9/21.2         | 0.0185           | 39.9/48.7         |
| Post-hoc      | Post-hoc verifier       | 0.4799         | 21.0/39.8         | 0.0468           | 20.9/31.3         | 0.1171           | 20.9/29.2         | 0.0159           | 40.9/69.3         |
| Self-critique | Self-critique prompting | 0.4978         | 20.0/ <b>20.4</b> | <b>0.0006</b>    | 19.9/ <b>15.0</b> | <b>0.0242</b>    | 19.9/ <b>10.6</b> | 0.0168           | 39.9/ <b>24.5</b> |
| Ours          | C2-FAIC                 | <b>0.8344</b>  | 21.1/29.9         | 0.0106           | 25.1/29.7         | 0.0408           | 28.0/28.0         | <b>0.0019</b>    | 42.1/49.9         |

Table 2: Main process-metric comparison on the full manifest. Arrows indicate metric direction; Calls/Tok.(k) reports mean calls and thousand-token use.

RTD 0.8344, against 0.5348 for the strongest reminder baseline, an absolute gain of 0.2996. Exact tracer checks reinsert missing constraints before commit; failures remain when preserving one marker cannot repair the visible trace. C2-FAIC also reduces CPR from 0.1285 to 0.0408 and reaches CLC 0.0019, the best leakage score in the comparison. Evidence and privacy contracts act on exposed claims or fields before shared-state admission. Self-critique obtains lower CPR (0.0242), while C2-FAIC has higher OQS (38.21 versus 35.27), placing the two controls at different process-quality operating points.

C2-FAIC reduces instruction decay from 0.0455 (no intervention) to 0.0106, but self-critique prompting reaches IDR 0.0006. Because IDR is diffuse, local revision can preserve a threatened constraint without fully redirecting the agent’s generation tendency.

| Baseline                 | Implementation and cited motivation                                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| No intervention          | Original AgentCollabBench scheduled collaboration (Mazumder et al., 2026).                                                                |
| Reminder controls        | Static, periodic, always-on; motivated by prompt-based control (Liu et al., 2023).                                                        |
| Post-hoc verifier        | One verifier turn after collaboration; motivated by LLM-as-judge designs (Cobbe et al., 2021; Zheng et al., 2023).                        |
| Self-critique prompting  | Pre-finalization critique; motivated by self-correction methods (Gou et al., 2024; Madaan et al., 2023; Shinn et al., 2023).              |
| Stateless Contract Guard | Matched structured boundary check motivated by runtime-enforcement systems (Xiang et al., 2024; Wang et al., 2025; Rebedea et al., 2023). |

Table 3: Baseline families and motivating prior work.

| Metric | Method                   | $n$ | Process | Calls | Tokens |
|--------|--------------------------|-----|---------|-------|--------|
| CLC    | C2-FAIC                  | 228 | 0.0056  | 51.62 | 51074  |
| CLC    | Stateless Contract Guard | 228 | 0.0026  | 51.70 | 50442  |
| CPR    | C2-FAIC                  | 226 | 0.1476  | 31.25 | 31529  |
| CPR    | Stateless Contract Guard | 226 | 0.0909  | 26.25 | 23798  |
| IDR    | C2-FAIC                  | 230 | 0.0238  | 24.79 | 29026  |
| IDR    | Stateless Contract Guard | 230 | 0.0175  | 23.30 | 24540  |
| RTD    | C2-FAIC                  | 232 | 0.8126  | 20.66 | 28724  |
| RTD    | Stateless Contract Guard | 232 | 0.6956  | 20.59 | 25825  |

Table 4: Matched public-context comparison. RTD is higher-is-better; the other process metrics are lower-is-better.

## 5.2 Matched Structured-Enforcement Comparison

The matched comparison contains 1,832 successful runs (916 per method) with identical induced-contract hashes and no unresolved errors (Table 4).

The full C2-FAIC controller improves RTD by 0.1170 over the stateless guard (unadjusted 95% CI [0.0655, 0.1692]; Holm-adjusted  $p < 10^{-4}$ ). This gain reflects the full controller, not persistence alone. IDR and CLC differences are not significant, and CPR numerically favors the stateless guard ( $\Delta_{\text{C2-FAIC}-\text{stateless}} = +0.0566$ ; unadjusted 95% CI [0.0139, 0.0989]; Holm-adjusted  $p = 0.054$ ). The observed advantage is specific to tracer retention rather than uniform across process failures.

## 5.3 Final-Output Quality

Mean OQS is 38.21 for C2-FAIC and 39.58 for no intervention (Table 5); the task-balanced difference is  $-1.22$  (95% CI  $[-2.91, 0.48]$ ), but endpoint differences make this comparison descriptive. C2-FAIC also exceeds self-critique by 2.99 task-balanced OQS points while delivering stronger RTD and CLC, though suppression effects remain unresolved.

Mean outcome quality score

| Method                  | RTD          | IDR          | CPR          | CLC          | Overall      |
|-------------------------|--------------|--------------|--------------|--------------|--------------|
| No intervention         | 52.16        | 37.72        | 41.54        | 26.70        | 39.58        |
| Static reminder         | 51.03        | 39.31        | 42.85        | 27.57        | 40.23        |
| Periodic reinjection    | 52.53        | 37.08        | 43.01        | 25.80        | 39.65        |
| Always reminder         | 51.23        | 40.65        | 44.77        | 26.60        | 40.85        |
| Post-hoc verifier       | <b>57.46</b> | <b>48.44</b> | <b>49.35</b> | <b>29.52</b> | <b>46.24</b> |
| Self-critique prompting | 38.74        | 36.47        | 44.43        | 21.45        | 35.27        |
| C2-FAIC                 | 50.65        | 39.10        | 39.65        | 23.24        | 38.21        |

Table 5: Final-output quality on the full manifest (916 slices; higher is better). Against no intervention, C2-FAIC has a task-balanced difference of  $\Delta = -1.22$  (95% CI  $[-2.91, 0.48]$ ; 10,000 task-clustered replicates), which does not resolve a difference under task resampling. Cross-method means are descriptive because provider endpoints differ.

Panel A: Official metric value, mean calls/task, and mean tokens/task

| Method             | RTD $\uparrow$ |               | IDR $\downarrow$ |               | CPR $\downarrow$ |               | CLC $\downarrow$ |               |
|--------------------|----------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|
|                    | Score          | Calls/Tok.(k) | Score            | Calls/Tok.(k) | Score            | Calls/Tok.(k) | Score            | Calls/Tok.(k) |
| C2-FAIC            | <b>0.8344</b>  | 21.1/29.9     | <b>0.0106</b>    | 25.1/29.7     | 0.0408           | 28.0/28.0     | <b>0.0019</b>    | 42.1/49.9     |
| w/o revision       | 0.8096         | 21.0/28.7     | 0.0177           | 23.4/26.9     | <b>0.0346</b>    | 25.7/23.7     | 0.0039           | 41.9/50.5     |
| w/o fallback       | 0.8223         | 21.1/29.2     | 0.0154           | 25.0/29.8     | 0.0454           | 28.1/29.2     | 0.0084           | 43.7/55.7     |
| w/o contract check | 0.3555         | 20.0/36.3     | 0.0445           | 19.9/27.7     | 0.1226           | 19.8/27.5     | 0.0190           | 39.9/64.9     |

Panel B: Mean intervention diagnostics per task

| Method             | RTD            | IDR            | CPR            | CLC            |
|--------------------|----------------|----------------|----------------|----------------|
| C2-FAIC            | 2.70/0.08/0.92 | 2.61/0.96/0.28 | 5.35/1.75/0.12 | 3.64/0.19/1.58 |
| w/o revision       | 2.80/0.00/0.82 | 2.51/0.00/0.33 | 5.44/0.00/0.13 | 3.83/0.00/1.71 |
| w/o fallback       | 2.71/0.11/0.74 | 2.53/0.83/0.63 | 5.59/1.68/1.17 | 3.44/1.86/1.29 |
| w/o contract check | 0.00/0.00/0.00 | 0.00/0.00/0.00 | 0.00/0.00/0.00 | 0.00/0.00/0.00 |

Table 6: C2-FAIC component ablation. Panel A reports official metrics and runtime cost; Panel B reports blocked messages / revision attempts / unresolved risks per task.

## 5.4 Mechanism Ablation

Table 6 separates detection, model-based repair, and safe fallback. Removing contract checking returns all four metrics close to no intervention, while the full method reaches 0.8344, 0.0106, 0.0408, and 0.0019; extra calls alone therefore do not explain the gain. Without revision, the controller relies more heavily on conservative handling; without fallback, unresolved IDR/CPR risks are more likely to remain after the permitted intervention path. Panel B shows that semantic IDR/CPR cases more often require repair, whereas RTD/CLC admit direct object-level actions.

**Hard-case model generalization.** Figure 2 tests whether the same handoff gate transfers as backbone reliability changes. RTD provides the clearest result: C2-FAIC raises tracer retention from 0.0000 to 0.6733 on gpt-4o-mini, 0.2107 to 0.9100 on gpt-5.4-mini, and 0.3987 to 0.8167 on deepseek-v4-flash. It also reduces CPR for every backbone, with the largest gains where no-control propagation is high. IDR and CLC gains shrink when no control is already near zero, while

object-level tracer and evidence failures transfer more consistently. The side table shows about five additional calls per task; token use is lower for gpt-4o-mini and gpt-5.4-mini but higher for deepseek-v4-flash.

## 5.5 Runtime Cost

The cost profile follows failure type. RTD and CLC keep calls near base collaboration because the decision is relatively hard-edged: preserve a marker or prevent a private field from crossing the boundary. IDR and CPR require the controller to read the candidate, assess instruction or evidence consistency, and sometimes request a repair. This explains their additional calls and makes the cost interpretable as targeted semantic work rather than uniform controller overhead. The method is most attractive when one contaminated handoff can influence several downstream turns.

Cost should therefore be read together with residual risk. A cheaper controller is not necessarily more efficient if it forwards unresolved violations, while extra semantic calls are useful only when they change admission or repair a threatened con-

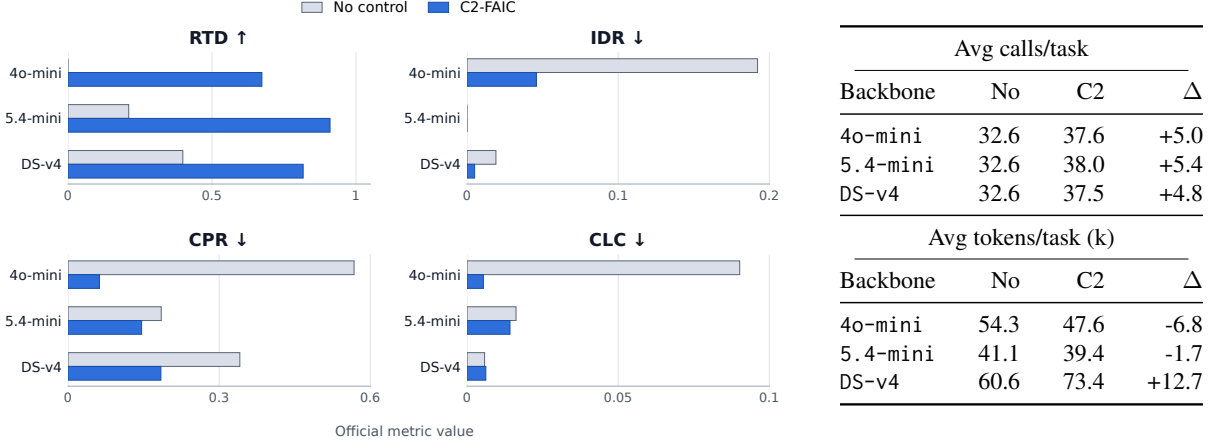


Figure 2: Targeted stress test across backbones on 200 selected hard tasks. Bars show process-metric scores for No control and C2-FAIC across backbones; the side table reports mean calls and tokens. Arrows indicate metric direction.

tract. Future evaluations should report wall-clock latency and parallel-check throughput in addition to calls and tokens.

## 6 Discussion

**Persistent state and the full controller.** Persistent state can represent obligations whose status depends on earlier handoffs. Under matched public contracts, the full C2-FAIC controller improves RTD by 0.1170 over the stateless guard, whereas IDR and CLC differences are not significant and CPR numerically favors the stateless alternative. Because intervention policies also differ, this comparison does not establish a separate effect of persistence.

**Checkability guides the choice of control.** An external ledger shifts reliability from model memory to boundary verification, complementing failure-analysis work (Cemri et al., 2025; Xie et al., 2026). Exact enforcement is strongest when a violation exposes an identifiable object, as in RTD and CLC; diffuse IDR behavior remains better suited to whole-response critique, and broad CPR correction can also benefit from that critic signal. These results motivate a hybrid design in which contract checks guard identifiable handoff objects and self-critique handles behavioral directives.

**Safety, quality, and cost must be considered jointly.** The process gains do not establish a terminal-quality improvement: OQS remains close to no intervention under task resampling, and endpoint differences preclude a causal quality comparison. Runtime work is also concentrated in seman-

tic IDR/CPR checks rather than exact RTD/CLC checks. A deployment can therefore keep exact monitors active while using  $\alpha_c$  to invoke semantic checking only for contracts in scope, preserving the empty-risk-set admission rule without paying uniform model-call overhead.

**Shared-memory architectures.** Immutable history can reduce omissions, but blackboard and long-context systems still have shared-memory write or tool/action commit points. C2-FAIC should therefore be read as pre-commit obligation tracking rather than a mechanism that requires point-to-point messages. Extending contract induction and state updates to these alternative commit surfaces is a concrete direction for broader validation.

## 7 Conclusion

Reliable multi-agent collaboration requires controlling what enters the shared trace, not only reminding agents or evaluating the final answer. C2-FAIC operationalizes this principle through persistent contract state and pre-commit checking at every handoff. On AgentCollabBench, it raises RTD from 0.5348 for the strongest reminder to 0.8344 and achieves the best CLC score in the comparison; under matched public contracts, the full controller improves RTD by 0.1170 over the stateless guard without supporting universal gains across metrics. Descriptive OQS remains close to no intervention ( $\Delta = -1.22$ , 95% CI  $[-2.91, 0.48]$ ), so process reliability and terminal utility should remain separate evaluation axes. The resulting design is best viewed as a complement to self-critique: contracts

protect identifiable trace objects, while critic-style correction addresses diffuse behavior. Future work should test this composition across architectures and optimize semantic-check latency.

## Limitations

The study deliberately centers on language-level handoffs in AgentCollabBench so that boundary control can be measured against a fixed process-failure taxonomy. Coverage includes the full 916-slice manifest, a 1,832-run matched structured comparison, and hard cases across three backbones; validation on additional benchmarks and repeated provider runs would further characterize external and stochastic variation. The structured comparison uses identical public-context contracts, while the broader baseline study uses harness-visible task text. It changes both persistent state and intervention policy; isolating persistence requires holding the latter fixed. Automatic induction produces valid empty bundles when public text contains no actionable obligation, but independent labeling of induction precision and recall would strengthen this component-level analysis. The evaluation targets one structured guard rather than exhaustively reimplementing every runtime framework. Finally, OQS outputs are deterministically reconstructed, but provider endpoints differ across methods; we therefore report quality descriptively and reserve causal quality claims for endpoint-matched reruns.

## Ethics Statement

This study evaluates publicly available benchmark tasks and LLM-generated collaboration traces and does not involve human participants or collection of personal data. The intended benefit is more reliable and accountable use of multi-agent LLM systems in language-centered tasks. Contract definitions can encode designer priorities and assumptions; before deployment, they should be documented, audited by domain experts, and paired with appropriate safety review.

## Acknowledgments

This work was supported by the Pioneer Fund of the System of Systems and Artificial Intelligence Key Laboratory of Sichuan Province (Grant No. SSAI202509).

## References

- Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. [Why do multi-agent LLM systems fail?](#) *Preprint*, arXiv:2503.13657.
- Kaiyan Chang and Yiwen Geng. 2025. [SagaLLM: Context management, validation, and transaction guarantees for multi-agent LLM planning](#). *Proceedings of the VLDB Endowment*, 18(12):4874–4886.
- Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, Alekhya Gampa, Beto de Paola, Dominik Gabi, James Crnkovich, Jean-Christophe Testud, Kat He, Rashnil Chaturvedi, Wu Zhou, and Joshua Saxe. 2025. [LlamaFirewall: An open source guardrail system for building secure AI agents](#). *Preprint*, arXiv:2505.03574.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujia Yang, Nan Duan, and Weizhu Chen. 2024. [CRITIC: Large language models can self-correct with tool-interactive critiquing](#). In *International Conference on Learning Representations*.
- Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. 2025. [The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 29680–29697. Association for Computational Linguistics.
- Jin Jia, Zhiling Deng, Zhuangbin Chen, Yingqi Wang, and Zibin Zheng. 2026. [MAS-FIRE: Fault injection and reliability evaluation for LLM-based multi-agent systems](#). *Preprint*, arXiv:2602.19843.
- Dongyun Jin, Patrick O’Neil Meredith, Choonghwan Lee, and Grigore Roşu. 2012. [JavaMOP: Efficient parametric runtime monitoring framework](#). In *Proceedings of the 34th International Conference on Software Engineering*, pages 1427–1430.
- Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. 2025. [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*, pages 2061–2070.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. [Pre-train, prompt, and predict: A systematic survey of](#)

- prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.
- Qingzhou Luo, Yi Zhang, Choonghwan Lee, Dongyun Jin, Patrick O’Neil Meredith, Traian Florin Șerbănuță, and Grigore Roșu. 2014. [RV-Monitor: Efficient parametric runtime verification with simultaneous properties](#). In *Runtime Verification*, pages 285–300. Springer.
- Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2025. [DoVer: Intervention-driven auto debugging for LLM multi-agent systems](#). *Preprint*, arXiv:2512.06749.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-Refine: Iterative refinement with self-feedback](#). *Preprint*, arXiv:2303.17651.
- Aritra Mazumder, Shubhashis Roy Dipta, Nusrat Jahan Lia, Tanzila Khan, Kainat Raisa Hossain, Nehaa Shri, Shubhrangshu Debsarkar, Humayra Tasnim, Gour Gupal Talukder Shawon, Debjoy Mitra, Sumaiya Ahmed Rani, Al Jami Islam Anik, and Al Nafeu Khan. 2026. [AgentCollabBench: Diagnosing when good agents make bad collaborators](#). *Preprint*, arXiv:2605.08647.
- Bertrand Meyer. 1992. [Applying “design by contract”](#). *Computer*, 25(10):40–51.
- Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. [NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445. Association for Computational Linguistics.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language agents with verbal reinforcement learning](#). *Preprint*, arXiv:2303.11366.
- Haoyu Wang, Christopher M. Poskitt, and Jun Sun. 2025. [AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents](#). *Preprint*, arXiv:2503.18666.
- Zhaohui Geoffrey Wang. 2026. [AgentTrace: Causal graph tracing for root cause analysis in deployed multi-agent systems](#). *Preprint*, arXiv:2603.14688.
- Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. 2024. [GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning](#). *Preprint*, arXiv:2406.09187.
- Yizhe Xie, Congcong Zhu, Xinyue Zhang, Tianqing Zhu, Dayong Ye, Minfeng Qi, Huajie Chen, and Wanlei Zhou. 2026. [From spark to fire: Modeling and mitigating error cascades in LLM-based multi-agent collaboration](#). *Preprint*, arXiv:2603.04474.
- Heng Zhang, Yuling Shi, Xiaodong Gu, Haochen You, Zijian Zhang, Lubin Gan, Yilei Yuan, and Jin Huang. 2025. [GraphTracer: Graph-guided failure tracing in LLM agents for robust multi-turn deep search](#). *Preprint*, arXiv:2510.10581.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-Bench and chatbot arena](#). *Preprint*, arXiv:2306.05685.
- Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Daisy Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. 2025. [MultiAgentBench: Evaluating the collaboration and competition of LLM agents](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 8580–8622.