# CATC: CLINICAL AGENT TEAM COORDINATION VIA RUNTIME LOCAL–GLOBAL TRACE AUDIT

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

## ABSTRACT

Clinical large language model agent teams coordinate specialized roles to solve complex clinical tasks. Existing clinical agent systems commonly evaluate individual responses or terminal outputs, leaving failures that arise during handoffs and propagate through shared context, shared state, and dependency relations difficult to detect during execution. Multiple agents may also reproduce the same unsupported claim, so agreement alone does not guarantee consistency with the available evidence, state, or reasoning dependencies. We introduce Clinical Agent Team Coordination (CATC), a framework for coordinating clinical agent teams through runtime local–global trace auditing. CATC follows a two-stage design. An independent auditor combines dynamically generated code checks with semantic reasoning over an incremental interaction graph to identify local agent failures and global inconsistencies in communication and context propagation. CATC converts each structured audit finding into a provenance-grounded coordination constraint for a reachable subsequent invocation. The agent team then produces a bounded correction, and newly generated events are fed back into the audit stage for continued monitoring. We evaluate CATC across multiple clinical datasets spanning perception-oriented and cognition-oriented tasks, examining audit quality, final task performance, intervention safety, and component-wise ablations. These results demonstrate that runtime trace auditing can selectively convert additional agent-team computation into more reliable clinical reasoning while making audit quality and intervention safety explicit.

## 1 INTRODUCTION

Clinical multi-agent systems assign complementary roles to language-model agents and combine their responses into a final decision (Du et al., 2023, Xiong et al., 2023). This structure can expose useful evidence, but it also creates handoffs at which a supported finding may be omitted, distorted, or repeated without its source. A clinical synthesis can therefore sound coherent while losing a consequential observation from an earlier agent or propagating an unsupported claim through later handoffs and aggregation (Figure 1). Agreement among agents is consequently insufficient evidence that the collaboration process preserved clinical evidence and dependency consistency. Detecting and correcting such failures at runtime, before they reach the final synthesis, requires observing the collaboration process itself rather than its final output.

We introduce Clinical Agent Team Coordination (CATC), a two-stage method for auditing an execution prefix and coordinating a bounded response (§3). An independent, read-only auditor connects claims, evidence, and handoffs in an incremental graph and checks local failures together with cross-agent inconsistencies. When it finds a witnessed anomaly, the team receives a targeted constraint at a subsequent invocation. The auditor cannot see future events or gold labels. We evaluate CATC end to end on the six-dataset clinical suite, reporting terminal task performance, component ablations, and descriptive cross-condition analyses.

Table 1 reports terminal B0–B5 and CATC comparisons under five orchestration patterns, and a descriptive analysis summarizes strict monotonic improvement and model-dependent dataset differences (§4.4). Our contributions are threefold.



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.

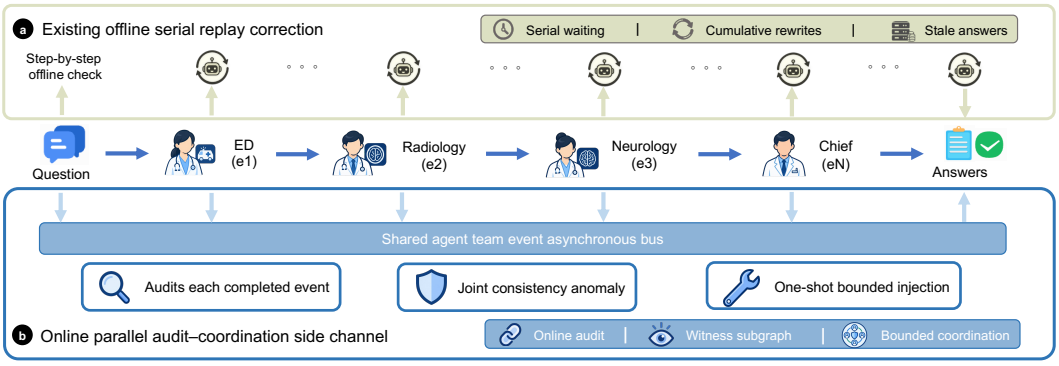



Figure 1, Offline serial correction versus online trace auditing. CATC observes completed events and emits a provenance-linked witness for targeted coordination. Gold labels and future events remain hidden.

1. **A bounded online coordination method.** CATC performs prefix-only, provenance-linked auditing of local and cross-agent failures, converts witnessed anomalies into targeted coordination constraints, and feeds newly generated events back into the audit loop (§3).
2. **An end-to-end clinical evaluation.** Across five orchestration patterns and six clinical datasets, CATC improves every orchestration–dataset cell over the vanilla multi-agent baseline with a mean gain of $+6.76$ pp, exceeds five verification baselines in four of five orchestrations, and component ablations attribute the gain to target routing, provenance, lifetime control, the paired graph views, and structured constraint injection (§4).
3. **A descriptive cross-condition analysis.** We report a strict monotonic-chain view of direct, iterative, and multi-agent inference together with model-dependent score differences across the six clinical datasets (§4.4).

## 2 Related Work

### 2.1 Medical Multi-Agent Collaboration and Consistency

Medical multi-agent systems coordinate debate, consultation, voting, and role specialization for clinical reasoning and diagnosis prediction (Du et al., 2023, Xiong et al., 2023, Chen et al., 2024, Kaesberg et al., 2025, Xie et al., 2025, Du et al., 2026). Surveys of multi-agent collaboration place these designs in a broader space of communication, role allocation, and failure attribution (Tran et al., 2025, Aratchige & Ilmini, 2025, Qi et al., 2026a, Chen et al., 2025, Cemri et al., 2025). Knowledge-enhanced systems add biomedical retrieval and structured evidence, while clinical agentic workflows organize heterogeneous signals through staged reasoning and structured handoffs (Aueawatthanaphisut, 2026, Wang et al., 2024).

Medical debate systems use answer agreement and voting as collaboration signals, while recent work measures whether agreement is accompanied by alignment in the underlying reasoning (Wang & Yang, 2026). Clinical evidence-auditing methods analyze evidence units, controlled evidence subsets, and answer–evidence consistency (Liao et al., 2026, Du et al., 2026, Afrasyab, 2026). Tool-grounded verification systems complement these approaches with database queries and explicit provenance for claim validation (Jeong et al., 2026, Theologitis & Suciu, 2026).

Medical multi-agent systems are commonly evaluated with diagnostic accuracy, answer agreement, and end-task utility (Du et al., 2023, Xiong et al., 2023, Chen et al., 2024, Kaesberg et al., 2025, Cui et al., 2025, Liu et al., 2026). Reasoning-alignment and evidence-use studies add checks on whether clinical conclusions are supported by compatible reasoning and available evidence (Wang & Yang, 2026, Liao et al., 2026, Afrasyab, 2026). Complementary studies examine self-verification and the gap between apparent reasoning quality and robust performance, motivating evaluation beyond final-answer accuracy (Chen et al., 2026c, Qi et al., 2026b, Zhang et al., 2026b, Zhou et al., 2026).

### 2.2 Agent Trace Auditing and Coordination

Agent auditing methods analyze execution prefixes or complete trajectories to identify risks and failures. AgentForesight audits execution prefixes and predicts early failures with agent and step localization (Zhang et al., 2026a). ProbGuard abstracts trajectories into probabilistic states for proactive safety auditing (Wang et al., 2026b). PrefixGuard learns typed adapters for heterogeneous trajectory prefixes (Huang et al., 2026), while Trajectory Guard uses sequence-aware anomaly scores to detect task–trajectory mismatch (Advani, 2026). Rule-based systems such as AgentSpec audit agent actions against predefined runtime specifications (Wang et al., 2026a). TraceElephant evaluates failure attribution using complete task inputs, execution contexts, and agent trajectories (Chen et al., 2026b). Observability toolkits and auditable-agent proposals extend this view from individual detectors to reusable failure records and recovery interfaces (Zhu et al., 2026, Nian et al., 2026, Patil et al., 2024).

Structured auditing methods transform execution traces into graph representations for failure localization. AgentTether constructs critical-transition graphs for diagnosis and runtime guidance (Zhao et al., 2026), and hierarchical causal graphs distinguish root causes from propagated symptoms (Wang et al., 2026d). Counterfactual attribution methods use interventions and replay to identify responsible agents and decisive steps (Zhang et al., 2025, Bonagiri et al., 2026). Visual and harness-level auditing work further emphasizes making intermediate reasoning and externalized research steps inspectable (Zhang et al., 2026c, Wang et al., 2026e).

Coordination methods guide subsequent agent behavior through runtime instructions and step-level rubrics (Zhao et al., 2026, Kang et al., 2026). Foundational agent frameworks use explicit reasoning–action traces, communicative roles, and verbal feedback to organize such coordination (Yao et al., 2023, Hong et al., 2024, Qian et al., 2024, Shinn et al., 2023). Guarded intervention methods evaluate candidate changes through deterministic checks, paired-trajectory audits, or regression-aware acceptance (Xia, 2026, Gao et al., 2026, Chen et al., 2026a). Independent verification evaluates candidate trajectories through decomposed criteria (Kwok et al., 2026). Iterative feedback and rubric distillation provide adjacent mechanisms for refining outputs while preserving a checkable criterion (Madaan et al., 2023, Fang et al., 2026). VRR-Stop models whether noisy intervention loops should continue, commit a candidate, or terminate (Wu et al., 2026).

Recovery-oriented systems add rollback, replay, versioning, and selective self-correction to the coordination loop (Wang et al., 2026c, Ma et al., 2026, Dubey, 2026, Zhang et al., 2026d, Yu et al., 2026). Record–replay and version-control abstractions make these interventions reproducible, while checkpoint defenses and trace redaction address integrity and disclosure risks (Feng et al., 2025, Li et al., 2025, Zheng et al., 2026, Xu et al., 2026). Runtime contracts and formal verification offer complementary ways to constrain execution before a repair is accepted (dos Santos Filho, 2026, Kouvaros & Lomuscio, 2014). Exploration systems also separate branching from commitment, a useful abstraction for deciding when a repaired trajectory should be retained (Wang & Zheng, 2026).

## 3 Method

We present *Clinical Agent Team Coordination (CATC)*, a two-stage online method for tracing and coordinating a clinical multi-agent system (MAS). CATC is layered on an otherwise unmodified runtime. A read-only audit layer observes completed runtime events, and a coordination layer can add a structured constraint immediately before a subsequent model request. CATC targets a failure mode in which an individually fluent claim is unsupported by the case, is distorted during a handoff, or is propagated through shared state until a later synthesis. Figure 2 illustrates this failure and the complete data path. The left panel shows a MedRBench trace in which radiology and neurology evidence of a frontal epidural empyema is lost before the chief's synthesis. The right panels show how CATC retains the location and propagation of that loss and turns it into a targeted pre-invocation constraint. We use "online" strictly. At time $t$ the protocol may use the task input $X$, the registered MAS structure and tools, and the completed prefix $E_{\leq t}$, but not future events, terminal outcomes, or an offline reference label.

### 3.1 Method overview

Let $X$ denote the task input, $R$ the participating agent roles, $G$ the observed orchestration and communication structure, and $U$ the registered read-only tools. The runtime appends an immutable

event to the observable prefix $E_{\leq t}$ after each completed response, handoff, tool call or result, aggregation, or shared-state update. The standardized event envelope and field availability rules are summarized in Appendix C, Table 4. CATC first compiles a task-conditioned audit codebook $K_X$ from $(X, R, G, U)$. For each new prefix, an external auditor generates and executes only code permitted by $K_X$ and emits a schema-constrained graph patch. The patches are merged into an incremental audit graph $AG_t$. Local and global consistency queries over $AG_t$ produce anomaly records and witness subgraphs. A coordination builder then derives a coordination graph $CG_t$, chooses the earliest reachable call that can handle a grouped anomaly, and renders a Coordination Constraint Injection (CCI) at the runtime's pre-invocation boundary. Events produced by that call re-enter the audit stage at the next time index.

The two ownership boundaries are summarized in the paired pseudocode below. Algorithm 1 owns codebook compilation, prefix observation, graph provenance, and anomaly decisions. Algorithm 2 owns grouping, routing, and constraint injection. It never re-detects an anomaly, consumes a future event, or reads a terminal score.

### 3.2 Task-conditioned audit codebook and prefix auditing

The codebook $K_X$ is compiled at task start from the actual task contract, roles, orchestration, dependency structure, and registered tools. It records which claims, entities, values, temporal relations, communication edges, and shared-state constraints are observable for this instance, together with the local and global queries that may be issued. The compilation is task-conditioned rather than a fixed universal rule list. A sequential planner–solver–critic topology exposes different handoff and aggregation relations from a parallel specialist topology. At the same time, $K_X$ is not arbitrary code. Its programs must conform to a frozen audit DSL/API, read only the current prefix, call only $U$, perform side-effect-free computation, and respect time and token budgets. The standardized fields and versioned vocabulary of the DSL, runtime mapping, and failure-state schemas are listed in Appendix C, Tables 5–7.

For each event, the auditor receives $P_t = (X, E_{\leq t}, S_{\leq t})$ where $S_{\leq t}$ contains only shared state exposed by the runtime through time $t$. It selects queries, generates a constrained audit program, and executes the program against registered tools. Deterministic filters can flag a missing required field, an empty rationale, an invalid option, or a contract violation without consuming model tokens. Other tools can recompute a value, check source grounding, map a label to an ontology, or test a semantic relation. Tool outputs are observations attached to the originating event and provenance. They are not independent votes. If a field, tool, or query is unavailable, the patch records an

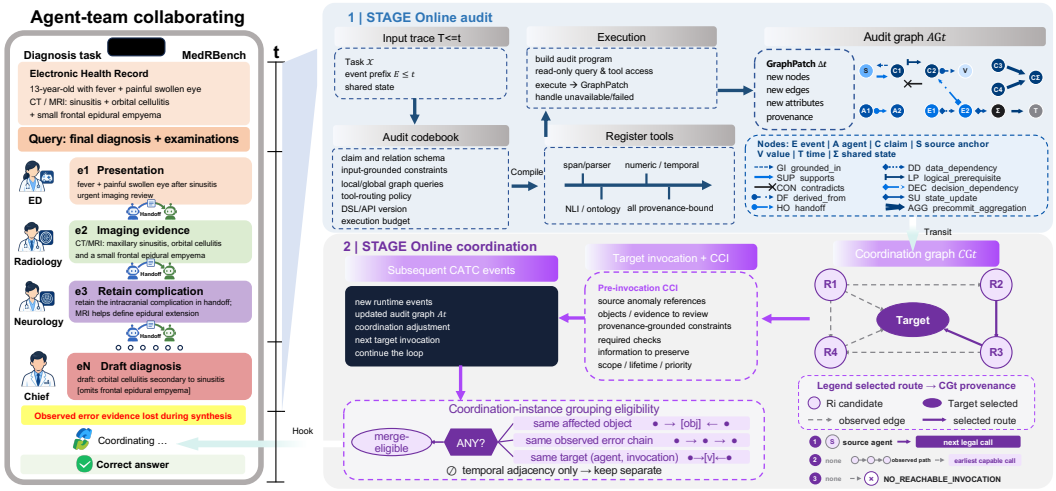



Figure 2, CATC's two-stage flow. Stage 1 audits the completed prefix for local and global inconsistencies. Stage 2 routes a witness-grounded constraint to a reachable invocation. Responses return to Stage 1.

indeterminate observation rather than assuming consistency. The observation-state vocabulary is defined in Appendix C, Table 7.

### 3.3 Incremental audit graph and joint consistency

Each audit round emits a graph patch $\Delta_t$ containing new or updated nodes, edges, observations, and provenance. The graph evolves by

$$AG_t = \text{Merge}(AG_{t-1}, \Delta_t), \qquad AG_0 = \text{Initialize}(X, R, G).$$

Nodes represent events, agents, claims, source anchors, values, times, and shared-state records. Semantic, communication, and dependency relations bind these nodes into the audit graph. The complete relation vocabulary and its directional meaning are defined in Appendix C, Table 6. Every observation retains the event and source context from which it was obtained, so an anomaly can be localized without rewriting the trace.

CATC runs two complementary views over the same graph. Local consistency checks an agent's claims against the input anchors and state visible to that agent. Global consistency checks cross-agent communication, shared state, dependency closure, and aggregation over the prefix. The final decision is joint. A tool observation contributes evidence to the relevant graph query, but no tool independently emits an anomaly label. For each decision, CATC stores the violating nodes and edges, input anchors, observations, unknown relations, and propagation extent as a witness subgraph $W_t$. The current codebook partitions records into eight families, thinking–output, evidence-grounding, coverage, numeric/formula, categorical/label, temporal, aggregation, and format/contract. These families describe the location and type of a trace-level inconsistency. They do not imply that a task-level answer is wrong or that a subsequent action will resolve it.

### 3.4 Coordination graph and target-invocation routing

The coordination layer consumes the active anomaly records and their witness subgraphs, not the raw trace as a second detector. It groups records only when they share an affected object, an observed error chain, or the same target invocation. Mere temporal adjacency is insufficient. Two anomalies next to each other in the event stream remain separate when their objects and dependencies differ. For each group $q$, the coordination graph $CG_t$ records the anomaly nodes, affected agents, candidate invocations, required checks, and provenance route from the witness to the target. $CG_t$ is a derived coordination view. It never replaces or edits $AG_t$.

Routing selects the earliest invocation that is both reachable in the observed orchestration and capable of performing the required check. If the source agent will not be called again, the route may continue

**Algorithm 1, Task-Conditioned Prefix Audit**

1. $K_X \leftarrow$ CompileCodebook$(X, R, G, U)$, $AG_0 \leftarrow$ InitializeGraph$(X, R, G)$.
2. **for** each completed event $e_t$ **do**
   (a) $P_t \leftarrow$ ObservablePrefix$(X, E_{\le t}, S_{\le t})$.
   (b) $Q_t \leftarrow$ SelectQueries$(K_X, e_t, AG_{t-1})$.
   (c) $p_t \leftarrow$ ConstrainedProgram$(K_X, P_t, Q_t)$, $o_t \leftarrow$ ExecuteReadOnly$(p_t, U)$.
   (d) $\Delta_t \leftarrow$ IndeterminatePatch$(e_t)$ if execution is unavailable or fails, otherwise $\Delta_t \leftarrow$ CompilePatch$(e_t, o_t, prov)$.
   (e) $AG_t \leftarrow$ Merge$(AG_{t-1}, \Delta_t)$.
   (f) $L_t \leftarrow$ LocalConsistency$(AG_t)$, $J_t \leftarrow$ GlobalConsistency$(AG_t)$.
   (g) $H_t \leftarrow$ JointDecision$(AG_t, L_t, J_t)$, $W_t \leftarrow$ WitnessSubgraphs$(AG_t, H_t)$.
   (h) Emit$(AG_t, H_t, W_t)$.
3. **return** $AG_t, H_t, W_t$.

**Algorithm 2, Audit-Guided Online Coordination**

1. $H_t \leftarrow$ ActiveAnomalies$(AG_t)$.
2. **for** each group $q$ in GroupRelated$(H_t, AG_t)$ **do**
   (a) $v^* \leftarrow$ EarliestReachableCapable $(q, AG_t, S_t)$.
   (b) **if** $v^* = \varnothing$, record $q$ as unroutable and continue.
   (c) $CG_t \leftarrow$ UpdateCoordinationGraph $(CG_t, q, v^*, AG_t)$.
3. **at the pre-invocation hook for** $v$**,**
   (a) $m \leftarrow$ MatchInvocation$(CG_t, v)$.
   (b) **if** $m = \varnothing$, return Invoke$(v)$.
   (c) $c \leftarrow$ RenderCCI$(m,$ Provenance$(AG_t))$.
   (d) $v \leftarrow$ MergeWithoutOverwrite$(v, c)$.
   (e) $y \leftarrow$ Invoke$(v)$, append events in $y$ and return.

The two CATC ownership boundaries. Standardized inputs, outputs, relations, and states are defined in Appendix C, Tables 4–7. Algorithm 1 audits only the completed observable prefix and incrementally updates $AG_t$. Algorithm 2 groups already located anomalies, derives $CG_t$, and injects a provenance-grounded CCI immediately before a matched model request. An unroutable group is recorded without creating a synthetic repair. The response and other runtime events from a coordinated call return to Algorithm 1.

to the earliest legal downstream owner. If no such invocation exists, CATC records the appendix-defined unroutable status and leaves the runtime unchanged. This explicit failure state prevents a coordinator from fabricating a call or silently treating an unrouteable anomaly as resolved.

### 3.5 Pre-invocation coordination constraint injection

The runtime hook occurs after the target agent and invocation have been selected and the original prompt/context has been assembled, but before the model request is sent. The interceptor matches the invocation identity in $CG_t$ and renders a CCI from its witness-grounded provenance. A CCI states the object to review, the source evidence and affected dependency, the constraints that must be checked, the information that must be preserved, and its scope, lifetime, and priority. These standardized CCI fields are listed in Appendix C, Table 8. It is a review constraint, not a supplied gold answer. The interceptor merges it into the runtime-managed context without overwriting the original request or earlier messages. If there is no matching coordination instance, the invocation proceeds unchanged.

This boundary separates diagnosis from intervention. Algorithm 2 does not re-run local/global consistency, inspect future events, consume terminal outcomes, or assert that the next response is correct. The response, handoff, tool events, and state updates produced after injection are appended as new events and are audited by Algorithm 1. Consequently, an injected constraint can influence a later call while the historical anomaly and its provenance remain immutable.

### 3.6 Online information boundary and method semantics

The online state consists of $(X, R, G, U, E_{\leq t}, AG_t, CG_t)$ and the runtime state exposed at the current invocation. Gold labels, future events, terminal outputs, retrospective counterfactuals, and post-task scores are outside this state and cannot flow into either stage. Missing evidence and unavailable runtime fields remain explicitly indeterminate or unobservable according to the standardized states in Appendix C, Table 7. Neither is silently converted to a positive or negative anomaly decision. Similarly, the online protocol identifies anomalies and routes review constraints, but it does not claim a definitive root cause, a repaired answer, or successful coordination from an injection record alone. Such outcomes are assessed by the separate offline evaluation reported in §4.

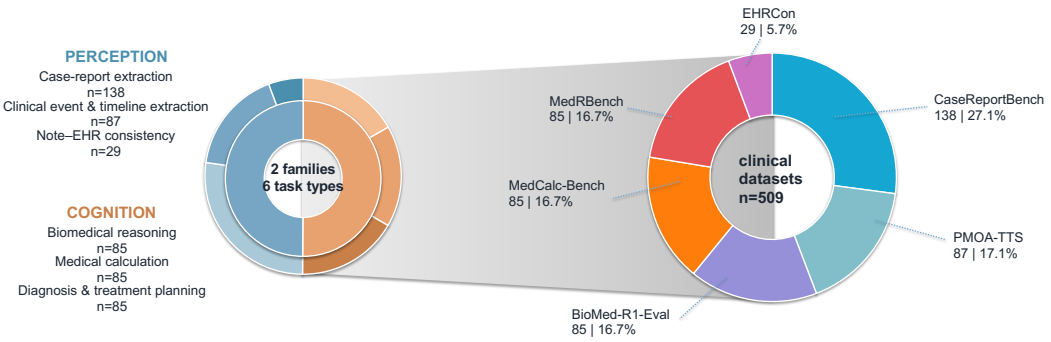
## 4 Experiments

### 4.1 Dataset

**Tasks.** $n$=509 test tasks from six clinical datasets split into a *perception* group (dense extraction and consistency, CaseReportBench, EHRCon, PMOA-TTS, proxy evaluators) and a *cognition* group (reasoning and calculation, BioMed-R1-Eval, MedCalc-Bench, MedRBench, native scoring), covering $49.9\%/50.1\%$ of the suite respectively (Table 2, Appendix A). The dataset counts are BioMed-R1-Eval (85), MedCalc-Bench (85), MedRBench (85), CaseReportBench (138), EHRCon (29), and PMOA-TTS (87). Three use proxy evaluators built for this program, so conclusions on them are paired-effect statements under the same evaluator, never official-benchmark claims. Each task has one frozen $[0, 1]$ score, and primary aggregation is the six-dataset equal-weight macro.

Figure 3 gives the complementary workload composition. The left taxonomy separates the suite into two families and six task types, perception contains case-report extraction ($n$=138), clinical event and timeline extraction ($n$=87), and note–EHR consistency ($n$=29), while cognition contains biomedical reasoning, medical calculation, and diagnosis and treatment planning ($n$=85 each). The right panel shows the six dataset shares of the 509-task suite. The counts are frozen evaluation items, not additional tasks beyond the 509-item total.

**Orchestration patterns and comparison arms.** Five MAS topologies instantiate the same backbone (GLM-5.2) through role prompts, M1 centralized ($K\in\{3, 5\}$), M2 sequential planner→solver→critic→aggregator ($K$=4), M3 routing with $\leq$1 handoff ($K\in\{3, 4, 5\}$), M4 parallel ensemble ($K$=4), M5 moderated collaborative debate ($K$=8), where $K$ is the per-task call budget of the pattern. A parallel run of the same topologies with Claude Sonnet 4.5 provides the model-comparison view in §4.4. Each topology is paired with a vanilla instantiation B0 and five verification baselines: B1 self-refinement, B2 single-critic judging, B3 debate and consensus, B4




(a) Task taxonomy (b) Dataset composition

Figure 3, Task taxonomy and dataset composition for the frozen clinical suite. The inner and outer rings on the left show two workload families and six task types, the right ring shows the six dataset shares. Perception is shown in blue and cognition in orange. Labels report frozen item counts and percentages of the 509-task suite.

attribution and repair, and B5 online auditor and repair. B0–B5 and CATC share task IDs, splits, context limits, and output contracts, so all comparisons within a topology are paired on the same tasks.

Table 1 reports the expanded B0–B5 baseline matrix together with the paired CATC result under each orchestration pattern. Relative to B0, CATC improves all 30 orchestration–dataset cells. The mean gain across M1–M5 is $+6.7624$ pp, with per-orchestration gains from $+3.9640$ pp (M4) to $+9.0114$ pp (M1). The largest dataset-level mean gain is on BioMed-R1 ($+25.6471$ pp), while the gains on EHRCon and CaseReport are smaller ($+0.3139$ and $+0.9027$ pp). The expanded rows make the B1–B5 verification baselines explicit alongside the paired CATC comparison.

### 4.2 Main Result

CATC's six-dataset mean gains over the vanilla baseline B0 are $+9.0$, $+8.2$, $+6.1$, $+4.0$, and $+6.5$ pp under M1–M5, respectively (Table 1). The gains concentrate on the cognition-oriented datasets. BioMed-R1 rises from $52.9$ to $89.4$ under M2, from $69.4$ to $88.2$ under M3, and from $56.5$ to $77.6$ under M5, and MedR-Bench rises from $68.2$ to $77.6$ under M3. PMOA-TTS improves by 2.8–7.2 pp under every orchestration, while EHRCon and CaseReport change by at most $+0.4$ and $+1.2$ pp. The verification baselines behave differently. Self-refinement (B1), single-critic judging (B2), attribution + repair (B4), and online auditor + repair (B5) remain within about $1.5$ pp of B0 on average, several of their per-dataset cells fall below B0, and under M3 all five verification baselines lower the mean score ($-1.5$ to $-0.4$ pp) while CATC adds $+6.1$ pp. Debate and consensus (B3) is the only baseline approaching CATC's scale, with mean gains of $9.3$, $3.2$, $-1.0$, $1.2$, and $2.7$ pp under M1–M5, driven mainly by large BioMed-R1 gains under M1, M2, and M5. CATC obtains the largest mean gain in four of the five orchestrations and trails only B3 on M1, by $0.3$ pp.

### 4.3 CATC Ablations

To isolate the components of CATC, we report a relative ablation on the full protocol. Full CATC is normalized to $100$ for each quality metric. Removing either the local or global graph view reduces audit quality, task score, and rescue rate, which shows that the two views provide complementary context. Removing task-routed tools has a smaller effect, but it weakens deterministic clinical observations. The largest protocol and CCI drops occur without target routing. Removing provenance weakens evidence anchoring, while removing lifetime control produces the highest damage rate. A text-only CCI retains some audit benefit but lowers task score.

### 4.4 Analysis

Figure 5 presents two descriptive analyses. Panel (a) shows the strict monotonic chain from direct single-agent inference to iterative single-agent inference and then to MAS. It includes only tasks

Table 1, Expanded baseline matrix of terminal scores (percent) for the vanilla multi-agent baseline (B0), five verification baselines (B1–B5), and CATC under each orchestration pattern on the six-dataset clinical suite. Each M header row reports the B1–B5 mean gain over the same-orchestration B0 baseline, and each method-row aggregate reports its six-dataset mean gain relative to B0.

| Verification method | MedCalc (Acc) | MedR-Bench (Reason.) | EHRCon (F1) | CaseReport (F1) | PMOA-TTS (Ev. F1) | BioMed-R1 (Score) | Avg. gain |
|---|---|---|---|---|---|---|---|
| **M1 – Centralized** | 2.4 | 1.2 | −1.1 | 0.3 | 0.1 | 10.1 | 2.1 |
| B0 Vanilla MAS (base) | 69.4 | 63.5 | 45.6 | 58.1 | 33.4 | 23.5 | 0.0 |
| B1 Self-Refine | 72.9 | 65.9 | 44.5 | 59.2 | 34.5 | 23.5 | 1.2 |
| B2 Single critic judge | 69.4 | 63.5 | 45.5 | 58.1 | 33.4 | 23.5 | −0.0 |
| B3 Debate / consensus | 77.6 | 67.1 | 44.0 | 58.5 | 32.9 | 69.4 | 9.3 |
| B4 Attribution + repair | 70.6 | 63.5 | 42.9 | 58.1 | 32.8 | 24.7 | −0.2 |
| B5 Online auditor + repair | 68.2 | 63.5 | 45.3 | 58.0 | 33.7 | 27.1 | 0.4 |
| **CATC** | **70.6** | **68.2** | **45.7** | **59.1** | **36.2** | **67.1** | **+9.0** |
| **M2 – Sequential (Pipeline)** | −0.7 | 0.7 | 0.0 | −0.2 | −0.3 | 3.3 | 0.5 |
| B0 Vanilla MAS (base) | 71.8 | 65.9 | 42.8 | 59.0 | 34.1 | 52.9 | 0.0 |
| B1 Self-Refine | 74.1 | 67.1 | 43.9 | 59.0 | 34.2 | 51.8 | 0.6 |
| B2 Single critic judge | 70.6 | 65.9 | 42.8 | 59.0 | 34.1 | 54.1 | 0.0 |
| B3 Debate / consensus | 74.1 | 67.1 | 43.9 | 57.9 | 33.4 | 69.4 | 3.2 |
| B4 Attribution + repair | 69.4 | 65.9 | 40.3 | 59.0 | 33.1 | 52.9 | −1.0 |
| B5 Online auditor + repair | 67.1 | 67.1 | 43.1 | 58.9 | 34.0 | 52.9 | −0.6 |
| **CATC** | **72.9** | **71.8** | **43.1** | **60.1** | **37.8** | **89.4** | **+8.2** |
| **M3 – Routing (Handoff)** | −4.0 | −0.9 | −2.0 | 0.4 | −0.4 | 0.2 | −1.1 |
| B0 Vanilla MAS (base) | 76.5 | 68.2 | 45.0 | 57.7 | 34.2 | 69.4 | 0.0 |
| B1 Self-Refine | 71.8 | 65.9 | 43.2 | 58.7 | 33.9 | 69.4 | −1.3 |
| B2 Single critic judge | 76.5 | 68.2 | 42.4 | 57.7 | 34.2 | 69.4 | −0.4 |
| B3 Debate / consensus | 74.1 | 64.7 | 44.6 | 59.1 | 33.1 | 69.4 | −1.0 |
| B4 Attribution + repair | 72.9 | 68.2 | 42.3 | 57.6 | 33.0 | 68.2 | −1.5 |
| B5 Online auditor + repair | 67.1 | 69.4 | 42.2 | 57.6 | 34.8 | 71.8 | −1.4 |
| **CATC** | **78.8** | **77.6** | **45.3** | **58.6** | **38.5** | **88.2** | **+6.1** |
| **M4 – Parallel (Ensemble)** | −0.9 | 2.1 | −0.9 | 0.6 | −0.1 | −0.0 | 0.1 |
| B0 Vanilla MAS (base) | 76.5 | 57.6 | 45.2 | 58.0 | 33.6 | 70.6 | 0.0 |
| B1 Self-Refine | 72.9 | 58.8 | 43.8 | 59.4 | 33.3 | 67.1 | −1.0 |
| B2 Single critic judge | 76.5 | 57.6 | 45.2 | 58.0 | 33.6 | 70.6 | 0.0 |
| B3 Debate / consensus | 74.1 | 67.1 | 44.2 | 59.4 | 33.4 | 70.6 | 1.2 |
| B4 Attribution + repair | 76.5 | 57.6 | 43.2 | 58.1 | 33.5 | 71.8 | −0.2 |
| B5 Online auditor + repair | 77.6 | 57.6 | 45.2 | 58.0 | 33.6 | 72.9 | 0.6 |
| **CATC** | **77.6** | **67.1** | **45.6** | **58.2** | **38.0** | **78.8** | **+4.0** |
| **M5 – Collaborative (Group Chat)** | 0.0 | 0.7 | −1.1 | 0.4 | 0.1 | 2.8 | 0.5 |
| B0 Vanilla MAS (base) | 72.9 | 63.5 | 45.1 | 58.4 | 34.1 | 56.5 | 0.0 |
| B1 Self-Refine | 71.8 | 67.1 | 41.8 | 59.5 | 33.9 | 55.3 | −0.2 |
| B2 Single critic judge | 72.9 | 63.5 | 45.1 | 58.4 | 34.1 | 56.5 | 0.0 |
| B3 Debate / consensus | 74.1 | 63.5 | 44.5 | 59.4 | 34.5 | 70.6 | 2.7 |
| B4 Attribution + repair | 72.9 | 63.5 | 43.3 | 58.2 | 34.1 | 56.5 | −0.3 |
| B5 Online auditor + repair | 72.9 | 63.5 | 45.2 | 58.3 | 34.1 | 57.6 | 0.2 |
| **CATC** | **74.1** | **70.6** | **45.5** | **59.6** | **41.3** | **77.6** | **+6.5** |

satisfying $S0 < \mathrm{mean}(S1) < \mathrm{mean}(MAS)$, so points describe the subset with continuous improvement rather than an unconditional dataset average. Panel (b) compares the dataset-level mean scores of Claude Sonnet 4.5 and GLM-5.2 on the six clinical datasets. The green labels report the Sonnet minus GLM difference in percentage points.

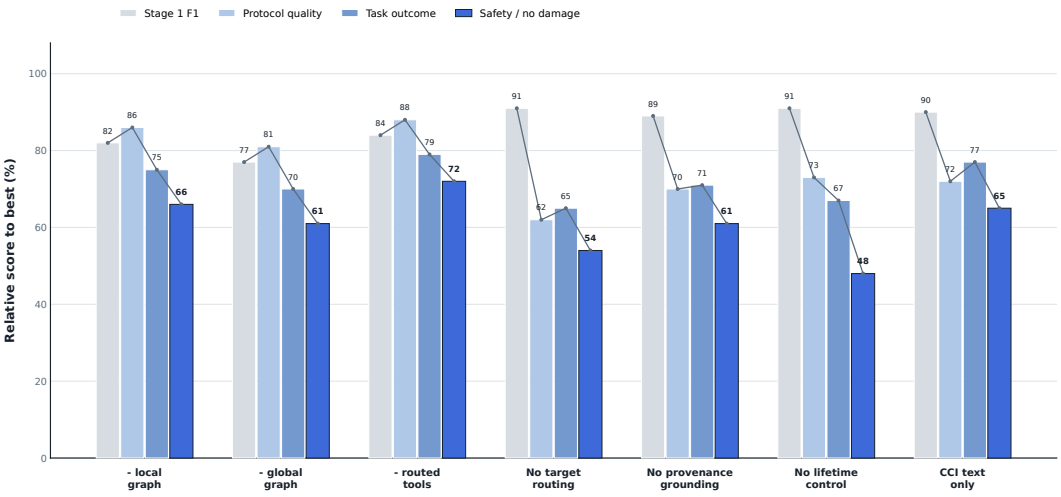



Figure 4, Relative CATC ablations normalized to Full CATC at 100. Target routing, provenance, lifetime control, graph views, task-routed tools, and structured CCI each contribute to reliable audit and bounded recovery.

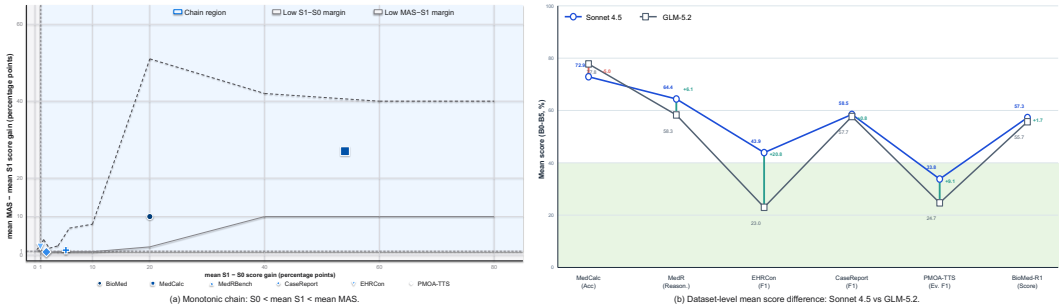



Figure 5, Descriptive analysis of monotonic improvement and model-dependent dataset differences. Panel (a) retains tasks with $S0 < \text{mean}(S1) < \text{mean}(MAS)$. Panel (b) compares Sonnet 4.5 and GLM-5.2 means across the six clinical datasets.

## 5 Conclusion

We presented CATC, a two-stage online protocol for making clinical agent teams reliable at runtime. A read-only auditor combines code checks with semantic reasoning over an incremental interaction graph to localize local agent failures and cross-agent inconsistencies, each anchored to a witness subgraph with full provenance. Every witnessed finding is routed to the earliest reachable invocation capable of performing the required check and injected as a structured coordination constraint, and newly generated events re-enter the audit loop. Across six clinical datasets and five orchestration patterns, CATC improved all 30 orchestration–dataset cells over the vanilla multi-agent baseline, with a mean gain of $+6.76$ pp that surpassed five verification baselines in four of five orchestrations and concentrated on cognition-oriented tasks. Ablations showed that target routing, provenance, lifetime control, the paired graph views, and structured constraint injection each contribute to reliable auditing and bounded recovery. These results indicate that runtime local–global trace auditing can selectively convert additional agent-team computation into more reliable clinical reasoning.

## Ethics Statement

This work does not involve human subjects, sensitive personal information, or the collection of new user data. The datasets used in our experiments are publicly available and are used in accordance with their respective licenses and terms of use. To the best of our knowledge, this work does not introduce significant risks to privacy, security, or individual safety. We have also considered potential

misuse of the proposed method and do not identify any specific high-risk application arising directly from this work.

## Reproducibility Statement

To facilitate reproducibility, we provide detailed information about the publicly available datasets, data preprocessing, dataset splits, model configurations, and evaluation procedures in the paper and supplementary materials. The evaluation materials used for the reported experiments will be released publicly upon publication. All important evaluation details, including model configurations, hyperparameters, and execution procedures, are described in the paper and supplementary materials. CATC and its evaluation are described in the paper and the supplementary materials.

## References


Laksh Advani. Trajectory guard – a lightweight, sequence-aware model for real-time anomaly detection in agentic ai, 2026. URL https://arxiv.org/abs/2601.00516.

Koyar Afrasyab. Judge-dependent safety gains and model-specific helpfulness costs of evidence-sufficiency prompting in clinical llms, 2026. URL https://arxiv.org/abs/2607.18086.

R. M. Aratchige and W. M. K. S. Ilmini. Llms working in harmony: A survey on the technological aspects of building effective llm-based multi agent systems, 2025. URL https://arxiv.org/abs/2504.01963.

Aueaphum Aueawatthanaphisut. Aerospectra sentinel: An auditable llm prompt-chaining decision-support workflow for acute asthma risk assessment from respiratory sounds and clinical signals, 2026. URL https://arxiv.org/abs/2606.08247.

Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. Causalflow: Causal attribution and counterfactual repair for llm agent failures, 2026. URL https://arxiv.org/abs/2605.25338.

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. Why do multi-agent llm systems fail?, 2025. URL https://arxiv.org/abs/2503.13657.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms, 2024. URL https://arxiv.org/abs/2309.13007.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable llm agents: Diagnosing and repairing harness flaws, 2026a. URL https://arxiv.org/abs/2606.06324.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the whole elephant: A benchmark for failure attribution in llm-based multi-agent systems, 2026b. URL https://arxiv.org/abs/2604.22708.

Shuaihang Chen, Yuanxing Liu, Wei Han, Weinan Zhang, and Ting Liu. A survey on llm-based multi-agent system: Recent advances and new frontiers in application, 2025. URL https://arxiv.org/abs/2412.17481.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, Yaorui Shi, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Learning to self-verify makes language models better reasoners, 2026c. URL https://arxiv.org/abs/2602.07594.

Yu Cui, Hang Fu, Haibin Zhang, Licheng Wang, and Cong Zuo. Free-mad: Consensus-free multi-agent debate, 2025. URL https://arxiv.org/abs/2509.11035.

Elzo Brito dos Santos Filho. Parcer as an operational contract to reduce variance, cost, and risk in llm systems, 2026. URL https://arxiv.org/abs/2603.00856.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Yuexi Du, Jinglu Wang, Shujie Liu, Nicha C. Dvornek, and Yan Lu. Care: Towards clinical accountability in multi-modal medical reasoning with an evidence-grounded agentic framework, 2026. URL https://arxiv.org/abs/2603.01607.

Sunny Dubey. Real-time detection and repair of llm agent failures. *arXiv preprint arXiv:2608.02464*, 2026.

Junfeng Fang, Zhepei Hong, Mao Zheng, Mingyang Song, Gengsheng Li, Houcheng Jiang, Dan Zhang, Haiyun Guo, Xiang Wang, and Tat-Seng Chua. Rubric-based on-policy distillation, 2026. URL https://arxiv.org/abs/2605.07396.

Erhu Feng, Wenbo Zhou, Zibin Liu, Le Chen, Yunpeng Dong, Cheng Zhang, Yisheng Zhao, Dong Du, Zhichao Hua, Yubin Xia, and Haibo Chen. Get experience from practice: Llm agents with record & replay, 2025. URL https://arxiv.org/abs/2505.17716.

Haowen Gao, Haoran Chen, Can Wang, Shasha Guo, Liang Pang, Zhaoyang Liu, Huawei Shen, and Xueqi Cheng. Skillaudit: Ground-truth-free skill evolution via paired trajectory auditing, 2026. URL https://arxiv.org/abs/2606.14239.

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 J
urgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Xinmiao Huang, Jinwei Hu, Rajarshi Roy, Changshun Wu, Yi Dong, and Xiaowei Huang. Prefixguard: From llm-agent traces to online failure-warning monitors, 2026. URL https://arxiv.org/abs/2605.06455.

Seyeon Jeong, Yeonjun Choi, JongWook Kim, and Beakcheol Jang. Tool-mad: A multi-agent debate framework for fact verification with diverse tool augmentation and adaptive retrieval, 2026. URL https://arxiv.org/abs/2601.04742.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11640–11671. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.606. URL http://dx.doi.org/10.18653/v1/2025.findings-acl.606.

Jiazheng Kang, Bowen Zhang, Zixin Song, Jiangwang Chen, Xiao Yang, Da Zhu, and Guanjun Jiang. Co-react: Rubrics as step-level collaborators for react agents, 2026. URL https://arxiv.org/abs/2605.23590.

Panagiotis Kouvaros and Alessio Lomuscio. Automatic verification of parameterised interleaved multi-agent systems, 2014. URL https://arxiv.org/abs/1301.6431.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. Llm-as-a-verifier: A general-purpose verification framework, 2026. URL https://arxiv.org/abs/2607.05391.

Yang Li, Siqi Ping, Xiyu Chen, Xiaojian Qi, Zigan Wang, Ye Luo, and Xiaowei Zhang. Agentgit: A version control framework for reliable and scalable llm-powered multi-agent systems, 2025. URL https://arxiv.org/abs/2511.00628.

Junchi Liao, Jiawen Deng, and Fuji Ren. Auditing evidence use in medical llm diagnosis. *arXiv preprint arXiv:2607.20848*, 2026.

Yiqing Liu, Hantao Yao, Wu Liu, Allen He, and Yongdong Zhang. Hcp-mad:heterogeneous consensus-progressive reasoning for efficient multi-agent debate, 2026. URL https://arxiv.org/abs/2604.09679.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Dover: Intervention-driven auto debugging for llm multi-agent systems, 2026. URL https://arxiv.org/abs/2512.06749.

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, 2023. URL https://arxiv.org/abs/2303.17651.

Yi Nian, Aojie Yuan, Haiyue Zhang, Jiate Li, Li Li, Xiyang Hu, Hua Wei, Xiongye Xiao, Chaowei Xiao, and Yue Zhao. Auditable agents, 2026. URL https://arxiv.org/abs/2604.05485.

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

Shihao Qi, Jie Ma, Rui Xing, Wei Guo, Xiao Huang, Zhitao Gao, Jianhao Deng, Jun Liu, Lingling Zhang, Bifan Wei, Boqian Yang, Pinghui Wang, Jianwen Sun, Jing Tao, Yaqiang Wu, Hui Liu, Yu Yao, and Tongliang Liu. Beyond individual intelligence: Surveying collaboration, failure attribution, and self-evolution in llm-based multi-agent systems, 2026a. URL https://arxiv.org/abs/2605.14892.

Zhenting Qi, Susanna Maria Baby, Stefanie Anna Baby, Kan Yuan, Andrew Tomkins, Tu Vu, Da-Cheng Juan, and Cyrus Rashtchian. On the generalization gap in self-evolving language model reasoning, 2026b. URL https://arxiv.org/abs/2606.01075.

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, 2024. URL https://arxiv.org/abs/2307.07924.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Michael Theologitis and Dan Suciu. Thucy: An llm-based multi-agent system for claim verification across relational databases, 2026. URL https://arxiv.org/abs/2512.03278.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms, 2025. URL https://arxiv.org/abs/2501.06322.

Cong Wang and Yusheng Zheng. Fork, explore, commit: Os primitives for agentic exploration, 2026. URL https://arxiv.org/abs/2602.08199.

Haotian Wang, Xiyuan Du, Weijiang Yu, Qianglong Chen, Kun Zhu, Zheng Chu, Lian Yan, and Yi Guan. Learning to break: Knowledge-enhanced reasoning in multi-agent debate system, 2024. URL https://arxiv.org/abs/2312.04854.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable llm agents. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*, ICSE '26, pp. 2938–2950. ACM, April 2026a. doi: 10.1145/3744916.3764546. URL http://dx.doi.org/10.1145/3744916.3764546.

Haoyu Wang, Christopher M. Poskitt, Jiali Wei, and Jun Sun. Probguard: Proactive runtime monitoring for llm agent safety via probabilistic prediction, 2026b. URL https://arxiv.org/abs/2508.00500.

Pan Wang, Yihao Hu, Hang Wang, Zirui Lv, Xin Zhang, Jianshe Li, Jiang-Ming Yang, Wei Wu, and Yongqi Tong. Diagnosis before recovery: Turning agent failures into selective self-correction, 2026c. URL https://arxiv.org/abs/2608.11772.

Xiaoyang Wang and Christopher C Yang. The consistency illusion: How multi-agent debate hides reasoning misalignment. *arXiv preprint arXiv:2606.08457*, 2026.

Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for llm-based multi-agent systems, 2026d. URL https://arxiv.org/abs/2602.23701.

Zijian Wang, Hanqi Li, Ziyue Yang, Zijian Hu, Shenghan Zuo, Yunzhe Zhang, Da Ma, Danyu Luo, Chenrun Wang, Jing Peng, Tiancheng Huang, Sijia Guo, Huayang Wang, Zichen Zhu, Senyu Han, Yilu Cao, Bo Chen, Xin Chen, Kai Yu, and Lu Chen. Externalizing research synthesis and validation in ai scientists through a research harness, 2026e. URL https://arxiv.org/abs/2606.18874.

Yitao Wu, Si Shen, Rui Yang, Hong Peng, and Bin Hu. Verify, repair, repeat, or stop? robust stopping for noisy verify-repair loops in llm agents. *arXiv preprint arXiv:2607.17641*, 2026.

Haizhou Xia. Guarded repair for harm-aware post-hoc replacement of llm mathematical reasoning. *arXiv preprint arXiv:2605.24613*, 2026.

Yuzhang Xie, Hejie Cui, Ziyang Zhang, Jiaying Lu, Kai Shu, Fadi Nahab, Xiao Hu, and Carl Yang. Kerap: A knowledge-enhanced reasoning approach for accurate zero-shot diagnosis prediction using multi-agent llms, 2025. URL https://arxiv.org/abs/2507.02773.

Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. Examining inter-consistency of large language models collaboration: An in-depth analysis via debate. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 7572–7590. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.508. URL http://dx.doi.org/10.18653/v1/2023.findings-emnlp.508.

Shuwen Xu, Zhitao He, and Yi R. Fung. Redact: Redacting agent capability traces for procedural skill protection, 2026. URL https://arxiv.org/abs/2606.10813.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Caili Yu, Yiqi Wang, Jiaqi Zhang, Yiqun Duan, Mingkai Zheng, Zhangkai Wu, Kaize Shi, and Taotao Cai. From faulty memories to corrected actions: Dependency-guided rollback repair for memory-augmented agents, 2026. URL https://arxiv.org/abs/2608.10502.

Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. Agentforesight: Online auditing for early failure prediction in multi-agent systems. *arXiv preprint arXiv:2605.08715*, 2026a.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems?, 2025. URL https://arxiv.org/abs/2509.03312.

Haoran Zhang, Yafu Li, Zhi Wang, Zhilin Wang, Shunkai Zhang, Xiaoye Qu, and Yu Cheng. Characterizing, evaluating, and optimizing complex reasoning, 2026b. URL https://arxiv.org/abs/2602.08498.

Jun Zhang, Jiasheng Zheng, Boxi Cao, Yaojie Lu, Hongyu Lin, Jia Zheng, Xianpei Han, and Le Sun. Reasoninglens: Hierarchical visualization and diagnostic auditing for large reasoning models, 2026c. URL https://arxiv.org/abs/2606.23404.

Zhisong Zhang, Tianqing Fang, Kaixin Ma, Wenhao Yu, Hongming Zhang, Haitao Mi, and Dong Yu. Webrollback: Enhancing web agents with explicit rollback mechanisms, 2026d. URL https://arxiv.org/abs/2504.11788.

Chenyu Zhao, Shenglin Zhang, Wenwei Gu, Yongqian Sun, Dan Pei, Chetan Bansal, Saravan Rajmohan, and Minghua Ma. Agenttether: Graph-guided diagnosis and runtime intervention for reliable llm agent operation. *arXiv preprint arXiv:2607.06273*, 2026.

Yusheng Zheng, Yiwei Yang, Wei Zhang, and Andi Quinn. Acrfence: Preventing semantic rollback attacks in agent checkpoint-restore, 2026. URL https://arxiv.org/abs/2603.20625.

Weiyuan Zhou, Haiping Ma, Xiaoshan Yu, Changqian Wang, Shangshang Yang, and Xingyi Zhang. Agentcat: Simulating computerized adaptive testing via multi-agent large language models, 2026. URL https://arxiv.org/abs/2606.21832.

Kunlun Zhu, Xuyan Ye, Zhiguang Han, Yuchen Zhao, Bingxuan Li, Weijia Zhang, Muxin Tian, Xiangru Tang, Pan Lu, James Zou, Jiaxuan You, and Heng Ji. Agentdebugx: An open-source toolkit for failure observability, attribution, and recovery in llm agents, 2026. URL https://arxiv.org/abs/2607.18754.

# A Datasets

Table 2, The six clinical datasets used in the audit (frozen test slices). Evaluator type *native* denotes an official or faithful local implementation of the published metric, *proxy* denotes a program-built evaluator for which only paired effects under the same evaluator are claimed.

| Dataset | Work type | $n$ | Metric (frozen $[0, 1]$) | Evaluator |
|---|---|---|---|---|
| BioMed-R1-Eval | biomedical MCQ reasoning | 85 | accuracy | native |
| MedCalc-Bench | medical calculation | 85 | interval/rule accuracy | native |
| MedRBench | diagnosis + treatment | 85 | diagnosis/treatment accuracy | native |
| CaseReportBench | case-report extraction | 138 | category–span match | proxy |
| EHRCon | note–EHR consistency | 29 | entity-level consistency | proxy |
| PMOA-TTS | event timeline extraction | 87 | event/time match | proxy |
| **Total / macro** | | **509** | six-dataset equal weight | |

BioMed-R1-Eval aggregates biomedical multiple-choice reasoning. MedCalc-Bench requires parameter extraction, formula selection, unit conversion, and numeric computation scored by published interval rules. MedRBench cases require final diagnosis, recommended examinations, and treatment plans. CaseReportBench is dense structured extraction from case reports (18 categories, e.g. history, labs/imaging, neuro, cardiovascular). EHRCon judges consistency between clinical notes and structured EHR fields, and PMOA-TTS extracts clinical events with relative times in hours. The perception group (CaseReportBench, EHRCon, PMOA-TTS) uses proxy evaluators with JSON contracts, while the cognition group (BioMed-R1-Eval, MedCalc-Bench, MedRBench) uses native scoring. All conditions share task IDs, splits, context limits, output contracts, and evaluators. The evaluator freeze predates all score-blind budget calibration.

# B Orchestration Arms and Execution

Table 3, Orchestration arms. All instantiate the same backbone (GLM-5.2, thinking disabled, temperature 0). $K$ is the per-task call budget of the pattern. Each topology is paired with the B0–B5 verification baselines and CATC of Table 1 under the same $K$.

| Arm | Mechanism | Calls |
|---|---|---|
| M1 | centralized orchestrator $\rightarrow$ specialists A/B/C $\rightarrow$ synthesis | $K \in \{3, 5\}$ |
| M2 | sequential planner $\rightarrow$ solver $\rightarrow$ critic $\rightarrow$ aggregator | $K=4$ |
| M3 | router $\rightarrow$ specialist ($\leq$1 handoff) $\rightarrow$ aggregator | $K \in \{3, 4, 5\}$ |
| M4 | parallel specialists A/B/C $\rightarrow$ aggregator | $K=4$ |
| M5 | moderator, two debate rounds, vote, consensus | $K=8$ |

**Budget.** Each topology block shares a per-task call budget $K$ and a cumulative token allowance per (dataset, topology), with per-call caps adapting to the remaining ledger and a fraction reserved

for finalization. B0–B5 and CATC run under the same budgets, so within-topology comparisons are paired on both tasks and compute. Execution audits show zero missing, invalid, duplicate, or allowance-exceeding records.

**Configuration.** All conditions run thinking-disabled at temperature 0, and every arm shares task IDs, splits, context limits, output contracts, and evaluators. The evaluator freeze predates all runs.

# C CATC STANDARDIZED INTERFACES AND STATES

This appendix defines the machine-readable vocabulary referenced by the method body. The main text explains the semantic decisions, while these tables fix the names, directions, fields, and boundary states used to represent them.

## C.1 RUNTIME EVENT INTERFACE

Table 4, Standardized runtime event envelope. Event payloads are type-specific, an absent field is not synthesized.

| Field | Meaning |
|---|---|
| `event_id` | Immutable identifier for a completed runtime action. |
| `event_type` | Response, handoff, tool call/result, aggregation, or state update. |
| `agent_id` | Agent responsible for or receiving the action, when exposed. |
| `visible_context` | Task, message, tool, and shared-state context visible at that action. |
| `payload` | Type-specific content. Fields are optional and availability is explicit. |
| `provenance` | Source spans, parent events, dependency links, and runtime origin. |

## C.2 AUDIT CODEBOOK AND GRAPH PATCH INTERFACE

Table 5, Standardized audit-stage inputs and outputs.

| Interface | Standardized content |
|---|---|
| Codebook input | Task $X$, roles $R$, orchestration $G$, registered read-only tools $U$. |
| Codebook output $K_X$ | Claim/relation schema, input-grounded constraints, local/global queries, tool routing, DSL/API version, and execution budget. |
| Prefix input | $X$ and the runtime-observable prefix $E_{\leq t}$ plus exposed shared state. |
| Graph patch $\Delta_t$ | New or updated nodes, relations, observations, provenance, and availability state for one completed event. |
| Audit graph $AG_t$ | $\mathrm{Merge}(AG_{t-1}, \Delta_t)$ with active anomaly records and witness subgraphs. |

## C.3 AUDIT GRAPH RELATION VOCABULARY

Table 6, Standardized audit-graph relations. Each relation is directed from the first node to the second.

| Relation | Meaning |
|---|---|
| `grounded_in` | A claim or observation points to its source anchor. |
| `supports` | One claim, observation, or state supports another. |
| `contradicts` | One claim, observation, or state conflicts with another. |
| `entails` | The source relation semantically entails the target relation. |
| `derived_from` | A value, claim, or state is derived from an upstream item. |
| `handoff` | A message or result is transferred between agents. |
| `data_dependency` | A downstream item consumes an upstream item. |
| `logical_prerequisite` | One item is required before another can be justified. |
| `state_update` | An event changes the represented shared state. |
| `aggregation` | A result is formed from a set of upstream items. |

Table 7, Standardized observation and coordination states. These states describe what the protocol observed or executed, not whether a clinical answer is correct.

| State | Meaning |
|---|---|
| `UNKNOWN` | A relevant observation could not be established because evidence or execution was insufficient. |
| `NOT_OBSERVABLE` | The runtime did not expose the information at the current boundary. |
| `COORDINATED` | A matched CCI was merged at the pre-invocation boundary. Outcome is not implied. |
| `NO_REACHABLE_INVOCATION` | No legal, reachable, capable target invocation exists. |
| `INVALID_INJECTION_POINT` | The requested runtime insertion point is unsupported. |
| `CONSTRAINT_CONFLICT` | Constraints cannot be merged under the defined priority policy. |
| `INVOCATION_TIMEOUT` | The target invocation did not arrive within its allowed window. |
| `INJECTION_FAILED` | The CCI was not successfully merged into the target context. |

### C.4 Observation and coordination states

### C.5 Coordination graph and CCI interface

Table 8, Standardized coordination-stage inputs, coordination graph fields, and constraint-injection fields.

| Interface or field | Standardized content |
|---|---|
| Coordination input | Current $AG_t$, active anomaly records, witness subgraphs, and exposed runtime state. |
| `source_anomaly_refs` | References to the audit records that motivate coordination. |
| `affected_agent_refs` | Agents, states, or communication edges affected by the witness. |
| `target_invocation` | Earliest reachable invocation capable of performing the required check. |
| `injection_point` | Runtime context layer at which CCI is merged before the model request. |
| `scope/lifetime` | Object and duration for which the constraint applies. |
| `priority` | Conflict-resolution order for multiple constraints. |
| `provenance` | Route from witness subgraph to target invocation. |
| CCI content | Objects to review, evidence and constraints, checks before output, and upstream information to preserve. |

### C.6 Permission and information boundary

Table 9, Stage ownership and forbidden information flow.

| Capability | Online Audit | Online Coordination |
|---|---|---|
| Observe completed prefix | Read-only | Reads current audit state |
| Update audit provenance | Append-only | Does not rewrite |
| Declare or route anomaly | Declares witness | Routes from witness |
| Inject CCI | No | Only before matched request |
| Read gold, future events, terminal outcome | Forbidden | Forbidden |

The standardized vocabulary above is intentionally separate from retrospective evaluation labels. In particular, a coordination status records whether the runtime boundary was reached and processed. It is not a claim that the next answer is correct or that an anomaly disappeared.

# D USE OF LLMS

We used AI-assisted language tools only for copyediting, including grammar, spelling, and clarity. They were not used to generate data, run experiments, perform analyses, or determine scientific claims or conclusions. The authors reviewed the manuscript and take responsibility for its content.

## D.1 CATC REPRODUCTION SYSTEM PROMPTS

The following prompt scaffolds collect the system-level constraints needed to reproduce the CATC protocol. They define online information boundaries and mechanism statuses.

### D.1.1 OVERALL PROTOCOL

```
Implement exactly one online CATC loop:

runtime event prefix E<=t
-> Stage 1 Online Audit
-> GraphPatch
-> audit graph AG_t
-> Stage 2 Coordination Graph CG_t
-> pre-invocation interceptor
-> Coordination Constraint Injection (CCI)
-> new runtime events
-> Stage 1

Stage 1 must be external, read-only, and prefix-only.
Stage 2 may consume only anomalies already recorded in AG_t.
Do not add candidate branches, checkpoints, rollback, verifier gates,
post-hoc replacement, or multi-round repair.
```

### D.1.2 INPUTS AND ONLINE SAFETY

```
Require task input X, agent roles/topology, registered tools,
and a runtime event adapter.

Normalize every event as:

{
  "event_id": "...",
  "event_type": "response|handoff|tool_call|tool_result|aggregation|state_update",
  "agent_id": "...",
  "context": {},
  "payload": {},
  "provenance": {}
}

Preserve missing fields as unavailable or UNKNOWN.
Never infer absent thinking, evidence, reason, or tool output.

Before every audit or coordination step, reject or quarantine records
containing future_events, gold, terminal_score, final_reference,
root_cause, or an unobserved decisive step.

The online auditor may read only X, frozen K_X, and E_<=t.
```

### D.1.3 STAGE 1 AUDIT

```
For every newly appended event:

1. Run constrained audit code in a read-only sandbox.
2. Produce observations, not independent votes.
3. Compile a fixed GraphPatch.
4. Merge it immutably into AG_t and increment the graph version.
5. Preserve event and tool provenance on every node and edge.
6. Query the same graph at local, global, and joint levels.
7. Treat these as views of one graph, not three voters.

Emit an anomaly only when it contains:
claim/edge references, broken invariant, source spans,
observed step, possible upstream sources, severity,
confidence, unknown, and a witness subgraph.

Insufficient evidence or tool failure must be UNKNOWN, never agreement.
```

### D.1.4 AUDIT TOOLS

```
Use the common checks:

- check_source_attribution
- check_claim_source_binding
- check_claim_context_alignment
- check_thinking_output_alignment

Use task-specific checks:

- MedCalc:
  check_value_consistency
  check_aggregation_fidelity

- CaseReport:
  check_category_mapping

- EHRCon:
  check_label_field_match

- PMOA:
  check_temporal_consistency
  check_aggregation_fidelity

Keep deterministic format/structure checks as fast filters.
Semantic tools must remain provenance-bound observations.
```

### D.1.5 STAGE 2 COORDINATION

```
Build CG_t only from the current AG_t.

Merge anomalies only if they:
- share the same claim, message, handoff, or edge;
- lie on observed dependency descendants; or
- target the same later invocation.

Temporal proximity alone is insufficient.

Route the target invocation first to the source agent's next legal call.
If that agent will not execute again, follow observed
dependency/communication edges to the earliest reachable downstream
agent with authority to check or block the error.

If no legal target exists, record:
NO_REACHABLE_INVOCATION.
```

### D.1.6 CCI INTERCEPTOR

```
Invoke the interceptor after the target prompt/context is assembled
and before the model request.

The interceptor must:

- match both agent_id and invocation_id;
- construct a CCI containing:
  object to re-check,
  confirmed constraints,
  pre-output checks,
  preserved upstream information,
  source anomaly references,
  scope, lifetime, priority, and injection point;
- merge the CCI into an independent runtime namespace;
- never overwrite the original prompt, context, or history;
- be idempotent for (coordination_id, invocation_id);
- log provenance;
- return a mechanism status, not a claim that the clinical error is solved.
```

### D.1.7 STATUS AND ONLINE/OFFLINE BOUNDARIES

```
Use exactly these mechanism statuses:

COORDINATED
NO_REACHABLE_INVOCATION
INVALID_INJECTION_POINT
CONSTRAINT_CONFLICT
INVOCATION_TIMEOUT
INJECTION_FAILED
```

```
New response, handoff, tool, aggregation, and state events must return
to Stage 1. Do not inspect future events during the current decision.

Online artifacts may contain graph versions, anomaly provenance,
target invocation, CCI hash, injection status, scope/lifetime,
latency, and prompt growth.

Online artifacts must not contain gold labels, terminal scores,
future events, or offline references.

Evaluate final task score, rescue/damage, W/L/tie, tokens, and calls
only in a separate offline evaluator.
Missing downstream behavior must be NOT_OBSERVABLE.
```