RESOLVING AGENT UNCERTAINTY AND HANDOFF ERRORS UNDER SCARCE HUMAN ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

When an agent in a multi-agent system (MAS) encounters uncertainty, requesting clarification is preferable to silently guessing, particularly in cases where human expertise is required. However, an agent's uncertainty reflects information missing from its local view, not necessarily a system-wide need to query a human. Consequently, clarification requests can escalate system-resolvable uncertainty to humans, while silent errors in inter-agent handoffs may go unreported. A MAS can scale faster than available human attention, amplifying avoidable human queries and costly downstream revisions. We introduce ARBITER, a lightweight plug-in meta-agent that jointly handles agent-raised clarification requests and terminal handoffs at which errors may otherwise remain silent. Its policy decides whether and how to resolve them within the system or by querying a human while treating human attention as scarce. We train the policy by modeling the trade-off between end-to-end task success and human-query count across feasible resolution choices. Because multiple choices can achieve the same outcome, ARBITER retains this flexibility during training rather than imposing a unique target. We evaluate a 4B ARBITER on math task across MAS configurations, and 30B agent models. Our experiments show that ARBITER reduces human queries by up to 9.1% compared with strong prompt-only baselines while maintaining end-to-end task success. Specifically, we find that 21.4% of baseline human queries occur even when a non-human resolution succeeds. Overall, ARBITER reduces avoidable human queries by 8.7%, enabling more efficient allocation of human attention without parametric updates to the underlying MAS agents.

1 INTRODUCTION

Autonomous agents promise to convert inexpensive computation into useful work while freeing human time. Yet agent execution can scale with compute, whereas the human capacity to inspect outputs, answer questions, and make consequential decisions cannot. We refer to this finite oversight capacity as *human attention*. Catalini et al. (2026) characterize this divergence as a *measurability gap*: as execution becomes cheaper, human verification bandwidth becomes the binding constraint on useful deployment. This burden is already visible in workplace AI. Business Insider's coverage of Glean's 2026 Work AI Index (Spirlet, 2026) calls this form of human oversight "botsitting." Some of this supervision is indispensable when a task requires human preferences or high-stakes verification. Yet botsitting can also include avoidable supervision triggered by uncertainty that could be resolved without querying a human. Most existing clarification frameworks (Vijayvargiya et al., 2026; Pu et al., 2026; Dong et al., 2026; Trinh et al., 2026; Gulati et al., 2026) treat the human as the sole response channel when escalation occurs. We study this distinction in multi-agent systems (MAS). Our goal is to distinguish cases that genuinely require a human query from those resolvable within the agent system, thereby reserving scarce human attention for the former.

An agent's clarification request signals a perceived information gap, not necessarily a need for human input: the answer may already exist in local context, a tool, or another agent. Conversely, a terminal handoff can silently omit or distort context, leaving downstream agents with an incomplete view (Lin et al., 2026). Such errors may later require human diagnosis or rework. Since clarification-seeking does not consistently reflect model uncertainty (Testoni & Fernández, 2024), requests alone are an incomplete signal for intervention. We therefore monitor both agent-raised clarification requests and terminal handoffs (Figure 1).

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.

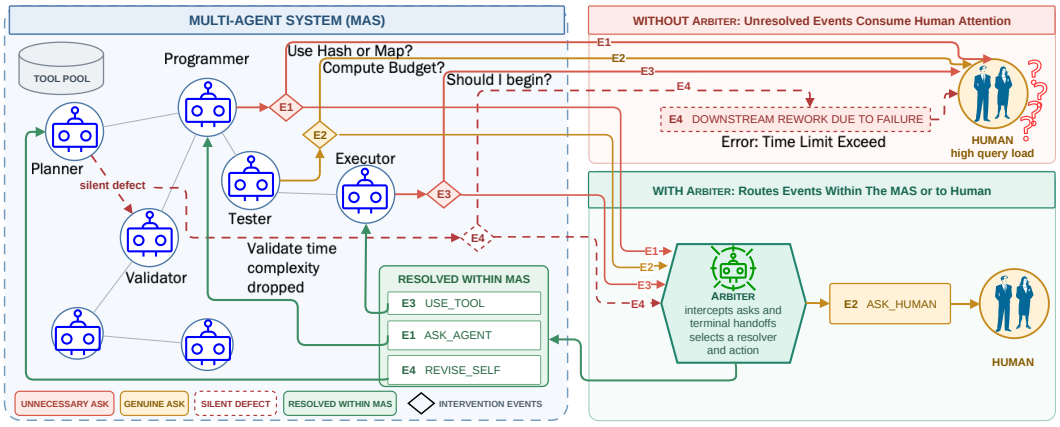

Figure 1: Overview of ARBITER. Without ARBITER, agent-raised requests may be sent directly to a human, while silent handoff defects can cause downstream rework. ARBITER intercepts both surfaces and routes each event to a suitable within-system resolution or to human input. The depicted routes illustrate the general interface rather than an exhaustive action set; each experimental setting instantiates the feasible subset supported by its data and executor.

These surfaces identify when intervention may be useful, but not what to do. We organize the resolution space by *what needs resolution*—a defect in present information or a gap in required information—and *how to resolve it*—which resolver to use and what action to take. These categories describe the choice space; ARBITER does not predict them as diagnostic labels.

Prior work addresses clarification requests and collaboration failures separately (Liang et al., 2024; Shinn et al., 2023; Madaan et al., 2023; Wu et al., 2024). AgentAsk (Lin et al., 2026) inspects handoffs by querying endpoint agents, but does not process agent-raised clarification requests or support human resolution. We introduce ARBITER, a lightweight plug-in meta-agent that intercepts both surfaces and chooses among feasible within-system and human resolutions without retraining the MAS agents. To our knowledge, it is the first such meta-agent to jointly handle agent-raised requests and potentially silent handoff errors while treating human attention as scarce.

A defect or gap does not determine which feasible resolution choice will improve the final task: multiple choices may succeed, while a plausible one may fail end to end. We therefore derive ARBITER's supervision from the downstream outcomes of feasible choices. At each recorded intervention event, we save the pre-intervention MAS state, execute each feasible choice from that state, and continue the workflow until termination. We record end-to-end task success and a human-query indicator for each choice. Comparing these outcomes lets us identify acceptable routes under the trade-off between task success and human attention, without assuming a single correct resolution choice. Section 3 defines the deployment objective, and Section 4 describes the learning methods. We evaluate ARBITER on GSM8K (Cobbe et al., 2021) across MAS configurations built with MasRouter (Yue et al., 2025) and multiple actor model families.

Our contributions are threefold: **(1)** a unified resolution problem spanning agent-raised clarification requests and terminal handoffs while treating human attention as scarce; **(2)** an outcome-grounded data-generation pipeline that produces set-valued acceptable-route targets and RL trajectories under the same trade-off between end-to-end task success and human attention; and **(3)** ARBITER, a lightweight plug-in meta-agent that applies this formulation across heterogeneous MASs. We instantiate ARBITER with a 4B-parameter Qwen3 model, smaller than the 24B–30B models it coordinates, without updating the underlying MAS agents.

Empirically, ARBITER reduces human queries by 9.1% relative to the prompt-only baseline and by 3.2% relative to a prompt-only baseline calibrated to ask less, while satisfying the predeclared one-percentage-point non-inferiority margin for task success. We find that 21.4% of baseline human queries occur at events where a non-human route succeeds end to end; ARBITER reduces these avoidable queries by 8.7%. Together, these results are consistent with better allocation of human attention beyond inference-time calibration alone.

2 Related Work

Clarification and silent failures. Clarification work studies when an agent should ask a user for missing information (Vijayvargiya et al., 2026; Trinh et al., 2026; Dong et al., 2026). It generally treats the user as the response channel. Separately, debate, reflection, and self-correction address errors in agent outputs (Liang et al., 2024; Shinn et al., 2023; Madaan et al., 2023; Wu et al., 2024). AgentAsk (Lin et al., 2026) is closest to our handoff surface: it inspects inter-agent messages and queries endpoint agents before errors propagate, but does not process agent-raised requests or support human resolution. ARBITER brings explicit requests and terminal handoffs into one policy.

Routing scarce assistance. HILA (Yang et al., 2026) learns when to use a peer solution, generate a new one, or defer to a human; CascadeDebate (Chang et al., 2026) escalates through deliberation and larger models before human fallback. Like these systems, ARBITER treats external help as costly. It applies this choice at both intervention surfaces while leaving the host MAS agents unchanged.

Learning from downstream outcomes. AgentAsk (Lin et al., 2026) uses task-rewarded RL, while Learning to Interrupt (Wang et al., 2026) and CollabLLM (Wu et al., 2025) use downstream consequences to guide communication decisions. ARBITER likewise grounds training in executed outcomes. Its offline data pipeline compares every feasible route from the same intercepted state using end-to-end task success and human-query use, allowing multiple routes to be acceptable rather than imposing a unique route label. Appendix A gives further comparisons.

3 Problem Formulation

Intervention events. A human gives a task x to a host MAS with agents $\mathcal{A} = \{A_1, \ldots, A_M\}$. The agents have fixed roles $\mathbf{r}$, policies, and communication topology G, and exchange outputs to complete the task. During execution, the human remains available as an external source of information. ARBITER, a meta-agent, decides how the workflow should proceed at two types of intervention events: (i) an explicit-ask event, triggered when an agent raises a clarification request, and (ii) a terminal-handoff event, triggered when an agent's output is passed to another agent or returned as the final answer. An output that satisfies both definitions creates a single intervention event. These event types instantiate the two intervention surfaces introduced in Section 1: agent-raised clarification requests produce explicit-ask events, and terminal handoffs produce terminal-handoff events. The t-th event e_t is triggered by an agent output y_t; we call the agent that produced y_t the *actor*. The event occurs after y_t is generated but before it is consumed, so ARBITER can preserve or replace the output without changing the underlying agents. The visible collaboration history h_t and feasible resolution-choice set $\mathcal{U}_t$ vary by event. With static context $c = (x, \mathcal{A}, \mathbf{r}, G)$, ARBITER observes $o_t = (c, h_t, y_t, \mathcal{U}_t)$. This gives ARBITER a system-level coordination view combining the visible history, MAS topology, and feasible choices. However, o_t contains neither a label indicating whether y_t has a defect or gap nor the counterfactual outcomes of unexecuted choices. Training allows ARBITER to estimate which choices are more likely to yield end-to-end task success given o_t, but the realized outcome of each choice remains unknown at decision time. Thus, ARBITER selects among feasible resolution choices using their expected downstream consequences, rather than predicting a defect, gap, or resolver label.

Feasible choices. Each feasible resolution choice $u \in \mathcal{U}_t$ specifies an action $a(u)$ and, when required, a resolver. A choice can preserve the candidate, retry or revise the current actor, query a human, or query a named non-self agent. With parameters θ, the meta-agent policy $\pi_\theta(u \mid o_t)$ has support only on $\mathcal{U}_t$. Section 4 explains how these choices are represented and executed, including any generated question or feedback. Within-system resolution executes a non-human choice, whereas human resolution executes ASK-HUMAN. For outcome analysis, we call an event *system-resolvable* when at least one feasible non-human choice attains end-to-end task success.

Constrained objective. Human attention is scarce, which we operationalize as the number of human queries in a complete task episode. Minimizing this count alone yields the trivial policy that never queries the human, whereas a weighted objective requires an arbitrary exchange rate with end-to-end task success. We therefore minimize expected human queries per episode subject to preserving the end-to-end task success of the prompt-only baseline.

For a task x drawn from deployment distribution $\mathcal{D}$, policy π and policy-execution randomness ω induce a complete trajectory $T_\pi(x, \omega)$. Its intervention-event indices form $\mathcal{E}(T_\pi)$. At each $t \in \mathcal{E}(T_\pi)$, choice u_t has action $a(u_t)$. Binary end-to-end task success is $S(T_\pi) \in \{0, 1\}$, and the cumulative human-query count is

$$H(T_\pi) = \sum_{t \in \mathcal{E}(T_\pi)} \mathbf{1}\{a(u_t) = \text{ASK-HUMAN}\}.$$

Every executed query counts once, regardless of its response. The quantity $J_S(\pi)$ measures expected end-to-end task success, while $J_H(\pi)$ measures expected human queries per episode. Formally,

$$J_S(\pi) = \mathbb{E}_{x \sim \mathcal{D},\, \omega}[S(T_\pi(x, \omega))], \qquad J_H(\pi) = \mathbb{E}_{x \sim \mathcal{D},\, \omega}[H(T_\pi(x, \omega))].$$

We evaluate the frozen prompt-only policy π_{base} on matched tasks and seeds under the same MAS and serving contract. For a predeclared non-inferiority margin $\delta \geq 0$, the deployment objective is

$$\min_{\pi} J_H(\pi) \quad \text{subject to} \quad J_S(\pi) \geq J_S(\pi_{\text{base}}) - \delta, \tag{1}$$

We set $\delta = 0.01$, permitting at most a one-percentage-point reduction in end-to-end task success, and also report the strict case $\delta = 0$. Among policies meeting this reliability requirement, the objective selects the one with the fewest expected human queries per episode.

4 ARBITER

ARBITER is a plug-in meta-agent that intercepts consequential agent outputs and decides whether the MAS should proceed, self-correct, consult another agent, or ask the human. We train its policy under the objective in Section 3: reducing human queries while preserving end-to-end task success. Both learning methods observe an intercepted MAS state and its feasible resolution choices. The offline method reuses recorded outcomes for every feasible choice; the online method samples decisions from its current policy and evaluates them through fresh MAS continuations.

4.1 SHARED DECISION AND EXECUTION INTERFACE

At each intervention event, ARBITER freezes the MAS immediately after the actor generates a candidate and before that candidate is consumed. It observes the task and visible message history, the actor's role, its candidate output, and a manifest of feasible resolution choices. Gold answers, route outcomes, and diagnostic labels are never policy inputs. Figure 1 illustrates the broader architecture: the host MAS may expose any executable intervention through this manifest. We implement each feasible resolution choice $u \in \mathcal{U}_t$ as a route $\rho \in \mathcal{R}(o_t)$, specifying an action and, when required, a resolver. In the datasets and systems evaluated here, the event-specific route set is drawn from five families: CONTINUE, SELF-RETRY, REVISE-SELF, ASK-HUMAN, and one ASK-AGENT:v route for each eligible non-self agent v. Thus, these routes define our experimental instantiation, not a limitation of the ARBITER interface.

To execute a route, ARBITER also generates route-conditioned content p when required. The resulting structured intervention pairs ρ with p. Continue accepts the candidate; self-retry resamples the actor; revise-self reruns it with grounded feedback; and query routes append a resolver's answer before rerunning the actor. Execution then resumes the host-MAS workflow. Constrained decoding limits the route prefix to the feasible set, while the model generates the question or feedback. Appendix Section C gives the complete interface and execution contract.

4.2 SHARED LEARNING SETUP

Both methods train on individual interventions. Executing a choice and continuing the MAS from a fixed intercepted state yields final-answer task success and a human-query indicator for that choice. These one-intervention outcomes provide tractable surrogates for the episode-level constrained objective in Eq. 1.

4.3 EXACT OFFLINE CONTEXTUAL-BANDIT LEARNING

Under this shared setup, the offline method obtains outcomes for every feasible resolution choice once during data generation and reuses them throughout optimization. For each of the N admitted

training events e, we checkpoint the pre-intervention state and execute every feasible route from that same state. This matched replay turns route learning into a finite-action, full-information contextual-bandit problem: for every $\rho \in \mathcal{R}(o_e)$, we observe stored end-to-end task success $z_{e\rho} \in \{0, 1\}$ and human-query indicator $h_{e\rho} = \mathbf{1}\{\rho = \text{ASK-HUMAN}\}$. Unlike standard logged bandit feedback, the learner therefore has one outcome for every feasible route at each recorded context.

Let $g_\theta(\rho, o)$ be the log probability of the canonical route prefix and let $q_\theta(\rho \mid o)$ be its softmax over feasible routes. We call q_θ the router policy: it is the route-level implementation of the general policy π_θ in Section 3. With the frozen prompt-only router policy q_0, the route-level implementation of π_{base}, we minimize

$$\mathcal{L}_{\text{off}}(\theta) = \frac{1}{N} \sum_e \Bigg[- \sum_{\rho \in \mathcal{R}(o_e)} q_\theta(\rho \mid o_e) z_{e\rho} + \lambda_h \sum_{\rho \in \mathcal{R}(o_e)} q_\theta(\rho \mid o_e) h_{e\rho} + \beta D_{\text{KL}}(q_\theta(\cdot \mid o_e) \| q_0(\cdot \mid o_e)) - \tau \mathcal{H}(q_\theta(\cdot \mid o_e)) \Bigg]. \tag{2}$$

The first two terms favor successful routes while penalizing human queries; the remaining terms anchor the learned router policy and maintain exploration. The expectation is *exact* with respect to the finite feasible-route set and its frozen recorded outcomes because it sums over every route rather than sampling actions from the current policy. It is not an exact estimate of fresh deployment end-to-end task success. We use set-valued supervised learning to initialize and benchmark the router policy; its construction and the offline constrained variant are given in Appendix C.2.

4.4 Online primal–dual learning

Reusing stored outcomes makes offline optimization efficient, but those outcomes cannot evaluate content newly generated by the current policy. The online method replaces this frozen feedback with fresh execution. It is a separate alternative to Eq. 2, although the evaluated run uses a separately trained offline query-constrained checkpoint as a warm start for stable route exploration (Appendix C.3). For each sampled context, the policy generates a group of complete route–payload interventions; each is executed once to obtain binary end-to-end task success S for the fresh continuation and human-query indicator H for the selected route. In dual batch k, each sample receives

$$r_{\mu_k} = -H + \mu_k S, \tag{3}$$

and GRPO (Shao et al., 2024) updates both route and payload tokens. The multiplier is updated after the complete batch as

$$\mu_{k+1} = \text{clip}_{[0,\mu_{\max}]} \left(\mu_k + \eta [S_{\text{target}} - \overline{S}_k] \right), \tag{4}$$

where $\overline{S}_k$ is mean task success from fresh continuations and S_{target} is fixed before training from the initializer's development performance. When end-to-end task success falls below target, the multiplier increases its weight in the next batch; once the target is met, the dual update again emphasizes avoiding human queries. Thus online learning directly addresses the route–payload and continuation mismatch that cannot be measured by frozen offline replay.

4.5 Policy selection and inference

The offline and online methods produce alternative ARBITER policies under the same execution interface. At inference, the same feasible-route mask is enforced and a development-tuned penalty may calibrate only the ASK-HUMAN score. The checkpoint, prompt, decoding contract, and penalty are frozen before evaluation. Appendix Sections C.3 and C.4 provide the optimization and selection details.

5 Outcome-Grounded Training Data

Paired tasks. We use GSM8K (Cobbe et al., 2021) because numeric answers let us measure task success after an intervention. Each fully specified problem is paired with an underspecified version. Qwen and Nemotron independently identify the problem's facts; we remove one load-bearing fact

from their intersection while preserving the question and unrelated details. We keep a pair only when each of the three host-agent models in Section 6 fails all three trials without the fact and successful solving returns after reinsertion. During MAS execution, only the proxy human (Appendix B.3) has access to the withheld fact. The two arms test whether ARBITER avoids unnecessary escalation while preserving useful queries; matched route outcomes, not task arm, determine whether a query was system-resolvable. We assign each source problem, its variants, and their branches to one split.

Intervention outcomes. We run heterogeneous host MASs on both task arms and save the state at each explicit ask or terminal handoff, after the actor generates an output but before it is consumed. The policy sees the task, visible history, actor role and output, topology, and feasible routes; it does not see the gold answer, withheld fact, task-arm label, or route outcomes. From this same state, we execute each feasible route to task completion and record final-answer success $z_{e\rho}$ and the human-query indicator $h_{e\rho}$. The common preceding trajectory makes the route outcomes comparable.

Targets and scale. The acceptable target contains every successful non-human route when one exists; otherwise it contains a successful ASK-HUMAN route. This retains equally effective choices without imposing a unique label. The stored-outcome training split has 1,003 saved intervention states (contexts). Across them, 2,463 context–route pairs are acceptable targets, since a context can admit several successful routes. Of 3,168 test contexts, 1,978 have a successful recorded route; the other 1,190 remain in the denominators of the complete policy comparison. On the test split, actors raise explicit asks in 60.5% of underspecified contexts versus 26.7% of fully specified contexts. Appendix Sections B and B.5 give the construction, execution, and count details.

6 EXPERIMENTAL SETUP

Our experiments test whether ARBITER reduces human queries while preserving task success. An unmodified prompt-only policy and a calibrated prompt-only policy distinguish learning from inference-time query suppression. We compare supervised, exact offline, and online primal–dual learning under the shared intervention interface of Section 4.

6.1 BENCHMARK AND HOST MULTI-AGENT SYSTEMS

GSM8K provides numeric exact-match task success. Each source problem yields paired fully specified and one-fact-withheld tasks; all variants and branches from a source remain in one split (Section 5). We use 10% of the training split for checkpoint, hyperparameter, prompt, decoding, and calibration choices, then freeze those choices for the held-out comparison on 3,168 intervention contexts. The fully specified arm tests whether the policy avoids unnecessary queries; the withheld-fact arm tests whether it preserves queries that can recover needed information.

To test a plug-in policy across different hosts, MasRouter (Yue et al., 2025) constructs task-specific MASs with varying agent counts, roles, and communication topologies. Reachable resolvers also vary by intervention event. The actors are drawn from Qwen3-30B-A3B, Nemotron-30B, and Devstral-24B; ARBITER is Qwen3-4B. Every policy receives the same visible history and event-specific feasible routes, with unavailable routes masked.

6.2 POLICIES COMPARED

The *prompt-only baseline* uses the frozen 4B model without parameter updates and the prompt specified in Appendix B.2. The *calibrated prompt-only baseline* uses the same model and prompt with a development-tuned penalty on ASK-HUMAN. This comparator tests whether query reduction comes from learning which events the MAS can resolve, beyond globally lowering the propensity to ask. We also report two supervised fine-tuning (SFT) policies trained on set-valued acceptable-route targets. *Conservative SFT* is an early checkpoint with near-baseline human-query use and coverage of all five action families. *Maximum-Utility SFT* is the supervised candidate chosen by the development score of task-success rate minus 0.05 times human-query rate, without a hard query budget. This score selects a checkpoint; it is not the SFT training objective. The exact offline contextual-bandit policy uses stored outcomes for every feasible route; online primal–dual GRPO evaluates newly generated route–payload interventions through fresh continuations. The reported offline operating

Table 1: Policy comparison under stored-route replay on 3,168 held-out contexts. A selected ASK-HUMAN route counts as one query; Δ queries is relative to prompt-only. "Yes" requires paired, source-clustered 95% lower confidence bounds above -1 percentage point for the success difference and above zero for query reduction.

Policy	Successes	Human queries	Δ queries	Qualifies vs. base
Prompt-only	1,715	911	—	Reference
Calibrated prompt-only	1,700	855	-56	Yes
Conservative SFT	1,715	915	$+4$	No
Maximum-utility SFT	1,853	2,170	$+1{,}259$	No
Online primal–dual, step 15	1,698	845	-66	Yes
Exact offline contextual bandit (calibrated)	1,694	828	-83	Yes

point uses a development-tuned inference-time ASK-HUMAN score penalty; the reported online step-15 point uses no such penalty. These alternatives compare imitation, complete offline feedback, and fresh online interaction. Training and decoding details appear in Appendix C.2–C.4.

6.3 Metrics, selection, and statistical comparisons

For each held-out context, reliability is numeric exact-match success of the stored continuation for the selected route; selecting ASK-HUMAN counts as one human query. These intervention-level measures do not capture complete policy-induced episodes. We fix a one-percentage-point success non-inferiority margin relative to the unmodified prompt-only baseline. On the 10% development subset, we select the lowest-query candidate whose paired success interval clears this margin, then freeze its configuration before test evaluation. Requiring success first prevents a policy from earning credit for reducing queries simply by refusing useful escalation. We also report a strict zero-loss sensitivity check.

Policies are paired on the same contexts. We aggregate variants and branches within their GSM8K source problem and obtain 95% confidence intervals by bootstrapping source clusters, preserving dependence among contexts from the same problem. On the held-out comparison, the success gate requires the lower bound of the paired success-difference interval to exceed -1 percentage point; a query-reduction claim also requires the lower bound of the paired query-reduction interval to exceed zero. We apply both criteria against the calibrated prompt-only policy as well. Secondary diagnostics include avoidable human queries, structured-output validity, and feasible-route coverage.

7 Results: Preserving Success with Fewer Human Queries

We evaluate the frozen policies on the same 3,168 held-out GSM8K intervention contexts using stored outcomes for their selected routes. Our primary question is whether offline and online policies reduce human attention while satisfying the predeclared one-percentage-point non-inferiority margin. Table 1 summarizes the operating points that answer this question; additional operating points and source-clustered interval details appear in Appendix D.

7.1 Both RL policies reduce queries under the success constraint

Both RL policies pass the predeclared success and query-reduction gates. The calibrated exact offline policy selects ASK-HUMAN on 828 contexts, compared with 911 for the prompt-only policy—a **9.1% reduction**. The online step-15 policy, evaluated without an inference-time ASK-HUMAN score penalty, selects it on 845 contexts—a **7.2% reduction**. For the offline point, the source-macro task-success change is -0.51 percentage points, with a 95% confidence interval of $[-0.91, -0.14]$, entirely within the predeclared one-point non-inferiority margin. The corresponding source-macro reduction in query rate is 1.92 percentage points, with a 95% confidence interval of $[1.24, 2.61]$. Both RL policies preserve the required level of task success under stored-route replay while using fewer human queries. Under the stricter zero-loss criterion, the prompt-only policy remains preferred (Appendix D).

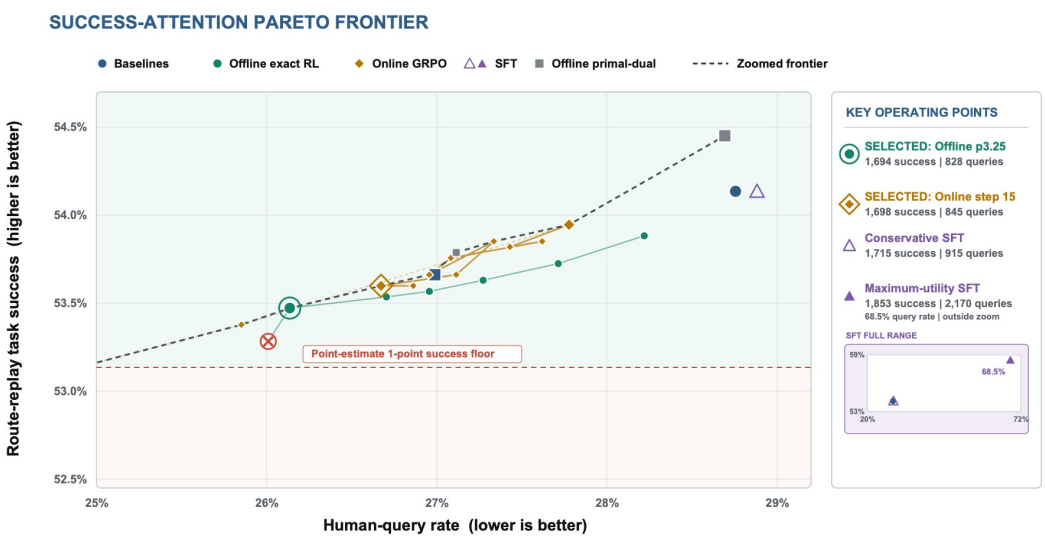

Figure 2: Success–attention operating points on the common 3,168-context test set. The main panel zooms to a 25–29.2% human-query rate; its dark dashed curve traces the observed frontier. The inset shows the prompt-only baseline and SFT points across the 20–72% range. The horizontal dashed line marks the point-estimate success floor, one percentage point below prompt-only; qualification uses paired, source-clustered 95% confidence intervals. Moving up and left improves task success and reduces query rate. The outlined offline p3.25 and online step 15 points are selected and pass the required comparisons. Offline uses a tuned ASK-HUMAN score penalty; online step 15 uses none at inference.

7.2 Learning contributes beyond inference-time calibration

A score penalty applied to the frozen prompt-only policy reduces its query count from 911 to 855 while yielding 1,700 successes. This shows that part of the baseline's query use can be controlled at inference time. Under the same structured-output and serving contract, the learned-and-calibrated exact offline policy reaches 828 queries, 27 fewer than this strongest calibration-only operating point. The online step-15 policy, with no inference-time score penalty, reaches 845 queries, 10 fewer than the calibrated prompt-only point. Both paired comparisons pass the task-success and positive query-reduction gates against the calibrated baseline (Appendix D). These results show an incremental contribution from learning beyond inference-only calibration without attributing the effect to a single mechanism.

7.3 Avoidable human queries decline in count

An event is system-resolvable when at least one feasible non-human route succeeds from the same intervention state. The prompt-only policy asks the human at 195 such events, or 21.4% of its 911 queries. The calibrated exact offline policy makes 178 such queries, 17 fewer (8.7% relative), while making 83 fewer queries overall. Avoidable queries constitute 21.5% of its 828 human queries, nearly the same share as for prompt-only. The results therefore show fewer avoidable queries in count, but do not establish that the policy preferentially removes them.

7.4 Offline calibration and online training yield qualifying points

Figure 2 shows the measured success–attention trade-off. For the frozen offline checkpoint, increasing the inference-time human-query penalty reduces queries while also lowering success. The reported 828-query point is the lowest-query point in that tested offline penalty sweep that passes the interval-based success gate; a stronger penalty fails it. This identifies a qualifying point within the tested offline calibration sweep.

Online primal–dual learning provides a complementary route to this frontier. As training proceeds, its checkpoints generally move toward lower human-query use while remaining near the required task-success level. At step 15, the online policy attains 1,698 successes with 845 queries and passes the required comparisons with both the unmodified and calibrated prompt-only policies, without an inference-time ASK-HUMAN score penalty. The two reported RL points differ by 17 queries at their tested settings. Because the calibration settings differ and no later online checkpoints were evaluated, this gap does not establish an inherent advantage for either learning method. Additional online training or calibration might change the operating point; those outcomes are not measured here. Additional tested points appear in Appendix D.

7.5 SUPERVISED LEARNING EXPOSES A HUMAN-HEAVY SHORTCUT

The SFT operating points explain why valid route imitation is insufficient for our objective. Conservative SFT yields 1,715 successes and 915 queries, near the prompt-only counts. Selecting SFT for maximum scalar utility moves in the opposite direction, reaching 1,853 successes with 2,170 human queries. The large maximum-utility shift reveals the central optimization risk: supervised targets admit a human-heavy shortcut, whereas imitation alone does not specify which point on the success–attention frontier is acceptable. Outcome-grounded RL represents this trade-off directly, and both tested RL methods yield policies that satisfy the success and query-reduction criteria.

8 CONCLUSION

Human attention becomes a bottleneck when multi-agent systems escalate questions they could resolve internally or leave handoff errors for people to diagnose. We propose ARBITER, a plug-in meta-agent that intercepts clarification requests and terminal handoffs and learns from downstream outcomes when to resolve an issue within the MAS or ask a human. On held-out GSM8K intervention contexts, this approach reduces human queries while meeting the predeclared task-success constraint, without retraining the host agents.

9 LIMITATIONS

The evaluation uses GSM8K numeric exact-match tasks. Each underspecified problem is a fully specified source with one load-bearing fact removed. That construction gives a known missing fact and a paired control, but it does not cover naturally occurring information gaps, several missing facts at once, or open-ended tasks whose success is not a single number.

Human attention is counted as one unit per ASK-HUMAN selection. The answer comes from a Qwen3-4B proxy (Appendix B.3), not from a person. The proxy supplies the withheld fact when the question asks for it and otherwise replies that it does not know. This cost model does not capture the varying burden of real responses. A query is one component of human time: the evaluation does not measure waiting, response effort, context switching, or other uses of attention. No study with live users is reported, so the results do not establish how much human time fewer queries would save.

ARBITER is one 4B backbone. The host actors are three model families in MasRouter teams. Transfer to unseen teams, other actor families, or another meta-agent backbone is untested.

The headline comparison replays the selected route from a fixed intervention state and scores the stored continuation. It does not rerun the task from the start, and it does not measure outcome variability when the current policy generates a new question, feedback, or continuation.

IMPACT STATEMENT

ARBITER reduces human queries while meeting a task-success constraint. That reduction is not a proposal to drive queries to zero or to replace people. Humans remain essential when the system cannot resolve an event. A lower query count is not, by itself, a measured saving of human time. Queries omit waiting, response effort, context switching, and other uses of attention, and this paper reports no study with live users. We therefore do not claim how much human time is saved.

Reproducibility Statement

Section 6 states the benchmark, host models, policies, selection rule, and statistical gates. Section 5 summarizes the outcome-grounded data, and Section 4 states the shared interface and learning objectives. Appendix B, including the prompts in Appendix B.2 and the split manifest in Appendix B.5, specifies how paired tasks, intervention contexts, and stored route outcomes are built. Appendix C gives the execution interface and the exact offline, online primal–dual, and calibration procedures. Appendix B.3 specifies the proxy human. Appendix C.5 lists the hyperparameters of the reported policies, Appendix C.6 gives the vLLM inference configuration, and Appendix C.7 states the cluster and wall-clock times. Appendix D gives additional operating points and the source-clustered bootstrap. We will release the source code, environment lockfile, and regeneration scripts upon publication.

AI Use Statement

We used Claude (Anthropic) and ChatGPT (OpenAI) as coding and writing assistants during the Arbiter project. They supported implementation and debugging of the Arbiter framework and analysis code, and drafting and editing of manuscript text. The authors reviewed and verified the work and are responsible for all design decisions, scientific claims, analyses, and conclusions. The language models evaluated in this study are research subjects, not authorial assistants.

References

Christian Catalini, Xiang Hui, and Jane Wu. Some simple economics of AGI. *arXiv preprint arXiv:2602.20946*, 2026. doi: 10.48550/arXiv.2602.20946. URL https://arxiv.org/abs/2602.20946.

Raeyoung Chang, Dongwook Kwon, Jisoo Lee, and Nikhil Verma. CascadeDebate: Multi-agent deliberation for cost-aware LLM cascades. *arXiv preprint arXiv:2604.12262*, 2026. doi: 10.48550/arXiv.2604.12262. URL https://arxiv.org/abs/2604.12262.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

Yijiang River Dong, Tiancheng Hu, Zheng Hui, Caiqi Zhang, Ivan Vulić, Andreea Bobu, and Nigel Collier. Value of information: A framework for human–agent communication. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42879–42896, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1987. URL https://aclanthology.org/2026.acl-long.1987/.

Aijing Gao, Yiming Kang, Mengdie Flora Wang, and Jae Oh Woo. Knowing when to ask: Self-gated clarification for hierarchical language agents. *arXiv preprint arXiv:2606.11349*, 2026. doi: 10.48550/arXiv.2606.11349. URL https://arxiv.org/abs/2606.11349.

Anmol Gulati, Hariom Gupta, Elias Lumer, Sahil Sen, and Vamse Kumar Subbiah. Ask early, ask late, ask right: When does clarification timing matter for long-horizon agents?, 2026. URL https://arxiv.org/abs/2605.07937.

Xu He, Di Wu, Yan Zhai, and Kun Sun. SentinelAgent: Graph-based anomaly detection in LLM-based multi-agent systems. *arXiv preprint arXiv:2505.24201*, 2025. doi: 10.48550/arXiv.2505.24201. URL https://arxiv.org/abs/2505.24201.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, 2024. doi: 10.18653/v1/2024.emnlp-main.992. URL https://aclanthology.org/2024.emnlp-main.992/.

Bohan Lin, Kuo Yang, Zelin Tan, Yingchuan Lai, Chen Zhang, Guibin Zhang, Xinlei Yu, Miao Yu, Xu Wang, Yudong Zhang, and Yang Wang. AgentAsk: Multi-agent systems need to ask. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28055–28077, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1294. URL https://aclanthology.org/2026.acl-long.1294/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

So Yeon Min, Yue Wu, Jimin Sun, Max Kaufmann, Fahim Tajwar, Yonatan Bisk, and Ruslan Salakhutdinov. Self-regulation and requesting interventions. *arXiv preprint arXiv:2502.04576*, 2025. doi: 10.48550/arXiv.2502.04576. URL https://arxiv.org/abs/2502.04576.

George Pu, Michael S. Lee, Udari Madhushani Sehwag, David J. Lee, Bryan Zhu, Yash Maurya, Mohit Raghavendra, Yuan Xue, and Samuel Marc Denton. LHAW: Controllable underspecification for long-horizon tasks. *arXiv preprint arXiv:2602.10525*, 2026. doi: 10.48550/arXiv.2602.10525. URL https://arxiv.org/abs/2602.10525.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Thibault Spirlet. The rise of the 'botsitters'. *Business Insider*, June 2026. URL https://www.businessinsider.com/botsitting-ai-hidden-human-labor-at-work-2026-6. Accessed 2026-08-06.

Alberto Testoni and Raquel Fernández. Asking the right question at the right time: Human and model uncertainty guidance to ask clarification questions. *arXiv preprint arXiv:2402.06509*, 2024. doi: 10.48550/arXiv.2402.06509. URL https://arxiv.org/abs/2402.06509.

Tu Trinh, Mohamed Elfeki, Guangze Luo, Kelvin Luu, Nathan Hunt, Ernesto Hernandez, Nandan Marwaha, Yannis Yiming He, Charles Wang, Fernando Carabedo, Alessa Castillo, and Bing Liu. Hil-bench (human-in-loop benchmark): Do agents know when to ask for help?, 2026. URL https://arxiv.org/abs/2604.09408.

Sanidhya Vijayvargiya, Xuhui Zhou, Akhila Yerukola, Maarten Sap, and Graham Neubig. AmbigSWE: Interactive agents to overcome underspecificity in software engineering. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=X2yzXtH4wp.

Danqing Wang, Da Yin, Ruta Desai, Lei Li, Asli Celikyilmaz, and Ansong Ni. Learning to interrupt in language-based multi-agent communication. *arXiv preprint arXiv:2604.06452*, 2026. doi: 10.48550/arXiv.2604.06452. URL https://arxiv.org/abs/2604.06452.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. CollabLLM: From passive responders to active collaborators. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://arxiv.org/abs/2502.00640.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12846–12867, 2024. doi: 10.18653/v1/2024.emnlp-main.714. URL https://aclanthology.org/2024.emnlp-main.714/.

Shuang Xie, Yunan Lu, Han Li, and Lingyun Wang. EARS: Explanatory abstention for reliable sub-agent modeling in large-scale multi-agent systems. *arXiv preprint arXiv:2606.18668*, 2026. doi: 10.48550/arXiv.2606.18668. URL https://arxiv.org/abs/2606.18668.

Wei Yang, Defu Cao, Jiacheng Pang, Muyan Weng, and Yan Liu. Adaptive collaboration with humans: Metacognitive policy optimization for multi-agent LLMs with continual learning. *arXiv preprint arXiv:2603.07972*, 2026. doi: 10.48550/arXiv.2603.07972. URL https://arxiv.org/abs/2603.07972.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. *arXiv preprint arXiv:2502.11133*, 2025. doi: 10.48550/arXiv.2502.11133. URL https://arxiv.org/abs/2502.11133.

A Additional related work

Clarification for expressed uncertainty. Ambig-SWE (Vijayvargiya et al., 2026), LHAW (Pu et al., 2026), and HiL-Bench (Trinh et al., 2026) evaluate whether agents detect missing information, ask useful questions, and use the answers without over-asking. Gulati et al. (2026) studies how the timing of clarification affects a longer trajectory. VoI (Dong et al., 2026) weighs a question's expected utility against user effort, while Gao et al. (2026) makes asking compete with task actions. These methods clarify when a user query is useful; they do not select among a human, another MAS agent, and an internal repair when an agent raises a request.

Handoff monitoring and agent repair. AgentAsk (Lin et al., 2026) places a clarification module at inter-agent handoffs and decides whether, what, and which endpoint agent to ask to prevent error propagation. EARS (Xie et al., 2026) trains sub-agents to replace unsupported answers with explanatory abstentions that a coordinator can use for clarification or rerouting. SentinelAgent (He et al., 2025) monitors execution graphs for anomalous node, edge, and path behavior and can apply policy-based interventions. Debate (Liang et al., 2024), reflection (Shinn et al., 2023; Madaan et al., 2023), and verification-based self-correction (Wu et al., 2024) provide ways to repair agent outputs. These approaches make failures visible or correct them, but do not jointly route agent-raised requests and terminal handoffs to within-system or human resolution through a plug-in policy.

Routing under costly assistance. HILA (Yang et al., 2026) chooses among adopting a peer solution, generating a new solution, and deferring to a human. Its continual-learning loop also uses human feedback to update the agents. Self-Regulation and Requesting Interventions (Min et al., 2025) learns when help from a stronger model or additional test-time computation is worth its cost. CascadeDebate (Chang et al., 2026) uses debate and larger models before a human fallback. These systems establish the value of treating external assistance as scarce; ARBITER applies that trade-off to explicit requests and handoffs without updating the host agents.

Learning from executed outcomes. AgentAsk (Lin et al., 2026) supplements teacher demonstrations with task-rewarded RL, and HILA (Yang et al., 2026) evaluates strategic actions through their consequences and costs. Learning to Interrupt (Wang et al., 2026) branches conversations at candidate interruption points and derives labels from final task performance and communication cost. CollabLLM (Wu et al., 2025) forward-simulates candidate responses to estimate their downstream value. ARBITER also uses downstream outcomes, but its offline data generation evaluates every feasible resolution route from the same intercepted event and records both task success and whether the route queries a human. This supports a set of acceptable routes rather than a unique diagnostic or route label.

B GSM8K DATA CONSTRUCTION

B.1 PAIRED TASK GENERATION AND VALIDATION

We start from a fully specified GSM8K source problem and construct one paired underspecified variant. Qwen and Nemotron independently enumerate the facts in the original problem. We select one load-bearing fact from the intersection of their lists and remove it while preserving the requested quantity, the entities involved, and all unrelated details. Selecting an agreed fact and removing only that fact avoids arbitrary deletions or multiple simultaneous gaps. The removed content is stored as task-authority metadata and is never included in the policy-visible state.

Candidates pass a removal-and-restoration check before entering the corpus. Without the selected fact, Qwen3-30B-A3B, Nemotron-30B, and Devstral-24B must each fail on three trials under numeric exact match. We then append the removed fact to the underspecified problem; Qwen and Nemotron must both match the gold answer using the usual GSM8K solver prompt. We also reject rewrites that change the requested quantity, introduce a second ambiguity, reveal the answer, or make the original wording internally inconsistent. These checks make the two arms interpretable: fully specified tasks provide a control with no deliberately hidden information, whereas underspecified tasks test whether a policy preserves useful human queries. Empirical route outcomes, not task-arm identity alone, determine whether a query was avoidable.

The proxy human represents the task authority and is available only after ASK-HUMAN is selected, so neither its response nor the withheld fact can leak into route selection. On underspecified tasks it receives the withheld fact; on fully specified tasks it receives no withheld fact and declines. Appendix B.3 specifies the Qwen3-4B model and prompts. This controlled contract isolates the allocation question studied here—whether human-held information is required—from variation in user availability or expertise.

B.2 PROMPT AND RESPONSE CONTRACTS

The following templates give the instructions used to construct and check the paired tasks. Braced fields are filled with the corresponding source problem, gold solution, or removed fact. The two fact-inventory judges receive the same system and user prompts.

Fact enumeration. System:

```
Output only valid JSON.
```

User:

```
You analyze a FULLY specified grade-school math word problem.
List EVERY atomic, load-bearing fact required to compute the answer.
A fact is one quantity, rate, price, count, or condition.
Do NOT merge independent numbers into one fact.
Do NOT include the final question itself as a fact.
PROBLEM:
{original_prompt}
GOLD SOLUTION (for reference only; do not invent facts absent from the problem):
{canonical_solution}
Respond exactly:
{
  "facts": [
    {
      "fact_id": "f1",
      "removal_type": "quantity|rate|price|count|condition",
      "text": "complete sentence stating the fact",
      "answer": "concise value/condition a human would supply if asked",
      "span_hint": "short verbatim span from PROBLEM if possible, else null"
    }
  ]
}
```

Fact removal and rewrite. The rewrite step receives the selected fact from the intersection of the two inventories. Its system message is:

```
Output only valid JSON.
```

Its user template is:

```
Rewrite the PROBLEM so it is missing EXACTLY the fact below.
Keep the final question identical.
Keep every other fact intact so the problem still reads as a natural word problem.
Do not add new information.
CRITICAL | how to omit the fact (human-like underspecification):
- Delete or paraphrase the clause that states the fact so the sentence still reads
  as something a person might say (e.g. "accommodate people" or drop the count
  clause entirely | NOT "accommodate ___ people").
- NEVER use fill-in-the-blank markers: underscores (___), question-mark runs (???),
  bracket placeholders ([missing], [blank], [unknown]), or the words unknown /
  missing / blank / TBD / N/A as stand-ins for the removed value.
- Do not write "the number is unknown" or similar worksheet phrasing.
PROBLEM:
{original_prompt}
FACT TO REMOVE:
id={fact_id}
text={fact_text}
answer={fact_answer}
type={removal_type}
Respond exactly:
{
  "underspecified_question": "...",
  "removal_type": "...",
  "removed_fact": "...",
  "removed_fact_answer": "..."
}
```

Solver and restoration gates. The three models use the following GSM8K system prompt for the no-fact trials. Qwen and Nemotron also use it for the restoration gate:

```
You are a math tutor. Solve the problem step by step. End your response with the final numeric
     answer on its own line in exactly this form: #### <number>
```

For the restoration gate, the user message is:

```
{underspecified_prompt}
Additional information: {removed_fact}
```

Both restoration judges must produce the gold answer under numeric exact match.

Host actors and ask detection. Each host actor receives a system message formed from its role description, the collaboration-specific reasoning instruction, the role's output-format instruction when present, and the following ask instruction. The user message contains the task followed by visible same-round and previous-round agent outputs.

```
If an outcome-relevant fact required for your assigned deliverable is missing, ask for only
     one primary missing fact in <ask>...</ask>. Do not ask about information visible in the
     task or handoff, optional behavior, invented requirements, or work outside your role.
     Otherwise produce your assigned deliverable without an ask block.
```

The GSM8K clarifier uses the following system instruction when assessing an actor output for a blocking issue:

```
You review grade-school math reasoning in a multi-agent system. Output ONLY JSON:
     needs_clarify (bool), clarify_question (str), type ("DG"|"RD"|"SC"|"CG"|""). Optional:
     issues, ask_to, ask_reason, confidence, fix_hint. Clarify ONLY when a reliable numeric
     solution cannot proceed. Error Type Codes:
DG Data Gap: A required quantity, rate, unit, condition, or relationship is missing.
RD Referential Drift: A pronoun, quantity, unit, or entity reference is ambiguous.
SC Signal Corruption: Arithmetic, equation, unit, interpretation, or logical reasoning is
     incorrect or contradictory.
CG Capability Gap: The requested mathematical deliverable is outside the current role.
No blocking issue: needs_clarify=false, clarify_question="", type="".
```

The clarifier user template is:

```
Roles snapshot:
{system_roles_desc}

Topology:
{topology_desc}

Current role:
{current_agent_role}
Next roles:
```

```
{next_agent_roles}

Role prompt / expected deliverable:
{role_prompt}

Agent output (math reasoning / answer):
{output_text}

Checklist:
1. Required quantity, rate, unit, or condition absent? (DG)
2. Ambiguous entity, quantity, or reference? (RD)
3. Arithmetic, equation, unit, or interpretation error? (SC)
4. Role or requested-format mismatch? (CG)
5. If not blocking a reliable solution, do NOT clarify.

Output rules:
- issues: ["DG: missing unit price", "SC: multiplication error"] etc.
- clarify_question: only if blocking; single closed-ended <=25 words.
- type: dominant blocking code else "".
Return JSON ONLY.
```

The final GSM8K answer is generated under the following system and user instructions, with the original task and all agent outputs inserted into the user message before its final instruction:

```
System: You are the final math decision-maker. Reconcile the candidate solutions, check the
     interpretation and arithmetic, and give one final answer. Do not invent missing facts.
     End the response with exactly one line in the form: #### <number>
User suffix: Solve the stated problem using the useful work above. Show concise reasoning and
     end with exactly one final line: #### <number>
```

Prompt-only baseline. The unmodified prompt-only baseline runs the frozen Qwen3-4B ARBITER with no parameter updates and no ASK-HUMAN score penalty. It uses the following system message. The calibrated prompt-only baseline uses the same model and messages; only its inference-time ASK-HUMAN score penalty differs.

```
Choose one available KAGURA route using only the actor-visible context. Return exactly one
     JSON object with exactly action, target, question, and feedback, in that key order. Never
      close the object after target; always emit all four keys. Use null question and feedback
      for continue/self_retry. For ask_human/ask_agent, emit a nonempty context-specific
     question and null feedback. For revise_self, emit null question and feedback exactly
     shaped as {"issues":[{"description":string,"evidence":[{"source_id":string,"quote":string
     }]}]}; every list must be nonempty and every string must be nonempty. Evidence must quote
      the actor-visible context and identify its source; do not copy illustrative payload text
     . Example output shapes follow. continue: {"action":"continue","target":null,"question":
     null,"feedback":null}. self_retry: {"action":"self_retry","target":"self","question":null
     ,"feedback":null}. revise_self: {"action":"revise_self","target":"self","question":null,"
     feedback":{"issues":[{"description":"Correct the identified reasoning error.","evidence
     ":[{"source_id":"actor_output","quote":"Quote the exact problematic statement."}]}]}}.
     ask_human: {"action":"ask_human","target":"human","question":"Which missing fact resolves
      the uncertainty?","feedback":null}. ask_agent: {"action":"ask_agent","target":"agent_0
     ","question":"Can you independently verify the uncertain step?","feedback":null}. Do not
     add prose or markdown.
```

The user message is a compact JSON serialization of the following template, expanded here for readability; keys are sorted and no extra whitespace is inserted in the actual message. The route list contains only feasible action–target pairs for that event. Braced fields below stand for their event-specific values, which may themselves be structured objects.

```
{
  "actor_visible_context": {
    "actor_id": {actor_id},
    "actor_output": {actor_output},
    "actor_role": {actor_role},
    "generation_phase": {generation_phase},
    "observed_task": {observed_task},
    "role_contract": {role_contract},
    "role_prompt": {role_prompt},
    "visible_current_round_handoffs": {visible_current_round_handoffs},
    "visible_prior_round_handoffs": {visible_prior_round_handoffs}
  },
  "available_routes": [
    {"action": {action}, "target": {target}}, ...
  ]
}
```

The same route-prefix mask and payload checks apply to prompt-only and learned policies (Appendix C).

B.3 PROXY HUMAN

The proxy human stands in for the user who posed the grade-school problem. It is called only after a policy selects ASK-HUMAN. It sees the task given to the system, at most one forgotten fact, and the agent's question. When that question asks for the forgotten fact, the proxy answers in one short sentence using only the fact. It does not solve the problem or introduce numbers the fact does not contain. When no fact was withheld, or the question is about anything else, it replies with exactly "I don't know." It does not say that it is a model or that a fact was withheld. Later questions in the same dialogue keep the earlier turns and are answered under the same rule.

Reported runs instantiate this proxy with Qwen3-4B. Its system prompt is:

```
Role: You are simulating the user who gave a grade-school math word problem to
a multi-agent AI system.

Sometimes you forget to mention a fact. An agent may ask you for clarification.
Your job is to answer only when the question is about that forgotten fact;
otherwise decline.

Task:
- If you were given a forgotten fact (not NULL), and the question is asking for
  that fact or information directly contained in it, reply in one short natural
  sentence using only that fact. Do not solve the full problem. Do not invent
  numbers or facts that are not in the forgotten fact.
- If the forgotten fact is NULL, or the question is about anything else, reply
  with exactly this sentence and nothing else:
I don't know.

Do not mention that you are an AI, a model, or a proxy. Do not mention
"forgotten fact" or "private fact" in your reply.
```

The first user message is:

```
The task you gave the system:
{posed_task}

Forgotten fact:
{removed_fact_or_NULL}

Question from the agent:
{actual_MAS_question}

Your reply:
```

Later questions in the same dialogue keep the chat history and send only:

```
Question from the agent:
{actual_MAS_question}

Your reply:
```

Fully specified tasks always set the forgotten fact to `NULL`. Underspecified tasks insert `removed_fact`, the withheld fact text, and not `removed_fact_answer`. That answer field is used only by a deterministic mock proxy, which is not the proxy in the reported runs.

B.4 CONTEXT EXTRACTION AND MATCHED ROUTE EXECUTION

For each source problem and actor model, a trained MasRouter checkpoint uses the fully specified problem and a source-derived routing seed to select the host team's agent count, roles, and collaboration pattern; the latter determines its topology. We save that team specification and reuse it for the paired underspecified problem and all route continuations. Source problems and actor models therefore yield heterogeneous hosts while both arms of a pair share the same host configuration. We run these hosts on both members of every pair and intercept explicit-ask and terminal-handoff events. At each event, we persist the state after the actor has produced its candidate but before that candidate is consumed. The saved state contains the task, visible collaboration history, actor role, candidate output, topology, and feasible-route manifest. It omits the gold answer, withheld fact, task-arm label, and all route outcomes.

From this common checkpoint, we execute every feasible route once: CONTINUE, SELF-RETRY, REVISE-SELF, one ASK-AGENT:v route per eligible peer, and ASK-HUMAN. Each branch resumes the host workflow and runs to termination, at which point we record numeric exact-match success and whether the route consumed human attention. Sharing the pre-intervention trajectory makes route outcomes matched: differences after the checkpoint are attributable to the selected intervention rather than to a different history leading to the event.

B.5 SPLIT MANIFEST AND TARGET DISTRIBUTION

All variants, intervention contexts, and counterfactual branches derived from one GSM8K source are assigned to the same split. This source-grouped rule is necessary because randomly splitting branches would expose nearly identical histories and source answers across training and test. The manifest records a stable source identifier, task arm, host-MAS configuration, actor family, event surface, feasible routes, branch outcomes, and split assignment for every context.

Table 2 expands the main-paper data summary. A unique target is the sole acceptable route for its context; a set-member target is one of several acceptable routes. Ask-agent targets remain distinct by resolver even though they are grouped into one action family in the table.

Table 2: Set-valued route-target membership by split. Parentheses give the fraction of all target memberships in that split.

	Train			Test		
Action family	Unique	Set member	Total	Unique	Set member	Total
ASK-HUMAN	487	0	487 (19.8%)	515	0	515 (10.2%)
ASK-AGENT	8	611	619 (25.1%)	29	1,697	1,726 (34.3%)
REVISE-SELF	12	439	451 (18.3%)	3	62	65 (1.3%)
SELF-RETRY	10	446	456 (18.5%)	39	1,320	1,359 (27.0%)
CONTINUE	7	443	450 (18.3%)	36	1,325	1,361 (27.1%)
All memberships	524	1,939	2,463	622	4,404	5,026

All 1,003 training contexts have at least one successful recorded route: 459 are fully specified and 544 underspecified. Actors raise explicit asks in 66 and 240 of these contexts, respectively. The test split contains 3,168 contexts: 1,442 fully specified and 1,726 underspecified, with 385 and 1,044 explicit asks. Of these test contexts, 1,978 have at least one successful recorded route and therefore contribute the 5,026 acceptable-route targets above. The remaining 1,190 have no successful route in the recorded feasible set. They have no positive SFT target, but remain in the denominators for policy task success and human-query use; dropping them would make policies that encounter difficult contexts appear artificially strong.

C DETAILED ARBITER INTERFACE AND LEARNING METHODS

The offline and online methods share one resolution-choice interface, executor, and intervention-level training unit. They differ in whether outcomes come from complete stored route execution or from fresh policy-sampled continuations, and in how those outcomes update the policy. We use *route* for the action–resolver representation of a feasible resolution choice and *payload* for any route-conditioned question or feedback.

C.1 SHARED DECISION AND EXECUTION INTERFACE

The policy emits one structured object with fields `action`, `target`, `question`, and `feedback`. Continue and self-retry require null payloads; query routes require a nonempty question; revise-self requires a nonempty list of issues supported by evidence. A token-level trie masks the `action`/`target` prefix to the event-specific feasible route set $\mathcal{R}(o)$. Payload tokens remain model-generated, and the executor separately rejects malformed objects, infeasible targets, or payloads that do not satisfy the selected route's schema. The structured envelope matters because route quality is meaningful only when the intervention can be executed unambiguously.

Execution has the same semantics for every policy. Continue commits the intercepted candidate. Self-retry restores the current actor's pre-generation request and resamples without feedback. Revise-self appends the candidate and ARBITER's grounded feedback before rerunning the actor. Ask-human and ask-agent send the generated question to the selected resolver, append its response to the actor's context, and rerun the actor. The resulting candidate then returns to the remaining host-MAS workflow. Resolver responses are revealed only after a query route is selected.

Both learning methods train from a recorded pre-intervention state, execute one resolution choice, and continue the workflow to obtain end-to-end task success and a human-query indicator. This intervention-level unit makes counterfactual training tractable. The offline method executes every feasible route during data generation and reuses those outcomes; the online method samples interventions from the current policy and obtains a fresh continuation for each sample.

C.2 EXACT OFFLINE CONTEXTUAL-BANDIT DETAILS

For each event e, routes are enumerated from the policy-visible manifest without consulting the gold answer, task arm, or route outcomes. Feasible routes are executed from the same checkpoint under a shared continuation randomness contract. Let $z_{e\rho} \in \{0, 1\}$ denote the resulting end-to-end task success and let $h_{e\rho} = \mathbf{1}\{\rho = \text{ASK-HUMAN}\}$. These are stored outcomes for the matched branches, rather than population success probabilities.

The empirical successful and acceptable route sets are

$$\widehat{\mathcal{S}}_e = \{\rho \in \mathcal{R}(o_e) : z_{e\rho} = 1\}, \qquad \widehat{\mathcal{A}}_e = \begin{cases} \arg\min_{\rho \in \widehat{\mathcal{S}}_e} h_{e\rho}, & \widehat{\mathcal{S}}_e \neq \varnothing, \\ \varnothing, & \widehat{\mathcal{S}}_e = \varnothing. \end{cases} \tag{5}$$

Thus every successful non-human route is acceptable whenever one exists; otherwise a successful human route is retained. Exact ties remain set-valued, which avoids introducing an arbitrary preference among routes with the same observed success and human-attention cost.

For supervised initialization and SFT baselines, canonical route-prefix scores are normalized over the feasible set,

$$q_\theta(\rho \mid o) = \frac{\exp g_\theta(\rho, o)}{\sum_{\rho' \in \mathcal{R}(o)} \exp g_\theta(\rho', o)}.$$

On events $\mathcal{E}_+$ with nonempty acceptable sets, we minimize

$$\mathcal{L}_{\text{route}}(\theta) = -\frac{1}{|\mathcal{E}_+|} \sum_{e \in \mathcal{E}_+} \log \sum_{\rho \in \widehat{\mathcal{A}}_e} q_\theta(\rho \mid o_e). \tag{6}$$

This objective rewards probability assigned anywhere in the acceptable set. A separate token-normalized negative-log-likelihood term trains questions and feedback on acceptable payload-bearing routes; null-payload routes contribute no payload loss.

The conservative SFT operating point is an early checkpoint after 25 optimizer updates: it makes 915 human queries versus 911 for prompt-only and retains all five action families. "Conservative" describes this measured behavior, not a separate query-constrained training loss. The maximum-utility SFT operating point comes from a separate supervised variant that adds a 0.1-weighted KL term toward uniform probability within each acceptable set; this term is zero for singleton sets. Among the tested SFT candidates, its checkpoint had the highest development score of task-success rate minus 0.05 times human-query rate. This scalar score is used for selection, not for the supervised parameter updates.

The exact offline objective in Eq. 2 instead uses all contexts and sums over their complete finite feasible-route sets. On an all-zero context, the success term vanishes but the human-cost, KL, and entropy terms remain active. Only canonical route-prefix tokens receive this objective; stored payload tokens are not optimized. The expectation is therefore exact over the frozen recorded route outcomes, but remains a route-level surrogate for the outcomes of newly generated payloads or changed downstream continuations. We train separate human-query cost profiles to expose the attainable success–attention frontier rather than committing to one arbitrary scalar exchange rate.

For comparison, the offline primal–dual variant minimizes empirical human queries subject to a baseline-relative success target. On the frozen subset with at least one successful route, its Lagrangian replaces the first two terms of Eq. 2 with

$$\widehat{J}_H(q_\theta) + \mu \left[\widehat{J}_S(q_0) - \delta - \widehat{J}_S(q_\theta) \right], \qquad \mu \geq 0,$$

and updates μ by projected dual ascent after each batch. Restricting this variant to contexts with a nonempty successful set prevents an all-zero route set from creating an infeasible per-context success target.

C.3 Online primal–dual details

Online GRPO samples complete route–payload sequences under the same feasible route-prefix mask used at inference. Every sampled intervention receives a fresh MAS continuation and the reward in Eq. 3. Group-relative advantages update the full generated sequence, with a frozen-reference KL penalty and entropy bonus. In contrast to the route-only offline objective, online learning can therefore assign credit to both route selection and newly generated content.

The online run is initialized from a separately trained offline query-constrained checkpoint, which provides a stable mixture of feasible routes before fresh exploration. Its success target is fixed before training from the initializer's comparison performance minus the one-point non-inferiority margin. The recorded online run samples from a separate pool of 24,387 intervention contexts. Offline training needs complete stored outcomes for every feasible route and thus uses the smaller 1,003-context labeled split. Online training executes only its sampled route from a saved state, so it can draw from the larger pool without first labeling every alternative route. The online pool and its holdout remain source-disjoint. The dual variable is updated only after every sample in a complete batch has been scored, following Eq. 4. A malformed output executes no route and receives $S = 0$ and $H = 0$; this prevents parser failure from being mistaken for successful avoidance of human attention.

C.4 Policy selection and calibration details

At inference, a nonnegative calibration penalty κ may be subtracted only from the ASK-HUMAN route score,

$$g_{\theta,\kappa}(\rho, o) = g_\theta(\rho, o) - \kappa \mathbf{1}\{\rho = \text{ASK-HUMAN}\}.$$

Scores are then renormalized over the event's feasible routes. We sweep this penalty for the prompt-only and exact offline policies to distinguish learning from query reduction obtainable through inference-time calibration alone. The reported offline point uses penalty 3.25; the calibrated prompt-only point uses penalty 1.5; online step 15 uses no inference-time ASK-HUMAN score penalty. Appendix C.5 lists the hyperparameters of all six Table 1 policies. Candidate checkpoints, prompts, structured envelopes, decoding settings, and penalty values are fixed before the paired comparison. Within the offline penalty sweep, we first apply the one-point success gate and then choose the passing penalty with the fewest human queries. The zero-margin rule is reported separately and retains the prompt-only policy when no learned candidate passes.

C.5 Hyperparameters of the compared policies

Table 3 gives the training and inference settings for the six policies in Table 1. Prompt-only rows leave Qwen3-4B unchanged. Supervised and online rows update that model with LoRA on the attention projections; host actors are not updated.

The online row is initialized from a separately trained offline query-constrained checkpoint, as stated in Appendix C.3.

C.6 Policy inference

The decisions in Table 1 are generated from Qwen3-4B with vLLM 0.23.0. The server uses tensor parallelism 2, GPU memory utilization 0.75, at most 32 concurrent sequences, model length 10,531, prefix caching disabled, and seed 0. Each request uses temperature 0, seed 0, and at most 256

Table 3: Hyperparameters for the policies in Table 1. κ is the inference-time ASK-HUMAN score penalty. Both supervised rows use LoRA rank 16 and $\alpha = 32$. Maximum-utility SFT also uses learning rate 10^{-5}, context batch size 2, and seed 0.

Policy	Update	Training settings	κ
Prompt-only	None	Frozen Qwen3-4B	0
Calibrated prompt-only	None	Same frozen model	1.5
Conservative SFT	LoRA	Preferred-set KL weight 0; learning rate 10^{-6}; step 25	0
Maximum-utility SFT	LoRA	Preferred-set KL weight 0.1; 502 steps	0
Online primal–dual, step 15	LoRA	LoRA rank 16, $\alpha = 32$; learning rate 10^{-6}; KL coefficient 0.02; entropy coefficient 0.002; batch 16; mini-batch 8; 16 rollouts per context at temperature 4; dual step size 2, initial value 2, cap 10, success target 0.5317; seed 1756; checkpoint step 15	0
Exact offline contextual bandit	Trained	Learning rate 3×10^{-5}; training human-cost weight 0.5; checkpoint step 10	3.25

generated tokens. A logits processor restricts the route prefix to that event's feasible routes. Learned checkpoints are loaded as LoRA adapters. Requests run at concurrency 16, with a 300-second timeout and one retry.

Stored route outcomes come from separate vLLM servers for the host actors. Each uses tensor parallelism 2, maximum model length 32,768, and GPU memory utilization 0.90. The Nemotron server disables prefix caching, and the Devstral server is restricted to its language-model weights. Table 1 does not call these servers again; it replays the stored route outcome.

C.7 HARDWARE

The runs used a cluster of four nodes, each with eight 40GB A100 GPUs. Each node served one model and replicas of that model. With these servers running in parallel and replicas available, data generation took about three days and online training took about one week.

D ADDITIONAL RESULTS AND ANALYSIS

D.1 POLICY OPERATING POINTS

Table 4 reports the named operating points on the common 3,168-context GSM8K comparison using stored route outcomes. Success counts include all contexts, including the 1,190 without a successful recorded route; human-query counts also use all 3,168 contexts. "Base only" means that a learned policy passes the success and positive query-reduction gates against the unmodified prompt-only baseline but not the incremental comparison against the calibrated baseline. "Both" denotes passage of both comparisons.

Relative to the unmodified baseline, the calibrated exact offline policy changes source-macro success by -0.510 percentage points (95% CI $[-0.912, -0.135]$) and reduces the source-macro query rate by 1.918 points ($[1.237, 2.611]$). Its 828 queries are 83 fewer than the baseline's 911, a 9.1% relative reduction. Relative to the calibrated prompt-only policy, its source-macro success difference is -0.072 points ($[-0.346, 0.220]$) and its query-rate reduction is 0.557 points ($[0.079, 1.048]$), corresponding to 27 additional queries saved. The online step-15 checkpoint is the only reported online checkpoint to pass the calibrated comparison: its success difference is -0.045 points ($[-0.411, 0.309]$) and query-rate reduction is 0.657 points ($[0.055, 1.282]$) relative to that comparator. The online point uses 845 queries versus 828 for the calibrated offline point; only the latter uses an inference-time ASK-HUMAN score penalty.

Table 4: Named policy operating points on the common test cohort. Coverage counts how many of the five label-supported action families appear in valid outputs.

Policy	Task successes	Human queries	Validity	Route coverage	Decision
Prompt-only baseline	1,715	911	100%	5/5	Reference
Calibrated prompt-only	1,700	855	100%	5/5	Base comparator
Conservative SFT	1,715	915	100%	5/5	Fails query gate
Maximum-utility SFT	1,853	2,170	100%	4/5	Fails query gate
Exact offline RL, maximum utility	1,792	1,206	100%	5/5	Fails query gate
Exact offline RL, penalty 3.25	1,694	828	100%	5/5	Both
Exact offline RL, lower-query boundary	1,685	809	100%	5/5	Fails success gate
Offline primal–dual, empirical Pareto	1,725	909	100%	5/5	Not selected
Offline primal–dual, closest joint point	1,704	859	100%	5/5	Not selected
Online primal–dual, step 8	1,709	880	100%	5/5	Base only
Online primal–dual, step 9	1,705	869	100%	5/5	Base only
Online primal–dual, step 10	1,706	875	100%	5/5	Base only
Online primal–dual, step 11	1,703	858	100%	5/5	Base only
Online primal–dual, step 12	1,700	854	100%	5/5	Base only
Online primal–dual, step 13	1,706	866	100%	5/5	Base only
Online primal–dual, step 14	1,700	859	100%	5/5	Base only
Online primal–dual, step 15	1,698	845	100%	5/5	Both
Online primal–dual, step 16	1,698	851	100%	5/5	Base only

D.2 HUMAN-ATTENTION ALLOCATION

An event is *system-resolvable* when at least one feasible non-human route achieves end-to-end task success. A human query at such an event is *avoidable*: the selected policy consumed human attention even though CONTINUE, SELF-RETRY, REVISE-SELF, or an eligible ASK-AGENT route succeeded from the same state. The avoidable-query fraction divides these queries by all human queries made by the policy. Under this definition, 195 of 911 prompt-only human queries (21.4%) are avoidable. The calibrated exact offline policy makes 178 avoidable queries among 828 human queries (21.5%): 17 fewer avoidable queries, an 8.7% relative reduction in their count. The fraction of human queries that are avoidable is essentially unchanged.

Task arm and empirical resolvability answer different questions. Fully specified versus underspecified is a property of how the task was constructed; system-resolvable versus not system-resolvable is determined by observed route outcomes at an intervention context. The task arms test whether queries are allocated in the intended direction, while the matched route definition measures whether a particular query could have been replaced by a successful internal resolution.

D.3 STATISTICAL PROCEDURE

The GSM8K source problem is the statistical unit. Policies are paired on the same contexts, and task variants and counterfactual branches are aggregated within their originating source before computing source-macro differences. We obtain 95% confidence intervals by resampling source clusters, thereby preserving dependence among branches that share a problem and pre-intervention history. For a margin $\delta = 0.01$, a policy passes non-inferiority only when the lower bound of its paired success-difference interval exceeds -1 percentage point. A query-reduction claim additionally requires the lower bound of the paired reduction interval to exceed zero. Repeating the decision with $\delta = 0$ yields the strict sensitivity result reported in Section 7.

Overall success under stored-route replay and selected human-query rates use all 3,168 contexts. Metrics that require an acceptable target, such as acceptable-set route accuracy, use only the 1,978 target-bearing contexts and are labeled conditional diagnostics. Structured-output validity uses every generated decision, while route coverage reports whether each supported action family appears at least once; coverage is not an accuracy measure.