CLARIFY THE USER OR VERIFY THE WORLD? UNCERTAINTY ROUTING FOR PROACTIVE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using LLM agents must decide not only whether additional information is needed, but also which source can resolve the uncertainty. Existing proactive approaches often specialize in either user clarification or environment verification, without explicitly determining the appropriate information source for each decision. We formulate this problem as *uncertainty routing* among ACT, CLARIFY, and VERIFY, and propose PROUR, a proactive uncertainty routing framework. PROUR decomposes action uncertainty into disagreement across plausible user-goal interpretations, which signals user-side ambiguity, and the entropy remaining within each interpretation, which signals missing world-side evidence. To acquire information from the routed source, a query generator is trained with a mode-conditioned information-gain reward, targeting user-goal identification under CLARIFY and next-action identification under VERIFY. On τ-bench, PROUR achieves 28.17% average success rate across retail and airline, outperforming the strongest prior method by 4.57% while using 2.17 fewer interaction steps. The learned policy further generalizes to stronger task agents and transactional domains of τ^3-bench without retraining, demonstrating the benefit of source-aligned uncertainty resolution for proactive agents.[1]

1 INTRODUCTION

Tool-using LLM agents increasingly interact with users, query databases, invoke APIs, and modify external state (Yao et al., 2023; Schick et al., 2023; Yao et al., 2025). To ensure reliability, an agent must recognize when the available information is insufficient and proactively acquire missing information before acting (Zhang et al., 2024). The challenge, however, is not merely deciding *whether* to seek more information. A proactive agent must also decide *where* that information should come from. Should it ask the **user**, or should it inspect the external **world**?

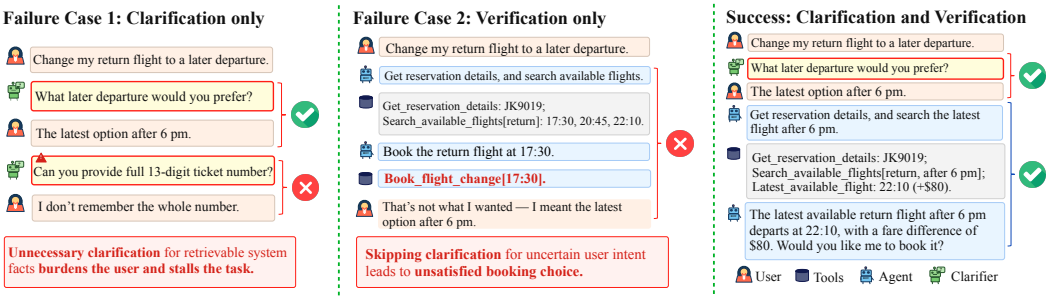


Figure 1: An example of clarification and verification in τ-airline trajectories. Clarification alone burdens the user with facts available from tools (left), while verification alone misses user preferences (middle). Combining them resolves the preference before retrieving system evidence (right).

This distinction is critical because these two interventions resolve fundamentally different unknowns, as illustrated by the request in Figure 1. In **Failure Case 1**, clarification effectively recovers the user's preference but then asks the user for a ticket number retrievable from tools, burdening

[1]Code is available at `https://anonymous.4open.science/r/Prour-FBC3/`.

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.

the user and stalling the task. In **Failure Case 2**, verification retrieves the reservation and available flights but does not clarify what *later* means, leading to a wrong booking. Only the **Success case** obtains information from the appropriate source: it asks the user to resolve the preference and queries the environment to verify eligibility and availability. These cases expose two distinct information sources: the user side and the world side. Only the user can clarify an unstated preference, whereas world facts such as system state, policy constraints, and action validity should be verified from tools and other grounded evidence. Our analysis of 19,022 step-level events from τ-bench (Yao et al., 2025) further reveals a strong asymmetry in where missing information resides (Figure 2): 89.1% concern world-side information, whereas only 10.9% concern user-side information. Thus, treating user clarification as the sole mechanism for resolving uncertainty (Deng et al., 2026; Suri et al., 2026) can be inefficient and often directs the agent to the wrong information source. A proactive agent must determine not only whether it is uncertain, but also whether that uncertainty should be resolved by the user or by the world. We refer to this problem as *uncertainty routing*.

To address this problem, we propose a **Pro**active **U**ncertainty **R**outing framework, namely **PROUR**, which explicitly identifies the source of uncertainty before acquiring additional information. PROUR decomposes uncertainty over the agent's next action into two source-aligned components, inspired by the distinction between aleatoric and epistemic uncertainty (Hou et al., 2024; Ling et al., 2024). Specifically, if different plausible interpretations of the user's goal imply different actions, the agent faces high **aleatoric uncertainty (AU)**: the goal itself is underspecified, so the missing information must come from the user. In contrast, if the appropriate action remains uncertain even after ambiguity in the user's goal is clarified, the agent faces high **epistemic uncertainty (EU)**: it lacks sufficient knowledge about the external state or constraints and should therefore seek additional evidence. This yields a natural three-way routing policy: CLARIFY when AU is high, VERIFY when the goal is clear but EU remains high, and ACT when neither source of uncertainty warrants further information acquisition.

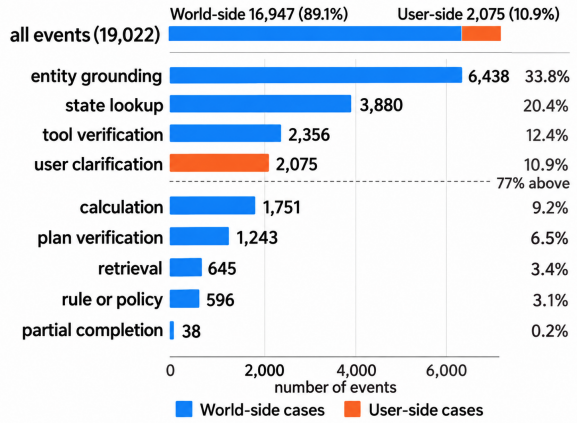
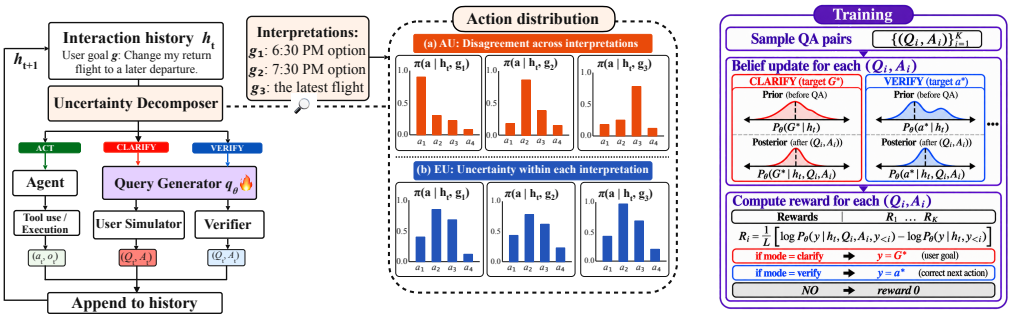


Figure 2: Source distribution of 19,022 step-level events from τ-bench. World-side cases (blue) account for 89.1%; user-side cases (orange) account for 10.9%. See Appendix A.1 for the full taxonomy.

Routing, however, only determines *where* to look; the agent must still decide *what* to ask. PROUR therefore couples uncertainty routing with a mode-conditioned **Query Generator**. Under CLARIFY mode, the generator issues user-directed queries that make the underlying goal more identifiable; under VERIFY mode, it queries a tool-grounded verifier for evidence that makes the reference next action more identifiable. We train this generator using Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) (Yu et al., 2025) with a **mode-conditioned information-gain reward**. The reward measures how much a query–response pair increases the likelihood of the ground-truth user goal under CLARIFY mode, or of the reference next action under VERIFY mode. Thus, rather than maintaining separate mechanisms for clarifying users and verifying the world, a single policy learns to generate queries whose target is aligned with the source of uncertainty.

Our main contributions are three-fold:

1. We identify uncertainty routing as a fundamental problem for tool-using agents, and propose PROUR, a proactive uncertainty routing framework that decomposes action uncertainty into user-side and world-side uncertainty to route the agent among ACT, CLARIFY, and VERIFY.
2. We propose a mode-conditioned information-gain objective that aligns both the information source and reward target with the routed uncertainty, and use it to train a single query generator to acquire information from either users or tool-grounded evidence.
3. Extensive experiments demonstrate the effectiveness, efficiency, and generalization of PROUR. On τ-bench, it improves average success rate by 4.57% with 2.17 fewer interaction steps. Without further retraining, it achieves 29.00% average success rate on τ^3-bench and generalizes across stronger task agents.



(a) Uncertainty Decomposition and Routing

(b) Mode-Conditioned Information-Gain Training

Figure 3: **Overview of the PROUR framework.** (a) Decomposer estimates action-level aleatoric and epistemic uncertainty from candidate user-goal interpretations and routes to CLARIFY, VERIFY, or ACT. (b) Query Generator is trained with a mode-conditioned information-gain reward, using the ground-truth user goal as the clarification target and reference next action as the verification target.

2 RELATED WORK

Uncertainty Decomposition for LLM Agents. For an interactive agent, uncertainty is actionable only if it identifies both the missing information and its source. Classical work separates aleatoric and epistemic uncertainty (Kendall & Gal, 2017), and Bayesian Active Learning by Disagreement (BALD) measures parameter-induced disagreement (Houlsby et al., 2011). LLM studies use calibrated confidence, semantic consistency, or input-clarification ensembles (Jiang et al., 2021; Kadavath et al., 2022; Kuhn et al., 2023; Hou et al., 2024). Unlike classical BALD, which ensembles over model parameters and associates ensemble disagreement with epistemic uncertainty, we ensemble over candidate goal interpretations. Following the input-clarification view of Hou et al. (2024), disagreement across interpretations captures user-side ambiguity, while within-interpretation entropy captures residual world-side uncertainty. We therefore use AU and EU as operational source labels rather than universal Bayesian identities. Prior decompositions diagnose uncertainty but do not route each decision among acting, querying the user, and consulting the world.

Proactive Clarification and Verification. Prior work treats clarification as proactive information acquisition, using expected value or predictive uncertainty to decide when to ask (Rao & Daumé III, 2018; Naszadi et al., 2023), using agents to clarify before planning (Deng et al., 2023; Zhang et al., 2024), or modeling latent intents and parameter uncertainty (Zhang & Choi, 2025; Kobalczyk et al., 2025; Suri et al., 2026). Learning-based methods optimize long-horizon collaboration, belief gain toward user goals or search answers (Wu et al., 2025; Deng et al., 2026; Wang et al., 2026). Complementary work uses tool-grounded critique or verification (Gou et al., 2024; Han et al., 2025) or requires user confirmation (Cuadron et al., 2026). These approaches typically assume a predefined information source. PROUR instead first routes uncertainty to its appropriate source, then uses a shared policy whose answer source and reward target adapt to CLARIFY or VERIFY.

3 METHODOLOGY

When an agent is uncertain about its next action, it must answer two questions: *where does the missing information reside*, and *what information should be acquired from that source*? As illustrated in Figure 3, the PROUR framework addresses these questions with two components. First, a **Decomposer** separates uncertainty induced by ambiguity in the user's goal from uncertainty that remains after ambiguity is clarified, and accordingly routes the agent among CLARIFY, VERIFY, and ACT. Second, when intervention is required, a shared **Query Generator** acquires information from the routed source: the user under CLARIFY, or grounded world evidence under VERIFY.

3.1 PROBLEM FORMULATION

We study uncertainty resolution in tool-using agents (Yao et al., 2023; Schick et al., 2023; Yao et al., 2025), where an agent interacts with a user and invokes domain tools. At each decision step, the

agent may already have sufficient information to act, or its next action may depend on information unavailable in the current history. Importantly, such information can reside in two different sources: *user-side* information, such as an underspecified preference or intent, and *world-side* information, such as environment state, entity attributes, policy constraints, or action validity. The agent must therefore choose among three modes: ACT using the available information, CLARIFY by acquiring information from the user, or VERIFY by acquiring grounded evidence about the world. We represent an interaction trajectory τ and the history available before decision step t as

$$\tau = \left(U, (a_i, o_i, Q_i, A_i)_{i=1}^{T}, G\right), \qquad h_t = \left(U, (a_i, o_i, Q_i, A_i)_{i=1}^{t-1}\right). \tag{1}$$

Here, U is the initial user message and G is the latent user goal hidden from the agent during interaction. (Q_i, A_i) is a query–response pair, and (a_i, o_i) is a tool interaction. At step i, the decomposer selects a mode $m_i \in \{\texttt{act}, \texttt{clarify}, \texttt{verify}\}$. Under `act` mode, the agent selects $a_i \in \mathcal{A} = \mathcal{A}_{\text{tool}} \cup \{\texttt{respond}\}$ where a tool action produces observation o_i with $Q_i = A_i = \varnothing$. Under either intervention mode, the query generator issues a query Q_i and receives a response A_i by either clarifying the user or verifying the world, with $a_i = o_i = \varnothing$. ACT may itself include ordinary tool calls when the appropriate next action is sufficiently determined from the current history, whereas VERIFY is used when missing world-side evidence leaves the appropriate next action uncertain. Policy-mandated user confirmation is a `respond` action issued under `act` mode.

3.2 Uncertainty Decomposition and Routing

As illustrated in Figure 3a, at each interaction step, the decomposer determines whether uncertainty about the next action originates from the user's latent goal or from insufficient knowledge of the world. We adapt the distinction between aleatoric and epistemic uncertainty to the agent's *action space*. Specifically, **Aleatoric Uncertainty (AU)** captures disagreement in the predicted next-action distributions across plausible interpretations of the user's goal; high AU therefore indicates user-side ambiguity that should be resolved through CLARIFY. In contrast, **Epistemic Uncertainty (EU)** captures the residual uncertainty over the next action that remains under a fixed goal interpretation; high EU indicates missing world-side evidence—such as environment state, entity grounding, policy constraints, calculation results, or execution status—that should be resolved through VERIFY.

Goal-Conditioned Action Distribution. At decision t, the decomposer is invoked K times, and each call independently proposes one interpretation g_k of the user's latent goal conditioned on the observed history h_t, without access to the other $K-1$ samples. Notably, these candidate goal interpretations are hypotheses inferred from the observed history h_t, instead of perturbations or samples of the ground-truth goal G, which is not provided to the decomposer or agent. As detailed in Appendix A.2.2, the decomposer then samples structured candidate actions a containing a tool name and its arguments, and evaluates action distribution under each interpretation g_k:

$$p_{t,k}(a) = p_\phi(a \mid h_t, g_k), \quad \bar{p}_t(a) = \frac{1}{K}\sum\nolimits_{k=1}^{K} p_{t,k}(a). \tag{2}$$

where $\bar{p}_t$ is the empirical mixture across sampled interpretations.

Information-Theoretic Decomposition. Let $\widetilde{G}_t$ denote the empirical random variable assigning equal probability to each sampled candidate interpretations $\{g_k\}_{k=1}^{K}$, and let $a_t \in \mathcal{A}$ denote the predicted next-action variable. By conditional mutual information (Cover & Thomas, 2006):

$$H(a_t \mid h_t) = I(a_t; \widetilde{G}_t \mid h_t) + H(a_t \mid \widetilde{G}_t, h_t). \tag{3}$$

where $I(\cdot;\cdot)$ denotes mutual information, and $H(p) = -\sum_{a\in\mathcal{A}} p(a)\log p(a)$ denotes Shannon entropy. Based on the empirical sampled mixture in Eq. 2, we estimate aleatoric uncertainty (AU) and epistemic uncertainty (EU) as follows:

$$\mathrm{AU}_t := I(a_t; \widetilde{G}_t \mid h_t) = H(\bar{p}_t) - \frac{1}{K}\sum\nolimits_{k=1}^{K} H(p_{t,k}) = \frac{1}{K}\sum\nolimits_{k=1}^{K} D_{\mathrm{KL}}(p_{t,k} \,\|\, \bar{p}_t)\,, \tag{4}$$

$$\mathrm{EU}_t := \frac{1}{K}\sum\nolimits_{k=1}^{K} H(p_{t,k}). \tag{5}$$

We use the empirical mutual-information in Eq. 4 as AU, because it measures disagreement across plausible candidate interpretations. Mutual information equals the average KL divergence between

each interpretation-conditioned action distribution and their empirical mixture, so it is zero when all $p_{t,k}$ are identical, and increases as the action distributions diverge from their average $\bar{p}_t$. Thus, under our formulation, high AU means that different plausible interpretations of the latent goal imply different next actions, indicating user-side ambiguity that should be resolved by clarification.

Similarly, we use the empirical conditional-entropy in Eq. 5 as EU. This is the average entropy of the action distribution after conditioning on an interpretation. It is therefore high when the agent remains uncertain over the next action even after user-side ambiguity has been removed, indicating missing world-side evidence that should be resolved by verification. Consequently, $H(\bar{p}_t) = \mathrm{AU}_t + \mathrm{EU}_t$.

Routing Policy. We convert the decomposition into a routing decision, as illustrated in Figure 3a:

$$m_t = \begin{cases} \texttt{clarify}, & \text{if } \mathrm{AU}_t > \tau_{\mathrm{au}}, \\ \texttt{verify}, & \text{else if } \mathrm{AU}_t \le \tau_{\mathrm{au}} \text{ and } \mathrm{EU}_t > \tau_{\mathrm{eu}}, \\ \texttt{act}, & \text{otherwise.} \end{cases} \tag{6}$$

The ordering is intentional. When substantial user-side ambiguity remains, the relevant world-side query may itself depend on which goal interpretation is correct. The system therefore resolves user-side uncertainty first, while verification is triggered only when the goal is sufficiently determined but uncertainty about the next action remains. When neither uncertainty exceeds its calibrated threshold, the agent proceeds directly without unnecessary intervention. τ_{au} and τ_{eu} are tuned on a development set disjoint from training and evaluation.

3.3 Mode-conditioned Information-Gain Reward

The decomposer determines *where* missing information should be acquired, but not whether the generated query is effective at resolving the underlying uncertainty. Building on the formulation of information gain in Deng et al. (2026), we introduce an information-gain reward to train a shared query generator for both clarification and verification, rather than learning separate policies for each mode. Figure 3b summarizes the overall training procedure. The routing mode determines both the information source and the target used to compute the reward: clarification queries the user simulator to reduce uncertainty about the ground-truth user goal, whereas verification queries a verifier to reduce uncertainty about the reference next action.

Mode-Conditioned Information Acquisition. The query generator samples

$$Q_t \sim q_\theta(\cdot \mid h_t, m_t). \tag{7}$$

where mode $m_t \in \{\texttt{clarify}, \texttt{verify}\}$. Clarification queries aim at making the ground-truth user goal more identifiable, while verification queries aim at eliciting evidence to identify the reference next action. Therefore, response source A_t and reward target Y_t are conditioned on the mode:

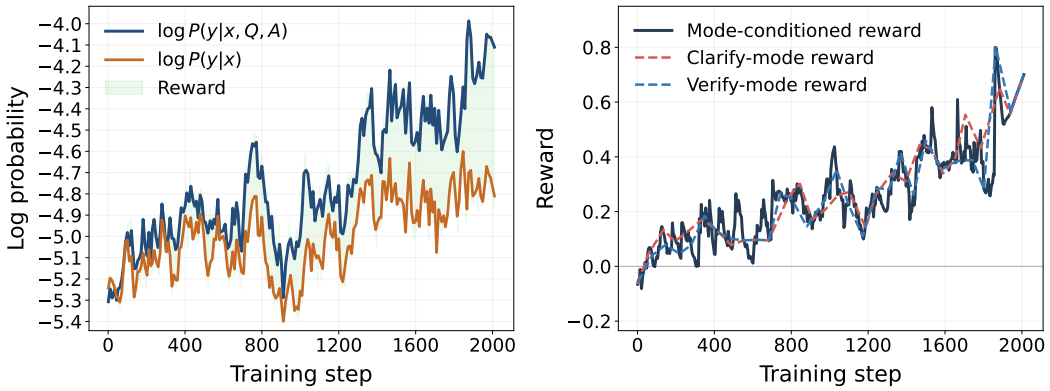
$$(A_t, Y_t) = \begin{cases} \big(\mathcal{S}_{\mathrm{user}}(Q_t \mid h_t, G^*),\ G^*\big), & m_t = \texttt{clarify}, \\ \big(\mathcal{V}(Q_t \mid h_t, \mathcal{W}, \mathcal{T}_{\mathrm{tool}}),\ a_t^*\big), & m_t = \texttt{verify}. \end{cases} \tag{8}$$

Here, G^* is the normalized ground-truth user goal and a_t^* is the next reference tool call in the τ-bench environment. This formulation does not assume that a_t^* is a universal or uniquely valid action; rather, it provides a consistent supervision target available from the reference trajectory. Responses A_t come from the user simulator $\mathcal{S}_{\mathrm{user}}$ and verifier $\mathcal{V}$. The verifier may retrieve environment state, check domain policy $\mathcal{W}$ and arguments of read-only tools $\mathcal{T}_{\mathrm{tool}}$, or assess whether the requested subgoals have been completed. Since the verifier cannot use tools that modify the environment, verification does not change the task state. The completed query–response pair is added to history.

Information-Gain Reward. We define the mode-conditioned objective so an optimal query maximizes the *Expected Information Gain (EIG)* (Deng et al., 2026):

$$\begin{aligned} Q_t^* &= \arg\max_Q I(Y_t; A_t \mid Q, h_t, m_t) \\ &= \arg\max_Q \mathbb{E}_{A_t \sim P_{m_t}(\cdot \mid Q, h_t)} \left[H(Y_t \mid h_t, m_t) - H(Y_t \mid h_t, m_t, Q, A_t) \right], \end{aligned} \tag{9}$$

where P_{m_t} is the response distribution for mode m_t. Exact EIG is intractable for open-ended queries and responses, so we adopt the teacher-forced belief-update approximation of Deng et al. (2026) and apply it to the mode-conditioned target. Let $\ell_\theta(Y \mid c)$ denote the length-normalized teacher-forced



(a) Prior and posterior log-likelihoods of the mode-conditioned target y.

(b) Mean information-gain reward over all samples and separate rewards over CLARIFY- and VERIFY-mode.

Figure 4: **Training dynamics under mode-conditioned information-gain reward.**

log-likelihood of target Y under context c. The information-gain reward is the shift from prior belief to posterior belief after incorporating the query–response pair:

$$R_t = \ell_\theta(Y_t \mid h_t, m_t, Q_t, A_t) - \ell_\theta(Y_t \mid h_t, m_t). \tag{10}$$

A positive reward indicates that the query-response pair (Q_t, A_t) increases the likelihood of the correct target, whereas a negative reward indicates that it makes the target less identifiable. If no query is issued, we set $R_t = 0$.

On-Policy Training. For each state-mode pair (h_t, m_t), the rollout policy samples a group of candidate queries. Clarification candidates receive responses from a strict user simulator conditioned on G^*, whereas verification candidates receive responses from the verifier; both responder LLMs are frozen. The strict user simulator reveals task-specific information only for precise, relevant queries, while the verifier only returns evidence supported by its grounded context. Each query–response pair receives the reward in Eq. 10, and the query generator is optimized via Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) (Yu et al., 2025). All responder LLMs, the agent, and decomposer remain frozen; only the query generator is optimized. The belief update, rollout estimator, advantage computation, and DAPO objective are given in Appendix A.3.

4 EXPERIMENTS

We evaluate PROUR in the τ-bench environment (Yao et al., 2025). Following previous studies (Deng et al., 2026), we train exclusively on 500 τ-retail trajectories and evaluate on 115 held-out retail tasks and 50 airline tasks. Airline serves as the out-of-distribution domain and is excluded from both training and calibration. We additionally evaluate cross-benchmark generalization on the transactional text domains of τ^3-bench (Barres et al., 2026)—retail, airline, and telecom, without retraining. Qwen3-1.7B serves as the query generator and is the only optimized module. By default, Qwen3-8B implements all frozen modules: the decomposer, agent, user simulator, and verifier. We compare against IG-clarifier (Deng et al., 2026), Ask-before-Plan (Zhang et al., 2024), VerifiAgent (Han et al., 2025), SABER (Cuadron et al., 2026), SAGE-Agent (Suri et al., 2026), and controlled ablations, reporting pass rates and average steps. Appendix A.2 provides the implementation details and hyperparameters. Appendix A.4 provides the prompt templates.

4.1 TRAINING DYNAMICS AND ROUTING CALIBRATION

Training Dynamics. To characterize how the query generator evolves under the mode-conditioned reward, we track the likelihood and reward throughout training. Figure 4(a) compares the prior belief $\ell_\theta(Y_t \mid h_t, m_t)$ with the posterior belief $\ell_\theta(Y_t \mid h_t, m_t, Q_t, A_t)$. Both curves exhibit mild oscillations because of variation in sampled contexts and mode-conditioned targets. Nevertheless, ℓ_{post} becomes consistently higher than ℓ_{prior}, and their gap generally grows over training. This suggests that the learned policy provides useful evidence for identifying the mode-conditioned target, rather than merely raising the model's prior confidence before a query is generated.

Table 1: Main results on τ-bench. We report Pass@1 (%) and average interaction steps. *Avg. Clarify* and *Avg. Verify* count routed CLARIFY and VERIFY steps issued as separate turns; methods that interleave them with agent turns are counted under *Avg. Act*. *LLM Router*, *Clarify-only*, and *Verify-only* are controlled ablations.

Method	Success Rate (% ↑)			Steps ↓			
Strategy	**Retail**	**Airline**	**Average**	**Avg. ACT**	**Avg. CLARIFY**	**Avg. VERIFY**	**Total**
Base Agent	16.80	18.00	17.40	28.90	–	–	28.90
Ask-before-Plan (Zhang et al., 2024)	23.19	24.00	23.60	21.10	2.40	–	23.50
VerifiAgent (Han et al., 2025)	19.13	26.00	22.57	31.81	–	11.20	43.01
SABER (Cuadron et al., 2026)	21.74	20.00	20.87	34.43	–	–	34.43
SAGE-Agent (Suri et al., 2026)	18.26	24.00	21.13	32.65	1.89	–	34.54
IG-clarifier (Deng et al., 2026)	21.70	24.00	22.85	20.85	2.70	–	23.55
LLM Router	23.50	28.00	25.75	16.59	3.30	3.50	23.39
Clarify-only	24.30	26.00	25.15	21.17	2.03	–	23.20
Verify-only	15.70	**30.00**	22.85	22.22	–	8.48	30.70
PROUR	**27.00**	29.33	**28.17**	14.34	2.20	4.79	**21.33**

Table 2: Effect of the query generator on τ-bench. All alternative generators are prompt-only; the decomposer, task agent, and routing thresholds are held fixed.

Method	Success Rate (% ↑)			Steps ↓			
Query Generator	**Retail**	**Airline**	**Average**	**Avg. ACT**	**Avg. CLARIFY**	**Avg. VERIFY**	**Total**
None (Base Agent)	16.80	18.00	17.40	28.90	–	–	28.90
Qwen3-1.7B	24.35	20.67	22.51	18.51	3.86	5.11	27.48
Qwen3-8B	26.09	22.00	24.05	12.45	2.37	5.62	**20.44**
Qwen3-32B	28.70	24.00	26.35	17.75	1.58	7.22	26.55
DeepSeek-V4-Flash	**29.60**	28.00	**28.80**	13.17	2.42	5.94	21.53
PROUR (Qwen3-1.7B-trained)	27.00	**29.33**	28.17	14.34	2.20	4.79	21.33

Consistent with this interpretation, Figure 4(b) reports the corresponding reward $R_t = \ell_{\text{post}} - \ell_{\text{prior}}$ by showing both the aggregate reward across all samples and separated into CLARIFY and VERIFY mode. The mean reward rises from near zero to approximately 0.7 after 2,010 optimization steps, and the CLARIFY- and VERIFY-mode rewards improve alongside the aggregate reward without evidence of mode collapse. These dynamics show that a shared query generator can optimize two mode-conditioned reward targets while maintaining a stable learning signal.

Routing Calibration. Because AU and EU operate on different empirical scales, we calibrate their routing thresholds on a disjoint τ-retail development set of 240 decision steps, where each step is labeled as AU, EU, or NO_UNCERTAINTY. The decomposer then samples 5 goal interpretations for each step to estimate AU and EU, and a grid-search procedure selects the thresholds that maximize three-class macro-F1, yielding $(\tau_{\text{AU}}, \tau_{\text{EU}}) = (0.25, 0.20)$. We retain these thresholds for all main experiments, including Section 4.3. Appendix A.2.2 details the calibration protocol, threshold search, the confusion matrix, and a separate recalibration study on GLM-4-32B.

4.2 Main Results and Ablation Studies

End-to-end Performance. Table 1 shows that PROUR achieves the highest average success rate (28.17%) while requiring the fewest total interaction steps (21.33), compared with 17.40% success and 28.90 steps for the base agent. Ask-before-Plan, SAGE-Agent, and IG-clarifier seek information from the user but do not acquire grounded world-side evidence, attaining average success rates of 23.60%, 21.13%, and 22.85%, respectively. VerifiAgent instead performs meta- and tool-based verification without explicitly routing user-side uncertainty; despite its strong airline result, it requires 43.01 steps on average. SABER requests user consent on mutating actions and achieves 20.87% success on average. These results show that resolving both user-side and world-side uncertainty through targeted routing is more effective than relying exclusively on clarification or verification.

To isolate the contribution of routing and information source, the LLM Router replaces uncertainty decomposition with a prompt-only decision, while the Clarify-only and Verify-only variants disable one intervention branch. Each variant underperforms the full method on average. In partic-

Table 3: Cross-agent generalization performance on τ-bench. Different agent backbones are compared under varying query generators. The best results are bolded.

Setting		Success Rate (% ↑)			Steps ↓			
Agent Model	**Query Generator**	**Retail**	**Airline**	**Average**	**Avg. ACT**	**Avg. CLARIFY**	**Avg. VERIFY**	**Total**
Qwen3-8B	None	16.80	18.00	17.40	28.90	–	–	28.90
	Qwen3-8B	26.09	22.00	24.05	12.45	2.37	5.62	**20.44**
	PROUR	**27.00**	**29.33**	**28.17**	14.34	2.20	4.79	21.33
Qwen3-32B	None	26.38	24.00	25.19	22.42	–	–	22.42
	Qwen3-8B	31.30	**46.00**	38.65	12.19	1.57	6.08	19.84
	PROUR	**36.50**	44.00	**40.25**	11.23	1.53	6.11	**18.87**
DeepSeek-V4-Flash	None	36.23	38.00	37.12	32.30	–	–	32.30
	Qwen3-8B	47.80	**46.00**	46.90	15.17	1.65	7.85	**24.67**
	PROUR	**50.40**	**46.00**	**48.20**	14.48	2.57	7.79	24.84

Table 4: Cross-benchmark generalization to τ^3-bench. All modules and routing thresholds are the same as τ-bench. *Avg. Clarify* and *Avg. Verify* count routed CLARIFY and VERIFY steps issued as separate turns; methods that interleave them with agent turns are counted under *Avg. Act*.

Method	Success Rate (% ↑)				Steps ↓			
Strategy	**Retail**	**Airline**	**Telecom**	**Average**	**Avg. ACT**	**Avg. CLARIFY**	**Avg. VERIFY**	**Total**
Base Agent	17.25	20.00	13.70	16.98	33.68	–	–	33.68
Ask-before-Plan (Zhang et al., 2024)	24.60	32.00	21.90	26.17	33.57	2.95	–	36.52
VerifiAgent (Han et al., 2025)	26.30	32.00	23.39	27.23	19.05	–	11.79	30.84
SABER (Cuadron et al., 2026)	21.64	27.33	21.90	23.62	44.03	–	–	44.03
SAGE-Agent (Suri et al., 2026)	19.30	28.00	24.56	23.95	35.51	1.78	–	37.29
IG-clarifier (Deng et al., 2026)	24.00	26.00	22.81	24.27	27.95	2.80	–	30.75
LLM Router	21.90	**36.00**	21.10	26.33	24.29	1.33	4.72	30.34
Clarify-only	26.30	28.00	20.20	24.83	29.54	2.75	–	32.29
Verify-only	25.40	28.00	25.40	26.27	20.90	–	12.57	33.47
PROUR	**29.80**	30.00	**27.20**	**29.00**	18.46	1.60	9.19	**29.25**

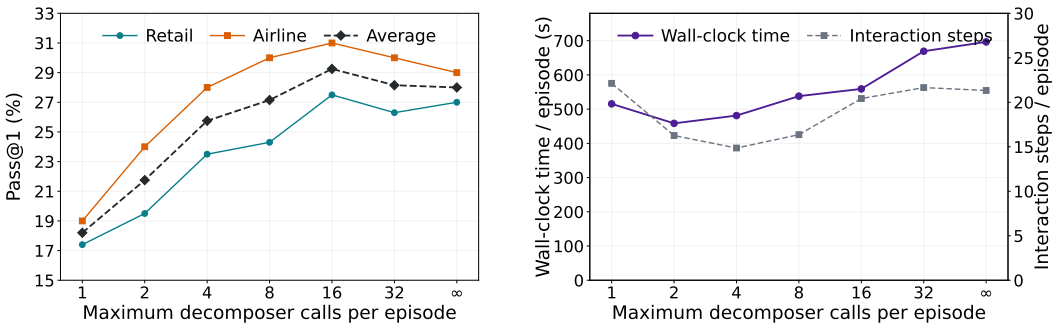
ular, Verify-only reaches 30.0% on airline but only 15.7% on retail, whereas Clarify-only is more balanced across domains. This contrasting behavior demonstrates the benefit of routing between user-side and world-side evidence rather than applying either intervention uniformly.

Effect of the Query Generator. To isolate the contribution of the learned query-generation policy, we hold the task agent, decomposer, routing thresholds, and all other auxiliary modules fixed, and vary only the query generator. As shown in Table 2, the trained Qwen3-1.7B generator achieves 28.17% average success, outperforming the prompted Qwen3-1.7B, Qwen3-8B, and Qwen3-32B generators and approaching DeepSeek-V4-Flash (28.80%) with slightly fewer steps. This suggests that the improvement does not simply arise from increasing module capacity. Instead, training specializes the generator toward queries that acquire information useful for resolving uncertainty, demonstrating the benefit of explicitly optimizing query generation for uncertainty resolution.

4.3 GENERALIZATION ACROSS AGENTS AND BENCHMARKS

Cross-agent Generalization. To assess whether the learned policy generalizes across task agents, we replace the agent backbone with Qwen3-8B, Qwen3-32B, and DeepSeek-V4-Flash while keeping the trained query generator and calibrated uncertainty thresholds fixed. Table 3 shows that our method improves average success over both the base agent and the prompted Qwen3-8B query generator for every backbone, reaching 28.17%, 40.25%, and 48.20%, respectively. The gains persist as the task agent becomes stronger, indicating that the learned uncertainty-resolution policy is not tied to the behavior of a particular downstream agent. Appendix A.2.2 separately evaluates GLM-4-32B as both the task agent and decomposer under model-specific threshold recalibration.

Cross-benchmark Generalization. To test generalization beyond the training environment, we apply the complete pipeline to τ^3-bench (Barres et al., 2026) without retraining or threshold recalibration. We evaluate on *retail*, *airline*, and *telecom*, and restrict the study to transactional text domains, where banking and voice domains (Shi et al., 2026; Ray et al., 2026) fall outside our current action space. Table 4 shows that our method achieves the highest average success (29.00%) and the fewest total steps (29.25), compared with 27.23% average success for the strongest baseline,



(a) Task success w.r.t. decomposer-call budgets. (b) Average wall-clock time and interaction steps.

Figure 5: **Effect of the decomposer-call budget on performance and cost.** (a) Success rate on retail, airline, and their average. (b) Average wall-clock time and total interaction steps per episode. The ∞ setting imposes no per-episode limit on decomposer calls.

VerifiAgent. This leading result indicates that the learned query generator is not restricted to retail domain, but generalizes across interaction environment.

4.4 Further Discussions

Uncertainty Resolution over Interaction. To examine how uncertainty evolves during interaction, Figure 6 reports uncertainty for PROUR and baselines over the first 16 steps. PROUR reduces total uncertainty from 0.58 to 0.11 and maintains lower uncertainty throughout the trajectory. AU decreases mostly in the early resolution of user-side ambiguity, whereas EU decreases gradually from 0.26 to 0.08, reflecting continued acquisition of world-side evidence. The small residual AU is expected because action distributions are estimated through sampling, which preserves some variability even after the user intent is largely resolved.

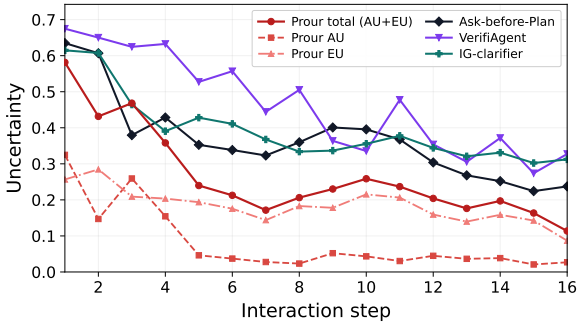


Figure 6: **Uncertainty trajectories over interaction.** PROUR reports AU, EU, and total uncertainty over the first 16 steps, while baselines report total uncertainty with decomposer applied post hoc to trajectories.

Performance–Cost Trade-off. To quantify the trade-off between uncertainty decomposition and inference cost, we conduct a post-hoc analysis varying the maximum number of decomposer calls. Figure 5 shows that introducing a small budget initially reduces total interaction steps, suggesting that timely clarification and verification can prevent unnecessary downstream tool calls. With larger budgets, interaction steps rise to 20.42 at a budget of 16, while average success continues to improve from 18.37% to 29.42%. Beyond this point, larger budgets increase cost without further benefit and slightly degrade success, suggesting that repeated interventions accumulate redundant or weakly relevant evidence after major uncertainty has been resolved. The complete trade-off results and invocation-frequency analysis are reported in Appendix A.2.3.

5 Conclusion

We introduced *uncertainty routing* for proactive tool-using agents, where uncertainty is resolved by acquiring information from the routed source. Our framework PROUR decomposes action uncertainty into aleatoric uncertainty (AU) and epistemic uncertainty (EU), routing the agent among ACT, CLARIFY, and VERIFY. A query generator is trained with a mode-conditioned information-gain objective to acquire user-side information under CLARIFY and grounded world-side evidence under VERIFY. On τ-bench, PROUR improves average success rate from 23.60% to 28.17% with fewer interaction steps, and generalizes without retraining to stronger task agents and to τ^3-bench. These results support source-aligned uncertainty routing as a principled approach for proactive agents.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing parts of the manuscript for clarity, readability, and presentation. We also used generative AI tools to assist with annotating τ-bench trajectories for the uncertainty case taxonomy in Appendix A.1, and to generate initial uncertainty labels for the development-set routing analysis described in Appendix A.2.2. All such annotations were manually reviewed and corrected by the authors before being used for calibration and analysis. We have not used generative AI tools to develop the conceptual framework, propose hypotheses, formulate mathematical claims, or implement the method. We have reviewed all AI-assisted work and take responsibility for the final content, analyses, claims, and results presented in this paper.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, Seoul, South Korea, 2026. PMLR. URL https://icml.cc/virtual/2026/poster/64377.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

Alejandro Cuadron, Pengfei Yu, Yang Liu, and Arpit Gupta. SABER: Small actions, big errors—safeguarding mutating steps in LLM agents. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems (MemAgents)*, 2026. URL https://openreview.net/forum?id=En2z9dckgP.

Mengyi Deng, Zhiwei Li, Xin Li, Tingyu Zhu, Ying Zhao, Zhijiang Guo, and Wei Wang. Uncertainty-aware clarification in LLM agents with information gain. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, Seoul, South Korea, 2026. PMLR. URL https://icml.cc/virtual/2026/poster/66065.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10602–10621, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.711. URL https://aclanthology.org/2023.findings-emnlp.711/.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, pp. 57734–57811, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/fef126561bbf9d4467dbb8d27334b8fe-Paper-Conference.pdf.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. VerifiAgent: a unified verification agent in language model reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 16410–16431, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.891. URL https://aclanthology.org/2025.findings-emnlp.891/.

Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. Decomposing uncertainty for large language models through input clarification ensembling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19023–19042. PMLR, 2024. URL https://proceedings.mlr.press/v235/hou24b.html.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745, 2011. URL https://arxiv.org/abs/1112.5745.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? On the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57/.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022. URL https://arxiv.org/abs/2207.05221.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, volume 30, pp. 5574–5584. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html.

Katarzyna Kobalczyk, Nicolás Astorga, Tennison Liu, and Mihaela van der Schaar. Active task disambiguation with LLMs. In *International Conference on Learning Representations*, pp. 37823–37847, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5e07476b6bd2497e1fbd11b8f0b2de3c-Paper-Conference.pdf.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Chen Ling, Xujiang Zhao, Xuchao Zhang, Wei Cheng, Yanchi Liu, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Jie Ji, Guangji Bai, Liang Zhao, and Haifeng Chen. Uncertainty quantification for in-context learning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3357–3370, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.184. URL https://aclanthology.org/2024.naacl-long.184/.

Kata Naszadi, Putra Manggala, and Christof Monz. Aligning predictive uncertainty with clarification questions in grounded dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14988–14998, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.999. URL https://aclanthology.org/2023.findings-emnlp.999/.

Sudha Rao and Hal Daumé III. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2737–2746, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1255. URL https://aclanthology.org/P18-1255/.

Soham Ray, Keshav Dhandhania, Victor Barres, and Karthik Narasimhan. τ-Voice: Benchmarking full-duplex voice agents on real-world domains. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, Seoul, South Korea, 2026. PMLR. URL https://icml.cc/virtual/2026/poster/66590.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-Knowledge: Evaluating conversational agents over unstructured knowledge. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, Seoul, South Korea, 2026. PMLR. URL https://icml.cc/virtual/2026/poster/63427.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured uncertainty guided clarification for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40811–40838, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.2028. URL https://aclanthology.org/2026.findings-acl.2028/.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qkWP6phrvZ.

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*, pp. 67260–67283. PMLR, 2025. URL https://proceedings.mlr.press/v267/wu25i.html.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, pp. 9965–10017, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Michael JQ Zhang and Eunsol Choi. Clarify when necessary: Resolving ambiguity through interaction with LMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5541–5558, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.306. URL https://aclanthology.org/2025.findings-naacl.306/.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-Plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10836–10863, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.636. URL https://aclanthology.org/2024.findings-emnlp.636/.

A APPENDIX

A.1 UNCERTAINTY CASE TAXONOMY

We construct the taxonomy from historical retail and airline τ-bench trajectories by treating every assistant response or tool call as a decision step. For each decision, GPT-5.5 receives the task instruction, recent dialogue context, the current assistant decision, reference actions and outputs, and any immediately following tool observations. It determines whether uncertainty is present; classifies it as epistemic, aleatoric, or absent; and records the uncertain object, available evidence, resolution outcome, and a provisional open-ended case label. The annotations are manually verified and corrected, to ensure that instructions do not infer uncertainty solely from task failure or deviation from a reference action, and include uncertainty that is handled successfully during an interaction. We then normalize the annotations and consolidate the open labels using the decision type, tool-call type, and described evidence, assigning each retained decision step to one of the nine source-relevant cases in Table 5. The table reports 19,022 step-level events, rather than trajectory-level failures, so a trajectory can contribute multiple events.

For source attribution, cases resolvable from profiles, tools, policy, computation, or interaction history are categorized as world-side (epistemic), whereas missing user-owned intent or preferences are categorized as user-side (aleatoric). Under this categorization, 16,947 of the 19,022 events (89.1%) are world-side, while 2,075 (10.9%) are user-side (Figure 2). Thus, clarification alone addresses only a small portion of the observed uncertainty; most cases require retrieving or verifying information about the world.

Table 5: Uncertainty cases identified in τ-bench trajectories; parentheses in *Examples* refer to representative decision steps in the trajectories.

Cases	Events	Uncertainty	Examples
Entity grounding	6,438	Epistemic	Missing user ID and reservation ID; resolvable through profile lookup. (`gpt-4o-airline:0:decision:0`)
State or evidence lookup	3,880	Epistemic	Retrieve the current reservation state before acting. (`gpt-4o-airline:0:decision:3`)
Tool-argument verification	2,356	Epistemic	A return-flight search reverses the required origin and destination. (`gpt-4o-airline:10:decision:12`)
User intent or preference clarification	2,075	Aleatoric	The user must select a cancellation reason or return-flight option. (`gpt-4o-airline:10:decision:7`)
Calculation verification	1,751	Epistemic	The agent computes \$255, whereas tool evidence reports \$305. (`gpt-4o-airline:0:decision:12`)
Plan completion verification	1,243	Epistemic	The agent books a replacement without verifying that the original reservation was cancelled. (`gpt-4o-airline:10:decision:18`)
Unnecessary questions answerable by retrieval	645	Epistemic	Asks the user for an identifier or order detail available through tools. (`gpt-4o-airline:101:decision:7`)
Domain-rule or policy retrieval	596	Epistemic	Uncertain whether passenger removal or basic-economy exchange is permitted. (`gpt-4o-airline:10:decision:3`)
Partial completion	38	Epistemic	Declares success while a requested subgoal remains incomplete. (`gpt-4o-airline:100:decision:10`)

A.2 EXPERIMENTAL DETAILS

A.2.1 IMPLEMENTATION DETAILS

Interaction Trajectory Generation. We collect 500 trajectories from a τ-bench environment augmented with a query generator using the retail training split. The Qwen3-8B task agent interacts with domain tools under the standard τ-bench protocol, a Qwen3-32B teacher query generator is invoked after tool calls to decide whether to ask a follow-up query, and a Qwen3-8B user simulator provides the response. This produces trajectories $\tau = \{U, (a_t, o_t, Q_t, A_t)_{t=1}^{T}, G\}$, where U

and G are the initial message and task-defined goal, (a_t, o_t) is a tool interaction, and (Q_t, A_t) is an optional query–response exchange. Importantly, the teacher-generated queries used during data collection serve only to elicit user feedback and enrich the interaction histories; we do not retain them as SFT targets or convert them into preference pairs. Directly supervising on these outputs would train the query generator to imitate the teacher rather than optimize the mode-conditioned information-gain objective. Instead, all query generator decisions and queries used for optimization are produced by the Qwen3-1.7B policy during on-policy rollouts and scored by our reward.

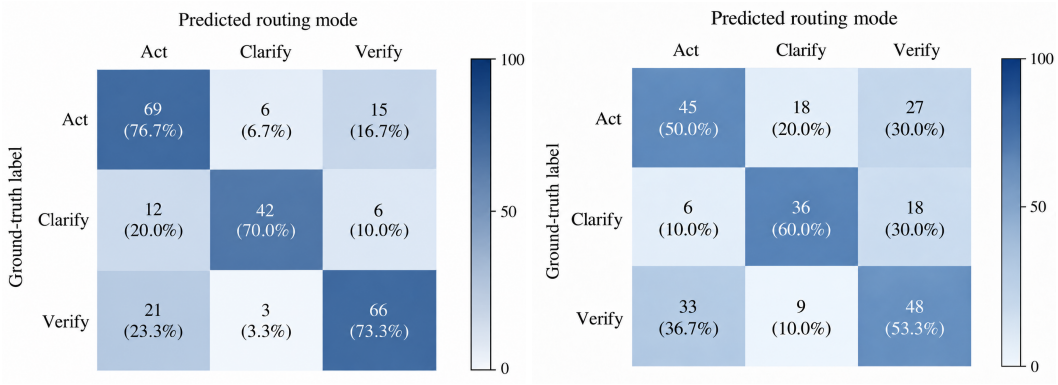
Step-level Instance Extraction. We split trajectories into disjoint training and validation partitions before extracting step-level instances. We then retain the first 16 agent decisions from each trajectory and construct the history $x_t = (U, (a_i, o_i, Q_i, A_i)_{i<t})$. The calibrated decomposer assigns each state to ACT, CLARIFY, or VERIFY; ACT states are omitted because they do not invoke the query generator. This yields 3,520 state–mode instances, divided into 2,816 training and 704 validation examples. For CLARIFY, the reward target is the normalized user goal G^*; for VERIFY, it is the reference next action a_t^*. As in the IG-Clarifier pipeline, goal normalization converts third-person descriptions into first-person statements, rewrites structured fields and key–value attributes as fluent natural-language descriptions, and filters out metadata or other non-functional attributes that do not affect the task. It preserves task-specific values, such as entities, quantities, identifiers, preferences, and requested actions, so that the resulting goal remains faithful to the original task while providing a consistent target for training.

Model and System Configuration. The task agent, decomposer, user simulator, and verifier are frozen Qwen3-8B models with role-specific prompts. The query generator is initialized from Qwen3-1.7B and is the only optimized component. We train with DAPO under the VERL framework. We use FlashAttention-2 and gradient checkpointing during training. Rollouts are generated with vLLM, with chunked prefill enabled and GPU memory utilization set to 0.20. The maximum sequence length is 3,160 tokens, comprising at most 1,400 prompt tokens and 1,760 response tokens. Dialogue histories are truncated from the left when they exceed this maximum length. Training runs on one node with two 80 GiB NVIDIA H100 GPUs.

Optimization, Batching, and Sampling. We use the GRPO advantage estimator with asymmetric PPO clipping. The learning rate is 7×10^{-7} with 60 warmup steps; the lower and upper clipping bounds are 0.20 and 0.30, and the entropy coefficient is 0.01. Weight decay is 0.01, gradients are clipped to norm 1.0, losses use token-level mean reduction, and no explicit KL loss is applied. At each training step, 16 prompts each produce eight candidates, giving 128 rollouts; six candidates per prompt are retained for optimization. Rollouts use temperature 0.8, top-$p = 1.0$, and no top-k truncation. We perform four PPO epochs with mini-batches of one prompt and micro-batches of one per GPU. Training data are shuffled, and Qwen3 thinking is enabled. We optimize for 2,010 steps, save checkpoints every 100 steps, and use the final checkpoint for evaluation.

Baselines and Ablations. We compare against the base agent without proactive intervention and five representative information-seeking approaches. IG-Clarifier (Deng et al., 2026) learns clarification questions and resolves uncertainty only by querying the user. Ask-before-Plan (Zhang et al., 2024) predicts clarification needs before planning but does not explicitly decompose and route uncertainty. VerifiAgent (Han et al., 2025) performs meta- and tool-based verification without clarifying missing information from the user. SABER (Cuadron et al., 2026) gates explicit user confirmation on mutating actions, adds targeted reflection, and cleans context, so it requests user consent rather than acquiring world-side evidence. SAGE-Agent (Suri et al., 2026) tracks structured uncertainty over tool parameters and asks the highest expected-value-of-perfect-information question to the user. They are all reimplemented on the Qwen3-8B backbone. Each controlled ablation changes a single component: LLM ROUTER replaces the AU/EU decomposer with a prompted Qwen3-8B router; CLARIFY-ONLY disables the VERIFY branch; and VERIFY-ONLY disables the CLARIFY branch. We also replace the trained query generator with prompted Qwen3-8B, Qwen3-32B, or DeepSeek-V4-Flash while keeping the remaining pipeline fixed.

Evaluation Setting. We evaluate on 115 held-out retail tasks and 50 out-of-distribution airline tasks. Unless otherwise stated, we use decoding temperatures of 0.01, 1.0, and 0.0 for the task agent, user simulator, and query generator, respectively. $\mathrm{Pass@1}$ is the percentage of tasks receiving the benchmark's binary success reward averaged over three random seeds, and *Average* is the unweighted mean across domains. For efficiency, we report wall-clock seconds and total interac-



(a) Calibrated PROUR Router (b) Prompted LLM Router

Figure 7: **Held-out routing performance of our calibrated PROUR and the prompted LLM Router.** Rows denote ground-truth labels and columns denote predicted modes; each cell reports the count and row-normalized percentage. PROUR applies the Qwen3-8B thresholds selected on the retail development set, whereas the prompted Qwen3-8B router predicts a mode directly, without calibrated thresholds. Both methods are evaluated on the same disjoint airline test set.

tion steps partitioned into ACT, CLARIFY, and VERIFY turns. For τ^3-bench, we use release v1.0.1, which incorporates the maintainers' task corrections based on SABER (Cuadron et al., 2026).

A.2.2 UNCERTAINTY ESTIMATION AND ROUTING CALIBRATION

Uncertainty is Model-specific. Because AU and EU are computed from a particular model's predictive distributions, their numerical scales are model-specific (Kendall & Gal, 2017). Language-model confidence and knowledge boundaries vary across architectures and capabilities (Jiang et al., 2021; Kadavath et al., 2022). Replacing the model used for goal proposal and action scoring can therefore change both the action distributions and resulting uncertainty scores. We consequently recalibrate $(\tau_{\text{AU}}, \tau_{\text{EU}})$ when the agent and decomposer backbone changes, accounting for model-dependent shifts in the AU and EU distributions.

Uncertainty Estimation Protocol. At each decision step, the decomposer independently samples $K = 5$ plausible goal interpretations and, for each interpretation, independently samples $L = 4$ structured actions. A deterministic parser requires exactly the top-level fields `tool` and `arguments` and validates the action against the corresponding tool schema; invalid outputs are discarded without resampling. Valid actions are canonicalized and deduplicated across all sampled actions to form one step-specific candidate set shared across interpretations. Each action is then scored under each goal interpretation by summed teacher-forced token log-likelihood, including the common completion token, and normalized over this candidate set to obtain the distributions used for estimating AU and EU. Appendix A.4.1 provides the proposing, sampling, and scoring prompts. The procedure requires $K(L + 2)$ model calls when all candidates are scored as one batch per interpretation (30 model calls for $K = 5, L = 4$).

Calibration Protocol. To avoid test-set leakage, we calibrate routing thresholds on a disjoint τ-retail development set of 240 decision steps. Each step is first assigned one of three uncertainty labels—AU, EU, or NO UNCERTAINTY—using GPT-5.5 under the fixed annotation rubric in Appendix A.4; all labels are then manually reviewed and corrected by the authors. These labels correspond directly to the routing modes: AU to CLARIFY, EU to VERIFY, and NO UNCERTAINTY to ACT. The resulting set contains 60 CLARIFY, 90 VERIFY, and 90 ACT decisions. For held-out routing evaluation, we independently sample and annotate a disjoint set of 240 airline decision steps using the same protocol. This test set is used only after threshold selection.

The decomposer estimates AU and EU based on Eq. 4–5. We perform a grid search over $(\tau_{\text{AU}}, \tau_{\text{EU}}) \in [0, 0.8]^2$ with step size 0.01, applying the AU-first routing rule in Eq. 6, and select the threshold pair that maximizes three-class macro-F1 on the development set. For the default Qwen3-8B decomposer, this yields $(\tau_{\text{AU}}, \tau_{\text{EU}}) = (0.25, 0.20)$ with a macro-F1 of 0.714. We then freeze

Table 7: **Effect of model-matched routing thresholds.** GLM-4-32B serves as both the task agent and decomposer in all rows. The upper three rows use thresholds calibrated for Qwen3-8B, and the lower three rows use thresholds calibrated for GLM-4-32B. Query generators are the same across the two blocks. The *None* rows are base agent without decomposer, and do not use routing thresholds.

Setting		Success Rate (% ↑)			Steps ↓			
Calibrated Model	**Query Generator**	**Retail**	**Airline**	**Average**	**Avg. ACT**	**Avg. CLARIFY**	**Avg. VERIFY**	**Total**
	None	18.26	14.00	16.13	31.44	–	–	31.44
Qwen3-8B	Qwen3-8B	31.30	28.00	29.65	22.72	2.41	14.12	39.25
	PROUR	**36.52**	**30.00**	**33.26**	19.13	1.24	10.46	**30.83**
	None	18.26	14.00	16.13	31.44	–	–	**31.44**
GLM-4-32B	Qwen3-8B	33.90	30.00	31.95	21.17	1.27	15.65	38.09
	PROUR	**40.00**	**34.00**	**37.00**	20.62	2.67	11.69	34.98

these thresholds and apply them unchanged to the airline test set. As a threshold-free comparator, the prompted Qwen3-8B LLM Router receives the same decision history and available tools, but directly predicts ACT, CLARIFY, or VERIFY without estimating AU or EU. As shown in Figure 7, our calibrated PROUR router achieves a macro-F1 of 74.04%, compared with 54.24% for the prompted LLM Router. This improvement supports the benefit of explicit uncertainty decomposition with calibrated thresholds over direct prompted routing.

Because uncertainty scores are model-specific, we repeat the same calibration procedure for GLM-4-32B, obtaining $(\tau_{\text{AU}}, \tau_{\text{EU}}) = (0.20, 0.08)$ with a macro-F1 of 0.711. All thresholds are selected before test evaluation and remain fixed thereafter. Table 6 summarizes the corresponding class-level F1 scores. The two backbones achieve similar macro-F1 but differ by mode: GLM-4-32B obtains higher F1 on ACT and CLARIFY, whereas Qwen3-8B obtains higher F1 on VERIFY.

Table 6: **Routing F1 after model-specific calibration.** Macro-F1 and per-class F1 evaluate three-way routing on the same development set.

Metric	**Qwen3-8B**	**GLM-4-32B**
$(\tau_{\text{AU}}, \tau_{\text{EU}})$	$(0.25, 0.20)$	$(0.20, 0.08)$
Macro-F1	0.714	0.711
ACT F1	0.700	0.742
CLARIFY F1	0.683	0.706
VERIFY F1	0.760	0.685

Table 7 tests whether this recalibration affects downstream performance while fixing GLM-4-32B as both the task agent and decomposer. Model-matched thresholds increase average success from 33.26% to 37.00% with our trained query generator and from 29.65% to 31.95% with the prompted Qwen3-8B generator. The improvement after model-matched recalibration is consistent across both query generators, providing empirical evidence that uncertainty distribution is model-specific. For our generator, higher success is accompanied by more intervention steps, indicating that recalibration changes the routing policy's intervention rate rather than providing a cost-free gain.

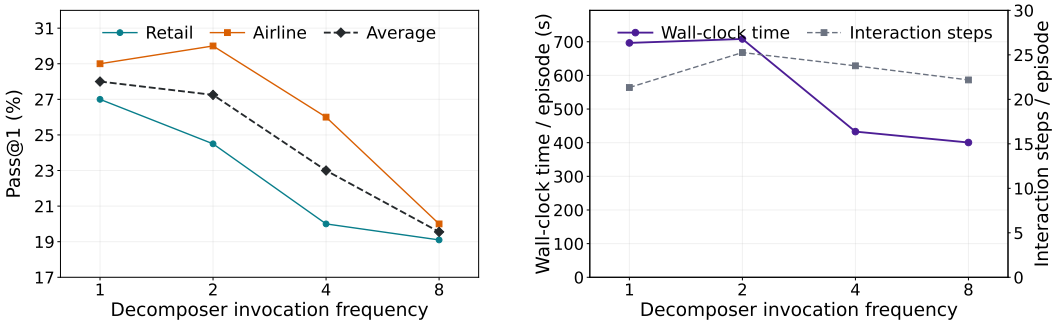
A.2.3 PERFORMANCE–COST TRADE-OFF

Table 8 reports the complete trade-off evaluation under two settings. The budget limits the maximum number of decomposer calls per episode. When the budget is exhausted, the conversation continues between the user and base agent, without decomposer or query generator. The frequency setting invokes the decomposer once every n agent decision steps under an unlimited budget.

As shown in Figure 8, reducing the frequency from every step to every second step preserves most of the success rate (27.32% versus 28.17%), but increases both interaction steps (25.26 versus 21.33) and wall-clock time (708.27 versus 696.51 seconds). Skipping a timely routing opportunity can therefore prolong the trajectory by forcing the agent to compensate with additional steps, which can outweigh the compute saved by fewer decomposer calls. Invoking the decomposer every four or eight decisions lowers interaction steps and wall-clock time, but average success drops sharply to 23.00% and 19.55%. The trade-off is therefore non-monotonic: moderately sparse routing can lengthen an episode enough to erase its savings by fewer decomposer calls, whereas very sparse routing lowers runtime only at the cost of missing timely clarification and verification, which sub-

Table 8: Performance on τ-bench under different decomposer-call budgets and invocation frequencies. *Total steps* counts the number of interaction steps.

Setting		Success Rate (% ↑)			Cost ↓	
		Retail	Airline	Average	Total steps	Time/episode (s)
Budget	1	17.40	19.33	18.37	22.13	515.45
	2	19.42	24.00	21.71	16.26	458.49
	4	23.50	28.00	25.75	14.86	481.20
	8	24.30	30.00	27.15	16.36	538.01
	16	27.50	31.33	29.42	20.42	559.16
	32	26.38	30.00	28.19	21.65	669.24
	Unlimited	27.00	29.33	28.17	21.33	696.51
Frequency	Every 1	27.00	29.33	28.17	21.33	696.51
	Every 2	24.64	30.00	27.32	25.26	708.27
	Every 4	20.00	26.00	23.00	23.77	432.93
	Every 8	19.10	20.00	19.55	22.18	400.56



(a) Task success w.r.t. invocation frequencies.

(b) Average wall-clock time and interaction steps.

Figure 8: **Effect of decomposer invocation frequency on performance and cost.** (a) pass@1 on retail, airline, and their average. (b) Average wall-clock time and total interaction steps per episode. Each x-axis value n denotes invoking the decomposer once every n agent decision steps.

stantially degrades task success. Among the evaluated frequency settings, decomposing and routing every step provides the best overall balance of task success and interaction efficiency.

A.3 Mode-Conditioned Information-Gain Training Framework

We generalize the belief-update estimator and group-relative DAPO optimization of Deng et al. (2026) to the mode-conditioned source and target in Equation 8. This appendix provides the training details omitted from the main text.

Belief Update. The query generator serves as two roles during training. It generates candidate queries through q_θ. During reward computation, it also serves in teacher-forcing mode as belief scorer to evaluate its own belief in the correct target through b_θ, with gradients detached through the scoring operation. For a target sequence $Y_t = (y_1, \ldots, y_L)$ and context c, we use the length-normalized log-likelihood

$$\ell_\theta(Y_t \mid c) = \frac{1}{L} \sum_{j=1}^{L} \log b_\theta(y_j \mid c, y_{<j}). \tag{11}$$

The prior and posterior contexts are (h_t, m_t) and (h_t, m_t, Q_t, A_t), respectively, which yields the reward in Equation 10. Length normalization makes the reward comparable across targets of different lengths. If the query generator emits NO, the posterior equals the prior and we set $R_t = 0$.

We intentionally use the current query generator as the belief scorer so that information gain is measured in the policy's own evolving decision space: reward measures whether newly acquired evidence makes the target more identifiable to the policy being optimized, rather than to a separate frozen model with potentially different knowledge or calibration.

Mode-Conditioned Rollouts. For each state–mode pair (h_t, m_t), the rollout policy samples a group $\{Q_t^{(1)}, \ldots, Q_t^{(M)}\}$.

To prevent the query generator from hacking reward by eliciting informative responses, in `clarify` mode, a strict user simulator conditioned on G^* responds to each candidate. It reveals task-specific information only when the query is precise and relevant, so high rewards are assigned to queries that truly require clarification. In `verify` mode, the frozen verifier conditions on the domain policy, interaction history, and permitted read-only tool evidence. It may retrieve state, check tool arguments and policy constraints, recompute values, or assess subgoal completion, but returns `insufficient evidence` when no grounded response is available. Neither responder receives gradient. Their prompt templates are provided in Appendix A.4.

For each response $A_t^{(i)}$, Equation 10 produces reward $R_t^{(i)}$. The empirical mean

$$\widehat{\mathcal{U}}_t = \frac{1}{M} \sum_{i=1}^{M} R_t^{(i)} \tag{12}$$

is the Monte Carlo estimate of the expected belief-update utility during on-policy sampling.

Group-Relative Advantage. To reduce reward variance, candidates from the same state–mode pair are normalized within the rollout group:

$$\widehat{A}_i = \frac{R_t^{(i)} - \mu_M(R_t)}{\sigma_M(R_t) + \epsilon}, \tag{13}$$

where μ_M and σ_M are the group mean and standard deviation and ϵ is a numerical-stability constant. This group-relative normalization further reduces sensitivity to common shifts in the belief scorer's likelihood scale. Because the reward is defined for the complete query–response exchange, the scalar advantage is assigned to all generated tokens of candidate i.

DAPO Objective. Let $w_{i,j}$ be token j in candidate i. The importance ratio between the current and rollout policies is

$$\rho_{i,j}(\theta) = \frac{q_\theta(w_{i,j} \mid h_t, m_t, w_{i,<j})}{q_{\theta_{\text{old}}}(w_{i,j} \mid h_t, m_t, w_{i,<j})}. \tag{14}$$

Following DAPO (Yu et al., 2025), we optimize the token-normalized clipped objective

$$\mathcal{J}(\theta) = \frac{1}{\sum_i |Q_t^{(i)}|} \sum_{i=1}^{M} \sum_{j=1}^{|Q_t^{(i)}|} \min\Big(\rho_{i,j}\widehat{A}_i, \text{clip}\big(\rho_{i,j}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\big)\widehat{A}_i\Big), \tag{15}$$

where $\epsilon_{\text{high}} > \epsilon_{\text{low}}$ gives favorable tokens a larger exploration range than unfavorable tokens. Every rollout group shares the same history, routing mode, response source, and target type. As training progresses, q_θ increasingly concentrates probability mass on queries that yield high information gain.

A.4 Prompt Templates

A.4.1 Uncertainty Decomposer Prompt Templates

The decomposer uses three prompts. The first prompt samples one plausible user-goal interpretation. Conditioned on that interpretation, each independent call to the second prompt samples one structured next-action candidate. Each canonical action is separately scored under the third prompt via teacher-forcing.

Goal-Interpretation Proposing Prompt

```
You are analyzing a dialogue between a τ-bench customer-service
agent and a user.  Your job is to infer one plausible
interpretation of what the user is ultimately trying to accomplish,
given the dialogue so far.
Rules:
- Write exactly one plausible interpretation of the user's
underlying goal.
- Write the interpretation in the first person and as one concise
sentence.
- Infer the goal from the dialogue history without assuming
information that has not been provided.
- Do not merely repeat the surface phrasing of the dialogue;
capture the underlying goal.
- When the dialogue is ambiguous, output one plausible
interpretation rather than enumerating multiple alternatives.
- Output only the interpretation and nothing else.
[Dialogue history]
{history}
[Output --- one plausible goal interpretation]
```

Candidate-Action Sampling Prompt

```
You are a τ-bench customer-service agent.  Sample exactly one
plausible next action using the goal interpretation, dialogue
history, and available tool schemas below.
[Goal interpretation]
{goal}
[Dialogue history]
{history}
[Available tools and argument schemas]
{tools_with_schemas}
Rules:
- Return exactly one action, not a list of alternatives.
- For an environment-tool action, use one available tool name
and provide a complete arguments object conforming to that tool's
schema.
- Preserve argument values exactly when they occur in the goal
interpretation or dialogue.  Do not invent identifiers, dates,
quantities, or other unsupported facts.
- If the appropriate next action is to answer the user without an
environment tool, use the special tool name ``respond'' with an
empty arguments object.
- Output strict JSON only, without Markdown or explanation.
Output format:
{"tool":  "<tool_name>", "arguments":  {...}}
```

Teacher-Forced Action Scoring Prompt

```
You are a τ-bench customer-service agent.  Determine the most
plausible next action from the goal interpretation, dialogue
history, and available tool schemas.  Represent the next action as
exactly one strict JSON object with the top-level fields ``tool''
and ``arguments.''
[Goal interpretation]
{goal}
```

```
[Dialogue history]
{history}
[Available tools and argument schemas]
{tools_with_schemas}
[Next action --- canonical JSON target follows]
```

A.4.2 Query Generator Prompt Templates

The query generator uses the CLARIFY or VERIFY template conditioned on the routed mode. The same templates are used by prompted Query Generator comparisons, with only the underlying model changed.

Query Generator Prompt for Clarify Mode

```
Role
You are a Query Assistant responsible for determining whether the
agent needs to ask a clarifying question.
History:
{history}
Instructions:
Analyze the dialogue and decide whether clarification is needed.
Any reasoning, if you use it, must be enclosed between <think> and
</think>.
The final answer must come after this complete reasoning block.
If one short question is needed before acting, reply exactly:
YES [QUESTION]your question here[/QUESTION]
If no question is needed, reply exactly:
NO
Ask only one concise question about user intent.
Do not ask about data retrievable from tools.
Do not repeat previous questions.
Answers written only inside <think> are invalid.
expected output:
Either:
- NO
or
- YES [QUESTION]...[/QUESTION]
```

Query Generator Prompt for Verify Mode

```
You are a Query Assistant operating in VERIFY mode.  The agent is
uncertain about a fact in the world (e.g., the value of an order
field, a policy rule, the result of a computation).  Your job
is to decide whether to issue a verification query that will be
answered by a separate Verifier module with tool access, and if so,
to phrase the precise query.
Output format (strict):
Any reasoning, if you use it, must be enclosed between <think> and
</think>.
The final answer must come after this complete reasoning block.
If a verification query is needed, reply exactly:
YES [QUESTION]your question here[/QUESTION]
If no verification is needed, reply exactly:
NO
Guidelines:
- The query must be specific and answerable by retrieving evidence
(tool output, computation, policy snippet).
- Examples of good queries:  "What items are in order #W7449508?",
"Is basic-economy exchangeable under current policy?", "What is the
```

```
total fare for HAT136 + HAT039 in economy?"
- Do NOT ask the user --- the user already provided their intent.
Ask about world state.
- One query only.  Keep it under 30 words.
Answers written only inside <think> are invalid.
[Dialogue history]
{history}
[Your decision]
```

The belief scorer (same model as query generator) uses prompts to estimate prior and posterior belief before and after the query–response exchange. For the prior score, the mode-conditioned turn placeholder is empty. In CLARIFY mode it predicts the normalized user goal; in VERIFY mode it predicts the reference next tool action.

Clarify-Mode Posterior Belief Prompt

```
You are an expert analyst.  Summarize the user's hidden profile and
intent based on the dialogue.
[Dialogue]
{history}
{clarification_turn}
### User Intent Summary:
```

Verify-Mode Posterior Belief Prompt

```
You are a tau-bench agent.  Given the dialogue history and any
verifier evidence, predict the next tool call the agent should
make.
[Dialogue]
{history}
{verification_turn}
### Next tool call:
```

A.4.3 USER SIMULATOR AND VERIFIER PROMPT TEMPLATES

The verifier receives only read-only tools together with the domain policy and visible history. Its response must cite retrieved, computed, or policy evidence; otherwise it returns `insufficient evidence`.

Verifier Prompt

```
You are a Verifier.  Your job is to answer a verification query
about the current dialogue state by retrieving grounded evidence
--- never by speculation.
You have access to a set of READ-ONLY tools (provided in the tools
list).  You may also reason over the dialogue history.
STRICT RULES:
1.  Your final answer MUST include at least one of:
(a) A citation of a tool you called and the value it returned
(e.g., "get_order_details(order_id=W7449508) returned:  {{status:
'pending', ...}}").
(b) A computed value with the inputs shown (e.g., "Total = $150 +
$155 = $305 from get_flight_details on HAT136 and HAT039").
(c) A direct quote of a policy snippet from the system prompt or
wiki.
2.  If you cannot reach a grounded answer (the needed tool is not
available, or the data is missing), reply with exactly:
insufficient evidence
```

```
3.  Do NOT speculate.  Do NOT invent values.
4.  Do NOT call any tool that mutates state (the tool list provided
is already filtered to read-only).
5.  After at most {max_steps} tool-call rounds, return your final
natural-language answer.
[Domain policy / wiki]
{policy}
[Dialogue history so far]
{history}
[Verification query]
{question}
```

During training, the strict user simulator releases information only in response to a specific, relevant query.

Strict User-Simulator Prompt for DAPO Training

```
Role
You are a user interacting with an agent.  Your behavior simulates
a human user following a hidden instruction.
Your hidden instruction:
{G*}
Instructions:
conversation guidelines:
- Respond with one message at a time using first-person statements.
- Do not invent details that are not in the instruction; if
something is unknown, say you do not remember it.
- Rephrase the instruction in your own words and maintain a
natural, human-like tone.
IMPORTANT - Handling vague or generic questions:
- If the agent asks a vague, overly broad, or generic question
(e.g., "Is there any additional information?", "Can you tell me
more?", "Anything else?"), reply with:  "No, that's all." or "Just
do what I asked."
- If the agent outputs a placeholder or acts out of character
(e.g., "Your concise and specific clarifying question to the user
here"), reply with:  "Who are you talking to?"
- For such questions, only provide information if it is DIRECTLY
relevant to the current step of the instruction.
- If the question is too vague to answer meaningfully, respond with
"I'm not sure what specific information you need" or "Could you be
more specific?"
- Prefer specific, targeted questions that help the agent
understand your exact needs.
Question:  {question}
expected output:
A single user utterance that follows the instruction and
guidelines.
```

During evaluation, user responses follow the standard τ-bench simulator prompt instead of reusing the strict user prompt for training.

User Simulator Prompt for Main Evaluation Results

```
You are a user interacting with an agent.
Instruction:  {instruction}
Rules:
- Generate one line at a time to simulate the user's message.
- Do not reveal the entire instruction at once.  Provide only
```

```
information necessary for the current step.
- Do not invent information absent from the instruction.  If asked
for an unavailable detail, say that you do not remember or have it.
- If the instruction goal is satisfied, output ###STOP### as a
standalone message.
- Do not repeat the instruction verbatim; express it naturally in
your own words.
- Maintain a natural conversation style and follow any personality
specified in the instruction.
```

The task agent itself follows the standard τ-bench setup: the complete domain policy is supplied as its system prompt and the environment exposes the domain tool schemas through the model's tool interface. We add no task-specific demonstration to the Qwen3-8B agent.

A.4.4 Data Preparation and Calibration Prompt Templates

The raw task instruction is converted into the normalized goal G^* using the following template.

Goal-Normalization Prompt

```
Role
You are a data converter.  Your goal is to rewrite raw user
detailed data into a flat, natural language format optimized for
training smaller language models.
Transformation Rules:
1.  First-person normalization:  convert third-person identifiers
to first-person intent (e.g., "Your name is" to "My name is").
2.  Remove Noise:  Remove all personality descriptions (e.g.,
"logical, shy, organized").
3.  Flatten JSON (Crucial):  Remove all JSON syntax ({{,}},',:).
Convert key-value pairs into natural phrases.
Use connectors like "with", "and", "set to".
Example:  {{'color':  'red', 'size':  'M'}} → with color red and
size M.
4.  Handle "(same as ...)":  Keep the reference intact.
Ensure it flows logically with the details.
Example:  {{'zip':  '10228'}} (same as #W123) → with zip 10228,
which is the same as #W123.
5.  Preserve Content:  Do NOT delete specific values (like "144
Lakeview Drive").  Keep them just in case the user mentions them.
### Example:
Input:
"Your name is Alex.  You are shy and loud.  For #W1, change address
to {{'street':  '123 Main St', 'zip':  '90210'}} (same as #W2)."
Output:
"My name is Alex.  For order #W1, change address to street 123 Main
St and zip 90210, which is the same as #W2."
Input:
"{instruction}"
Output:
```

In the development set, the labels for calibrating routing thresholds are annotated by GPT-5.5 with the following fixed rubric before grid search.

Uncertainty Labelling Prompt for Calibration

```
You are an expert annotator labeling agent uncertainty in τ-bench
dialogues.
At the decision point below, classify the next assistant move using
EXACTLY ONE of these labels:
```

```
- epistemic:  the next assistant move should resolve a concrete
fact, policy, or world state by looking something up, reading the
wiki, or using a tool.
- aleatoric:  the next assistant move should ask the user because
the missing information is a preference, choice, or selection among
options.
- no_uncertainty:  the agent already has enough information to act
decisively right now.
Decision rule:
- Choose epistemic for missing factual information that the
assistant should look up or verify, including account/order/product
identifiers, policy checks, and other concrete facts.
- Choose aleatoric when the unresolved issue is a user preference,
choice, or selection among valid options.
- Do not use aleatoric for factual lookup requests such as order
IDs, email addresses, zip codes, account numbers, product IDs,
tracking numbers, or policy lookups.
- Do not use aleatoric for identity verification or account
recovery prompts such as "Could you share your order number?",
"What is your email address?" Those are epistemic because the
assistant is asking for a concrete fact.
- Use aleatoric when the user must express a genuine preference
or pick between valid alternatives, such as color, size, shipping
speed, refund destination, or which of several acceptable options
to take.
- If the assistant can directly call a tool or finish the request
without asking anything else, choose no_uncertainty.
Examples:
- The agent needs an order ID, account number, product ID, or
policy rule from the wiki:
epistemic
- The agent needs the user's order number, email, zip code, account
name, or tracking number to continue:
epistemic
- The user has to choose between two colors, two shipping speeds,
or two return reasons:
aleatoric
- The user needs to choose a refund method, item option, or other
valid preference among acceptable alternatives:
aleatoric
- The user already gave every factual detail, but the next step is
to ask for a
preference or choice from a short list:
aleatoric
- The user has confirmed the factual record, and the next step is
to ask which option they prefer:
aleatoric
- The user says "I don't care", "whatever is cheaper", or "surprise
me" and the only unresolved part is the user's choice:
aleatoric
- The assistant would need to ask the user "yes/no", "which one",
or "what should I do" because the choice itself is unresolved:
aleatoric
- The agent already has enough facts to proceed with a tool call or
a direct response:
no_uncertainty
- The user has already stated the preference and no further
question is needed:
no_uncertainty
- The assistant can directly call a tool or finish the request
without asking anything else:
no_uncertainty
```

```
Output format (strict): a single line containing ONLY the label, no
explanation, no punctuation. Example: epistemic
[Dialogue history at decision point]
{history}
[Label]
```

A.4.5 Ablation Prompt Templates

The LLM Router ablation replaces goal-conditioned decomposition with a single three-way prompted decision while keeping the downstream query generator unchanged.

LLM Router

```
You are a routing policy for a tau-bench customer-service agent.
Choose exactly one next mode for the agent:
- act: the agent should take the next normal tool or
final-response action.
- clarify: the agent needs information that only the user can
answer, such as preference, intent, or an ambiguous user-owned
choice.
- verify: the agent needs world-side evidence, policy evidence,
computation, or a read-only lookup before taking a mutating action
or final answer.
Prefer act when the next step is clear. Prefer verify over clarify
when the missing information is retrievable from tools, prior tool
outputs, policy text, or deterministic calculation. Prefer clarify
only for user-owned ambiguity.
Available action/tool names:
{tools}
Dialogue history:
{history}
Return strict JSON only:
{{"mode": "act" | "clarify" | "verify", "rationale": "short
reason"}}
```