Ask or Act: Uncertainty-Aware Reinforcement Learning for Tool-Using Language Agents

Anonymous ACL submission

Abstract

Large language model (LLM) agents have shown strong potential in multi-turn tool-use tasks, yet they often act before resolving task-relevant uncertainty, causing premature decisions to cascade into long-horizon failures. Existing work mainly improves tool use through supervised fine-tuning or outcome-based reinforcement learning (RL), but largely treats action generation as a single objective and overlooks a key decision boundary: whether the agent should answer, retrieve, call a tool, or first *ask* for clarification. We introduce **AoA-RL** (Ask-or-Act RL), an RL framework that explicitly trains language agents to optimize this ask-or-act boundary under partial observability. AoA-RL treats uncertainty as an actionable control signal and models clarification as a first-class action alongside answering, retrieval, and tool use. To learn under sparse feedback, it decomposes delayed task reward into uncertainty-reduction and task-completion terms, enabling the model to distinguish useful clarifications from gratuitous turns. Across three representative agentic benchmarks covering multi-turn tool use, web navigation, and information retrieval, AoA-RL improves average task success by 5.5 points over the strongest agentic-RL baseline and reduces error propagation from 19.8% to 13.6%.

1 Introduction

Large language model (LLM) agents are increasingly deployed in settings that require interleaving reasoning with external actions, such as multi-turn tool use, web navigation, and information retrieval (Yao et al., 2023; Acikgoz et al., 2025; Xi et al., 2023). A growing body of work shows that pairing chain-of-thought reasoning with environment interaction substantially improves task completion over static question answering, and recent agentic benchmarks (Patil et al., 2025; Yao et al., 2024) report steady gains in function-calling accuracy and multi-step orchestration. As a result, language agents are no longer evaluated solely on whether they can *produce* an answer, but on whether they can *act* reliably across long horizons of partial information.

Yet a fundamental weakness persists across this progress: agents tend to act before their uncertainty is resolved. In realistic settings, user instructions are routinely incomplete or ambiguous, and the environment is only partially observable; acting on under-specified information at an early step can quietly commit the agent to an incorrect trajectory, where every subsequent tool call or retrieval compounds the original error. Three lines of work each address part of this picture but leave the core gap open. Tool-use methods focus on *how* to invoke tools more accurately (Qian et al., 2025; Dong et al., 2025; Feng et al., 2025), clarification methods study *whether* a question should be asked but typically as a prompting or scaffolding strategy (Kuhn et al., 2023; Andukuri et al., 2024; Zhang et al., 2025), and uncertainty-quantification work treats uncertainty as a *post-hoc* diagnostic rather than a control signal (Duan et al., 2025; Oh et al., 2026). What none of them optimizes directly is the decision boundary itself: at every step, should the agent answer, retrieve, call a tool, or first ask?

We address this gap with **AoA-RL** (Ask-or-Act RL), a reinforcement learning framework that trains language agents to optimize the ask-or-act boundary explicitly under partial observability (Figure 1). AoA-RL recasts each step of agent interaction as a choice over a unified action space in which clarification is a *first-class* action alongside answering, retrieval, and tool use, and uses the agent's task-relevant uncertainty as the control signal that gates this choice. To make this behavior learnable from sparse outcome feedback, we further introduce a hierarchical credit-assignment scheme that decomposes the delayed task reward into an uncertainty-reduction term, which provides per-

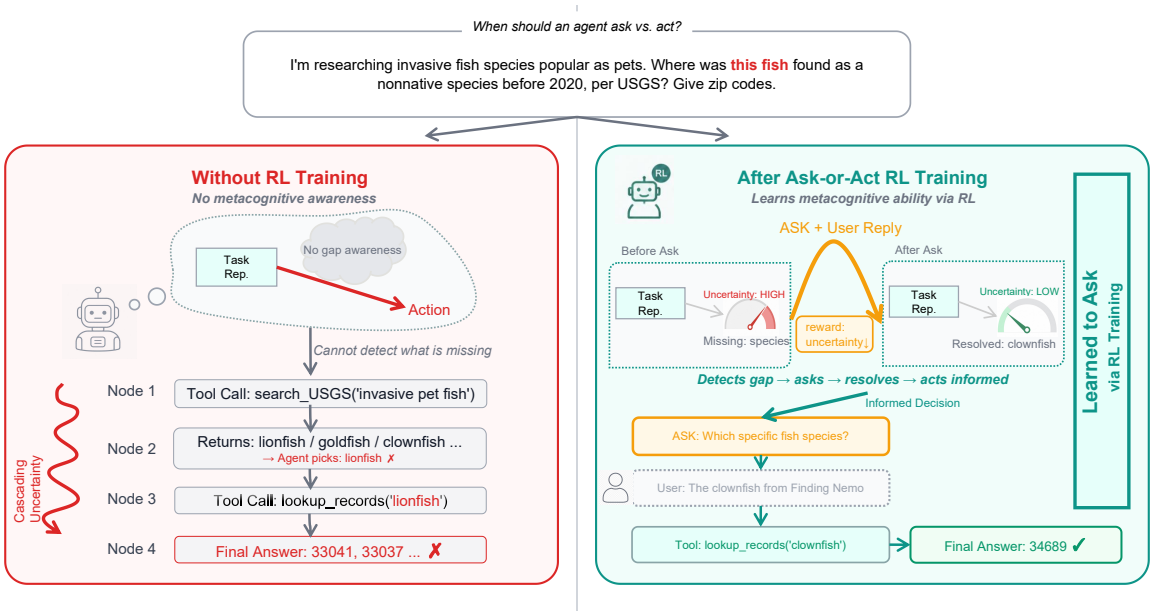

Figure 1: **When should an agent ask vs. act?** Given an under-specified user instruction (top), a standard tool-using agent (left) commits to an action without recognising the missing referent (*this fish*), and the initial uncertainty cascades into an incorrect final answer. AoA-RL (right) learns to detect the information gap, issue a targeted clarification, and act only after uncertainty is resolved.

step credit for clarifications that actually close information gaps, and a task-completion term, which keeps the policy aligned with the final objective. An interaction-cost term then prevents the agent from defaulting to over-cautious questioning.

Our contributions are threefold. **(i) Conceptual unification:** we formalise the ask-or-act decision boundary as a unified policy problem in a POMDP, treating clarification as a first-class action alongside answering, retrieval, and tool use. **(ii) Hierarchical credit assignment:** we decompose delayed task reward into uncertainty-reduction, task-completion, and interaction-cost terms, making indirectly useful actions learnable from sparse outcome feedback. **(iii) Empirical validation:** on three agentic benchmarks covering multi-turn tool use, web navigation, and information retrieval, AoA-RL achieves 76.7 average task success ($+5.5$ over the strongest agentic-RL baseline) and reduces error propagation from 19.8% to 13.6%, with the largest gains on highly ambiguous inputs.

2 Related Work

Our work draws on three lines of research that, taken together, frame the ask-or-act problem: agentic language models that *act* via tool use, methods that study *when to ask* and how to model agent *uncertainty*, and reinforcement learning approaches that train *multi-step* policies on top of language models.

2.1 Tool-Using and Agentic Language Models

A first line of work focuses on equipping language models with the ability to invoke external tools and orchestrate multi-step interactions. Early work interleaves reasoning with acting (Yao et al., 2023), while subsequent efforts unify dialogue management with agentic tool use (Acikgoz et al., 2025) and build benchmarks that stress-test function calling, parameter binding, and long-horizon orchestration in increasingly realistic settings (Patil et al., 2025; Yao et al., 2024). Recent agentic benchmarks reveal that even strong models remain brittle on tasks that require deviating from a default action trajectory, particularly when user instructions are under-specified (Yu et al., 2026). These works substantially advance *how* agents act, but largely treat the action space as fixed and leave open the question of whether the agent should act at all in the current step.

2.2 Clarification and Uncertainty in Language Agents

A second line of work asks when an agent should request more information rather than commit to an answer. Early studies show that model uncertainty does not directly correspond to human clarification behaviour, so simply supervising "when to ask" on

human-annotated data is insufficient (Kuhn et al., 2023); recent work therefore models the *future* value of a clarifying question, training agents to ask only when the question would improve downstream turns (Andukuri et al., 2024), or scopes clarification to the parameter space of the tool actually being called (Zhang et al., 2025). In parallel, a body of work on uncertainty quantification for language agents argues that single-token confidence is a poor signal in interactive settings, and proposes to decompose uncertainty across reasoning steps (Duan et al., 2025) or to track how it propagates through tool calls (Lymperopoulos and Sarathy, 2025). These two threads have largely developed in parallel: clarification is studied as a prompting or scaffolding mechanism, while uncertainty is studied as a diagnostic to be measured. What remains less explored is treating uncertainty as the *control signal* that decides between asking and acting within a unified policy.

2.3 Reinforcement Learning for Multi-Step Agents

A third line of work brings reinforcement learning to language agents that interact with tools and environments over many steps. Outcome-based RL has been shown to substantially improve tool-use accuracy and search behaviour over supervised baselines (Singh et al., 2025; Feng et al., 2025), and recent methods push further by analysing the design space of tool-use rewards (Qian et al., 2025) or by performing step-level credit assignment along long tool-use trajectories (Dong et al., 2025). These methods establish that RL is a viable training signal for agentic behaviour, and that finer-grained credit assignment matters once trajectories grow beyond a few steps. Their reward structures, however, are still organised around the quality of the action that was taken—which tool, which argument, which search query—rather than around the prior question of whether an action should have been taken at all. As a result, an action type whose value is largely *indirect*, such as asking a clarifying question that does not itself complete the task, fits awkwardly into existing reward formulations.

3 Method

3.1 Overview

We present **AoA-RL**, a reinforcement learning framework that trains a tool-using language agent to make an *ask-or-act* decision at every interaction step. At each step, the agent first encodes its history into a state representation, estimates a task-relevant uncertainty signal from that representation, and then samples a high-level action type from a unified action space that includes `Answer`, `Retrieve`, `Tool`, and `Ask`. The chosen action type conditions the generation of the action's content—a final answer, a retrieval query, a tool call, or a clarifying question—which is executed in the environment and produces the next observation. Training proceeds in two stages: a cold-start supervised stage exposes the policy to all four action types through demonstration trajectories, and a reinforcement learning stage optimises the ask-or-act decision under a hierarchical reward that decomposes the delayed task signal into an uncertainty-reduction term, a task-completion term, and an interaction-cost term. Figure 2 summarises the framework. The remainder of this section formalises the policy in Section 3.2 and the reward in Section 3.3.

3.2 POMDP Formulation and Ask-or-Act Policy

Interaction as a POMDP. We model the agent's interaction with a user, a set of tools, and an external knowledge source as a partially observable Markov decision process (POMDP)

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \gamma \rangle, \tag{1}$$

where $\mathcal{S}$ is the latent task state (the user's true intent together with relevant world facts), $\mathcal{O}$ is the space of observations the agent can access, $\mathcal{A}$ is the action space defined below, P is the environment transition distribution, R is the reward, and $\gamma \in (0, 1)$ is a discount factor. Partial observability arises naturally in our setting: user instructions can be incomplete or ambiguous, and the agent never directly observes the parameters needed by a downstream tool—it can only infer them from the trajectory so far. We write the trajectory up to step t as

$$\tau_t = (o_1, a_1, o_2, a_2, \ldots, o_t), \tag{2}$$

which concatenates the user instruction, prior actions, tool outputs, retrieved evidence, and any clarifications already received.

State and uncertainty. A pretrained language model encoder $f_\phi : \mathcal{O}^* \to \mathbb{R}^d$ compresses the trajectory into a state representation

$$h_t = f_\phi(\tau_t), \tag{3}$$

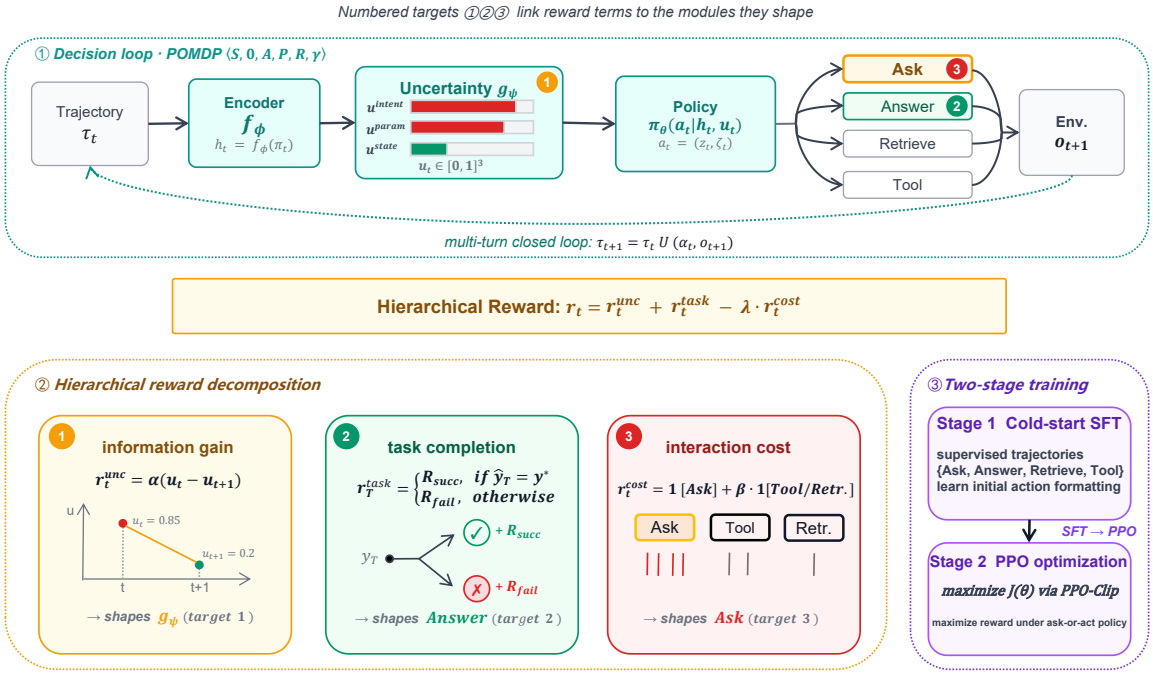

Figure 2: **The AoA-RL framework.** Top: at each step, the trajectory τ_t is encoded into a state representation h_t from which a structured uncertainty u_t is estimated; the policy $\pi_\theta(a_t \mid h_t, u_t)$ then selects one of four high-level action types (`Ask`, `Answer`, `Retrieve`, `Tool`) before generating the action content, closing the loop with the next observation o_{t+1}. Bottom: the hierarchical reward $r_t = r_t^{\text{unc}} + r_t^{\text{task}} - \lambda\, r_t^{\text{cost}}$ provides per-component supervision, and training is organised into a cold-start SFT stage followed by PPO optimisation.

which serves as the agent's belief over the latent state at step t. On top of h_t, a lightweight head g_ψ predicts a structured uncertainty vector

$$u_t = g_\psi(h_t) = \left[\, u_t^{\text{intent}},\ u_t^{\text{param}},\ u_t^{\text{state}} \,\right] \in [0,1]^3, \tag{4}$$

whose three coordinates capture, respectively, residual uncertainty about the user's intent, about parameters required by the next tool call, and about the world state queried by the task. Unlike token-level entropy from the language modelling head, u_t is task-relevant by construction: it answers the question *which kind of information is the agent still missing*, rather than how confident it is in the next token. In practice, g_ψ is a shared two-layer MLP ($d \to d/2$ with LayerNorm, GELU, and dropout) feeding three independent heads for $u_t^{\text{intent}}, u_t^{\text{param}}, u_t^{\text{state}}$. The three coordinates are aggregated as $\|u_t\|_1$ for use in both the uncertainty-reduction reward (Equation 9) and the calibration evaluation in Section 4.3; g_ψ is supervised jointly with the policy in Section 3.3.

Action space. A central design choice in AoA-RL is to admit clarification as a *first-class* action alongside answering, retrieval, and tool use. We therefore define the high-level action space as

$$\mathcal{A} = \{\, \texttt{Answer},\ \texttt{Retrieve},\ \texttt{Tool},\ \texttt{Ask} \,\}, \tag{5}$$

where `Ask` issues a clarifying question to the user, `Retrieve` queries an external knowledge source, `Tool` invokes a task tool with structured arguments, and `Answer` terminates the episode with a final response. Existing tool-using agents typically expose only the latter three; including `Ask` makes the choice between *asking* and *acting* an explicit object of policy learning rather than a side effect of prompting.

Hierarchical action selection. Each action factorises into a high-level *type* and its *content*:

$$a_t = (z_t, \xi_t), \quad z_t \in \mathcal{A}, \tag{6}$$

where ξ_t is the natural-language realisation of the action (a clarifying question, a retrieval query, a tool call with arguments, or a final answer). The policy correspondingly decomposes as

$$\pi_\theta(a_t \mid h_t, u_t) = \pi_\theta(z_t \mid h_t, u_t)\, \pi_\theta(\xi_t \mid h_t, u_t, z_t), \tag{7}$$

so that the type-level decision z_t is conditioned directly on the uncertainty signal u_t, while the content-level generation ξ_t is conditioned on the chosen type. Once a_t is executed, the environment returns o_{t+1}, the trajectory is updated to $\tau_{t+1} = \tau_t \cup \{(a_t, o_{t+1})\}$, and the encoder recomputes h_{t+1} for the next step. The closed-loop process terminates when $z_t = \texttt{Answer}$ or a step budget is reached.

Implementation note. In practice this factorisation is executed in a single autoregressive pass by the backbone language model: z_t is generated first, constrained to one of four action-type control tokens, and ξ_t is the immediately subsequent free-form generation conditioned on z_t.

3.3 Hierarchical Reward Design and RL Optimisation

Hierarchical reward. A central difficulty of training the ask-or-act policy is that Ask actions rarely produce immediate task progress: a clarifying question, by itself, does not retrieve evidence or return a tool result. If the only learning signal is the final task outcome, the model has no way to tell a useful clarification apart from a redundant turn—a credit-assignment problem that becomes more severe as trajectories grow longer. We address this with a hierarchical reward that decomposes the total per-step return into three complementary terms:

$$r_t = r_t^{\text{unc}} + r_t^{\text{task}} - \lambda\, r_t^{\text{cost}}, \tag{8}$$

where $\lambda > 0$ trades interaction budget against task utility; specific values for all reward coefficients are given in Section 4.1.

The *uncertainty-reduction* term rewards actions that close information gaps detected by g_ψ:

$$r_t^{\text{unc}} = \alpha\,\big(\,\|u_t\|_1 - \|u_{t+1}\|_1\,\big), \tag{9}$$

where $\alpha > 0$ scales the per-step credit. A clarifying question that genuinely resolves a missing intent or parameter drives the structured uncertainty u_{t+1} below u_t and receives positive credit; an uninformative question, or a tool call that does not address the gap, receives little or none. This term is the mechanism by which an action with only *indirect* task value, such as Ask, becomes learnable from sparse outcome rewards.

The *task-completion* term keeps the policy aligned with the final objective. Letting T denote the terminal step,

$$\begin{aligned} r_T^{\text{task}} &= \begin{cases} R_{\text{succ}}, & \text{if } \hat{y}_T = y^*, \\ R_{\text{fail}}, & \text{o.w.}, \end{cases} \\ r_t^{\text{task}} &= 0 \text{ for } t < T, \end{aligned} \tag{10}$$

with $R_{\text{succ}} > 0 \geq R_{\text{fail}}$. This terminal signal prevents the policy from inflating r_t^{unc} indefinitely by asking ever more questions without ever committing to an answer.

Finally, the *interaction-cost* term discourages unnecessary turns,

$$\begin{aligned} r_t^{\text{cost}} = \mathbb{I}[\,z_t = \texttt{Ask}\,] \\ + \beta\,\mathbb{I}[\,z_t \in \{\texttt{Retrieve}, \texttt{Tool}\}\,], \end{aligned} \tag{11}$$

where $\mathbb{I}[\cdot]$ is the indicator function and $\beta \in (0, 1]$ weights tool and retrieval calls relative to clarifications. Pairing r_t^{unc} with r_t^{cost} is what stops the policy from collapsing into either over-cautious questioning or premature acting: each Ask must pay for itself in measurable uncertainty reduction.

Policy optimisation. We optimise the joint policy π_θ together with the uncertainty head g_ψ to maximise the expected discounted return

$$J(\theta, \psi) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=1}^{T} \gamma^{\,t-1}\, r_t\right], \tag{12}$$

where r_t is the hierarchical reward of Equation 8. Because actions are discrete high-level types coupled with free-form content, and trajectories are long and partially observable, we use Proximal Policy Optimisation (PPO) (Schulman et al., 2017) as the underlying optimiser. Letting $\rho_t = \pi_\theta(a_t \mid h_t, u_t)/\pi_{\theta_{\text{old}}}(a_t \mid h_t, u_t)$ be the importance ratio and $\hat{A}_t$ a generalised advantage estimate, the per-step surrogate objective is

$$\mathcal{L}_{\text{RL}} = \mathbb{E}_t\Big[\min\big(\rho_t \hat{A}_t,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\Big]. \tag{13}$$

The uncertainty head g_ψ is updated by the same gradient: because u_t enters the policy through Equation 7 and the uncertainty-reduction reward through Equation 9, the credit-assignment machinery of PPO automatically shapes g_ψ toward representations whose changes track useful information gain, without requiring a separate auxiliary loss.

Two-stage training. Optimising Equation 13 from a randomly initialised action-type distribution is unstable, since most rollouts from an untrained

policy do not produce well-formed tool calls or clarifying questions. We therefore train in two strictly sequential stages. In **Stage 1 (cold-start SFT)**, the policy is optimised solely via the token-level cross-entropy loss $\mathcal{L}_{\text{SFT}}$ on demonstration trajectories that cover all four action types, until the agent produces syntactically valid actions of every type. In **Stage 2 (PPO)**, we switch entirely to PPO with $\mathcal{L}_{\text{RL}}$ and the hierarchical reward of Equation 8 to refine *when* each action type should be chosen. Implementation details, including optimiser, sequence length, and rollout setup, are deferred to Section 4.

4 Experiments

4.1 Experimental Setup

Datasets and tasks. We evaluate AoA-RL on three categories of agentic tasks that together stress different aspects of the ask-or-act decision: multi-turn tool use, information retrieval, and—as an *out-of-distribution generalisation test*—web navigation. For tool use and retrieval we start from public benchmarks (see Appendix B for details) and construct *ambiguous* variants by removing slots that are needed to solve the instance—tool arguments such as time, location, or entity scope; or retrieval constraints such as date range or source. For each removed slot we also collect the user's canonical follow-up answer, which is replayed by the environment when the agent issues an `Ask`. For web navigation we use the full **WebArena** test suite as a held-out OOD benchmark: no method (ours or baselines) is trained on WebArena trajectories, and ambiguous variants are constructed offline on top of its 812 tasks following the same slot-removal procedure. To probe behaviour at different levels of difficulty, we further stratify the evaluation set into *low*, *medium*, and *high* ambiguity tiers according to how many critical slots are missing. The resulting evaluation suite contains both information-sufficient examples (where asking is unnecessary and penalises the agent) and information-insufficient examples (where acting without asking is likely to fail).

Implementation. All methods share an identical backbone, the open-weight **Qwen2.5-7B-Instruct** model (7.62B parameters)[1], and are trained with the two-stage recipe of Section 3.3. Training uses 8×A100 80GB GPUs with DeepSpeed ZeRO-2 and bfloat16. Stage 1 SFT runs for 3 epochs (∼96 GPU-h); Stage 2 PPO runs for 1,000 updates with 8 rollouts per prompt at 16,384-token context, dominated by the rollout phase, totalling ∼1,920 GPU-h per seed. The three-seed total is ∼6,048 GPU-h. Optimisation uses AdamW (learning rate 1×10^{-6}, weight decay 0.01, β_1=0.9, β_2=0.95), mini-batch size 512, and a maximum sequence length of 16,384 tokens to accommodate long agent rollouts. PPO uses clip ratio ϵ=0.2, GAE λ_{GAE}=0.95, discount γ=0.99, and a KL coefficient of 0.05 against the SFT checkpoint. Reward coefficients of Equations 8–11 are set to α=1.0, β=0.5, λ=0.1, R_{succ}=10.0, and R_{fail}= − 5.0, all selected on the validation split of each task category via a small grid sweep ($\alpha \in \{0.5, 1.0, 2.0\}$, $\lambda \in \{0.05, 0.1, 0.2\}$); the full search grid and configuration files are released with our code. Source code, training configs, and the cold-start SFT and final RL checkpoints will be publicly released under the **CC BY 4.0** license upon publication; the camera-ready will include the repository URL.

Metrics. We report five complementary metrics. **Task Success Rate (TSR)** is the fraction of episodes whose final output matches the reference. **Clarification F1** treats each step as a binary decision—ask vs. act—against an oracle label that flags steps with missing critical slots, and reports the F1 of the agent's `Ask` actions. **Average Turns** measures interaction cost as the mean number of agent steps per episode. **Error Propagation Rate (EPR)** is computed by a deterministic, code-level procedure rather than human or LLM-judge inspection. For each failed episode ($\hat{y}_T \neq y^*$), let $\mathcal{S}^\star$ be the set of critical slots removed by the ambiguous-variant procedure (Section 4.1) and $v_s^\star$ the canonical user reply for slot s. We mark a step t as a *premature commitment* on s iff: (i) $z_t \in \{\texttt{Retrieve}, \texttt{Tool}, \texttt{Answer}\}$; (ii) ξ_t binds a value $\hat{v}_s$ to s, recovered by a typed slot-to-surface matcher (JSON-key lookup for tool and retrieve calls, slot-typed extractor for answers, with Unicode NFKC, case- and whitespace-normalisation, and unit-normalised equality for numeric and temporal slots); (iii) $\hat{v}_s \neq v_s^\star$ under that normalisation; and (iv) no earlier `Ask`–reply pair in τ_t has already produced $v_s^\star$. EPR is the fraction of failed episodes containing at least one such step. Slot inventories, matchers, and the exact normalisation rules are released with the code; the full procedure is given in Appendix A. **Expected Calibration Error (ECE)**

[1] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct, released under the Apache-2.0 license.

measures, on the same oracle ask/act labels, how well the predicted uncertainty $\|u_t\|_1$ aligns with the true need to ask. All metrics are reported with a single-sample decoding budget per episode and averaged over three random seeds. ECE uses the standard 15-bin formulation of Guo et al. (2017) on the binary ask/act target.

4.2 Main Results

We compare against five baselines spanning the three threads of Section 2: SFT, a vanilla supervised baseline; TOOLRL (Qian et al., 2025), outcome-based RL for tool use; ASKTOACT (Zhang et al., 2025), a self-correcting clarification framework; UPROP (Duan et al., 2025), which uses step-level uncertainty as an inference-time abstention gate; and ARPO (Dong et al., 2025), a recent agentic RL method with step-level credit assignment. Methods without a native `Ask` action (SFT, TOOLRL, ARPO) are augmented with a fixed Ask-emission template at inference so that Clarification F1 is well-defined. All systems share the same Qwen2.5-7B-Instruct backbone, training data, and inference budget; EPR is computed by the same deterministic procedure (Appendix A) for all rows.

Table 1 reports task success rate on each of the three categories, together with clarification F1, average turns, and EPR. AoA-RL achieves the strongest average TSR (76.7, $+5.5$ over ARPO) and improves on all three categories simultaneously, with the largest gains on multi-turn tool use ($+5.4$) and on the held-out WebArena OOD benchmark ($+5.9$); the latter is achieved without any WebArena trajectories at training time, suggesting that the learned ask-or-act policy transfers across domains rather than memorising in-distribution slot patterns. Crucially, AoA-RL drops EPR from 19.8 to 13.6 while using fewer turns (3.27) than every clarification-aware baseline—the gain comes from acting on better information rather than acting more aggressively. The Clarification F1 gap ($+13.3$ over ASKTOACT, $+8.9$ over UPROP) mirrors the long-standing observation that deciding *when* to clarify is harder than generating a clarifying utterance (Aliannejadi et al., 2019).

4.3 Robustness and Calibration

A natural worry about any clarification-aware agent is that its gains might come from easy cases where asking is cheap. To test this, we stratify the evaluation set by ambiguity tier and report TSR within each tier (Table 2, left). The pattern is clear: every method degrades as ambiguity increases, but AoA-RL degrades the least. The gap to ARPO widens from $+1.9$ on low-ambiguity inputs to $+7.8$ on high-ambiguity inputs —the regime where acting on incomplete information is most likely to derail the trajectory. The improvement therefore does not come from sharpening already-easy decisions; it comes from *recovering trajectories* that other agents lose at the first step.

A second question is whether the uncertainty signal u_t that drives the ask-or-act choice is actually well-calibrated, or whether the gains could be obtained with any auxiliary score. We evaluate $\|u_t\|_1$ as a binary predictor of the oracle ask/act label and report Expected Calibration Error (ECE) and Brier Score in Table 2 (right). AoA-RL achieves the lowest ECE (0.091) and Brier (0.138) by a clear margin, including over UPROP, which uses uncertainty only at inference time. Together, the two halves of Table 2 suggest that the hierarchical reward not only teaches the policy *when* to ask but also reshapes the underlying uncertainty estimate so that it carries genuine decision-relevant information.

4.4 Ablation Studies

To isolate the contribution of each component of AoA-RL, we ablate the four design elements that distinguish it from a generic agentic-RL recipe: the structured uncertainty signal, the uncertainty-reduction reward, the interaction-cost penalty, and the explicit `Ask` action. We additionally compare against an SFT-only variant trained on the same trajectories to verify that RL itself is necessary, not just the trajectory schema. Results are reported in Table 3.

Three observations stand out. First, removing the `Ask` action causes the largest drop in average TSR (-5.9) and the largest jump in EPR ($+8.1$); without a way to pause and gather information, the agent has no choice but to commit to whichever action looks best under partial observability, and the resulting failures cascade. Second, removing the uncertainty-reduction reward is the largest hit on Clarification F1 among the RL ablations (-10.3): the agent retains the ability to ask, but loses the ability to tell useful clarifications from redundant ones, confirming that the indirect value of asking cannot be learned from the terminal reward alone. The SFT-only variant suffers a comparable F1 drop (-10.8) for a different reason—it never

Method	Task Success Rate (TSR) ↑			Avg. TSR ↑	Clarif. F1 ↑	Avg. Turns ↓	EPR ↓
	Tool Use	Web Nav.	Retrieval				
SFT	63.8	58.9 ±2.1	66.7	63.1	41.5	3.10	27.6
TOOLRL (Qian et al., 2025)	67.9	62.8 ±1.8	70.4	67.0	46.9	3.42	24.1
ASKTOACT (Zhang et al., 2025)	69.1	63.5 ±1.9	71.6	68.1	55.4	4.08	22.7
UPROP (Duan et al., 2025)	70.3	65.1 ±1.7	72.5	69.3	59.8	3.76	20.9
ARPO (Dong et al., 2025)	72.4	67.0 ±1.6	74.1	71.2	57.6	3.58	19.8
AoA-RL (ours)	**77.8**	**72.9** ±1.5	**79.3**	**76.7**	**68.7**	**3.27**	**13.6**

Table 1: **Main results** on three categories of agentic tasks. AoA-RL improves average task success by 5.5 points over ARPO while reducing error propagation (EPR) from 19.8 to 13.6 and lowering interaction cost (Avg. Turns) below every clarification-aware baseline. All results are reported as means over three random seeds. Best results in each column are bold. All numbers are percentages (including EPR) except Avg. Turns. ↑/↓ indicate higher/lower is better.

Method	TSR by ambiguity ↑			Calibration	
	Low	Med.	High	ECE ↓	Brier ↓
SFT	73.5	62.1	48.6	0.182	0.211
TOOLRL	76.2	65.8	52.9	0.164	0.198
ASKTOACT	77.0	68.9	57.3	0.151	0.186
UPROP	78.4	70.5	59.8	0.129	0.169
ARPO	80.1	72.4	61.7	0.137	0.176
AoA-RL (ours)	**82.0**	**77.8**	**69.5**	**0.091**	**0.138**

Table 2: **Robustness and calibration.** Left: task success rate (%) stratified by ambiguity tier. AoA-RL's lead grows with ambiguity, from +1.9 at low to +7.8 at high ambiguity over the strongest baseline. Right: calibration of the structured uncertainty $\|u_t\|_1$ as a predictor of the oracle ask/act label. Best in each column is bold; ↑/↓ indicate higher/lower is better.

Variant	TSR ↑	Clar. F1 ↑	Turns ↓	EPR ↓
Full model	**76.7**	**68.7**	**3.27**	**13.6**
w/o uncertainty signal	73.8	61.2	3.34	17.8
w/o r_t^{unc}	72.9	58.4	3.29	18.9
w/o interaction cost	75.1	66.8	4.02	15.4
w/o Ask action	70.8	–	2.91	21.7
SFT instead of RL	71.4	57.9	3.49	20.6

Table 3: **Ablation of the AoA-RL framework.** Removing the Ask action and removing the uncertainty-reduction reward (r_t^{unc}) each cause large drops, but in different metrics: the former in Avg. TSR and EPR, the latter in Clarification F1. Clarification F1 is undefined (–) when no Ask action is available; for that row, EPR reduces to conditions (i)–(iii) of Appendix A since no Ask–reply pair can exist. ↑/↓ indicate higher/lower is better; best in each column is bold.

receives any RL credit at all. Third, removing the interaction-cost penalty barely changes TSR but inflates Avg. Turns from 3.27 to 4.02, indicating that without an explicit cost the policy drifts toward over-cautious questioning even when it is not paying off in task success. SFT-only training, finally, recovers only a fraction of the full system's gains, confirming that demonstration trajectories provide a reasonable initialisation but cannot by themselves optimise the long-horizon ask-or-act trade-off. Additional ablations on the reward composition and on cross-task consistency are reported in Appendix C.

5 Conclusion

We introduced AoA-RL, a reinforcement learning framework that trains tool-using language agents to optimise an explicit ask-or-act decision under partial observability. Two design choices distinguish it from prior agentic RL: clarification is treated as a first-class action driven by structured, task-relevant uncertainty, and the delayed task reward is decomposed into uncertainty-reduction, task-completion, and interaction-cost terms, allowing indirectly useful actions to receive credit. Across multi-turn tool use, web navigation, and information retrieval, AoA-RL improves task success and reduces error propagation, with the largest gains on highly ambiguous inputs and consistently better-calibrated uncertainty than clarification-aware (ASKTOACT) and uncertainty-aware (UPROP) baselines. These results suggest that knowing *when not to act* is a learnable competence in its own right, and a useful target for future agentic training pipelines. We leave the study of robustness to noisy or adversarial clarification replies, transfer of u_t across backbones, and longer-horizon settings to future work; the corresponding constraints of our current evaluation are itemised in the Limitations section below.

Limitations

Our study has several limitations. First, the ambiguous variants used for both training and evaluation are constructed from existing benchmarks by removing critical slots; while this gives controlled comparisons across ambiguity tiers, the distribution of missing information may not fully match the ambiguities encountered in fully naturalistic user traffic. Second, clarifications in our environment are answered by a canonical, pre-collected user response; agents deployed against real users will face noisier, sometimes unhelpful answers, and the policy will need to remain robust when an `Ask` does not in fact reduce uncertainty. Third, our evaluation focuses on a single backbone size and a fixed step budget; how the ask-or-act trade-off scales with model capacity, with longer horizons, or with stronger external tools is left to future work. Finally, we evaluate on English-language tasks only, and generalising structured-uncertainty estimation to multilingual or speech-based interaction is an open question.

Ethics Statement

Our experiments are conducted on publicly available benchmarks that do not contain personally identifiable information, and the canonical user replies in the ambiguous-variant procedure are derived programmatically from existing benchmark annotations rather than collected from real users. The ask-or-act capability targeted by AoA-RL is intended to *reduce* premature, low-confidence actions in tool-using agents; we do not anticipate harms specific to this work beyond those already discussed in the broader literature on autonomous LLM agents. We caution that the clarification policy is trained on idealised user replies and may behave less reliably when deployed against real users who provide noisy or unhelpful answers, as also noted in Limitations.

Use of AI Assistants

In line with the ACL Policy on the Use of AI Assistants, we disclose the following. Large language models were used as *writing aids* for grammar polishing, paraphrasing, and LaTeX formatting of selected paragraphs in this manuscript; all technical content, claims, experimental design, results, and analysis were authored, verified, and are owned by the human authors. AI assistants were also used for boilerplate code suggestions during implementation; all reported numbers come from code that was reviewed and executed by the authors. We did not use AI assistants to generate or label any of the data on which experimental results are computed.

References

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Emmanouil Koukoumidis, Dilek Hakkani-Tür, and Gokhan Tur. 2025. Can a single model master both multi-turn conversations and tool use? coalm: A unified conversational agentic language model.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 475–484. ACM.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. 2024. Star-gate: Teaching language models to ask clarifying questions.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025. Agentic reinforced policy optimization.

Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. 2025. Uprop: Investigating the uncertainty propagation of llms in multi-step agentic decision-making.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Clam: Selective clarification for ambiguous questions with generative language models.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, and 1 others. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. volume 37, pages 54463–54482.

Panagiotis Lymperopoulos and Vasanth Sarathy. 2025. Tools in the loop: Quantifying uncertainty of llm question answering systems that use tools.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Xuefeng Du, Hamed Hassani, Paul Bogdan, Dawn Song, and Sharon Li. 2026. Uncertainty quantification in llm agents: Foundations, emerging challenges, and opportunities.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024. Taskbench: Benchmarking large language models for task automation. volume 37, pages 4540–4574.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2023. The rise and potential of large language model based agents: A survey.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Peijie Yu, Wei Liu, Yifan Yang, Jinjian Li, Zelong Zhang, Xiao Feng, and Feng Zhang. 2026. Benchmarking llm tool-use in the wild.

Xuan Zhang, Yongliang Shen, Zhe Zheng, Linjuan Wu, Wenqi Zhang, Yuchen Yan, Qiuying Peng, Jun Wang, and Weiming Lu. 2025. Asktoact: Enhancing llms tool use via self-correcting clarification.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

A EPR Computation Details

This appendix gives the full operational definition of the Error Propagation Rate (EPR) used in Section 4.2.

Inputs. For each failed episode ($\hat{y}_T \neq y^*$) we have: the rolled-out trajectory $\tau = (o_1, a_1, \ldots, o_T, a_T)$, the set $\mathcal{S}^\star$ of slots removed by the ambiguous-variant procedure, and the canonical user reply $v_s^\star$ for each $s \in \mathcal{S}^\star$.

Typed slot-to-surface matcher. For each $s \in \mathcal{S}^\star$, the matcher $\mathrm{Match}_s(\xi)$ returns the value bound to s inside an action content ξ, or $\perp$ if s is not consumed:

- **Tool / Retrieve:** ξ is parsed as a JSON object and Match_s returns the value at the tool-schema key bound to s, after type-aware normalisation: Unicode NFKC, lower-casing and whitespace collapsing for strings; unit-normalised equality for numeric and temporal slots; ISO-8601 canonicalisation for dates.
- **Answer:** Match_s runs a slot-typed extractor (regex for dates and numbers, gazetteer for entities, span tagger for free text) over ξ and returns the first matching span under the same normalisation.

Premature-commitment predicate. Step t is a premature commitment on slot s iff

1. $z_t \in \{\texttt{Retrieve}, \texttt{Tool}, \texttt{Answer}\}$;
2. $\hat{v}_s := \mathrm{Match}_s(\xi_t) \neq \perp$;
3. $\hat{v}_s \neq v_s^\star$ under the matcher's normalisation; and
4. no earlier step $t' < t$ contains an `Ask` whose environment reply yields $v_s^\star$.

Aggregation. EPR is the fraction of failed episodes containing at least one premature commitment. The same code path is used for AoA-RL and all baselines.

B Dataset Details

This appendix describes how the three task suites used in Section 4 are constructed from existing public benchmarks, and how their ambiguous variants are derived.

Multi-turn tool use. We build on the ambiguous-variant construction pipeline of **ASKTOACT** (Zhang et al., 2025), applied to two existing tool-learning corpora: **xlam-function-calling-60k** (Liu et al., 2024), which contains 60,000 function-calling samples over 3,673 executable Python/REST APIs spanning 21 functional categories, and **TaskBench** (Shen et al., 2024), which covers three application domains (Hugging Face tools, multimedia tools, and Daily-Life APIs) with 103 APIs and ∼28k samples. We sample 10,000 instances from xlam-function-calling-60k and 3,800 from TaskBench, split 8:1:1 into train/val/test (∼11k/1.4k/1.4k total) and chain consecutive single-turn calls into multi-step trajectories when the original example specifies inter-tool dependencies. xlam-function-calling-60k is released by Salesforce AI Research under the **CC BY 4.0** license; TaskBench is released under the **Apache-2.0** license; the ASKTOACT data-construction pipeline is released by the authors. Following the ASKTOACT recipe, we treat the original fully specified queries as the *information-sufficient* variants and the parameter-removed queries as the *information-insufficient* variants, replaying the held-out parameter values as the canonical user reply when the agent issues an `Ask`.

Web navigation (out-of-distribution). For navigation tasks we use the full **WebArena** test suite (Zhou et al., 2024) as an out-of-distribution benchmark: WebArena is an evaluation-only benchmark of long-horizon web trajectories situated in fully functional, self-hosted websites (e-commerce, social forum, software development, content management, and map domains), comprising 812 tasks in total. Each instance pairs a user goal with the DOM state of a starting page and a reference sequence of clicks/types. WebArena is released by CMU under the **Apache-2.0** license; we use the official release. **No method evaluated in this paper (ours or baselines) is trained on WebArena trajectories**; all systems are trained on the tool-use and retrieval splits described in Sections B above and evaluated zero-shot on WebArena. We construct ambiguous variants by removing one of (i) the navigation target's identifying attribute, (ii) a required filter, or (iii) the starting subpage, and record the removed information as the user's follow-up reply when the agent issues an `Ask`.

Information retrieval. For retrieval we build on **WebGPT** (Nakano et al., 2021), in which a language agent answers ELI5-style long-form questions by issuing `Search` and `Click` actions in a text-mode browser and collecting `Quote` citations before composing a final answer. We sample 5,000 questions from the public ELI5 question split released with the WebGPT comparisons dataset (**MIT** license), split 8:1:1 into train/val/test (4,000:500:500). Because the original WebGPT browser environment was not publicly released, we approximate it with an offline document store built from a static Wikipedia snapshot together with cached retrieval results released with the comparisons dataset; `Search` and `Click` actions are served from this offline store rather than from a live web API, which keeps the environment deterministic and reproducible. The ambiguous variants here remove either the temporal scope, the entity referent, or the source restriction of the question, all of which materially affect the retrieval trajectory; the removed information is again stored as the canonical user reply.

Ambiguity tiers. Each evaluation instance is assigned a tier based on how many critical slots are missing from the user instruction: *low* (zero or one non-critical slot missing), *medium* (exactly one critical slot missing), and *high* (two or more critical slots missing, or a missing slot together with an ambiguous referent). The information-sufficient instances are placed in the *low* tier. Tier assignments are produced by a rule-based pass over the slot-removal procedure above and verified on a held-out sample.

Splits and sizes. For each task category we use the standard train/validation/test splits of the underlying benchmark and apply the ambiguous-variant procedure to all three splits. Ambiguous and original instances appear in roughly a 1:1 ratio in training. All hyperparameters are selected on the validation split; numbers reported in the main paper are computed on the held-out test split. Concrete sizes are summarised in Table 4. The test split is further stratified by ambiguity tier: aggregated across the

three task categories, the *low* tier contains 1,085 instances (40%), *medium* 977 (36%), and *high* 650 (24%).

Task category	Train	Val	Test
Multi-turn tool use	11,000	1,400	1,400
Web navigation (OOD)	–	–	812
Information retrieval	4,000	500	500
Total	15,000	1,900	2,712

Table 4: **Dataset sizes** after the ambiguous-variant procedure is applied. Multi-turn tool use combines xlam-derived and TaskBench-derived splits. **Web navigation is evaluation-only**: the full WebArena test suite (812 tasks) is held out as an out-of-distribution benchmark—no method is trained on WebArena trajectories. Ambiguous and information-sufficient instances appear in roughly a 1:1 ratio in the training splits.

User simulator. When the agent issues an `Ask`, the environment returns the canonical user reply collected during slot removal. We do not simulate noisy or adversarial user replies in the main experiments; this idealised setting is a deliberate choice to isolate the ask-or-act decision from user-simulation noise, and its limitations are discussed in the Limitations section.

C Additional Ablations

C.1 Reward Composition Ablation

To further isolate the contribution of each reward term in the hierarchical decomposition of Equation 8, we sweep three reward configurations: (i) the terminal task reward only; (ii) task reward augmented with the uncertainty-reduction term r_t^{unc}; and (iii) the full hierarchical reward of Equation 8, which adds the interaction-cost penalty. We report Avg. TSR, Clarification F1, Avg. Turns, and a composite Utility score $\text{Utility} = \text{TSR} - \lambda_1 \cdot \text{Turns} - \lambda_2 \cdot \text{ToolCalls}$ that captures the effect of interaction cost on overall helpfulness.

The trend is consistent with the design intent of the reward. Adding r_t^{unc} on top of the terminal task reward improves Clarification F1 from 58.4 to 66.1, confirming that the uncertainty-reduction term provides the dense credit needed to learn when an `Ask` action is informative. However, the policy at this point becomes slightly more talkative (Avg. Turns rises from 3.29 to 3.51). Adding the interaction-cost term restores Avg. Turns to 3.27 and yields the best Utility, indicating that r_t^{unc} and r_t^{cost} act as complementary signals: one rewards information gain, the other suppresses gratuitous turns.

Reward setting	TSR ↑	Clar. F1 ↑	Turns ↓	Utility ↑
Task only	72.9	58.4	3.29	66.3
+ r_t^{unc}	75.6	66.1	3.51	68.9
+ r_t^{unc} + r_t^{cost}	**76.7**	**68.7**	**3.27**	**71.4**

Table 5: **Reward composition ablation.** Adding the uncertainty-reduction term yields the largest jump in clarification quality, while adding the interaction-cost term recovers a tighter Avg. Turns without sacrificing TSR.

Variant	Tool	Web	Retr.	Avg.
Full model	**77.8**	**72.9**	**79.3**	**76.7**
w/o uncertainty signal	74.2	69.6	77.5	73.8
w/o uncertainty-reduction reward	73.5	68.9	76.4	72.9
w/o `Ask` action	70.1	66.4	75.8	70.8

Table 6: **Cross-task consistency of ablations** (TSR, %). The three components benefit all task categories, with the largest absolute drops on multi-turn tool use and web navigation—tasks whose long, path-dependent trajectories make early uncertainty most costly.

C.2 Cross-Task Consistency of Ablations

Table 6 reports per-task TSR for the three most informative ablations of Section 4.4. The trend in the main ablation table is consistent across all three task categories, but the magnitude of the drop differs by task type.

The largest absolute drops come on multi-turn tool use and web navigation, where an early misstep is hard to recover from; the smallest drops come on information retrieval, where later retrieval calls can partially compensate for an underspecified earlier query. This pattern reinforces the claim of Section 4.3: the value of asking is highest precisely where trajectories are most path-dependent.