Exploring Agentic Tool-Calling Decisions via Uncertainty-Aligned Reinforcement Learning

Yijin Zhou[1,2,3], Linqian Zeng[3], Xiaoya Lu[1,3], Wenyuan Xie[3], Dongrui Liu[1], Junchi Yan [2,3], Jing Shao[1,†]
[1]Shanghai Artificial Intelligence Laboratory, China
[2]Shanghai Innovation Institute, China
[3]Shanghai Jiao Tong University, China
yijin.zhou2002@gmail.com, shaojing@pjlab.org.cn

Abstract

Large language model (LLM)-based agents often make suboptimal tool-use decisions, including unsupported tool invocation and hallucinated direct responses, which may accumulate errors throughout multi-step interactions. Existing approaches mainly improve these behaviors through inference-time correction or coarse-grained reward signals based on decision outcomes and structured checklists, leaving the uncertainty characteristics of agent decisions underexplored. We observe that decision-oriented reinforcement learning tends to weaken the uncertainty separation between correct and incorrect actions, resulting in overconfident mistakes and weaker exploration signals. Therefore, we propose TRUST, which incorporates uncertainty quantification into reward design as a repulsive force for maintaining uncertainty separation, and labels lightweight key-turn annotations for unified post-training of multi-turn trajectories. Experimental results across diverse tool-use benchmarks show that TRUST consistently enhances both decision quality and agent performance while maintaining more reliable uncertainty estimates during optimization. Code is released under https://github.com/yjzscode/TRUST.

1 Introduction

Large language model (LLM)-based agents extend the capabilities of language models by invoking external tools for knowledge retrieval, computation, and interaction with real-world environments (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024). However, agents frequently exhibit tool-calling decision failures at specific action turns, either invoking tools when tool use is unsupported or unnecessary, or fabricating direct answers without issuing the required tool call (Ross et al., 2025; Healy et al., 2026). Unlike conventional textual hallucinations in LLMs, tool-calling decision failures can corrupt intermediate states and propagate errors across subsequent interaction turns, thereby incurring financial costs, execution failures, and information leakage in real-world agentic tasks (Lin et al., 2025; Su et al., 2025; Zhang et al., 2025).

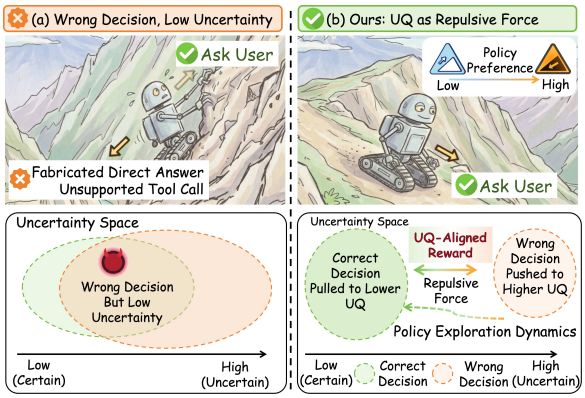

Figure 1: Comparison between tool-calling decision failure in wrong decision but low uncertainty, and our TRUST solution.

Recent studies address tool-calling decision failures by extracting internal uncertainty signals or structured reasoning patterns during inference (Zhang et al., 2026b; Stoisser et al., 2025). However, they primarily focus on post-hoc intervention in inference, instead of improving the agent's intrinsic decision capability through policy optimization. Research efforts by Ross et al. (2025); Suri et al. (2025) mitigate such failures through reinforcement learning (RL) for tool-calling decisions, typically relying on coarse-grained reward formulations from decision classification or rule-based checklists. Nevertheless, these approaches lack a quantitative analysis of the policy exploration dynamics underlying tool-calling decision optimization, leaving substantial room for improvement.

From the perspective of **U**ncertainty **Q**uantification (UQ) of model responses, we observe that existing RL-based approaches for

[†]Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 10734–10756
October 24-29, 2026 ©2026 Association for Computational Linguistics

optimizing tool-calling decisions tend to increase confidence in sampled high-reward actions, while unintentionally weakening the uncertainty separation between correct and incorrect decisions. As illustrated in Fig. 1(a), unsupported tool calls and unjustified direct answers increasingly overlap with low-uncertainty regions after decision-oriented RL optimization, impairing the model's original calibration that higher uncertainty should indicate a greater likelihood of incorrect decisions. Take Qwen3-4B-Thinking (Yang et al., 2025) as an example: the proportion of "Wrong Decision But Low Uncertainty" cases rises from 34.50% to 70.21% after RL optimization for tool-calling decisions. This observation suggests that existing decision-oriented objectives primarily optimize action correctness without preserving the uncertainty structure underlying those decisions. Consequently, overconfident but incorrect actions receive limited optimization pressure, reducing policy exploration toward more reliable alternatives.

To address this issue, we propose **T**ool-calling decision **R**eward with **U**ncertainty-**S**eparated post-**T**raining (TRUST), a framework that leverages *UQ as a repulsive force* within the reward, rather than using uncertainty merely as a post-hoc diagnostic. Specifically, we introduce a UQ-aligned reward that jointly models action correctness and a certainty margin derived from the uncertainty gap between negative and ground-truth decisions. As shown in Fig. 1(b), our reward encourages the model to assign lower uncertainty to correct decisions while maintaining comparatively higher uncertainty for incorrect or counterfactual decisions. This mechanism promotes exploration away from uncertain or unreliable decisions and provides a stronger optimization signal for policy updates.

Furthermore, we extend this training paradigm from standalone decision instances to multi-turn agent trajectories. Instead of exhaustively relabeling entire trajectories, we annotate lightweight key-turn decision points, enabling unified post-training for both trajectory-level task success and turn-level tool-calling calibration. The trajectory-level outcome reward supervises overall task completion and tool execution quality, while our turn-level UQ-aligned reward explicitly calibrates the timing and appropriateness of tool-calling decisions.

Empirically, TRUST yields substantial improvements across diverse tool-use benchmarks while preserving the uncertainty structure underlying agent decisions. In turn-level tool-calling decision optimization, TRUST improves by over 11% on the When2Call task accuracy, simultaneously strengthening performance in complex multi-turn interactions and tool-use trajectories. Cooperated with the trajectory-level tool-calling post-training, TRUST outperforms 6.33% on BFCL-V4 and 7.07% on ToolSandbox. The gains are particularly pronounced in challenging decision-intensive scenarios, including Multi-Turn and Irrelevance on BFCL-V4, as well as scenarios such as multiple user turns and distraction tools in ToolSandbox. Overall, TRUST jointly optimizes overall tool execution quality and turn-level decision appropriateness of tool use, providing stronger optimization signals for policy learning and enabling more reliable multi-turn agent behavior.

2 Related Work

Uncertainty quantification for language agents. UQ has been widely studied for estimating the reliability of language model outputs (Kadavath et al., 2022; Lin et al., 2022). Early work mainly focuses on output-level uncertainty through model probabilities, verbalized confidence, and sampling-based consistency signals (Kadavath et al., 2022; Lin et al., 2022; Manakul et al., 2023; Kuhn et al., 2023; Manakul et al., 2023). More recently, researchers argue that uncertainty in agentic systems extends beyond final responses to intermediate actions, environmental observations, and multi-step trajectories (Kirchhof et al., 2025; Duan et al., 2025; Oh et al., 2026; Zhang et al., 2026b; Stoisser et al., 2025; Lymperopoulos and Sarathy, 2025). Several approaches further exploit uncertainty to regulate agent behavior, such as triggering clarification under ambiguous instructions, controlling memory and reflection, or guiding exploration through structured rewards (Suri et al., 2025; Zhang et al., 2026b,a). Different from prior work that primarily treats uncertainty as a post-hoc signal for diagnosis or behavior control, we explicitly integrate uncertainty into policy optimization and align decision correctness with model certainty through uncertainty-aware rewards.

Tool-calling decision learning and optimization. Tool-augmented LLMs have been extensively studied for tool selection, argument generation, and large-scale API utilization (Schick et al., 2023; Qin et al., 2024; Patil et al., 2025; Lu et al., 2025). Beyond execution correctness, recent work increas-

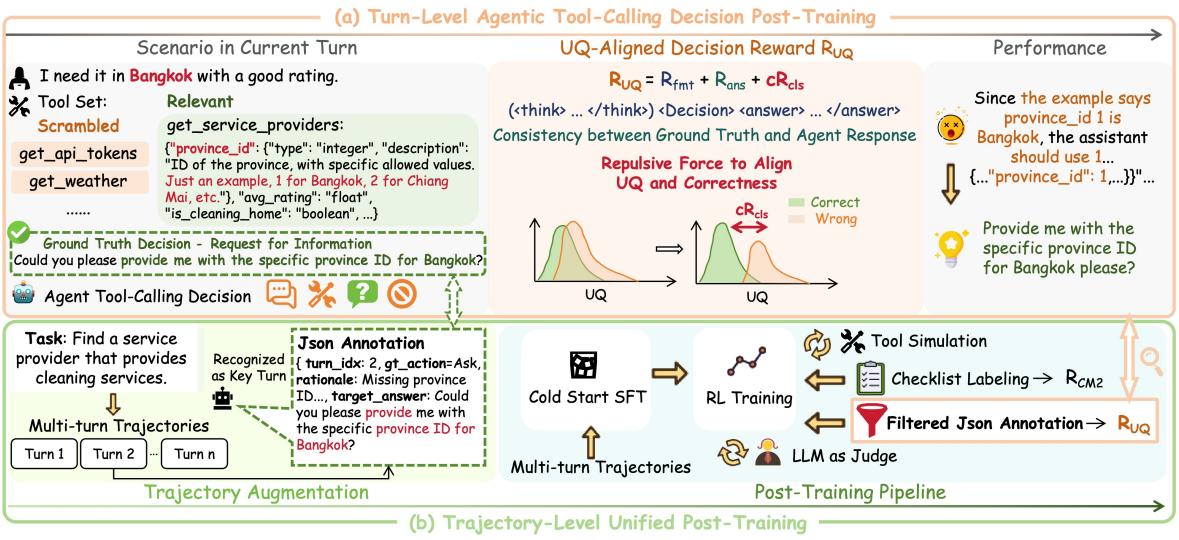

Figure 2: The overview of TRUST. It consists of two components: (a) A turn-level UQ-aligned reward uses uncertainty as a repulsive signal to align decision correctness and confidence. (b) A trajectory-level unified post-training framework augments key turns with decision annotations and integrates R_{UQ} with task-level rewards for joint optimization of execution quality and tool-calling decisions.

ingly focuses on the preceding decision process: whether an agent should invoke tools, ask follow-up questions, answer directly, or abstain under current observations. Benchmarks and analyses such as When2Call and related studies isolate this decision layer and reveal common failures, including unnecessary tool invocation and hallucinated direct answers (Ross et al., 2025; Wang et al., 2025; Wu et al., 2026; Sun et al., 2026). Concurrent work further explores RL-based optimization of tool-calling behavior using calibrated rewards or decision supervision (Suri et al., 2025; Modecrua et al., 2026; Zhong et al., 2026). In contrast to these approaches that mainly optimize decision correctness, our work studies how RL reshapes the uncertainty structure of tool-calling policies and leverages it to improve both decision quality and calibration.

3 TRUST Method

3.1 Problem Formulation

Given the state $\mathcal{S}$, the next-action decision is a four-way agentic action space (Ross et al., 2025):

$$\mathcal{A} = \{\textsc{Direct}, \textsc{Tool}, \textsc{Ask}, \textsc{Unable}\}, \quad (1)$$

where DIRECT means direct answer, TOOL means invoking an external tool, ASK means requesting missing information from the user, and UNABLE means admitting that the request can not be answered. Specifically, at decision turn t, the agent state is $s_t = (h_t, \mathcal{T})$, where $h_t = (o_1, a_1, ..., o_t, a_t)$ denotes the observed trajectory, and $\mathcal{T}$ denotes the available tool set with schemas. A policy $\pi_\theta : \mathcal{S} \to \mathcal{Z}$ maps from state to agent response, generating a structured response $z_t = (a_t, y_t)$, where $a_t \in \mathcal{A}$ and $y_t \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of surface realizations. For TOOL, y_t is a tool-call payload with a tool name and arguments; otherwise, y_t is natural language. The environment updates the state after tool execution or user interaction.

Each decision point provides a ground-truth pair $z_t^\star = (a_t^\star, y_t^\star)$. A tool-calling decision failure occurs when the selected action is unsupported by the current state, *i.e.* $a_t^\star \in \mathcal{A}/\{\textsc{Tool}\}, a_t = \textsc{Tool}$ or $a_t^\star = \textsc{Tool}, a_t = \textsc{Direct}$. Thus, the training objective is not only to increase the probability of correct final trajectories, but also to calibrate the policy over intermediate decisions:

$$\max_\theta \ \pi_\theta(a_t^\star, y_t^\star \mid s_t), \ \min_\theta \ \pi_\theta(a_t^-, y_t^- \mid s_t) \quad (2)$$

for unsupported or counterfactual decisions $z_t^- = (a_t^-, y_t^-)$. In our experiments, this policy is optimized with GRPO post-training. For each state s, the current model samples a group of responses $\mathcal{Z}^G = \{z_i\}_{i=1}^G$, and the policy is updated according to their relative rewards.

3.2 Turn-level UQ-Aligned Decision Reward

Directly rewarding the selected action can not make the agent's uncertainty better calibrated, as shown in Fig. 3. So we use UQ as the repulsive force in

reward as illustrated in Fig. 2(a). We instantiate uncertainty with sequence perplexity (Kuhn et al., 2023). Given a prompt state s and a candidate response $z = (a, y)$, the perplexity is

$$\mathrm{PPL}_\theta(z \mid s) = \exp\left(-\frac{1}{|z|}\sum_{j=1}^{|z|}\log p_\theta(z_{(j)} \mid s)\right), \quad (3)$$

where $z_{(j)}$ is the j-th sequence in z, p_θ calculates the average value of tokens in $z_{(j)}$.

For each decision point, we compare the perplexity of the ground-truth decision with that of a negative decision:

$$m(s) = \mathbb{E}_{\mathcal{Z}^G}(\mathrm{PPL}_\theta(z^- \mid s)) - \mathbb{E}_{\mathcal{Z}^G}(\mathrm{PPL}_\theta(z^\star \mid s)), \quad (4)$$

where $\mathbb{E}_{\mathcal{Z}^G}(\mathrm{PPL}_\theta(z^- \mid s)) = 1$ if $\{(a^\star, \cdot) \in \mathcal{Z}^G\} = \emptyset$. The margin is converted into a bounded certainty coefficient,

$$c(s) = \sigma\left(\frac{m(s)}{\tau}\right), \quad (5)$$

where $\tau = 0.1$ is a temperature and $\sigma(\cdot)$ is the sigmoid function. A large margin means that the model assigns lower perplexity to the correct decision than to the negative decision. Conversely, a small or negative margin indicates that uncertainty is not aligned with decision correctness.

The reward combines format validity, decision correctness, and the UQ margin:

$$R_{\mathrm{UQ}}(z) = R_{\mathrm{fmt}}(z) + R_{\mathrm{ans}}(z, z^\star) + cR_{\mathrm{cls}}(a, a^\star). \quad (6)$$

Here, R_{fmt} rewards structured output format (<think> ... </think>)<$\mathcal{A}$><answer>$\mathcal{Y}$</answer> and internal consistency between the suggested action a and final answer y. n trajectory-level unified training, R_{fmt} is set to 0. $R_{\mathrm{ans}}(z, z^\star)$ rewards consistency between z and z^*. R_{cls} rewards action correctness and certainty coefficient c itself:

$$R_{\mathrm{cls}}(a, a^\star) = 1 + \begin{cases} 2 & , a^* = a, \\ 1 & , \{a, a^*\} = \{\textsc{Direct}, \textsc{Tool}\} \text{ or } \{\textsc{Ask}, \textsc{Unable}\}, \\ 0 & , \text{otherwise.} \end{cases} \quad (7)$$

Exact action match receives the highest R_{cls}, while a weaker partial reward is assigned when the prediction preserves the coarse decision direction between action and non-execution. R_{UQ} can be obtained without a LLM judger if using structured output format (<think> ... </think>)<$\mathcal{A}$><answer>$\mathcal{Y}$</answer>; otherwise, it needs a lightweight LLM judger to judge the action $a \in \mathcal{A}$. Overall, cR_{cls} gives a repulsive force in R_{UQ}, to push wrong decisions to higher UQ and pull correct decisions to lower UQ, driving policy expands the exploration space in cases of incorrect decisions, and makes it easier to sample the right ones.

3.3 Unified Post-Training for Trajectory-Level Performance and Turn-Level Decision

Standalone decision examples provide clean supervision for when an agent should answer, call tools, ask the user, or stop. However, realistic agent failures often arise inside multi-turn trajectories, where an early unsupported decision can corrupt later states. Therefore, it is of great significance to unify the classical task performance-driven rewards and the next-action decision rewards for RL training. To realize this unification, we augment CM2 trajectories (Zhang et al., 2026c) with lightweight tool-call decision annotations at key turns, and plug in our R_{UQ} with the CM2 checklist reward R_{CM2}.

Trajectory augmentation with tool-call decision annotation. As shown in Fig. 2(b), we construct trajectory-level decision supervision from trajectories without relabeling the full conversation, only augmenting each trajectory with a small set of key-turn decision annotations.

A labeling model reads the full payload and selects no more than 2 decision-critical turns where a next-action $\mathcal{A}$ signal is useful. Each selected turn receives exactly one label from the action space $\mathcal{A}$. To reduce the annotation bias of trajectories toward tool execution TOOL, the prompt for the labeling model includes running label-count statistics and requests to prefer underrepresented valid actions when multiple turns are equally useful.

The annotation output is a strict JSON object. Each annotation has the following schema

$$\begin{aligned} \texttt{annotation} = \{ & \\ & \texttt{turn_idx} : \text{key turn idx}, \\ & \texttt{gt_action} = a^* \in \mathcal{A}, \\ & \texttt{rationale} : \text{Reason for } \texttt{gt_action}, \\ & \texttt{target_answer} = y^* \in \mathcal{Y} \\ \} & \end{aligned} \quad (8)$$

which is used by the mixed reward manager during GRPO post-training. For each annotated turn, the trajectory prefix up to the user message defines the state s_t, the annotated label defines $a_t^\star$, and the annotated target answer defines $y_t^\star$.

Reward combination for unified RL. To unify the next-action decision with the task completion

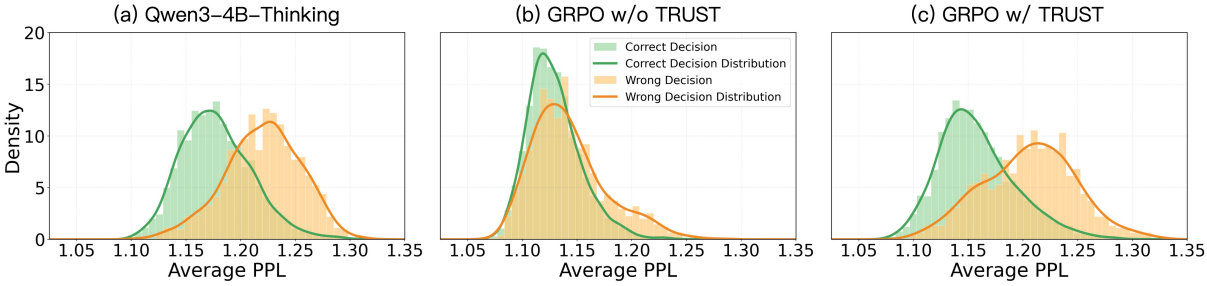

Figure 3: Uncertainty calibration of tool-calling decisions on When2Call. Green denotes correct decisions, and orange denotes wrong decisions. Lower PPL means higher certainty. Compared with Qwen3-4B-Thinking and decision GRPO training without TRUST, training with TRUST yields a clearer separation between correct and wrong decisions, assigning lower PPL to correct decisions and higher PPL to wrong ones.

and accuracy aggregation in one post-training process, trajectories are optimized with a mixed reward as shown in Fig. 2(b). The original checklist reward R_{CM2} continues to supervise task completion and tool execution quality over the full trajectory. Our turn-level decision reward R_{UQ} is added sparsely at the annotated turn end positions:

$$R = R_{\text{CM2}} + \sum_{t \in \mathcal{K}} R_{\text{UQ}}(z_t), \tag{9}$$

where $\mathcal{K}$ is the set of annotated key turns. This design preserves the original CM2 objective for execution quality while adding a targeted next-action decision signal for tool-call timing and action appropriateness. TRUST yields a lightweight trajectory-level training signal without relabeling entire conversations or redesigning the performance-driven rewards, realizing the unified training for task performance, efficiency for tool-calling, and tool-calling hallucination mitigation.

4 Experiments

4.1 Setup

Benchmarks. We conduct experiments on three benchmarks, namely When2Call (Ross et al., 2025) for turn-level when to call training and evaluation, and tool-use benchmarks ToolSandbox (Lu et al., 2025) and BFCL-V4 (Patil et al., 2025) to evaluate the multi-turn overall performance.

Baselines. Closed models MiniMax-M2.5 (MiniMax, 2026), GPT-4o-mini (Hurst et al., 2024), and open-source models Qwen3-235B-A22B, Qwen3-4B-Thinking, and Qwen3-8B-Thinking (Yang et al., 2025) are tested on three benchmarks for performance comparison of our post-trained model with TRUST. Moreover, we compare two representative UQ baselines for agentic tool-calling, AUQ (Zhang et al., 2026b) and SAGE-Agent (Suri et al., 2025), all implemented on Qwen3-4B-Thinking and Qwen3-8B-Thinking. Note that we do not report the results of the vanilla base model and training-free methods for Qwen3-8B-Base, since it is not instruction-tuned under this evaluation protocol (Zhang et al., 2026c). Instead, we provide the corresponding training-free experiments on Qwen3-8B-Thinking as informative references. For turn-level and trajectory-level post-training, GRPO is trained only by the tool-calling decision, and CM2 (Zhang et al., 2026c), from the same model checkpoint as TRUST, serves as a training-needed baseline. See Appendix A for further details of benchmarks and baselines.

Training setups. We select Qwen3-4B-Thinking and Qwen3-8B-Base as the agent backbone models (Yang et al., 2025) for group relative policy optimization (GRPO) post-training (Shao et al., 2024). *Qwen3-4B-Thinking* is trained on When2Call training dataset to validate the efficiency of calibrating *turn-level tool-calling decision* only, while *Qwen3-8B-Base* sees *a complete trajectory-level post-training*, including cold start SFT and the unified RL, following (Zhang et al., 2026c). In trajectory augmentation, we use Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025) as the labeler. In unified RL training, Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025) serves as an LLM judge. Prompts and more settings are in Appendix B.

4.2 TRUST Successfully Aligns Correctness and Uncertainty

To isolate whether the proposed reward calibrates the tool-calling decision itself, we train on When2Call training set and compare PPL distributions of correct and wrong decisions. As shown in Fig. 3(a), the original Qwen3-4B-Thinking al-

Table 1: Performance comparison of When2Call. FDAR is the False Direct Answer Rate. **Boldface** and underlining denote the best and second-best results, respectively, within each corresponding baseline group. The transparency intensity of green-colored cells is (Performance of TRUST) - (Performance of The Baseline with same "Training").

Model / Method	Model Base	Training	Acc Norm↑	F1↑	Tool Hall↓	FDAR↓
Closed-source Baselines						
MiniMax-M2.5	-	-	23.93	24.94	24.03	20.48
GPT-4o-mini	-	-	57.64	60.26	59.30	11.61
Claude-Sonnet-4	-	-	80.53	81.85	10.08	5.86
Open-source Baselines						
235B-A22B-Instruct	-	-	42.33	32.62	82.95	4.63
30B-A3B-Instruct	-	-	35.10	17.75	94.57	1.34
4B-Thinking	-	-	69.36	72.77	41.47	17.11
8B-Thinking	-	-	45.19	50.30	14.73	32.78
From Qwen3-4B-Thinking						
AUQ	4B-Thinking	Training-free	56.35	60.32	45.71	22.98
SAGE	4B-Thinking	Training-free	73.36	72.36	**0.00**	8.89
GRPO	4B-Thinking	Turn-level	72.46	76.06	5.73	24.76
Turn-level TRUST	4B-Thinking	Turn-level	**80.83**	**82.84**	17.83	**5.07**
From Qwen3-8B-Thinking						
AUQ	8B-Thinking	Training-free	34.34	37.91	37.21	36.99
SAGE	8B-Thinking	Training-free	**71.87**	**71.57**	**0.00**	11.04
From Qwen3-8B-Base						
Traj.-level CM2	8B-Base	Traj.-level	43.75	36.94	75.43	9.34
Traj.-level TRUST	8B-Base	Traj.-level	62.32	60.62	49.87	**8.39**

Table 2: Performance comparison of BFCL-V4.

Model / Method	Overall Score	Web Search	Memory	Multi-Turn	Live	Non-Live	Irrelevance
Open-source Baselines							
235B-A22B-Instruct	53.54	44.50	27.10	50.12	83.20	90.10	82.38
30B-A3B-Instruct	41.00	22.50	17.63	30.00	77.94	85.77	79.90
4B-Thinking	38.61	6.00	12.37	52.00	71.06	38.02	79.23
8B-Thinking	41.58	12.00	14.62	37.75	80.53	87.58	79.07
From Qwen3-4B-Thinking							
AUQ	35.21	3.00	20.43	23.50	62.40	77.08	81.37
SAGE	41.16	3.50	13.02	41.50	80.49	86.71	79.23
Turn-level GRPO	38.98	3.00	24.52	22.75	86.33	80.61	82.38
Turn-level TRUST	**48.04**	3.00	24.95	49.62	84.09	89.23	84.84
From Qwen3-8B-Thinking							
AUQ	29.08	14.00	15.70	10.00	75.42	84.46	40.14
SAGE	42.99	9.50	21.72	36.37	80.83	88.15	79.60
From Qwen3-8B-Base							
Traj.-level CM2	38.15	22.00	21.94	22.50	74.58	80.12	71.46
Traj.-level TRUST	**44.48**	27.00	30.57	29.88	75.57	82.27	79.33

ready exhibits a correlation between correctness and uncertainty, where correct decisions tend to have lower PPL than wrong ones, forming two distinct distributions. After GRPO decision training without TRUST, however, this separation becomes less reliable in Fig. 3(b). Wrong decisions are also assigned low PPL, indicating that direct optimization can make the model confident in both correct and incorrect decisions. In contrast, training with TRUST again produces a clearer uncertainty gap depicted in Fig. 3(c). Correct decisions concentrate in the lower-PPL region, while wrong decisions are shifted toward higher PPL.

We measure the distribution overlap in Fig. 3 by

$$\text{IoU} = \frac{\text{Correct} \cap \text{Wrong}}{\text{Correct} \cup \text{Wrong}}. \tag{10}$$

It yields an initial IoU of 34.50%, which increases to 70.21% following GRPO. Conversely, integrating our UQ-Aligned Reward leads to a reversion of IoU to 35.29% while delivering superior accuracy. This demonstrates that TRUST explicitly aligns the model's internal uncertainty with decision correctness, making erroneous tool-calling decisions easier to identify and less likely to be reinforced as confident actions. The uncertainty calibration can be generalized to a wide range of LLMs, as shown in Table 4.

4.3 Performance of TRUST on Turn-Level Tool-Calling Decision Post-Training

To evaluate the effectiveness of TRUST, we conduct turn-level GRPO and evaluation on Qwen3-4B-Thinking, where the results demonstrate substantial performance gains across all benchmarks.

First, on **When2Call** test set (Table 1), TRUST achieves the highest Acc Norm compared to various baseline methods, outperforming the runner-up by 7.47%, while yielding the low-

Table 3: Performance comparison of ToolSandBox benchmark across scenario categories and tool augmentations.

Model / Method	Overall Score	Scenario Categories							Tool Augmentations							
		STC	MTC	SUT	MUT	SD	C	II	0-DT	3-DT	10-DT	AT	TNS	TDS	ADS	ATS
Open-source Baselines																
235B-A22B-Instruct	69.88	62.46	62.68	63.85	59.48	62.68	57.63	90.98	73.84	73.44	74.98	70.75	74.85	70.99	74.73	74.79
30B-A3B-Instruct	61.57	62.08	60.07	62.88	54.09	62.34	57.24	58.45	64.68	62.12	62.25	61.72	63.61	64.94	63.54	63.55
4B-Thinking	52.89	57.68	38.09	43.15	38.18	53.03	28.71	90.82	58.71	54.75	53.21	52.61	53.82	53.64	56.40	60.54
8B-Thinking	59.74	64.32	44.79	50.16	44.06	61.55	35.76	94.93	59.05	62.95	65.13	60.74	62.35	61.08	65.23	64.00
From Qwen3-4B-Thinking																
AUQ	55.47	48.23	47.92	49.20	44.79	51.59	41.71	84.87	56.12	58.09	60.55	56.98	60.41	59.03	55.51	56.99
SAGE	52.54	48.22	46.93	49.14	42.04	54.38	40.99	71.06	50.14	56.03	54.33	51.60	60.46	51.49	55.90	55.45
Turn-level GRPO	39.37	46.92	21.15	28.67	19.04	23.37	19.05	95.98	43.36	45.57	40.93	38.86	44.28	37.90	41.82	43.71
Turn-level TRUST	**56.35**	51.18	46.26	52.83	37.68	55.59	42.27	94.44	57.42	58.53	59.05	58.68	57.66	56.90	58.02	58.77
From Qwen3-8B-Thinking																
AUQ	64.08	56.46	56.68	57.85	53.48	56.68	51.63	87.98	67.84	67.44	68.98	64.75	68.85	64.99	68.73	68.79
SAGE	59.72	54.85	51.56	53.95	47.56	59.41	43.68	83.48	59.70	65.72	60.27	58.29	62.97	66.61	62.22	65.52
From Qwen3-8B-Base																
Traj.-level CM2	61.21	70.57	58.55	64.22	51.93	56.64	53.70	55.60	60.12	65.04	65.02	65.03	61.80	61.30	64.14	64.51
Traj.-level TRUST	**68.28**	68.66	59.70	63.10	56.93	61.58	52.04	90.70	73.01	76.41	74.94	69.16	66.74	68.77	71.26	71.24

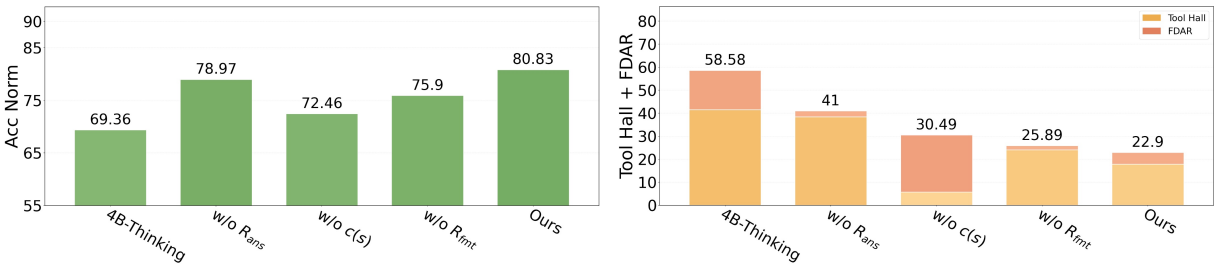

Figure 4: Ablation study of R_{UQ} on When2Call. Tool Hall + FDAR measures the overall tool-calling hallucination. The setting w/o $c(s)$ represents $c(s) = 1$, *i.e.* $R_{\text{UQ}} = R_{\text{fmt}} + R_{\text{ans}} + R_{\text{cls}}$.

est False Direct Answer Rate (FDAR) of 5.07%. Compared to the vanilla base model Qwen3-4B-Thinking, TRUST improves the Acc Norm by 11.47%. When compared directly against turn-level GRPO training, TRUST not only achieves an 8.37% absolute improvement in Acc Norm but also reduces the overall hallucination metric, defined as a sum of Tool Hallucination and FDAR, from GRPO's 30.49% down to 22.90%. Overall, TRUST surpasses most open-source and closed-source models, reaching a level comparable to the top-tier closed-source Claude-Sonnet-4.

Crucially, we discover that *optimizing tool-call decisions solely at turn level can directly catalyze and boost the generalized performance in complex multi-turn interactions and tool-use trajectories*. On **BFCL-V4** in Table 2, TRUST delivers an Overall Score of 48.04%, which represents a substantial 9.43% leap over the base model, completely dominates all baseline methods, and notably exceeds the performance of the much larger 30B-A3B-Instruct (41.00%). Specifically, TRUST exhibits a compelling advantage in dimensions like Multi-Turn (49.62%) and Irrelevance (84.84%), proving its high precision in handling multi-turn tool-calling and detecting irrelevant tools.

Similarly, on **ToolSandBox** in Table 3, TRUST establishes a state-of-the-art performance within its group, outperforming the base model by 3.46% and scoring over turn-level GRPO with a 16.98% margin. In contrast to TRUST, while turn-level GRPO has the highest performance in detecting Insufficient Information (II), it drives the agent towards a conservative tool-calling strategy, which affects the tool-calling and task completion capability. Furthermore, when subjected to various tool-level perturbations 3-DT and 10-DT as well as specialized robust evaluations (TNS, TDS, ADS, and ATS), TRUST maintains strong score stability and resilience. This firmly underscores the powerful cross-task benefits that precise turn-level decision training provides for long-dependency tool-chain workflows.

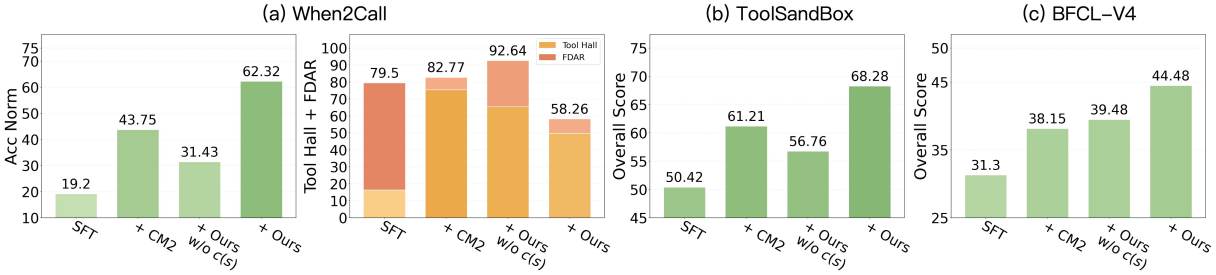

Figure 5: Ablation study results of the unified post-training across three evaluation benchmarks.

4.4 Performance of TRUST on Trajectory-Level Unified Post-Training

On **When2Call** in Table 1, TRUST demonstrates a commanding superiority over the trajectory-level RL baseline CM2. Specifically, TRUST advances the Acc Norm from 43.75% to 62.32% and the F1 score from 36.94% to 60.62%, marking a substantial leap in execution accuracy.

Furthermore, the experimental empirical results on multi-turn benchmarks confirm that TRUST, which applies GRPO on trajectories augmented with key turn decision annotations, utilizes tools more accurately and effectively. On **BFCL-V4** in Table 2, TRUST establishes an Overall Score of 44.48%, significantly outperforming Traj.-level CM2 (38.15%) and surpassing 30B-A3B-Instruct (41.00%). This outstanding performance is consistently mirrored in scenarios requiring higher agent capabilities, such as Web Search, Memory, and Multi-Turn dimensions. Similarly, on **ToolSandBox** in Table 3, TRUST delivers an Overall Score of 68.28%, outstripping the trajectory-level GRPO baseline by 7.07%. Remarkably, this score not only leaves the 30B model far behind but also closely approaches the performance of 235B-A22B-Instruct (69.88%). Moreover, it is highlighted that TRUST achieves an Insufficient Information (II) score of 90.70%, drastically mitigating the hallucination plagued by the GRPO baseline CM2 (55.60%).

Importantly, unlike training-free alternatives (*e.g.*, SAGE, AUQ) that often rely on multi-step prompting or trial-and-error reasoning loops, TRUST internalizes the decision-making logic directly into the model weights, thereby introducing zero additional inference latency or computational overhead during deployment.

4.5 Ablation Study

Ablation for R_{UQ}. As shown in Fig. 4, all experiments trained with GRPO demonstrate improvements over the vanilla model baseline, while our full method achieves the best overall performance. Among all reward components, removing the uncertainty repulsive reward *i.e.* w/o $c(s)$ leads to the most significant degradation, reducing Acc Norm from 80.83% to 72.46%. Furthermore, excluding the answer reward (w/o R_{ans}) or the format reward (w/o R_{fmt}) also results in varying degrees of performance decline, accompanied by lower accuracy and increased hallucination compared with the complete method. Notably, removing $c(s)$ also substantially increases the overall hallucination (Tool Hall + FDAR) from 22.90% to 30.49%, indicating that $c(s)$ plays a crucial role in suppressing unsupported tool calls and false direct answers. These results suggest that different reward components contribute complementary effects, where R_{ans} and R_{fmt} mainly improve answer quality and output validity, while $c(s)$ is particularly important for reliable tool-calling decisions.

Ablation for unified post-training. As shown in Fig. 5, in trajectory-level unified post-training, removing $c(s)$ also leads to a drastic performance degradation across all evaluated benchmarks, firmly validating that the repulsive force in TRUST is vital for calibrating tool-call decisions. Compared with SFT and the GRPO baseline without tool-calling decision annotation and a combination of rewards in TRUST, the full unified training framework achieves the highest overall scores and the lowest hallucination across the board. The empirical results consistently demonstrate that every designed module plays an indispensable role, and their unification yields the optimal performance. More experimental result details and case study can be found in Appendix C and Appendix D.

5 Conclusions

In this paper, we propose TRUST, an uncertainty-aware reward optimization framework that im-

proves agentic tool-calling decisions in both turn-level and trajectory-level unified post-training. By integrating uncertainty quantification information directly into the reward function, TRUST aligns model certainty with decision correctness and mitigates hallucinated tool-use behaviors. We further introduce lightweight key-turn annotations to unify turn-level decision calibration with trajectory-level task optimization. Extensive experiments on When2Call, BFCL-V4, and ToolSandbox demonstrate that TRUST consistently improves tool-calling reliability, multi-turn interaction quality, and overall agent performance without additional inference overhead.

Limitations

Despite its effectiveness, several limitations remain for future exploration. First, the current framework mainly relies on perplexity-based uncertainty estimation, while more advanced semantic or trajectory-level uncertainty modeling could further improve calibration. Second, our experiments focus on text-based tool-use benchmarks with predefined action spaces. Extending TRUST to more dynamic settings, such as embodied agents or open-world tool ecosystems, is an important direction for future work.

Ethical Considerations

This work aims to improve the tool-calling decision capability and reliability of LLM-based agents by proposing TRUST, an uncertainty-aware reward framework for post-training tool-calling decisions. By aligning decision confidence with correctness, our method improves the turn-level tool-calling decision accuracy and the trajectory-level task performance, while reducing hallucinated or unsupported tool use and mitigating error propagation in multi-turn interactions. All datasets and reproduced baselines used in this work are publicly available, properly cited, and follow their original licenses and terms of use. While we do not anticipate direct severe societal risks from the proposed method itself, more capable agentic systems may still be misused for generating misleading content or unsafe automated behaviors. Future work should further strengthen safety mechanisms such as content moderation and risk-aware deployment protocols.

Acknowledgements

This work is supported by Shanghai Artificial Intelligence Laboratory and Shanghai Innovation Institute. We would like to express our gratitude to our collaborators for their efforts.

References

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 117 others. 2025. Llama-nemotron: Efficient reasoning models. *Preprint*, arXiv:2505.00949.

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. *arXiv preprint arXiv:2506.17419*.

Gemma, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kait Healy, Bharathi Srinivasan, Visakh Madathil, and Jing Wu. 2026. Internal representations as indicators of hallucinations in agent tool selection. *arXiv preprint arXiv:2601.05214*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Michael Kirchhof, Gjergji Kasneci, and Enkelejda Kasneci. 2025. Position: Uncertainty quantification needs reassessment for large-language model agents. *arXiv preprint arXiv:2505.22655*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*.

Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Pengfei Cao, and 1 others. 2025. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv preprint arXiv:2509.18970*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, and 1 others. 2025. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.

Panagiotis Lymperopoulos and Vasanth Sarathy. 2025. Tools in the loop: Quantifying uncertainty of llm question answering systems that use tools. *arXiv preprint arXiv:2505.16113*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

Meta. 2024. Llama-3.2-3b-instruct. Hugging Face Repository.

MiniMax. 2026. Minimax-m2.5. Hugging Face Repository.

Wachiravit Modecrua, Krittanon Kaewtawee, Krittin Pachtrachai, and Touchapon Kraisingkorn. 2026. Multi-turn reinforcement learning for tool-calling agents with iterative reward calibration. *arXiv preprint arXiv:2604.02869*.

Dhruv Nathawani, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Ameya Sunil Mahabaleshwarkar, Jian Zhang, and Jane Polak Scowcroft. 2025. Nemotron-post-training-dataset-v1. Hugging Face Repository.

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. *arXiv preprint arXiv:2602.05073*.

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.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pages 9695–9717.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. 2025. When2call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3391–3409.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Josefa Lia Stoisser, Marc Boubnovski Martell, Lawrence Phillips, Gianluca Mazzoni, Lea Mørch Harder, Philip Torr, Jesper Ferkinghoff-Borg, Kaspar Martens, and Julien Fauqueur. 2025. Towards agents that know when they don't know: Uncertainty as a control signal for structured reasoning. *Preprint*, arXiv:2509.02401.

Hang Su, Jun Luo, Chang Liu, Xiao Yang, Yichi Zhang, Yinpeng Dong, and Jun Zhu. 2025. A survey on autonomy-induced security risks in large model-based agents. *arXiv preprint arXiv:2506.23844*.

Chung-En Sun, Linbo Liu, Ge Yan, Zimo Wang, and Tsui-Wei Weng. 2026. Llm agents already know when to call tools–even without reasoning. *arXiv preprint arXiv:2605.09252*.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A Rossi, and Dinesh Manocha. 2025. Structured uncertainty guided clarification for llm agents. *arXiv preprint arXiv:2511.08798*.

Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jentse Huang, Wenxiang Jiao, and Michael R. Lyu. 2025. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21773–21784. Association for Computational Linguistics.

Qinyuan Wu, Soumi Das, Mahsa Amani, Arijit Nag, Seungeon Lee, Krishna P. Gummadi, Abhilasha Ravichander, and Muhammad Bilal Zafar. 2026. To call or not to call: A framework to assess and optimize llm tool calling. *arXiv preprint arXiv:2605.00737*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Dengjia Zhang, Xiaoou Liu, Lu Cheng, Yaqing Wang, Kenton Murray, and Hua Wei. 2026a. SELAUR: Self evolving llm agent via uncertainty-aware rewards. *arXiv preprint arXiv:2602.21158*.

Jiaxin Zhang, Prafulla Kumar Choubey, Kung-Hsiang Huang, Caiming Xiong, and Chien-Sheng Wu. 2026b. Agentic uncertainty quantification. *arXiv preprint arXiv:2601.15703*.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, Chenyang Zhao, Henry Peng Zou, Haoyun Deng, Sathish Reddy Indurthi, Shujian Liu, Simin Ma, Xiaoyang Wang, Xin Eric Wang, and Song Wang. 2026c. Cm2: Reinforcement learning with checklist rewards for multi-turn and multi-step agentic tool use. *Preprint*, arXiv:2602.12268.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. 2025. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*.

Haitian Zhong, Jixiu Zhai, Lei Song, Jiang Bian, Qiang Liu, and Tieniu Tan. 2026. Rc-grpo: Reward-conditioned group relative policy optimization for multi-turn tool calling agents. *arXiv preprint arXiv:2602.03025*.

A Details of Benchmarks and Baselines

A.1 Benchmarks

We evaluate three benchmarks for TRUST framework:

- When2Call (Ross et al., 2025): a natural benchmark for next-action decision training and testing, as it explicitly evaluates whether an agent should answer directly, call a tool, ask a follow-up question, or admit that the provided tools cannot answer the request.
- ToolSandbox (Lu et al., 2025): evaluates stateful, conversational, and interactive tool use, with scenarios such as canonicalization, tool scrambled, and insufficient information that require the agent to decide when and how to execute tools versus ask or wait for more context.
- BFCL-V4 (Patil et al., 2025): evaluates function-calling reliability across single, multiple, parallel, multi-turn, and relevance-detection settings, where irrelevant or unsuitable function sets directly test whether the model can abstain from unsupported tool calls.

A.2 Baselines

We compare against two training-free uncertainty-aware agent baselines:

- **AUQ** (Zhang et al., 2026b): uses verbalized confidence and explanations to control long-horizon agent execution, propagating uncertainty through memory and invoking reflection only when confidence is low.
- **SAGE** (Suri et al., 2025): a structured uncertainty-guided clarification framework that models uncertainty over tool calls and argument domains, then uses EVPI-based scoring to decide whether to ask a targeted clarification question or execute the best tool call.

B Detailed Setups

B.1 Prompts

Prompts for the LLM labeler can be found in Box 1, and prompts for the LLM judger in the assigning tool-calling decision reward can be found in Box 2. The judger is aiming to judge the decision category in the response of agents, rather than obtain the reward R_{UQ} in one step.

B.2 Training Settings

For turn-level training based on Qwen3-4B-Thinking, we train 1 epoch GRPO. We sample 4 rollouts for each question as the group size. The learning rate is set to 3×10^{-5}, and the KL divergence loss coefficient is set to 0.001. For trajectory-level training based on Qwen3-8B-Base, we follow the training settings in Zhang et al. (2026c) for unified post-training, including cold-start SFT and RL. All training data is from the NVIDIA/NEMOTRON-POST-TRAINING-DATASET-V1 dataset Bercovich et al. (2025); Nathawani et al. (2025) and filtered by Zhang et al. (2026c). NVIDIA/NEMOTRON-POST-TRAINING-DATASET-V1 contains a large number of 310k synthetic tool-use dialogues across diverse domain. For SFT, we use the cold-start training set in Zhang et al. (2026c), which has 8k trajectories. We apply SFT as a cold start under a learning rate

3×10^{-6} with a warmup ratio of 0.1. For RL, the corresponding training set in Zhang et al. (2026c) is further annotated by TRUST and filtered for turn-level decision distribution balance. Finally, we get 4k trajectories from the original 8k data. Then we optimize from the cold-start SFT checkpoint using GRPO based on VeRL. The mini-batch size is set to 128, the KL divergence loss coefficient to 0.001, and the learning rate is 3×10^{-6} for GRPO. The group size is 16 for one trajectory. We train for 400 GPU hours for GRPO. All experiments are conducted on 8× NVIDIA H200 GPUs using DeepSpeed ZeRO-3 optimization.

C Additional Exprimental Results

Detailed experimental results on BFCL-V4 can be found in Tables 5–8, where BFCL-V4 is decomposed into four categories: Non-Live (Table 5), Live (Table 6), Multi-Turn (Table 7), and Agentic (Table 8). Ablation study results on When2Call are presented in Table 9. For trajectory-level post-training from Qwen3-8B-Base, ablation results on BFCL-V4 are further broken down into Non-Live (Table 10), Live (Table 11), Multi-Turn (Table 12), and Agentic (Table 13) categories. Ablation results on ToolSandBox are provided in Table 14.

C.1 Results for Uncertainty Calibration

Table 4 further demonstrates that the uncertainty calibration effect of TRUST generalizes consistently across different backbone LLMs. We evaluate four representative instruction-following models spanning the Qwen, Llama, and Gemma families. Across all backbones, conventional decision GRPO substantially increases the overlap between the PPL distributions of correct and wrong decisions, indicating that optimization solely for decision accuracy tends to produce overconfident incorrect predictions. In contrast, TRUST consistently reduces the distribution overlap while improving or preserving the normalized When2Call accuracy. Notably, on Qwen3-4B-Thinking, the PPL IoU decreases from 70.21% to 35.29% after applying TRUST , accompanied by an accuracy improvement from 72.46% to 80.83%. Similar trends are observed on Qwen3-8B-Thinking, Llama-3.2-3B-Instruct, and Gemma-3-4B-IT, demonstrating that TRUST is architecture-agnostic and consistently encourages lower uncertainty for correct decisions while assigning higher uncertainty to erroneous ones.

C.2 Results on BFCL-V4

Tables 5–8 report detailed results on BFCL-V4 across Non-Live, Live, Multi-Turn, and Agentic categories. Overall, TRUST consistently improves tool-calling performance under both turn-level and trajectory-level training settings, with particularly large gains on complex multi-turn and agentic scenarios.

Non-Live and Live tool calling. On relatively direct tool-calling categories, such as Non-Live and Live, the benefit of trajectory-level optimization is already evident. As shown in Tables 5 and 6, Traj.-level TRUST improves over Traj.-level CM2 from 80.12% to 82.27% on Non-Live, and from 74.58% to 75.57% on Live. The gains are moderate but consistent, suggesting that these categories can be effectively optimized with trajectory-level feedback.

The ablation results further show that $c(s)$ is less critical for these relatively simple single-turn settings. For example, in Tables 10 and 11, +TRUST w/o $c(s)$ already reaches 78.90% on Non-Live and 74.17% on Live, which are close to the SFT or CM2 baselines. On some subcategories, such as Live Parallel Multiple, +TRUST w/o $c(s)$ even matches the full TRUST score of 54.17%. These results suggest that for direct tool-calling tasks where the decision boundary is relatively clear, trajectory-level answer and format rewards can already provide useful learning signals, even without an explicit classification reward.

Multi-turn tool calling. The advantage of TRUST becomes more pronounced in Multi-Turn scenarios, which require maintaining context, resolving missing information, and making correct tool-use decisions across turns. As shown in Table 7, Traj.-level TRUST improves over Traj.-level CM2 from 22.50% to 29.88%, yielding a substantial gain of 7.38% points. The improvement is consistent across all subcategories: Multi Turn Base improves from 34.00% to 40.50%, Missing Function from 21.00% to 26.50%, Missing Parameter from 18.00% to 28.00%, and Long Context from 17.00% to 24.50%.

These gains indicate that TRUST is particularly effective in higher-order tool-calling settings where the model must reason over dialogue history and incomplete user intents. Compared with simpler Non-Live and Live categories, Multi-Turn tasks rely more heavily on correctly identifying whether,

when, and how to invoke tools. Therefore, the uncertainty repulsive reward cR_{cls} becomes more important: without it, the model may still learn from successful trajectories, but it lacks a direct signal for distinguishing tool-use versus non-tool-use decisions across turns.

Agentic tool calling. Large trajectory-level gains also appear in the Agentic category, which includes Web Search and Memory tasks. As shown in Table 8, Traj.-level TRUST significantly improves over Traj.-level CM2 from 21.97% to 28.79% on the overall Agentic score, corresponding to a 6.82% gain. The improvement is especially clear on Memory, where TRUST improves from 21.94% to 30.57%. Within Memory, Key Value Store increases from 9.68% to 21.00%, Vector Store from 14.19% to 24.55%, and Rec Sum from 41.94% to 46.16%.

These results highlight the importance of the proposed reward design for high-level agentic tool use. Agentic tasks require not only producing correct tool arguments, but also deciding when external search, memory retrieval, or memory update is necessary. In such settings, cR_{cls} provides an essential signal for tool-use decision-making. Compared with Non-Live and Live categories, where +TRUST w/o cR_{cls} can already obtain reasonable performance, Agentic tasks benefit much more from the full reward. This suggests that explicit classification supervision is critical for complex agent behaviors involving planning, memory, and external information access.

C.3 Ablation Study on When2Call

Table 9 presents the ablation study on the When2Call test set. Since When2Call directly evaluates whether the model calls tools at the right time, it provides a focused benchmark for analyzing the contribution of different reward components, especially $c(s)$.

Effect of reward components under turn-level training. Starting from Qwen3-4B-Thinking, turn-level TRUST achieves the best overall Acc Norm of 80.83% and F1 of 82.84%, outperforming the base checkpoint by 11.47 points in Acc Norm and 7.07% points in F1. Removing different reward components leads to different failure modes. Without R_{ans}, the model still obtains a strong Acc Norm of 78.97%, suggesting that classification and format rewards can already guide the model toward better tool-use decisions. However, without $c(s)$, Acc Norm drops to 72.46%, and FDAR increases to 24.76%, indicating that the model becomes worse at deciding when not to call tools. Replacing $c(s)$ with a random coefficient, Acc Norm decreases by 10.16%, and Tool Hall rises to 34.50%. Without R_{fmt}, the model obtains 75.90% Acc Norm, but Tool Hall remains relatively high at 24.03%.

These results show that each reward component contributes differently: R_{ans} improves final task correctness, R_{fmt} stabilizes valid tool-call formatting, and $c(s)$ is especially important for suppressing incorrect tool-use decisions. The sharp FDAR increase after removing $c(s)$ confirms that classification reward plays a key role in calibrating when tools should or should not be invoked.

Trajectory-level ablation from Qwen3-8B-Base. The importance of $c(s)$ is even clearer in the trajectory-level setting. Starting from Qwen3-8B-Base, SFT only achieves 19.20% Acc Norm and 27.29 F1%, showing that supervised fine-tuning alone is insufficient under this evaluation protocol. Adding Traj.-level CM2 improves Acc Norm to 43.75%, but it also produces a very high Tool Hall score of 75.43%, suggesting severe over-calling or hallucinated tool use.

In contrast, + Traj.-level TRUST w/o $c(s)$ only reaches 31.43% Acc Norm and 26.19% F1, with FDAR increasing to 27.14%. This indicates that trajectory-level optimization without classification reward provides limited guidance for deciding whether tools should be used. The full + Traj.-level TRUST substantially improves Acc Norm to 62.32% and F1 to 60.62%, while reducing Tool Hall to 49.87% and FDAR to 8.39%. Compared with the w/o $c(s)$ variant, the full reward improves Acc Norm by 30.89 points and F1 by 34.43 points.

These results demonstrate that $c(s)$ is crucial for When2Call-style decision-making. While w/o $c(s)$ can perform reasonably well on simpler BFCL categories such as Non-Live and Live, it is insufficient for benchmarks that explicitly require accurate tool-use timing. The full TRUST reward is therefore necessary for learning robust tool-use policies that balance calling tools when needed and avoiding unnecessary or hallucinated tool calls.

D Case Study

D.1 Case Study 1: Correct Tool Parameter Usage

As shown in Fig. 6, a failure mode is one in which the agent reaches a superficially correct fi-

nal decision while still failing the task at the trajectory level. After correctly entering the `test` directory and identifying `test_file1.txt` and `test_file2.txt`, the baseline incorrectly applies `wc` to the directory name `test` rather than to the two discovered files. As a result, the required intermediate evidence, namely the character counts of the relevant text files, is never obtained. Nevertheless, the model proceeds to update the ticket priority to 2, which happens to coincide with the correct final value because the true counts (20 and 18) are both not greater than 20. In contrast, TRUST executes the intended file-level counting operations and therefore reaches the same final priority assignment through a valid evidential chain.

D.2 Case Study 2: Timely Tool Invocation

The second case study (Fig. 7) demonstrates a complementary failure pattern in the baseline: the agent abstains from calling tools when tool use is necessary to advance the task. A missing tool call in an early turn can destabilize the entire downstream interaction and produce a globally incorrect final state. In the first turn, the baseline requests zip code information from the user instead of invoking the available city-to-zipcode and distance estimation tools. This omission prevents the model from grounding the subsequent reasoning about travel feasibility. The error compounds in later turns: the model does not execute the requested fuel-filling action, and by the final turn, the vehicle state still reports 10.0 gallons, directly contradicting the user's requirement to add 30 gallons and end with 40 gallons. By comparison, TRUST correctly performs the tool chain for geographic lookup, distance estimation, fuel feasibility assessment, tank filling, and engine start, uses the right tool in key turns, and completes the task.

D.3 Case Study 3: Deferring Tool Use under Missing Information

The example in Fig. 8 provides a clearer example of inappropriate tool invocation under incomplete user specification. In the first turn, the user expresses the intent to add a company's stock to the watchlist but does not provide the company name. The benchmark's reference trajectory, therefore, leaves this turn empty, indicating that the correct behavior is to defer action and elicit the missing information. Instead, the baseline hallucinates the company as Apple, invokes `get_symbol_by_name`, and adds `AAPL` to the watchlist. Although the model later adds `ZETA` after the user explicitly specifies "Zeta Corp", the watchlist has already been irreversibly contaminated with an incorrect extra entry. The agent trained by our method TRUST, in contrast, handles the underspecified first turn conservatively by requesting the missing company name and only performs tool calls after the parameter becomes available.

Box 1: Prompt for Labeling Json Annotation

You are annotating multi-turn tool-use trajectories for when2call training, namely supervision for deciding when to answer directly, invoke tools, request missing information, or abstain.

Task:
1. Read the full trajectory, the available tool specifications, and the checklist metadata.
2. Identify decision-critical turns at which a when2call-style supervision signal is both informative and well supported by the observed trajectory.
3. For each selected turn, assign exactly one ground-truth action from the following action space:
- ‘direct_answer‘: the request can be correctly addressed from the existing conversational context or general knowledge, without additional user input and without tool use.
- ‘tool_call‘: the appropriate next step is to invoke one or more tools, and the necessary arguments are already available.
- ‘request_for_info‘: the assistant lacks essential user-provided information, and clarification is the correct immediate next action.
- ‘cannot_answer‘: the assistant should refuse, abstain, or acknowledge incapability or lack of access, such that further clarification would not resolve the problem.
4. Prefer substantive reasoning over shallow lexical heuristics.
5. Maintain broad coverage over the four action categories; do not overproduce ‘tool_call‘ annotations merely because the data originate from a tool-use setting.

Annotation principles:
- Prioritize turns that are genuinely decision-critical, including missing arguments, unsupported or hallucinated tool usage, inappropriate tool invocation, clarification after tool failure, directly answerable requests, and authentic inability cases.
- When multiple ‘turn_idx‘ values are defensible within the same trajectory, prefer the one whose correct action would most improve action-space coverage under the provided balance guidance.
- Avoid producing many redundant annotations of the same class from a single trajectory, especially repeated ‘request_for_info‘ cases.
- Do not default to ‘request_for_info‘ as a conservative fallback. If the assistant already has sufficient information to answer, choose ‘direct_answer‘. If the assistant already has sufficient information to act, choose ‘tool_call‘. If no realistic clarification would resolve the issue, choose ‘cannot_answer‘.
- Use the entire trajectory, including earlier tool outputs and prior assistant behavior. Later turns often support ‘direct_answer‘ or ‘tool_call‘ because the required information has already been established upstream.
- The ‘target_answer‘ must remain compatible with the when2call reward interface:
(1) For ‘tool_call‘, return one or more concrete tool-call XML blocks in the form ‘<tool_call>{"name": "...", "arguments": {...}}</tool_call>‘.
(2) For ‘request_for_info‘, return the realized clarification question.
(3) For ‘direct_answer‘, return the realized answer text.
(4) For ‘cannot_answer‘, return the realized refusal or inability statement.

Illustrative cases:
- ‘direct_answer‘: the user requests an explanation, rewrite, or straightforward factual transformation that can already be completed without tools.
- ‘tool_call‘: the user has supplied all required slots, and the correct next action is immediate tool execution.
- ‘request_for_info‘: a decisive slot required for answering or tool execution is missing from the

user.
- ‘cannot_answer‘: the available toolset cannot solve the request, or the assistant lacks the necessary authority, capability, or access, and clarification would not remedy the limitation.
Return strict JSON:
{
"turn_idx": Current ‘turn_idx‘ in data,
"gt_action": Action for this turn in ‘direct_answer‘/‘tool_call‘/‘request_for_info‘/‘cannot_answer‘,
"rationale": Your explanation for ‘gt_action‘,
"target_answer": Real response in data
}

Here are two illustrative annotation examples:
- Example 1:
{
"turn_idx": 2,
"gt_action": "request_for_info",
"rationale": "Missing province ID required to continue the request correctly.",
"target_answer": "Could you please provide me with the specific province ID for Bangkok?"
}
- Example 2:
{
"turn_idx": 0,
"gt_action": "tool_call",
"rationale": "All required parameters are available. The assistant should call the tool immediately.",
"target_answer": "<tool_call>"name": "a", "arguments": "account_id": "123"</tool_call>"
}

Dataset balance guidance:
- Current accepted annotation counts: ‘direct_answer‘={DIRECT_ANSWER_COUNT}, ‘tool_call‘={TOOL_CALL_COUNT}, ‘request_for_info‘={REQUEST_FOR_INFO_COUNT}, ‘cannot_answer‘={CANNOT_ANSWER_COUNT}
- Currently underrepresented action categories: {UNDERREPRESENTED_ACTIONS}
- Currently overrepresented action categories: {OVERREPRESENTED_ACTIONS}
- If multiple ‘turn_idx‘ values within the same trajectory are equally well justified, prefer the one whose correct ‘gt_action‘ belongs to an underrepresented category.
- Prefer valid non-‘tool_call‘ annotations when ‘tool_call‘ is already overrepresented.
- Favor a small number of high-value annotations per trajectory rather than many redundant ones.
- Do not force an annotation whose action is not supported by the trajectory evidence.

Trajectory payload: {
"tools": {TOOLS_JSON},
"turn_summaries": {TURN_SUMMARIES_JSON},
"checklists": {CHECKLISTS_JSON}
}

Box 2: Prompt for LLM Judger in TRUST

You are a structured parser for assistant turns in multi-turn tool-use trajectories.

Given the history of completed assistant turns within a multi-turn trajectory:
1. Judge the assistant's next action from the current observed turn.
2. Map the action to exactly one normalized action category:
- ‘direct_answer‘: the assistant provides a substantive answer directly, without invoking a tool and without requesting additional user information.
- ‘tool_call‘: the assistant initiates one or more tool calls as the immediate next action.
- ‘request_for_info‘: the assistant asks the user for missing information that is necessary before answering or executing a tool call.
- ‘cannot_answer‘: the assistant indicates inability, refusal, lack of capability, or lack of access such that further clarification would not resolve the issue.
3. Extract the realized response content in a when2call-compatible representation:
- If the action is ‘tool_call‘, return only the tool-call JSON content, without any surrounding prose.
- Otherwise, return the assistant's realized natural-language response exactly as expressed in the turn.
4. Ground the decision in the realized turn behavior rather than in hypothetical alternatives.

The annotation of this turn:
{JSON_ANNOTATON}

If the assistant has conducted ‘gt_action‘ in ‘JSON_ANNOTATION‘ in this turn, choose ‘pred_action‘ the same as ‘gt_action‘.

Return strict JSON. You MUST follow this Output Schema:
{
"pred_action": Action for this turn in ‘direct_answer‘/‘tool_call‘/‘request_for_info‘/‘cannot_answer‘,
"pred_answer": Response of the assistant in this turn,
"rationale": "The turn is a clarification request because a required argument is not yet available. The assistant requests a missing identifier before any tool execution."
}

Here is an example:
{
"pred_action": "request_for_info",
"pred_answer": "Could you provide the account ID?",
"rationale": "The turn is a clarification request because a required argument is not yet available. The assistant requests a missing identifier before any tool execution."
}

Table 4: More results of uncertainty calibration for a wide range of backbone models.

Metric	Vanilla Model	GRPO w/o TRUST	GRPO w/ TRUST
From Qwen3-4B-Thinking			
PPL IoU	34.50	70.21	35.29
When2Call Acc Norm	69.36	72.46	80.83
From Qwen3-8B-Thinking			
PPL IoU	55.65	70.35	50.89
When2Call Acc Norm	45.39	60.32	70.00
From Llama-3.2-3B-Instruct (Grattafiori et al., 2024; Meta, 2024)			
PPL IoU	65.30	81.22	41.49
When2Call Acc Norm	34.54	28.45	45.39
From Gemma-3-4B-IT (Gemma et al., 2025)			
PPL IoU	61.40	70.40	60.88
When2Call Acc Norm	36.75	37.98	46.92

Table 5: Detailed comparison experiment results of the Non-Live category on BFCL-V4.

Models	Non-Live	Simple	Multiple	Parallel	Parallel Multiple
Open-source Baselines					
235B-A22B-Instruct	90.10	78.42	95.50	94.00	92.50
30B-A3B-Instruct	85.77	68.58	94.50	91.50	88.50
4B-Thinking	38.02	41.58	37.00	53.50	20.00
8B-Thinking	87.58	72.83	96.50	92.00	89.00
From Qwen3-4B-Thinking					
AUQ	77.08	66.83	81.50	80.00	80.00
SAGE	86.71	71.33	95.00	92.50	86.00
Turn-level GRPO	86.33	73.33	94.50	91.00	86.50
Turn-level TRUST	**89.23**	77.92	96.00	92.50	90.50
From Qwen3-8B-Thinking					
AUQ	84.46	70.83	90.00	92.50	84.50
SAGE	**88.15**	74.58	95.00	93.00	90.00
From Qwen3-8B-Base					
Traj.-level CM2	80.12	70.50	93.00	85.00	72.00
Traj.-level TRUST	82.27	71.08	94.50	85.00	78.50

Table 6: Detailed comparison experiment results of Live category on BFCL-V4.

Models	Live	Simple	Multiple	Parallel	Parallel Multiple
Open-source Baselines					
235B-A22B-Instruct	83.20	87.60	82.24	75.00	83.33
30B-A3B-Instruct	77.94	83.33	76.83	68.75	75.00
4B-Thinking	71.06	63.18	72.84	87.50	66.67
8B-Thinking	80.53	84.50	79.68	75.00	79.17
From Qwen3-4B-Thinking					
AUQ	62.40	65.89	61.06	87.50	66.67
SAGE	80.49	83.05	79.81	84.50	80.33
Turn-level GRPO	80.61	85.66	79.58	81.25	70.83
Turn-level TRUST	**84.09**	89.53	82.62	87.50	87.50
From Qwen3-8B-Thinking					
AUQ	75.42	80.56	74.26	79.25	68.83
SAGE	**80.83**	84.88	80.06	68.75	79.17
From Qwen3-8B-Base					
Traj.-level CM2	74.58	74.64	75.17	76.00	46.83
Traj.-level TRUST	75.57	75.97	75.97	75.00	54.17

Table 7: Detailed comparison experiment results of the Multi-turn category on BFCL-V4.

Models	Multi-Turn	Multi Turn Base	Multi Turn Missing Function	Multi Turn Missing Parameter	Multi Turn Long Context
Open-source Baselines					
235B-A22B-Instruct	50.13	63.50	45.00	38.50	53.50
30B-A3B-Instruct	30.00	43.50	10.50	25.00	41.00
4B-Thinking	52.00	63.50	58.50	35.00	51.00
8B-Thinking	37.75	46.50	38.00	36.00	30.50
From Qwen3-4B-Thinking					
AUQ	23.50	35.00	22.00	16.00	21.00
SAGE	<u>41.50</u>	59.00	55.50	19.00	32.50
Turn-level GRPO	22.75	31.00	21.50	25.00	13.50
Turn-level TRUST	**49.63**	60.50	58.00	45.50	34.50
From Qwen3-8B-Thinking					
AUQ	10.50	21.00	7.50	7.50	6.00
SAGE	**36.38**	47.00	44.00	29.50	25.00
From Qwen3-8B-Base					
Traj.-level CM2	22.50	34.00	21.00	18.00	17.00
Traj.-level TRUST	<u>29.88</u>	40.50	26.50	28.00	24.50

Table 8: Detailed comparison experiment results for the Agentic category on BFCL-V4. The Agentic category contains two subcategories, *i.e.*, Web Search and Memory.

Models	Agentic	Web Search	Snippet	No Snippet	Memory	Key Value Store	Vector Store	Rec Sum
Open-source Baselines								
235B-A22B-Instruct	35.80	44.50	57.00	32.00	27.10	14.19	16.13	50.97
30B-A3B-Instruct	20.07	22.50	21.00	24.00	17.63	9.03	9.03	34.84
4B-Thinking	9.19	6.00	8.00	4.00	12.37	10.65	15.81	10.65
8B-Thinking	13.31	12.00	15.00	9.00	14.62	5.16	7.10	31.61
From Qwen3-5B-Thinking								
AUQ	11.72	3.00	4.00	2.00	20.43	13.55	19.35	28.39
SAGE	8.26	3.50	4.00	3.00	13.02	6.57	15.81	16.67
Turn-level GRPO	<u>13.76</u>	3.00	4.00	2.00	24.52	12.90	20.00	40.65
Turn-level TRUST	**13.98**	3.00	4.00	2.00	24.95	12.90	23.87	38.06
From Qwen3-4B-Thinking								
AUQ	14.85	14.00	17.00	11.00	15.70	9.03	14.84	23.23
SAGE	15.61	9.50	12.00	7.00	21.72	7.10	17.42	40.65
From Qwen3-8B-Base								
Traj.-level CM2	<u>21.97</u>	22.00	31.00	13.00	21.94	9.68	14.19	41.94
Traj.-level TRUST	**28.79**	27.00	32.00	22.00	30.57	21.00	24.55	46.16

Table 9: Detailed ablation study results on When2Call test set. We do not report results for the base checkpoint of Qwen3-8B-Base since it is not instruction-tuned under this evaluation protocol. The setting w/o $c(s)$ represents $c(s) = 1$, *i.e.* $R_{\text{UQ}} = R_{\text{fmt}} + R_{\text{ans}} + R_{\text{cls}}$. Random c represents replacing $c(s)$ with a random coefficient from the matched distribution.

Models	Acc Norm	F1	Tool Hall	FDAR
From Qwen3-4B-Thinking				
Qwen3-4B-Thinking	69.36	75.77	41.47	17.11
w/o R_{ans}	<u>78.97</u>	79.96	38.37	2.63
w/o $c(s)$	72.46	76.06	5.73	24.76
Random c	70.67	75.08	34.50	12.19
w/o R_{fmt}	75.90	76.55	24.03	1.86
Turn-level TRUST	**80.83**	82.84	17.83	5.07
From Qwen3-8B-Base				
SFT	19.20	27.29	16.38	63.12
+ Traj.-level CM2	<u>43.75</u>	36.94	75.43	9.34
+ Traj.-level TRUST w/o $c(s)$	31.43	26.19	65.50	27.14
+ Traj.-level TRUST	**62.32**	60.62	49.87	8.39

Table 10: Detailed ablation study results of trajectory-level post-training on Non-Live category of BFCL-V4.

Models	Non-Live	Simple	Multiple	Parallel	Parallel Multiple
SFT	79.65	69.08	89.00	80.50	80.00
+ CM2	<u>80.12</u>	70.50	93.00	85.00	72.00
+ TRUST w/o $c(s)$	78.90	70.08	91.00	81.00	73.50
+ TRUST	**82.27**	71.08	94.50	85.00	78.50

Table 11: Detailed ablation study results of trajectory-level post-training on the Live category of BFCL-V4.

Models	Live	Simple	Multiple	Parallel	Parallel Multiple
SFT	72.17	76.36	71.70	75.00	45.83
+ CM2	<u>74.58</u>	74.64	75.17	76.00	46.83
+ TRUST w/o $c(s)$	74.17	75.58	74.36	68.75	54.17
+ TRUST	**75.57**	75.97	75.97	75.00	54.17

Table 12: Detailed ablation study results of trajectory-level post-training on the Multi-Turn category of BFCL-V4.

Models	Multi-Turn	Multi Turn Base	Multi Turn Missing Function	Multi Turn Missing Parameter	Multi Turn Long Context
SFT	19.10	24.00	18.75	14.13	19.50
+ CM2	<u>22.50</u>	34.00	21.00	18.00	17.00
+ TRUST w/o $c(s)$	17.37	15.50	17.50	18.50	18.00
+ TRUST	**29.88**	40.50	26.50	28.00	24.50

Table 13: Detailed ablation study results of Agentic category on BFCL-V4. The Agentic category contains two subcategories *i.e.* Web Search and Memory.

Models	Agentic	Web Search	Snippet	No Snippet	Memory	Key Value Store	Vector Store	Rec Sum
SFT	10.87	14.00	18.00	10.00	7.74	9.03	3.87	10.32
+ CM2	21.97	22.00	31.00	13.00	21.94	9.68	14.19	41.94
+ TRUST w/o $c(s)$	<u>28.20</u>	26.50	25.00	28.00	29.89	18.71	21.29	49.68
+ TRUST	**28.79**	27.00	32.00	22.00	30.57	21.00	24.55	46.16

Table 14: Detailed ablation study results on ToolSandBox.

Model / Method	Overall Score	Scenario Categories						Tool Augmentations								
		STC	MTC	SUT	MUT	SD	C	II	0-DT	3-DT	10-DT	AT	TNS	TDS	ADS	ATS
SFT	50.42	54.02	32.73	38.78	31.42	39.07	28.70	95.70	56.10	54.88	55.46	47.39	51.61	55.46	51.57	53.34
+ CM2	<u>61.21</u>	70.57	58.55	64.22	51.93	56.64	53.70	55.60	60.12	65.04	65.02	65.03	61.80	61.30	64.14	64.51
+ TRUST w/o $c(s)$	56.76	54.37	40.68	44.88	39.71	43.58	38.76	96.01	61.69	61.49	62.64	61.80	61.38	61.21	61.22	61.91
+ TRUST	**68.28**	68.66	59.70	63.10	56.93	61.58	52.04	90.70	73.01	76.41	74.94	69.16	66.74	68.77	71.26	71.24

Example 1

Turn 2 uses **wc(test)** instead of the two files. The final priority is **2**, but the benchmark still fails because the middle turn is wrong.

Failure Case in Baselines

Wrong target in turn 2: **test** is a directory; the correct calls are **test_file1.txt** and **test_file2.txt**.

Failure

TURN 1

USER

In my workspace folder, direct your attention to the initial directory we have access to and list out the files present including the hidden files. Should you stumble upon a directory named 'test', go into there, dive deep and identify any files with 'test' in their names using 'ls'. Keeping tabs on details is key.

ASSISTANT / TOOL_CALL

ls(a=True)

{"current_directory_content": ["assignment.docx","test","submissions","completed_tasks"]}

ASSISTANT / TOOL_CALL

cd(folder='test')

{"current_working_directory":"/test"}

ASSISTANT / TOOL_CALL

ls(a=True)

{"current_directory_content":["test_file1.txt","test_file2.txt"]}

ASSISTANT

Listed the files and entered /test.

TURN 2

USER

Subsequently, what's the character count of all text file with test?

ASSISTANT / TOOL_CALL

wc(file_name='test')

TOOL OUTPUT

{"error":"wc: test: No such file or directory"}

ASSISTANT

Wrong target: **test** is not the file to count.

TURN 3

USER

Open up ticket 654321. If the character count of any file is greater than 20, Set the priority to 3. Else, set to 2.

ASSISTANT / TOOL_CALL

get_ticket(ticket_id=654321)

TOOL OUTPUT

{"title":"Feature Request","priority":"Medium"}

ASSISTANT / TOOL_CALL

edit_ticket(priority=2)

TOOL OUTPUT

{"status":"Ticket 654321 has been updated successfully."}

ASSISTANT

Final priority is **2** (correct value, but the previous turn still failed).

Success in Ours

Turn 2 calls **wc** on the two real files, so turn 3 can update the ticket with proper evidence.

Success

TURN 1

USER

In my workspace folder, direct your attention to the initial directory we have access to and list out the files present including the hidden files. Should you stumble upon a directory named 'test', go into there, dive deep and identify any files with 'test' in their names using 'ls'. Keeping tabs on details is key.

ASSISTANT / TOOL_CALL

ls(a=True)

{"current_directory_content": ["assignment.docx","test","submissions","completed_tasks"]}

ASSISTANT / TOOL_CALL

cd(folder='test')

{"current_working_directory":"/test"}

ASSISTANT / TOOL_CALL

ls(a=True)

{"current_directory_content":["test_file1.txt","test_file2.txt"]}

ASSISTANT

Listed the files and entered /test.

TURN 2

USER

Subsequently, what's the character count of all text file with test?

ASSISTANT / TOOL_CALL

wc(file_name='test_file1.txt')

ASSISTANT / TOOL_CALL

wc(file_name='test_file2.txt')

ASSISTANT

Got the real counts: **20** and **18**.

TURN 3

USER

Open up ticket 654321. If the character count of any file is greater than 20, Set the priority to 3. Else, set to 2.

ASSISTANT / TOOL_CALL

get_ticket(ticket_id=654321)

ASSISTANT / TOOL_CALL

wc(file_name='test_file1.txt')

ASSISTANT / TOOL_CALL

wc(file_name='test_file2.txt')

ASSISTANT / TOOL_CALL

edit_ticket(priority=2)

ASSISTANT

Priority updated to **2** with the correct evidence.

Figure 6: Example of tool-calling parameter failure in baselines and success in ours method.

Example 2

Turn 1 should have called the distance tools; Failure Case in instead asked the user for zipcodes and never filled the tank, so the final fuel state stayed wrong. Success in Ours completes the whole chain.

Failure Case in Baselines

Should have used **get_zipcode_based_on_city** and **estimate_distance**, but it never did. Final fuel stays at **10.0 gallons**.

Failure

TURN 1

USER
I'm planning a road trip from Rivermist to Stonebrook and would love an estimate on how far this journey will take me.

ASSISTANT
Needs zipcode inputs instead of calling the distance tools.

TURN 2

USER
Great, now that I know the distance, please check if my current fuel level can handle the journey or if I need a refill.

ASSISTANT
Still asks for distance/fuel details instead of checking them.

TURN 3

USER
Ah, it looks like I might need some fuel. Could you go ahead and fill up the tank with 30 gallons of fuel so that I end with 40 gallons?

ASSISTANT
Explains the tool semantics, but still no actual fill action.

TURN 4

USER
Let's get the vehicle ready by starting the engine, please make sure everything's in place.

ASSISTANT / TOOL_CALL
startEngine(ignitionMode='START')

TOOL OUTPUT
{"error":"All doors must be locked before starting the engine."}

ASSISTANT / TOOL_CALL
lockDoors(...)

TOOL OUTPUT
{"lockStatus":"locked","remainingUnlockedDoors":0}

ASSISTANT / TOOL_CALL
startEngine(ignitionMode='START')

TOOL OUTPUT
{"error":"Brake pedal needs to be pressed when starting the engine."}

ASSISTANT / TOOL_CALL
pressBrakePedal(pedalPosition=1)

TOOL OUTPUT
{"brakePedalStatus":"pressed","brakePedalForce":1000.0}

ASSISTANT / TOOL_CALL
startEngine(ignitionMode='START')

TOOL OUTPUT
{"engineState":"running","fuelLevel":10.0,"batteryVoltage":12.6}

ASSISTANT
Engine starts, but the fuel is still **10.0 gallons**, not 40.

Success in Ours

Correctly computes distance, checks fuel, adds **30 gallons**, and starts the engine with **40 gallons** remaining.

Success

TURN 1

USER
I'm planning a road trip from Rivermist to Stonebrook and would love an estimate on how far this journey will take me.

ASSISTANT / TOOL_CALL
get_zipcode_based_on_city('Rivermist')

TOOL OUTPUT
{"zipcode":"83214"}

ASSISTANT / TOOL_CALL
get_zipcode_based_on_city('Stonebrook')

TOOL OUTPUT
{"zipcode":"74532"}

ASSISTANT / TOOL_CALL
estimate_distance(cityA='83214', cityB='74532')

TOOL OUTPUT
{"distance":750.0}

ASSISTANT
Distance found: **750 km**.

TURN 2

USER
Great, now that I know the distance, please check if my current fuel level can handle the journey or if I need a refill.

ASSISTANT / TOOL_CALL
displayCarStatus(option='fuel')

TOOL OUTPUT
{"fuelLevel":10.0}

ASSISTANT / TOOL_CALL
estimate_drive_feasibility_by_mileage(distance=466.03)

TOOL OUTPUT
{"canDrive":false}

ASSISTANT
Fuel is not enough; a refill is needed.

TURN 3

USER
Ah, it looks like I might need some fuel. Could you go ahead and fill up the tank with 30 gallons of fuel so that I end with 40 gallons?

ASSISTANT / TOOL_CALL
fillFuelTank(fuelAmount=30)

TOOL OUTPUT
{"fuelLevel":40.0}

ASSISTANT
Tank filled to **40 gallons**.

TURN 4

USER
Let's get the vehicle ready by starting the engine, please make sure everything's in place.

ASSISTANT / TOOL_CALL
startEngine(ignitionMode='START')

TOOL OUTPUT
{"error":"All doors must be locked before starting the engine."}

ASSISTANT / TOOL_CALL
lockDoors(door=['driver','passenger','rear_left','rear_right'])

TOOL OUTPUT
{"lockStatus":"locked","remainingUnlockedDoors":0}

ASSISTANT / TOOL_CALL
startEngine(ignitionMode='START')

TOOL OUTPUT
{"error":"Brake pedal needs to be pressed when starting the engine."}

ASSISTANT / TOOL_CALL
pressBrakePedal(pedalPosition=1.0)

TOOL OUTPUT
{"brakePedalStatus":"pressed","brakePedalForce":1000.0}

ASSISTANT / TOOL_CALL
startEngine(ignitionMode='START')

TOOL OUTPUT
{"engineState":"running","fuelLevel":40.0,"batteryVoltage":12.6}

ASSISTANT
Engine is running with **40 gallons** left.

Figure 7: Example of tool-Calling decision failure of missed tool invocation and downstream task collapse, and success in ours method.

Example 3

Turn 1 had insufficient information, yet Failure Case in Baselines still added **AAPL**. Success in Baselines first asks for the company name, then adds the right stock **ZETA**.

Failure Case in Baselines

The watchlist is polluted with **AAPL** even though the user only asked to track an unspecified company stock.

Failure

TURN 1

USER
I want to keep track of a specific company stock activities, so it would be helpful to have its stock added to my watchlist.

ASSISTANT / TOOL_CALL
`get_symbol_by_name(name='Apple')`

ASSISTANT / TOOL_CALL
`add_to_watchlist(stock='AAPL')`

ASSISTANT
Mistakenly adds **AAPL** before the user specifies the company.

TURN 2

USER
The company name is Zeta Corp

ASSISTANT / TOOL_CALL
`get_symbol_by_name(name='Zeta Corp')`

ASSISTANT / TOOL_CALL
`add_to_watchlist(stock='ZETA')`

ASSISTANT
Adds **ZETA**, but the watchlist is already polluted by **AAPL**.

TURN 3

USER
I've recently placed an order but can't seem to recall the details. It would be useful if you could retrieve the specifics of my last order to aid in planning my next investment move.

ASSISTANT / TOOL_CALL
`get_order_history()`

ASSISTANT / TOOL_CALL
`get_order_details(order_id=12345)`

ASSISTANT
Order 12345: **AAPL, $210.65, 10 shares, Completed.**

TURN 4

USER
Considering the latest developments, I (user id USR001) need to quickly communicate with Jane, one of my business partners, regarding the new stock inclusion and the status of my current order. Send the message 'What are the new stock inclusion and the status of my current order?'?

ASSISTANT / TOOL_CALL
`get_user_id(user='Jane')`

ASSISTANT / TOOL_CALL
`send_message(receiver_id='USR002', message='What are the new stock inclusion and the status of my current order?')`

ASSISTANT
Message sent to **Jane (USR002)**.

TURN 5

USER
Post sending that message to Jane, I would prefer to review all the messages I've sent, as there might be relevant follow-up or advice from my network concerning the stock and order decisions.

ASSISTANT / TOOL_CALL
`view_messages_sent()`

ASSISTANT
Shows the sent messages, but the watchlist remains wrong because of the earlier **AAPL** addition.

Success in Ours

The watchlist stays correct: **[NVDA, ZETA]** and later turns remain consistent.

Success

correct turn-level decision

TURN 1

USER
I want to keep track of a specific company stock activities, so it would be helpful to have its stock added to my watchlist.

ASSISTANT
Need the stock symbol or company name first.

TURN 2

USER
The company name is Zeta Corp

ASSISTANT / TOOL_CALL
`get_symbol_by_name(name='Zeta Corp')`

ASSISTANT / TOOL_CALL
`add_to_watchlist(stock='ZETA')`

ASSISTANT
Adds **ZETA**; watchlist is correct.

TURN 3

USER
I've recently placed an order but can't seem to recall the details. It would be useful if you could retrieve the specifics of my last order to aid in planning my next investment move.

ASSISTANT / TOOL_CALL
`get_order_history()`

ASSISTANT / TOOL_CALL
`get_order_details(order_id=12345)`

ASSISTANT
Order 12345: **AAPL, $210.65, 10 shares, Completed.**

TURN 4

USER
Considering the latest developments, I (user id USR001) need to quickly communicate with Jane, one of my business partners, regarding the new stock inclusion and the status of my current order. Send the message 'What are the new stock inclusion and the status of my current order?'?

ASSISTANT / TOOL_CALL
`get_user_id(user='Jane')`

ASSISTANT / TOOL_CALL
`send_message(receiver_id='USR002', message='What are the new stock inclusion and the status of my current order?')`

ASSISTANT
Message sent to **Jane (USR002)**.

TURN 5

USER
Post sending that message to Jane, I would prefer to review all the messages I've sent, as there might be relevant follow-up or advice from my network concerning the stock and order decisions.

ASSISTANT / TOOL_CALL
`view_messages_sent()`

ASSISTANT
Shows the sent messages, with the watchlist still correct.

Figure 8: Example of tool-Calling decision failure of unwarranted tool use under information insufficiency, and success in ours method.