R2IF: Aligning Reasoning with Decisions via Composite Rewards for Interpretable LLM Function Calling

Aijia Cheng[1], Kailong Wang[2], Ling Shi[3], Yongxin Zhao[1]*
[1]Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
[2]Huazhong University of Science and Technology
[3]Nanyang Technological University
51275902045@stu.ecnu.edu.cn, wangkl@hust.edu.cn,
ling.shi@ntu.edu.sg, yxzhao@sei.ecnu.edu.cn

Abstract

Function calling empowers large language models (LLMs) to interface with external tools, yet existing RL-based approaches suffer from misalignment between reasoning processes and tool-call decisions. We propose R2IF, a reasoning-aware RL framework for interpretable function calling, adopting a composite reward integrating format/correctness constraints, Chain-of-Thought Effectiveness Reward (CER), and Specification-Modification-Value (SMV) reward, optimized via GRPO. Experiments on BFCL/ACEBench show R2IF outperforms baselines by up to 34.62% (Llama3.2-3B on BFCL) with positive Average CoT Effectiveness (0.05 for Llama3.2-3B), enhancing both function-calling accuracy and interpretability for reliable tool-augmented LLM deployment.

1 Introduction

Large language models (LLMs) excel at open-domain text generation but lack reliability in real-time structured interactions (e.g., tool invocation) due to static offline training data (Liu et al., 2024b,a; Wang et al., 2024). Function calling addresses this by translating natural language queries into precise executable tool APIs, unlocking up-to-date knowledge and complex task-solving capabilities.

Early function calling research relied on supervised fine-tuning (SFT) (Hao et al., 2025) with annotated tool-use data, but was bottlenecked by scarce high-quality executable data. As training signals and evaluation protocols matured, reinforcement learning (RL) (Qian et al., 2025; Zhang et al., 2025) emerged as a powerful alternative, directly optimizing functional correctness and enabling accurate function calls across diverse benchmarks. Yet most RL-based methods remain outcome-driven, rewarding only final correctness via AST evaluation—correct calls (Patil et al., 2025) do not equate to sound reasoning, as generated reasoning often serves as post-hoc rationalization rather than guiding tool selection or parameter construction. For example, when queried about San Francisco's weather (Figure 1), the baseline GRPO model's reasoning only identifies #get_current_weather# but ignores parameter format (City, State) and default values, leading to incomplete tool calls. Given function calling's structural rigor and sensitivity to fine-grained arguments, accuracy alone fails to capture reasoning quality or system robustness.

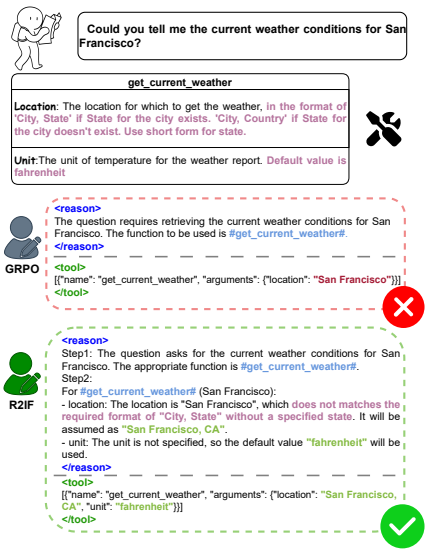

Figure 1: Our method correctly supplements parameter values (e.g., CA, fahrenheit) by aligning reasoning with tool specifications, while the baseline GRPO model fails to address parameter format requirements.

Directly rewarding reasoning without grounding in tool schemas is non-trivial. Unlinked to tool specifications and parameter transformations, process rewards prioritize verbosity over utility. Worse, unconstrained reasoning rewards may erode parameter precision by reinforcing plausible but tool-call decision-misaligned rationales, making "rewarding

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 36995–37008
July 2-7, 2026

reasoning" insufficient for reliable tool use.

Addressing this gap is pivotal to enhancing tool-augmented LLMs' interpretability and robustness: ungrounded reasoning complicates error diagnosis, causes failures in unseen parameter formats or queries, and undermines trust and generalization. Effective supervision must distinguish between reasoning that merely accompanies correct actions and reasoning that actively supports parameter specification, necessary modifications, and value instantiation—three pillars of reliable tool use.

Our Work. In this work, we propose R2IF, a reasoning-aware reinforcement learning framework for interpretable function calling in large language models. We formalize interpretability as the alignment between reasoning and executable tool-call decisions, grounded in parameter specification, modification, and value instantiation. Building on this, we design a composite reward integrating strict format/correctness constraints, a distribution-based Chain-of-Thought (Wei et al., 2023) Effectiveness Reward (CER) quantifying reasoning utility, and a Specification-Modification-Value (SMV) reward enforcing parameter-level reasoning alignment. Optimized via Grouped Proximal Policy Optimization (GRPO) (Shao et al., 2024), R2IF provides trajectory-level supervision to discourage correct-but-ungrounded tool calls and strengthen reasoning-tool call decision links.

Results and Findings. Across BFCL and ACEBench benchmarks, R2IF achieves top overall accuracy across Qwen2.5 (1.5B/3B/7B) and Llama3.2-3B backbones, with a +34.62% gain on Llama3.2-3B (BFCL) and +15.4% on Qwen2.5-7B (ACEBench). It maintains strong irrelevant tool-rejection performance ($\geq$96% accuracy) while boosting Atomic and Single-turn task gains, confirming balanced improvement over conservative strategies. R2IF's reasoning effectiveness (ACE) turns positive for 3B/7B models (e.g., 0.05 for Llama3.2-3B on BFCL), outperforming baselines with negative values. Notably, R2IF scales monotonically with model size and generalizes across model families, remaining competitive under real-user Live scenario distribution shifts.

Summary of contributions: Our contributions are summarized as follows:

- We propose a hybrid reward design that jointly optimizes reasoning quality and final function-call correctness, addressing the misalignment between reasoning processes and tool-call decisions in RL-based function calling.
- We introduce a distribution-based Chain-of-Thought Effectiveness Reward (CER), which enhances tool-call stability without relying on subjective reasoning scoring.
- We design a parameter-level Specification-Modification-Value (SMV) as an optimization of outcome-oriented RL methods, explicitly supervising parameter constraints, transformations, and value instantiation to improve interpretability and precision.

Table 1: Comparison of task categories in terms of tool availability and invocation patterns.

Task Category	Tool Set Size $\|T\|$	Distinct Tools Used	# Tool Calls $\|A\|$
Simple	1	1	1
Multiple	> 1	1	1
Parallel	1	1	> 1
Parallel Multiple	> 1	≥ 1	> 1
Irrelevance	$\mathbb{N}$	0	0

2 Background and Related Work

Formalization. In the function calling task, our goal is to evaluate the model's ability to deterministically transform user queries into executable actions. The core formalization of this task is as follows:

Given a user query Q and a tool set $T = \{t_1, t_2, ..., t_n\}$, the model performs a mapping function f, producing an ordered output tuple

$$(R, A) = f(Q, T).$$

Here, R represents the reasoning process, which is a text reasoning sequence; A represents the function call result, which can vary depending on the specific scenario, and it may be an empty action $\emptyset$, a single action, or a set of actions. The specific scenarios are all presented in Table 1.

Function call. Early work on LLM function calling primarily relied on supervised fine-tuning (SFT). To mitigate the lack of high-quality data, several studies proposed scalable synthetic data pipelines. Liu et al. (2024b) introduce APIGen, which generates verifiable function-calling data via hierarchical checks, while Liu et al. (2024a) propose ToolACE, an agentic synthesis framework that expands API coverage and scenario complexity. For standardized evaluation, Patil et al. (2025) present the Berkeley Function Calling Leaderboard (BFCL), which has become a widely adopted benchmark.

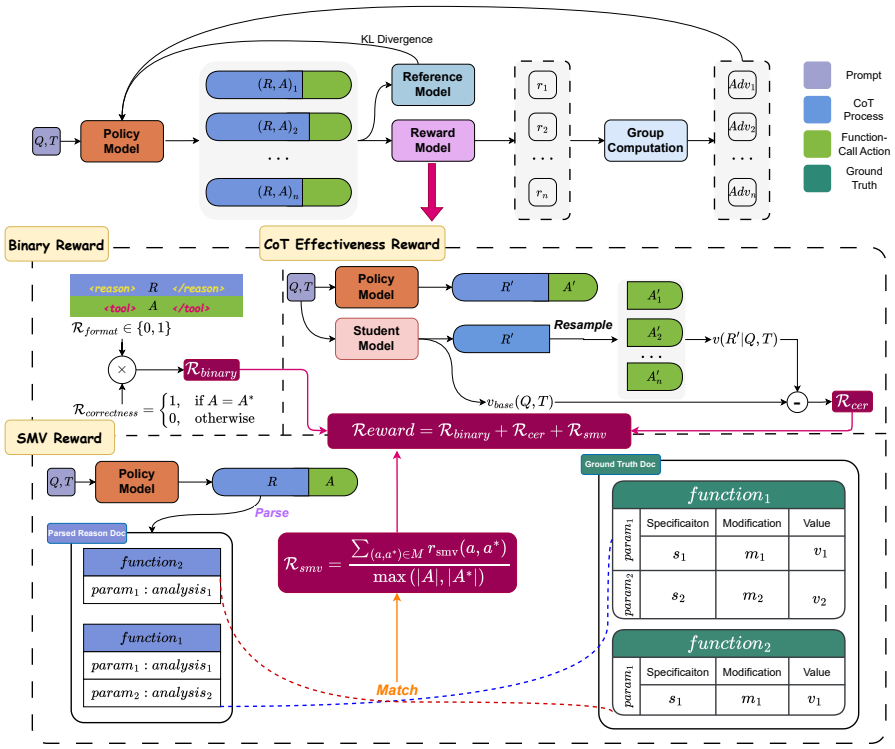

Figure 2: Overall pipeline of our reward optimization. We (i) compute a binary reward to enforce strict function-call correctness, (ii) compute a CER reward to measure whether the CoT supports the chosen actions, and (iii) compute an SMV reward to align the CoT with ground-truth parameter specification, modification, and value.

More recently, reinforcement learning (RL) has been used to frame function calling as a policy optimization problem, directly optimizing correctness under verifiable rewards. Inspired by R1-style reasoning models (DeepSeek-AI et al., 2025), RL has been adapted to tool use, with studies exploring reward design for tool selection and argument construction under GRPO (Qian et al., 2025), as well as rule-based RL with binary correctness and format rewards (Zhang et al., 2025).

RL for LLM Reasoning. Reinforcement learning has emerged as a core post-training paradigm for enhancing LLM reasoning by optimizing generation policies with outcome-based feedback and improving long-horizon credit assignment (Ouyang et al., 2022). In practice, PPO-style optimizers and their variants (e.g., GRPO) are widely used to stabilize and scale reasoning-oriented training (Schulman et al., 2017; Shao et al., 2024; DeepSeek-AI et al., 2025). Beyond answer-level rewards, prior work emphasizes process-aware supervision and step-level evaluation to provide denser signals for multi-step reasoning (Lightman et al., 2023; Zheng et al., 2025). Recent studies further extend RL to tool-augmented reasoning, rewarding correct tool usage under structured action spaces (Qian et al., 2025; Zhang et al., 2025).

GRPO. Each training instance consists of a user query Q and a tool set T. Given (Q, T), the policy generates multiple rollouts, each producing

$$(R, A) = f(Q, T),$$

where R is the reasoning sequence and A is the resulting function call. Collecting n rollouts for the same query yields

$$G_Q = \{((R_1, A_1), r_1), \ldots, ((R_n, A_n), r_n)\},$$

where r_i is the scalar reward of the i-th rollout.

We normalize rewards within each group. The mean and standard deviation are

$$\mu_Q = \frac{1}{n}\sum_{i=1}^{n} r_i, \quad \sigma_Q = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \mu_Q)^2}.$$

The normalized advantage is defined as

$$Adv_i(R_i, A_i \mid Q) = \frac{r_i - \mu_Q}{\sigma_Q + \eta},$$

where η is a small constant for numerical stability.

Policy optimization follows the clipped PPO objective:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{Q\sim\mathcal{D}}\,\mathbb{E}_{(R_i,A_i)\sim\pi_\theta}\Big[\min\Big(\rho_\theta^{(i)} Adv_i(R_i,A_i\mid Q),\ \text{clip}\left(\rho_\theta^{(i)},\,1-\epsilon,\,1+\epsilon\right)Adv_i(R_i,A_i\mid Q)\Big)\Big],$$

where

$$\rho_\theta^{(i)} = \frac{\pi_\theta(R_i,A_i\mid Q,T)}{\pi_{\text{old}}(R_i,A_i\mid Q,T)},$$

and ϵ is the PPO clipping hyperparameter. Optionally, a KL penalty to a reference policy can be added to further constrain updates.

3 Method

In this section, we describe our R2IF Method in detail, including the design of reward. An overview of the R2IF framework is presented in Figure 2.

3.1 Reward Function Design

To train the policy to produce both executable actions and well-grounded reasoning, we design a composite reward function that combines hard structural constraints with fine-grained reasoning supervision. For a rollout (R, A) generated from a query Q and tool set T, the overall reward is defined as

$$\mathcal{R} = \mathcal{R}_{\text{binary}} + \mathcal{R}_{\text{cer}} + \mathcal{R}_{\text{smv}},$$

where $\mathcal{R}_{\text{binary}}$ enforces both the format and the correctness of the tool calls, $\mathcal{R}_{\text{cer}}$ is used to reward the overall effectiveness of the reasoning, and $\mathcal{R}_{\text{smv}}$ provides a finer-grained reward signal by evaluating whether the reasoning correctly identifies the parameter specifications, applies the corresponding modifications, and ultimately derives the correct values. Each component is described below.

3.1.1 Binary Reward

The binary reward acts as a hard gate that ensures the model output is both well-formatted and exactly consistent with the ground truth. It is defined as

$$\mathcal{R}_{\text{binary}} = \mathcal{R}_{\text{format}} \cdot \mathcal{R}_{\text{correctness}},$$

where $\mathcal{R}_{\text{format}} \in \{0,1\}$ denotes format validity and $\mathcal{R}_{\text{correctness}} \in \{0,1\}$ denotes tool-call correctness. In the final reward aggregation, the binary reward is weighted by 3 to avoid the optimization focus shifting excessively toward the softer reward terms.

Format validity $\mathcal{R}_{\text{format}} = 1$ if and only if the output satisfies all of the following conditions: (i) it contains exactly one `<reason></reason>` block and one `<tool></tool>` block; (ii) the `<reason>` block precedes the `<tool>` block; (iii) response should not include any other text. Otherwise, $\mathcal{R}_{\text{format}} = 0$.

Tool-call correctness Let A^* denote the ground-truth action list and A the predicted action list, where each action is represented as a tuple $(\text{name}, \text{arguments})$. We set

$$\mathcal{R}_{correctness} = \begin{cases} 1, & \text{if } A = A^* \\ 0, & \text{otherwise} \end{cases}$$

For irrelevance-rejection instances, where no tool should be invoked, $\mathcal{R}_{\text{correctness}} = 1$ if the model explicitly outputs the predefined rejection string; otherwise $\mathcal{R}_{\text{correctness}} = 0$.

3.1.2 Chain-of-Thought Effectiveness Reward

While $\mathcal{R}_{\text{binary}}$ evaluates the structural correctness of format and tool call result, it does not directly capture whether the generated cot R is effective in supporting correct tool call. Thus, we introduce the Chain-of-Thought Effectiveness Reward (CER), which estimates the contribution of the reasoning prefix to successful tool call.

For each training instance, we maintain a precomputed success rate as the baseline value $v_{\text{base}}(Q,T)$ of a faithful student model which is finetuned by this faithfulness-enhancing method(Akter et al., 2025), representing the student model's own ability.

Then, for the given a rollout (R, A), we estimate the effectiveness of the reasoning prefix by letting the student model sample multiple continuations conditioned on the fixed prefix

$$\texttt{<reason>}R\texttt{</reason><tool>}.$$

Each sampled continuation produces a candidate action sequence, which is evaluated with the same ground truth. Finally, the success rate over these samples defines the effectiveness of the reasoning process $v(R\mid Q,T)$.

The chain-of-thought effectiveness reward is then defined as the advantage over the baseline:

$$\mathcal{R}_{\text{cer}} = v(R\mid Q,T) - v_{\text{base}}(Q,T). \quad (1)$$

More concretely, $\mathcal{R}_{\text{cer}}$ measures whether the reasoning R can search enough information so that it can not only guide the student model but also itself to generate the correct tool-call result.

3.1.3 Specification–Modification–Value Reward

In function calling, argument values often require non-trivial processing rather than direct extraction from the user query. Correct tool call therefore depends not only on producing the final argument values, but also on whether the model correctly identifies the underlying constraints and transformations implied by the tool specification.

To capture this structure, we introduce the **Specification–Modification–Value (SMV) reward**, which evaluates the extent to which the model's reasoning explicitly supports parameter-level decisions. Concretely, SMV measures whether the reasoning (i) correctly identifies the *specification* of an argument, (ii) describes the necessary *modification* from the user query to a valid argument value, and (iii) ultimately materializes the argument *value* in the tool call.

Documents and parsing. Given a response with predicted tool calls $A = \{a_i\}_{i=1}^{|A|}$, we parse the chain-of-thought into an *answer reasoning document* $\mathrm{Doc}^{\mathrm{ans}} = \{\mathrm{doc}_i^{\mathrm{ans}}\}_{i=1}^{|A|}$, where $\mathrm{doc}_i^{\mathrm{ans}}.\mathrm{arg}[p]$ is the reasoning snippet for argument p in call a_i. Meanwhile we provide a *ground-truth document* $\mathrm{Doc}^* = \{\mathrm{doc}_j^*\}_{j=1}^{|A^*|}$, where each argument has metadata $\mathrm{doc}_j^*.\mathrm{arg}[p] = \langle s_p, m_p, \cdot \rangle$ (specification s_p, modification m_p) which is generated by LLM. Details on how the ground-truth document is obtained are provided in Appendix G.2.

Exact-match alignment. We compute SMV only on strictly aligned calls. Let $M = \{(i, j)\}$ be the exact-match set, where $(i, j) \in M$ iff $a_i = a_j^*$ where $a_j^* \in A^*$. This prevents SMV credit for mismatched tools or schemas.

Parameter-level SMV. For a matched pair $(i, j) \in M$, let $R_p = \mathrm{doc}_i^{\mathrm{ans}}.\mathrm{arg}[p]$ denote the reasoning snippet for parameter p, and let $\langle s_p, m_p, \cdot \rangle = \mathrm{doc}_j^*.\mathrm{arg}[p]$ be the corresponding ground-truth specification and modification. We define the parameter-level SMV score as

$$r_{\mathrm{smv}}(p) = \tfrac{1}{3}(r_{\mathrm{spec}}(p) + r_{\mathrm{mod}}(p) + r_{\mathrm{val}}(p)), \tag{2}$$

which equally weights three complementary aspects of parameter-level correctness.

The value term provides a hard executability signal:

$$r_{\mathrm{val}}(p) = \mathbb{I}[p \in a_i.\mathrm{arg}], \tag{3}$$

indicating whether the parameter p is actually instantiated in the final tool call.

To assess specification and modification awareness, we compute sentence-level semantic similarity between the reasoning snippet R_p and the ground-truth text $t \in \{s_p, m_p\}$, with a threshold gate τ:

$$r_{\mathrm{sem}}(p, t) = \mathbb{I}[\mathrm{sim}(R_p, t) \geq \tau] \cdot \mathrm{sim}(R_p, t). \tag{4}$$

We then set $r_{\mathrm{spec}}(p) = r_{\mathrm{sem}}(p, s_p)$ and $r_{\mathrm{mod}}(p) = r_{\mathrm{sem}}(p, m_p)$ when the corresponding annotations are available, and 0 otherwise. This design rewards reasoning that explicitly acknowledges argument constraints and required transformations, rather than merely producing the correct final value.

Tool-call-level SMV. For $(i, j) \in M$, we average over supervised arguments P_j^* in doc_j^*:

$$r_{\mathrm{smv}}(a_i, a_j^*) = \frac{1}{|P_j^*|} \sum_{p \in P_j^*} r_{\mathrm{smv}}(p). \tag{5}$$

Action-level SMV. We aggregate over matched call pairs and normalize by call-count mismatch (as in the SMV Reward Part of Fig. 2):

$$\mathcal{R}_{\mathrm{smv}} = \frac{\sum_{(i,j) \in M} r_{\mathrm{smv}}(a_i, a_j^*)}{\max(|A|,\ |A^*|)}. \tag{6}$$

Irrelevance case For irrelevance instances where the correct action is an empty tool-call set, we prepare a reason why the tool can not be called and then compare it with the reasoning process R by similarity to check whether the problem is located or not.

4 Experiment

4.1 Training Dataset

For the training dataset, we select a subset of data from ToolACE dataset(Liu et al., 2024a). After data filtering and verification, we find that the number of Parallel case is relatively small, with no more than 500 instances. Thus, to avoid the influence of data imbalance, we sample 500 instances from the other four cases and finally get a balanced training set of 2,500 samples in total.

4.2 Experiment Setting

Training. We use Verl to train models on 6 NVIDIA RTX PRO 6000 GPUs. The details are listed in Table 15.

Baselines. We compare our method against the following baselines: (1) **Raw Instruct Model** is the original instruction-tuned model. (2) **SFT** fine-tunes the raw instruct model using the same training set, with the reasoning process distilled from GPT-4o (OpenAI et al., 2024). (3) **Binary reward**(Zhang et al., 2025) applies GRPO training with only binary reward. (4) **ToolRL**(Qian et al., 2025) applies GRPO training with the specific designed reward.

Benchmarks. We select the single-turn test case from several benchmarks to comprehensively evaluate the tool-calling performance of models. (1) **Berkeley Function Calling Leaderboard (BFCL)** (Patil et al., 2025) is a widely adopted benchmark designed to assess large language models' ability to invoke external functions. (2) **ACEBench** (Chen et al., 2025) is a more recent comprehensive benchmark for tool usage.

Metrics. We report two metrics to evaluate both tool call correctness and the utility of the generated reasoning.

Accuracy is the standard exact-match metric for tool calling. A prediction is counted as correct if its tool-call output matches the ground-truth under the official evaluation script; otherwise it is incorrect.

Average CoT Effectiveness is a metric to evaluate the overall utility of CoT reasoning process across multiple cases or tasks. It calculates the average effectiveness of the reasoning prefix in assisting tool-call decisions, based on its contribution to improving the model's tool-call performance.

For each test case i, we compute the Chain-of-Thought effectiveness using CER in Eq. 1:

$$R_{\text{cer}}^{(i)} = v(R^{(i)} \mid Q, T) - v_{\text{base}}(Q, T),$$

where $v(R^{(i)}|Q,T)$ is the student model's success rate conditioned on reasoning prefix $R^{(i)}$, and $v_{\text{base}}(Q,T)$ is its baseline success rate without it.

We report **Average CoT Effectiveness (ACE)** as the mean CER over all N cases:

$$\text{ACE} = \frac{1}{N} \sum_{i=1}^{N} R_{\text{cer}}^{(i)}.$$

4.3 Main Results

We report our method and baseline performance on BFCL and ACEBench with Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct (Team et al., 2024) and Llama3.2-3B (Dubey et al., 2024) as backbones in Table 2 and Table 3.

Table 2: The evaluation results on the BFCL (last updated November 19, 2025). For the best score, we use **boldface**, and for the second-best score, we use underline.

Model	Method	Overall			
		Non-live	Live	Irrelevance	Overall
Qwen2.5-1.5B-Instruct	Raw	30.75	37.12	78.62	48.83
	SFT	**33.63**	33.61	74.18	47.14
	Binary Reward	33.25	**38.41**	**98.51**	**56.72**
	ToolRL	31.19	35.80	95.15	54.04
	R2IF	29.38	37.62	96.17	54.39
Qwen2.5-3B-Instruct	Raw	30.50	35.09	92.44	52.68
	SFT	31.19	40.46	83.64	51.76
	Binary Reward	31.69	39.90	98.62	56.74
	ToolRL	28.00	27.68	**98.64**	51.44
	R2IF	**31.81**	**43.07**	98.25	**57.71**
Qwen2.5-7B-Instruct	Raw	55.50	56.07	87.85	66.47
	SFT	57.13	51.25	96.00	67.96
	Binary Reward	55.13	59.97	**96.85**	70.65
	ToolRL	57.50	61.64	95.57	71.57
	R2IF	**59.50**	**63.33**	93.59	**72.14**
Llama3.2-3B-Instruct	Raw	40.50	18.19	54.08	37.59
	SFT	60.19	33.45	75.61	56.41
	Binary Reward	65.44	37.06	90.27	64.26
	ToolRL	68.13	45.00	85.65	66.26
	R2IF	**69.44**	**56.23**	**90.95**	**72.21**

Result on BFCL. From an overall perspective, R2IF demonstrates clear and stable advantages on BFCL, exhibiting strong consistency across both model scales and model families. R2IF achieves the highest Overall accuracy on Qwen2.5-3B, Qwen2.5-7B, and Llama3.2-3B, and attains the second-best Overall result on Qwen2.5-1.5B.

As shown in the table 2, while maintaining strong performance on the Irrelevance subset, R2IF also brings consistent gains on Non-live settings. This indicates that the improvements enhance the end-to-end function-calling capability rather than a conservative rejection strategy.

Further comparison between the Non-live and Live splits reveals that the Live setting is substantially more challenging, with all models and training strategies suffering performance degradation due to the distribution shift from synthetic data to real user requests. Nevertheless, R2IF remains more competitive on Live-related metrics.

Result on ACEBench. Table 3 reports that from an overall perspective on ACEBench, R2IF achieves the highest Overall accuracy across all four backbones.

R2IF's gains are mainly concentrated in Atom and Single-turn, which more directly reflect tool decision quality. On Atom, R2IF clearly surpasses Raw and SFT for every backbone and generally outperforms Binary Reward and ToolRL, suggesting the proposed rewards improve atomic tool decisions. On Single-turn, R2IF also leads on

most backbones, with especially strong gains on Qwen2.5-7B and Llama3.2-3B, highlighting better support for end-to-end parameter construction.

Importantly, R2IF maintains strong rejection performance on the Irrelevant subset without trading off core tool-invocation accuracy. Its Overall improvements are therefore not driven by over-optimizing Irrelevant cases, but by more balanced gains on the primary invocation settings (Atom and Single-turn), aligning with ACEBench's goal of assessing comprehensive tool-use capability rather than single-metric optimization.

Why the Method with High Accuracy Performance gets low ACE. Comparing Table 2, Table 3, and Table 4, we observe some methods achieve high accuracy but relatively low ACE scores, such as Qwen2.5-3B with the Binary Reward method.

On BFCL, Qwen2.5-3B's Binary Reward method achieves high accuracy, especially in Non-live and Live settings, but its ACE score remains low, even negative. This suggests that while the model makes correct tool calls, its reasoning is shallow and lacks depth, relying on simple rules rather than thorough analysis.

Low ACE scores indicate inadequate reasoning support for tool-call decisions. While the model may produce correct results, its reasoning lacks transparency, which undermines support for more complex tasks. In contrast, R2IF not only focuses on accuracy but ensures reasoning stability and completeness.

4.4 Ablation Study

Table 5 presents ablation results showing how training pipeline components jointly enable robust, interpretable function-calling. Across Llama3.2-3B/Qwen2.5-7B and BFCL/ACEBench, removing any single component consistently degrades performance—indicating R2IF's gains stem from complementary, non-redundant signals.

Why we need SFT. SFT warm-start is more critical for smaller models. As shown in the results, removing SFT causes a larger accuracy drop for the 3B backbone. Compared to 7B models, smaller models possess weaker priors for instruction following and structured outputs, and thus are more prone to early-stage formatting instability, missing fields, or output drift, which reduces the fraction of executable trajectories. Moreover, binary rewards often act as a hard gate: malformed outputs receive little to no learning signal, pushing small models more frequently into a near-zero-reward regime and further impairing exploration efficiency. Therefore, SFT warm-start can establish basic output templates and tool-call syntax, enabling the policy to reliably produce evaluable trajectories; RL can then refine decision quality and reasoning–action alignment on top of this foundation.

Table 3: The evaluation results on the ACEBench

Model	Method	Atom	Single-turn	Irrelevant	Overall
Qwen2.5-1.5B-Instruct	Raw	43.70	13.50	30.00	29.07
	SFT	61.30	23.00	16.00	33.43
	Binary Reward	43.30	14.50	78.00	45.27
	ToolRL	52.30	26.00	88.00	55.43
	R2IF	64.70	20.50	88.00	**57.73**
Qwen2.5-3B-Instruct	Raw	52.00	25.00	90.00	55.67
	SFT	63.70	29.50	44.00	45.73
	Binary Reward	49.00	20.50	60.00	43.17
	ToolRL	58.00	31.50	96.00	61.83
	R2IF	70.70	26.00	90.00	**62.23**
Qwen2.5-7B-Instruct	Raw	57.30	40.00	88.00	61.77
	SFT	71.30	54.50	94.00	73.27
	Binary Reward	76.70	49.00	96.00	73.90
	ToolRL	76.30	56.50	96.00	76.27
	R2IF	78.00	57.50	96.00	**77.17**
Llama3.2-3B-Instruct	Raw	45.30	17.00	14.00	25.43
	SFT	67.30	34.50	54.00	51.93
	Binary Reward	72.00	47.50	90.00	69.83
	ToolRL	65.70	39.00	88.00	64.23
	R2IF	73.70	51.00	94.00	**72.90**

Effect of $\mathcal{R}_{\text{smv}}$. In contrast, the Specification–Modification–Value (SMV) reward primarily boosts fine-grained parameter decision accuracy. Ablating SMV consistently hurts overall accuracy and BFCL Live performance, reflecting its role in grounding reasoning to concrete parameter specifications and transformations. Moreover, removing SMV slightly increases Irrelevance scores in some cases, suggesting strict parameter-level supervision may marginally trade off with conservative rejection behavior. This highlights SMV targets argument construction precision rather than high-level tool selection.

Effect of $\mathcal{R}_{\text{cer}}$. The Chain-of-Thought Effectiveness Reward plays a critical role in stabilizing reasoning quality, particularly in challenging settings. Removing CER leads to a pronounced decline on BFCL Live and ACEBench Single-Turn subsets—known to require stronger generalization and precise reasoning under distribution shift. This indicates CER does not merely encourage verbose reasoning, but actively filters for reasoning prefixes useful for correct tool calls.

4.5 Further analysis

Table 4: Average CoT Effectiveness on BFCL and ACEBench. We test the ACE of the CoT generated during previous evaluations. The table reports the difference in success rates between the student model conditioned on the generated reasoning prefix and the baseline performance, across different models and methods. Positive values indicate that the reasoning improves tool-call performance, while negative values suggest it is not helpful.

Model	Method	BFCL				ACEBench			
		Overall	Non-live	Live	Irrelevance	Overall	Atom	Single Turn	Irrelevant
Qwen2.5-1.5B-Instruct	Raw	-0.14	-0.32	-0.26	0.17	-0.43	-0.64	-0.72	0.06
	SFT	-0.13	-0.31	-0.25	0.18	-0.48	-0.45	-0.69	-0.30
	Binary Reward	-0.15	-0.31	-0.22	0.09	-0.40	-0.69	-0.71	0.21
	ToolRL	-0.14	-0.16	-0.06	-0.18	-0.39	-0.68	-0.72	0.23
	R2IF	**-0.03**	-0.16	-0.05	0.13	**-0.34**	-0.45	-0.66	0.10
Qwen2.5-3B-Instruct	Raw	-0.07	-0.21	-0.09	0.09	-0.35	-0.54	-0.65	0.15
	SFT	-0.08	-0.12	-0.02	-0.09	-0.44	-0.42	-0.68	-0.21
	Binary Reward	-0.22	-0.57	-0.33	0.24	-0.41	-0.74	-0.75	0.25
	ToolRL	-0.23	-0.60	-0.32	0.22	-0.41	-0.71	-0.74	0.21
	R2IF	**0.02**	-0.11	-0.03	0.18	**-0.29**	-0.39	-0.68	0.21
Llama3.2-3B-Instruct	Raw	-0.20	-0.42	-0.27	0.09	-0.44	-0.67	-0.71	0.07
	SFT	-0.08	-0.08	-0.07	-0.10	-0.40	-0.43	-0.66	-0.10
	Binary Reward	-0.12	-0.30	-0.21	0.14	-0.39	-0.70	-0.71	0.23
	ToolRL	-0.18	-0.43	-0.30	0.20	-0.39	-0.70	-0.73	0.25
	R2IF	**0.05**	-0.05	0.07	0.14	**-0.28**	-0.38	-0.66	0.21

Table 5: Ablation results on BFCL and ACEBench. We remove individual components of R2IF, including the SMV reward (wo-smv), the CER reward (wo-cer), and the SFT warm-start (wo-sft).

Model	Method	BFCL				ACEBench			
		Overall	Non-live	Live	Irrelevance	Overall	Atom	Single Turn	Irrelevant
Llama3.2-3B-Instruct	R2IF	**72.21**	69.44	56.23	90.95	**72.90**	73.70	51.00	94.00
	wo-smv	69.26	67.69	52.54	87.55	72.40	73.70	45.50	98.00
	wo-cer	68.35	66.63	48.60	89.82	70.60	72.30	39.50	100.00
	wo-sft	66.00	65.00	45.46	87.55	67.33	65.00	45.00	92.00
Qwen2.5-7B-Instruct	R2IF	**72.14**	59.50	63.33	93.59	**77.17**	78.00	57.50	96.00
	wo-smv	71.99	60.19	59.85	95.94	76.90	81.70	53.00	96.00
	wo-cer	71.84	58.63	61.81	95.08	76.57	78.70	55.00	96.00
	wo-sft	68.91	57.56	55.33	93.84	74.93	76.30	52.50	96.00

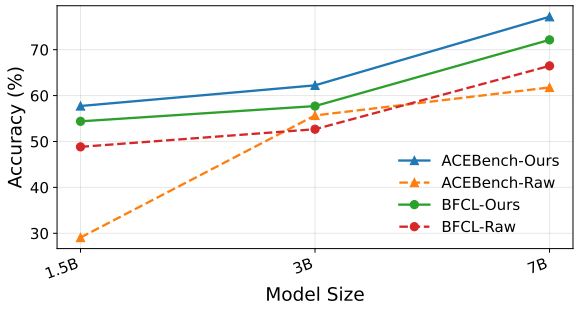

Figure 3: Scaling performance across model sizes using the Qwen2.5-Instruct series.

Scalability. Figure 3 summarizes how performance scales with backbone size on both ACEBench and BFCL. Overall, accuracy increases monotonically from 1.5B to 7B for both the raw instruct models and our trained models, indicating that our approach is compatible with standard scaling trends rather than trading off capacity for alignment. More importantly, our method consistently improves over the raw baselines across all sizes and both benchmarks.

Generalizability. Furthermore, our method exhibits strong *generalizability* across model families, not limited to the Qwen2.5 series. We observe consistent and substantial improvements on **Llama3.2-3B** as well. On BFCL, our approach boosts the Overall score from 37.59% to **72.21%** (+34.62%) while maintaining high Irrelevance performance. On ACEBench, we also improve the Overall accuracy. These results indicate that our reward design can transfer effectively across architectures and evaluation distributions.

5 Conclusion

We present R2IF, a reasoning-aware reinforcement learning framework that enhances the interpretability and effectiveness of function calling in LLMs. By aligning reasoning with tool-call decisions, R2IF ensures that predictions are based on valid

reasoning. Using a composite reward system, we strengthen the link between reasoning and decision-making. Our results demonstrate the value of structured reasoning supervision in improving both tool-call accuracy and reasoning stability, ensuring that outputs are both accurate and interpretable.

Limitations

Our framework has several limitations. First, the current evaluation of R2IF is limited to single-turn function query tasks, which restricts our understanding of its performance in multi-turn scenarios. In tasks involving multiple queries or interactions, where reasoning and function calls span across different steps, the model's ability to maintain coherent reasoning and accurate tool calls remains untested. This limitation suggests that the framework's generalizability to more complex, interactive environments is still unclear and requires further exploration.Second, the reward design of R2IF is highly tailored to specific function calling tasks, which may hinder its applicability to tasks that require different forms of reasoning or decision-making. Since the framework's rewards are optimized for particular task constraints and parameter specifications, its performance may vary significantly depending on the type of tool being invoked. As a result, R2IF's scalability and versatility in other LLM applications, where task structures or tools differ, may be limited.

Acknowledgments

This research is supported by National Science and Technology Major Project (No.2025ZD1606804), the Ministry of Industry and Information Technology of China, the Key Program of National Natural Science Foundation of China (No. 62432007) and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

References

Sanjeda Akter, Ibne Farabi Shihab, and Anuj Sharma. 2025. Inducing faithfulness in structured reasoning via counterfactual sensitivity. *Preprint*, arXiv:2509.01544.

Siddhant Bhambri, Upasana Biswas, and Subbarao Kambhampati. 2025. Do cognitively interpretable reasoning traces improve llm performance? *Preprint*, arXiv:2508.16695.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, Wulong Liu, Xinzhi Wang, Defu Lian, Baoqun Yin, Yasheng Wang, and Wu Liu. 2025. Acebench: Who wins the match point in tool usage? *Preprint*, arXiv:2501.12851.

DeepSeek-AI and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Bingguang Hao, ZengZhuang Xu, Maolin Wang, Yuntao Wen, Yicheng Chen, Cunyin Peng, Long Chen, Dong Wang, Xiangyu Zhao, Jinjie Gu, and 1 others. 2025. Funreason: Enhancing large language models' function calling via self-refinement multiscale loss and automated data refinement. *arXiv preprint arXiv:2505.20192*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2024a. Toolace: Winning the points of LLM function calling. *arXiv preprint arXiv:2409.00920*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. 2024b. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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 (ICML)*.

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. *arXiv preprint arXiv:2504.13958*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

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

Qwen Team and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3).

Jun Wang, Jiamu Zhou, Muning Wen, Xiaoyun Mo, Haoyu Zhang, Qiqiang Lin, Cheng Jin, Xihuai Wang, Weinan Zhang, and Qiuying Peng. 2024. Hammerbench: Fine-grained function-calling evaluation in real mobile device scenarios. *arXiv preprint arXiv:2412.16516*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. 2025. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*.

Xinyi Zheng, Ningke Li, Xiaokun Luan, Kailong Wang, Ling Shi, Meng Sun, and Haoyu Wang. 2025. Beyond correctness: Exposing llm-generated logical flaws in reasoning via multi-step automated theorem proving. *Preprint*, arXiv:2512.23511.

A Multi-turn Interactive Evaluation

To assess whether the proposed reward design generalizes beyond single-turn function calling, we conduct a multi-turn interactive case study on 50 samples, comparing Raw, SFT, Binary-Reward, ToolRL, and R2IF.

Table 6: The evaluation result on the multi-turn interactive case.

Model	Raw	SFT	Binary	ToolRL	**R2IF**
Llama3.2-3B	4.00	8.00	2.00	4.00	**10.00**
Qwen2.5-7B	8.00	10.00	14.00	**16.00**	**16.00**

Our method remains effective in multi-turn settings: it achieves the best performance on Llama3.2-3B and matches the strongest baseline on Qwen2.5-7B.

Overall, these results suggest that the benefits of R2IF are not limited to single-turn execution and persist when state updates accumulate over an interaction history.

B Dependency on Student Evaluator and Sampling Robustness of CER

B.1 Selection of student model

CER evaluates whether a reasoning prefix increases the downstream success rate of a student model. For this estimator to be meaningful, the student model must satisfy two conditions: (1) it must be able to produce executable tool calls when conditioned on a reasoning prefix, and (2) it must not systematically collapse to rejection behavior on non-irrelevance instances.

We evaluate three candidate student models on 200 samples: Qwen2.5-1.5B, Qwen2.5-3B, and Llama3.2-3B. Table 7 reports continuation-based success rates under reasoning prefixes.

Table 7: Validity check for student models.

Student Model	Non-Irrel.	Irrel.
Llama3.2-3B	**53.68**	**75.00**
Qwen2.5-1.5B	0.14	92.81
Qwen2.5-3B	0.73	98.75

The Qwen student models frequently default to the rejection response even when tool invocation is required, causing continuation-based success rates on non-irrelevance cases to degenerate to near-zero constants. Under this behavior, CER becomes largely uninformative. In contrast, Llama3.2-3B preserves meaningful continuation variability while

avoiding systematic refusal bias. We therefore adopt it as the student evaluator in our experiments.

B.2 Sampling Robustness of CER

We further examine whether CER is sensitive to sampling hyperparameters. Using the same student model and fixing the common instance set to 200 examples, we vary temperature, top-p, and the number of continuations K. Let C_0 denote the configuration used in the main experiments and other sampling configurations are listed in Table 8.

We then compare per-instance CER rankings against C_0.

Across configurations, CER exhibits strong ranking stability, limited numerical drift, and no dependence on a narrow temperature or top-p regime. Additional statistics further support this robustness: sign agreement ranges from 0.845 to 0.905, $\Pr(|\Delta| \leq 0.2) \approx 0.91\text{-}0.94$, and $\mathbb{E}[|\Delta|] \approx 0.052\text{-}0.076$.

Table 8: Sampling configurations for CER robustness analysis.

Configuration	Temperature	Top-p	K
C_0	0.4	1.0	5
C_1	0.4	0.9	5
C_2	0.7	1.0	5
C_3	0.7	0.9	5
C_4	0.4	1.0	10

Table 9: Ranking stability of CER under different sampling configurations.

Comparison	Spearman ρ	Kendall τ
C_0 vs C_1	0.944	0.897
C_0 vs C_2	0.915	0.854
C_0 vs C_3	0.896	0.833
C_0 vs C_4	0.936	0.884

C Incremental Contribution Beyond ToolRL

To isolate the contribution of CER and SMV beyond ToolRL-style outcome rewards, we replace the binary reward component in R2IF with ToolRL's correctness-plus-format reward, denoted as R2IF-toolrl.

As shown in Table 10, adding CER and SMV on top of ToolRL-style reward supervision yields consistent gains across backbones, which indicate that CER and SMV provide complementary process-level credit assignment beyond ToolRL-style outcome rewards.

Table 10: Incremental contribution of CER and SMV beyond ToolRL-style reward supervision.

Model	Method	BFCL	ACEBench
Llama3.2-3B	ToolRL	66.26	64.23
	R2IF-toolrl	66.73	72.50
	R2IF	**72.21**	**72.90**
Qwen2.5-3B	ToolRL	51.44	61.83
	R2IF-toolrl	57.14	61.73
	R2IF	**57.71**	**62.23**

D Computational Overhead Analysis

We analyze the computational overhead of our method from two perspectives: training cost and inference cost.

D.1 Training Overhead

We first measure the training overhead introduced by CER over 9 epochs and 36 steps. Table 11 reports the average training time per step.

Table 11: Average training time per step (seconds).

Model	With CER	Without CER
Llama3.2-3B	251.29	225.52
Qwen2.5-7B	612.64	595.19

Enabling CER increases per-step training time by 11.4% on Llama3.2-3B and 2.9% on Qwen2.5-7B. These results indicate that the computational cost is driven mainly by model scale, whereas CER introduces only a limited additional overhead.

D.2 Inference Overhead

We next examine the inference overhead by reporting average reasoning length on the BFCL test set (3,475 cases). Table 12 reports the average number of reasoning tokens across methods and backbones.

Table 12: Average reasoning length (tokens) on BFCL.

Model	Raw	SFT	Binary	ToolRL	R2IF
Llama3.2-3B	118.54	178.34	98.86	91.89	163.00
Qwen2.5-3B	245.33	376.92	142.83	147.37	278.03

The results in Table 12 suggest two points. First, the absolute reasoning length of R2IF remains moderate and stays within the same scale as SFT reasoning. Second, the increase relative to ToolRL is bounded: +71 tokens for Llama3.2-3B and +131 tokens for Qwen2.5-3B. This increase is expected, since the objective explicitly encourages exposing the chain

$$\text{Specification} \rightarrow \text{Modification} \rightarrow \text{Value},$$

which requires additional tokens to make otherwise implicit transformations explicit.

Overall, the additional inference cost remains controlled, while yielding more interpretable and decision-grounded reasoning for function calling.

E Human Evaluation of Reasoning Interpretability

To complement automatic evaluation, we conduct a human study of reasoning interpretability with the framework(Bhambri et al., 2025). We evaluate four dimensions: *Predictability*, *Comprehensibility*, *Interpretability*, and *Faithfulness*.

The study uses the Llama3.2-3B backbone and compares Raw, Binary-Reward, ToolRL, and R2IF. We recruited 8 evaluators, each of whom assessed 5 tasks under the Llama3.2-3B setting.

Table 13: Human evaluation results on reasoning interpretability.

Dimension	R2IF	ToolRL	Binary	Raw
Predictability	**4.850**	2.875	2.450	2.825
Comprehensibility	**4.825**	3.300	2.725	3.150
Interpretability	**4.800**	2.900	2.575	2.875
Faithfulness	**4.775**	2.650	2.275	2.775

As illustrated in Table 13, in all four dimensions, our method outperforms the baselines by a large margin. All pairwise comparisons between R2IF and the other methods are statistically significant ($p < 0.05$). In addition, the evaluation exhibits strong reliability, with Cronbach's $\alpha > 0.7$ for all dimensions. These results provide direct evidence that our method improves the interpretability of reasoning traces.

F Experiment

F.1 SFT Setting

We conduct supervised fine-tuning (SFT) using the TRL framework, and summarize the hyperparameter configuration in Table 14.

Table 14: Configuration for SFT training

Hyperparameter	Value
Train Batch Size	512
Max Prompt Length	3072
Max Response Length	1024
Learning Rate	1×10^{-5}
Total Epochs	2

F.2 GRPO Setting

The training was conducted using the Verl framework, and the configuration for GRPO training is summarized in Table 15. Below are the hyperparameters used for training:

Table 15: Configuration for GRPO training

Hyperparameter	Value
Train Batch Size	512
Validation Batch Size	128
Max Prompt Length	3072
Max Response Length	1024
Learning Rate	1×10^{-6}
PPO Mini Batch Size	128
GPU Memory Utilization	0.6
Number of Rollouts	5
Total Epochs	9

G Prompt

G.1 Prompt for LLM Inference

The System Prompt in Figure 4, with its predefined instructions and constraints, standardizes the model's behavior and output. It ensures consistent learning during training and uniform evaluation based on a unified set of rules.

G.2 Prompt for Construction of Ground-Truth Document

Under the prompt in Figure 5, a large language model is used to generate SMV annotations as the ground-truth document. Given the user query, tool documentation, and ground-truth tool-call result as input, the model produces parameter-level SMV annotations, including the relevant specification, necessary modification and final value for each argument.

Prompt for LLM Inference.

```
You are an expert in composing functions. You are given a question and a set of possible functions.
Based on the question, you will need to make one or more function/tool calls to achieve the purpose.

In each action step, you MUST:
1. Think about the 2-step reasoning process in the mind and enclosed your reasoning within
<reason></reason> tags.
    - Step1: Analyze the question and highlight the chosen function in the format of: #fun_name#
    - Step2: For each function you decide to call, you should analyze the value of each parameter
in the format of: - params_name1: analysis_process
2. Then, provide the final function call with function names and arguments in the format
of: <tool>[{"name": "func_name1", "arguments": {"params_name1": params_value1, "params_name2":
params_value2...}}, {"name": "func_name2", "arguments": {}}]</tool>
3. Make sure both the reasoning and the tool call steps are included together in one single reply.

Output format:
<reason>
Step1: ...
Step2:
For #fun_name1#:
- params_name1: analysis_process
...
</reason>
<tool>[{"name": "func_name1", "arguments": {"params_name1": params_value1, "params_name2":
params_value2...}}, {"name": "func_name2", "arguments": {}}]</tool>
You SHOULD NOT include any other text in the response.

Important Notes
1. If the given question lacks the arguments required by the function, point it out. If this
required parameter has enum option, you can choose a single unambiguous match from listed option.
2. If none of the function can be used, write it in <tool>None of function can be used</tool>.
```

Figure 4: Prompt for LLM Inference.

Prompt for Construction of Ground-Truth Document.

You are an expert in composing functions. You are given a question, a set of functions doc and a right reference of tool call.
You should explore the specification, modification and more argument value may exist for each toolcall in order.
And give me the reason why these specifications and modifications exist. no more than three sentences.
The specification are some special format requirements in the parameter description
- write no spec if here is no specification or not mentioned in the reason
- be careful with the e.g. tag that some parameters' description with it do not mean they have format specification
- no more than 15 words
- all the specification must be copied from description without any change

The modification is the action that convert the origin value (text from question) in the final state
- include change process from the original value to modified value
- write no modify if change nothing
- no more than 15 words

If there is a modification, then there must be a specification. If there is no modification, it doesn't necessarily mean there is no specification.

Note that the provided function is in Python 3 syntax.

The specification includes:
1 **units**: specify expected unit systems (e.g., SI units), allowable representations such as symbols vs. full names, and whether spacing is required between value and unit (e.g., "10 kg" vs "10kg").
2 **dates/times**: only choose one from following (e.g., YYYY-MM-DD, YYYY/MM/DD, DD/MM/YYYY, or "January 1, 2025").
3 **percentages**: specify whether values should be expressed as the percentage sign (e.g., "0.5" vs "50" vs "50%").
4 **place names / locations**: specify expected string patterns (e.g., "City, State" such as "San Francisco, CA" where applicable, or "City, Country" when no state exists), and indicate whether abbreviated forms of administrative regions should be used.
5 **precision**: specify how precision should be expressed (e.g., number of decimal places, significant digits).
......

Output in the json format:

```
[
    {
        "reason": "...",
    },
    {
       "name": "func_name1",
       "arguments": {
          "param_name1":{
             "specification": spec,
             "modification": mod,
          },
          "param_name2":{
             ....
          }
       }
    },
    {
       "name": "func_name2",
       "arguments": {
          ...
       }
    }
    ...
]
```

Figure 5: Prompt for Construction of Ground-Truth Document.