Tool Zero: Training Tool-Augmented LLMs via Pure RL from Scratch

Anonymous ACL submission

Abstract

Training tool-augmented LLMs has emerged as a promising approach to enhancing language models' capabilities for complex tasks. The current supervised fine-tuning paradigm relies on constructing extensive domain-specific datasets to train models. However, this approach often struggles to generalize effectively to unfamiliar or intricate tool-use scenarios. Recently, reinforcement learning (RL) paradigm can endow LLMs with superior reasoning and generalization abilities. In this work, we address a key question: Can the pure RL be used to effectively elicit a model's intrinsic reasoning capabilities and enhance the tool-agnostic generalization? We propose a dynamic generalization-guided reward design for rule-based RL, which progressively shifts rewards from exploratory to exploitative tool-use patterns. Based on this design, we introduce the *Tool-Zero* series models. These models are trained to enable LLMs to autonomously utilize general tools by directly scaling up RL from *Zero* models (i.e., base models without post-training). Experimental results demonstrate that our models achieve over 7% performance improvement compared to both SFT and RL-with-SFT models under the same experimental settings. These gains are consistently replicated across cross-dataset and intra-dataset evaluations, validating the effectiveness and robustness of our methods.

1 Introduction

Integrating LLMs with external tools has emerged as a pivotal advancement, significantly enhances their ability to address complex tasks (Qu et al., 2025; Wang et al., 2024). It opens up many practical uses across different fields. For example, it supports the automation of reasoning tasks (Jin et al., 2025; Manduzio et al., 2024), and enables Agent applications (Gunter et al., 2024; Chen et al., 2024). A tool-augmented model can respond to a user's query by invoking and executing external tools. In this paper, tools are used interchangeably with APIs, functions, and plugins.

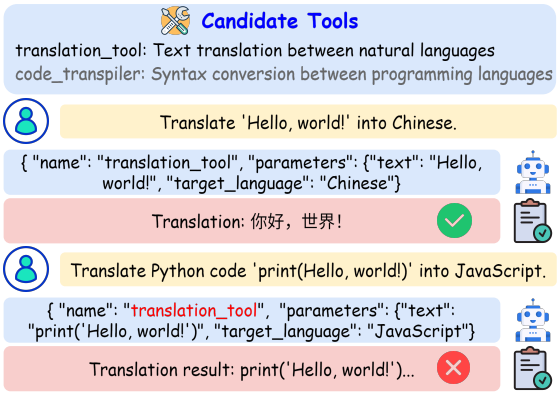

Figure 1: A response demonstration of a tool-augmented model trained in SFT paradigm. The model fails to recognize similar but unfamiliar task contexts (e.g., code transpilation), highlighting limited generalization to unseen tool-use scenarios.

Current approaches to enhance tool-use capability involve synthesizing extensive tool-use trajectories with advanced language models, followed by SFT on the generated data (Liu et al., 2024a; Lin et al., 2024; Chen et al., 2025). Under this paradigm, models achieves satisfactory performance in the scenarios that have the same distribution as the training data (Yan et al., 2024). However, these SFT-trained models primarily engage in imitating surface-level patterns rather than internalizing the reasoning process, tend to memorize the training trajectories rather than developing robust, intrinsic reasoning capabilities (Chen et al., 2025). Consequently, they exhibit limited generalization ability when applied to unseen scenarios, as elaborated in preliminary study in Section 3.

An illustrative example in Figure 1 demonstrates that while the model correctly addresses a natural language translation task, it fails to appropriately invoke tools for code transpilation. Specifically, the model interprets "translation" solely at the natural

language level, failing to recognize the code transpilation scenario implicit in the user's query. This mismatch underscores a critical limitation: current models lack the intrinsic reasoning abilities to discern nuanced task contexts. Enabling LLMs with genuine reasoning capabilities to overcome such generalization barriers has thus become an urgent research imperative.

Recent studies have demonstrated that simple rule-based R1-style RL (DeepSeek-AI, 2025), even without SFT, can significantly enhance LLMs' complex reasoning capabilities (Zeng et al., 2025a; Lu et al., 2025; Shen et al., 2025). This paradigm inspires us to extend pure RL to the tool learning domain, aiming to address the generalization limitations by eliciting models' intrinsic reasoning abilities. To this end, we propose a dynamic generalization-guided reward design for rule-based RL. This approach employs a progressive reward strategy: it first promotes early-stage exploratory behavior to cultivate intrinsic reasoning, then refines these capabilities into tool-use patterns focused on final-task precision. This design effectively resolves the exploration-exploitation dilemma in open-domain tool learning, bridging the gap between reasoning generalization and task-specific tool use.

To evaluate generalization, we conducted extensive experiments across diverse function-calling benchmarks. Results demonstrate that our proposed Tool-Zero 7B/32B models, trained using our method, significantly outperform both SFT models and RL-with-SFT baselines. For example, Tool-Zero-7B achieves a 7.14% performance improvement compared to SFT model ToolACE-8B. Notably, it also surpasses the RL-with-SFT model ToolRL-7B by 7.18%. Additionally, these gains are consistently replicated in both cross-dataset and intra-dataset evaluations.

2 Related Work

To contextualize our approach, we survey prior research on tool learning and its integration with large language models.

2.1 Tool Learning

Enhancing LLMs with external tools has emerged as a pivotal direction for addressing complex tasks in open domains (Qu et al., 2025; Wang et al., 2024). Typical applications include integrating LLMs with search engines (Zhang et al., 2024b; Lazaridou et al., 2022; Shuster et al., 2022), calculators (Nakano et al., 2021), and Python interpreters (Wang et al., 2024; Song et al., 2024; Chen et al., 2022). A dominant paradigm for equipping LLMs with external tools is imitation learning, where language models are trained via imitation on human-labeled datasets. This framework typically involves constructing large-scale supervised tool-use datasets (Prabhakar et al., 2025; Liu et al., 2024b,a) and applying either SFT (Zhang et al., 2024a,b; Qin et al., 2023) or direct preference optimization (DPO) reinforcement learning (Zeng et al., 2025b; Yu et al., 2024), enabling models to autonomously create and invoke tools. However, this paradigm faces challenges in enabling LLMs to generalize across diverse tools with varied argument structures and domains, highlighting a critical gap in tool-agnostic generalization.

2.2 Tool-Integrated Reasoning with Reinforcement Learning

RL has gained traction as a more scalable and generalizable training paradigm. Models like R1-Zero leverage group relative policy optimization (GRPO) (Shao et al., 2024) to unlock the model's reasoning capabilities at test time (DeepSeek-AI, 2025; Yu et al., 2025). This R1-style reasoning paradigm—marking a shift from train-time scaling to test-time scaling (Muennighoff et al., 2025; Xia et al., 2025)—has demonstrated success in mathematics (Shao et al., 2024), coding (Pan and Liu, 2025).

Recent studies (Jin et al., 2025; Qian et al., 2025) have explored unlocking tool-integrated reasoning for LLMs, with works like Torl (Li et al., 2025) and ReTool (Feng et al., 2025) achieving promising performance in mathematical tasks by integrating code tools. However, their training follows the SFT-then-RL paradigm and remains constrained to single-type tool-use scenarios. In contrast, our work aims to unlock the model's tool-agnostic (general-purpose tools) generalization capabilities via pure reinforcement learning scaled directly from *Zero* model.

3 Problem Statement and Analysis

Problem Formulation. We first provide the problem formulation of reasoning in tool augmented models. It formalizes the integration of external tools into the inference process to solve complex tasks. Given a tool set $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ and a

user query q, the reasoning trajectory up to step k is defined as:

$$\tau_k = [a_1(c_1), o_1], [a_2(c_2), o_2], \ldots, [a_k(c_k), o_k], \quad (1)$$

here, a_i denotes the model's reasoning action (natural language thought) at step i, $c_i \subseteq \mathcal{T}$ represents the subset of tools called at step i, and o_i denotes the observations received after tool execution, including environment and user feedback.

The model's policy is defined as $\pi : \tau_k \rightarrow a_{k+1}(c_{k+1})$. At each step $k+1$, the model must generate the next reasoning action a_{k+1}, select a tool subset $c_{k+1} \subseteq \mathcal{T}$, and formulate parameterized tool invocations for c_{k+1}. The goal is to enable LLMs with a generalized policy π that effectively addresses user queries by producing a sequence of action-observation pairs (a_t, o_t).

3.1 Preliminary Study

This section aims to show the generalization challenges faced by tool-augmented models trained in the SFT paradigm and presents the motivation of this paper. To this end, we conducted the following two preliminary studies:

(1) **Intra-Dataset Performance**. We compared two SFT-trained models: *ToolACE-8B* (Llama3.1-8b-inst finetuned on ToolACE (Liu et al., 2024a)) and *xLAM-7B-r* (Mistral-7b finetuned on xLAM (Liu et al., 2024b)), evaluated on the BECL benchmark (Yan et al., 2024) (comprising *Single-turn (Non-Live, Live)* and *Multi-turn* subsets). Notably, both training datasets use LLM-synthesized data to mimic real-world scenarios, with distributions aligned to BFCL-Live (details in Section 4.1). Results in Figure 2 show significant improvements on the Live metric (reflecting in-distribution performance), such as a improvement from 61.1 to 78.6. Conversely, exhibiting negligible gains or regressions on Non-Live and Multi-Turn subsets (e.g., 9.6 $\rightarrow$ 7.8). This suggests that SFT struggles with out-of-distribution generalization in open-domain settings. For instance, single-turn training data fails to transfer to multi-turn scenarios, and simple tool use patterns do not generalize to complex, interdependent tool chains.

(2) **Cross-Dataset Performance**. We extended our evaluation to diverse benchmarks (details in Appendix A.2), inspired by Lin et al., 2024. Notably, these benchmarks encompass varied tool-use scenarios(e.g., candidate tools, contextual domains, and invocation formats (JSON vs. Python code)). Results in Table 1 reveal inconsistent performance of existing tool-use models across benchmarks. For example, while xLAM-7B-fc achieved top performance on BFCL, it suffered significant degradation on two others, leading to the lowest overall average score. In contrast, the foundation models demonstrated more consistent cross-dataset performance. Therefore, this result highlighting a critical issue: SFT enhances in-distribution performance but weakens generalization to unseen scenarios. (e.g., novel tools, invocation formats).

In summary, our analysis reveals a fundamental trade-off in the SFT paradigm: while it enhances in-distribution tool-use accuracy, it severely limits generalization to unseen scenarios. To address this, we propose adopting a pure RL framework for tool learning, designed to dynamically balance exploration of new tool interactions with exploitation of task-relevant patterns.

4 Method

In this section, we provide a detailed introduction to our method. Figure 3 shows the overall architecture of our proposed dynamic Generalization-Guided reward strategy for GRPO (GG-GRPO).

4.1 Training Data Preparation

The following data are utilized in RL training. (1) ToolACE (Liu et al., 2024a): It is a general-purpose tool-use dataset, where the model learns when to invoke tools and when to respond directly, thereby enhancing decision-making in multi-step interactions. (2) xLAM (Liu et al., 2024b): This is a compositional dataset that requires one or multiple tool calls per turn. It encourages the model to reason about tool dependencies and actively plan diverse tool-calling actions. We also include an irrelevance-augmented subset[1] originating from xLAM.

Data Filtering. Since these datasets are generated by potentially unstable LLMs, they often contain non-standard formats for Abstract Syntax Tree (AST) parsing and GRPO training. We standardize the data by filtering out samples with invalid (1) tool calls (i.e., can be parsed in JSON or Python code format), (2) candidate tools (can be parsed in JSON format).

Multi-Turn Augment. Due to the lack of multi-turn tool-calling trajectories in xLAM, we have augmented it. The following four strategies were employed: (1) single-turn combination: concate-

[1] https://huggingface.co/datasets/MadeAgents/xlam-irrelevance-7.5k

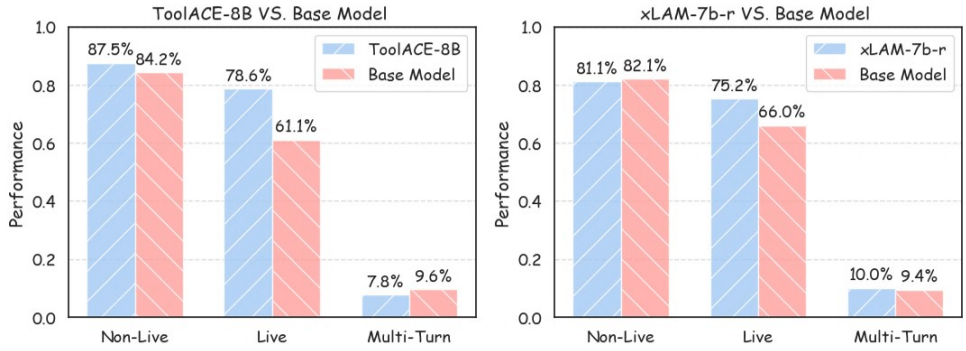

Figure 2: Intra-Dataset Performance. The improvement on metric (Live) with training-distributed data is significantly greater than that on other metrics. SFT struggles with out-of-distribution generalization in open-domain settings.

Models	BFCL-v3	API-Bank	SealTool	Tool-Alpaca	Nexus Raven	Avg.
♣ Granite-20B-FunctionCalling	③49.31	②68.53	①92.74	②58.03	①75.15	①**68.75**
♣ Gorilla-OpenFunctions-v2-7B	②52.10	③62.50	②91.12	③51.30	②68.46	②65.09
♣ xLAM-7B-fc	①**54.75**	①72.45	③76.90	①59.00	③57.50	③64.12
♠ Llama-3.1-8B-Instruct	③50.87	③69.92	③89.27	③59.36	③67.30	③67.34
♠ Qwen2.5-7B-Instruct	②53.69	①70.76	②91.07	②60.24	②72.24	②69.64
♠ GPT-3.5-Turbo-0125	①**53.91**	②70.71	①93.51	①62.50	①82.86	①**72.62**

Table 1: Cross-Dataset Performance of SFT models (♣) and foundation models (♠). Smaller ranking numbers (circled numbers) in each column indicate larger values. Inconsistent performance of SFT models across benchmarks, indicating SFT enhances in-distribution performance but weakens generalization to unseen scenarios.

nate related single-turn dialogs into multi-turn sequences. (2) tool removal: randomly remove one tool and reintroduce it in subsequent turns. (3) parameter clarification: randomly mask a parameter value to prompt user clarification. (4) result validation: randomly remove tools, delete parameters, or alter values in ground-truth answer to simulate user challenge the response. We present the statistical information of the proceeding data in Table 2.

	xLAM		ToolACE	
	Single-T.	Multi-T.	Single-T.	Multi-T.
Raw Data	67500	0	102154	2000
Multi-Aug.	0	10254	0	0
After	77754	10254	97300	1966

Table 2: Data statistics for xLAM and ToolACE in data processing.

Data Mask. Recent studies (Lin et al., 2024; Chen et al., 2025) have shown that naming preferences in tool descriptionss can significantly degrade model robustness when testing environments diverge from training conventions. To mitigate this issue, we adopt a name-masking strategy aligned with Lin et al., 2024, which masks function names (e.g., `calculate_sum` → `func_1`) and parameter names (e.g., `input_list` → `param_1`). It redirects the model's attention to tool descriptions and argument semantics, reduces overfitting to superficial naming patterns, thereby improving tool-agnostic reasoning in open-domain settings.

4.2 Generalization Guided Reward Design

To enhance tool generalization within the GRPO framework, we introduce a *dynamic generalization-guided reward design* that combines flexible exploration with structured convergence. Building on prior rule-based reward mechanisms (Qian et al., 2025; Li et al., 2025; Jin et al., 2025), our formulation decomposes the total reward $\mathcal{R}_{\text{final}}$ into two components:

$$\mathcal{R}_{\text{final}} = \mathcal{R}_{\text{format}} + \mathcal{R}_{\text{generalize}}, \quad (2)$$

where $\mathcal{R}_{\text{format}}$ enforces tool invocation format correctness, and $\mathcal{R}_{\text{generalize}}$ drives generalization through a *progressive reward strategy*.

Our strategy balances initial model exploration with final task precision via two stages:

1. General Rule-Based Reward. During early training iterations, we use a lenient, fine-grained reward to elicit the model's inherent generalization capabilities. This reward (r_{general}) measures the semantic overlap (e.g., tool name, argument

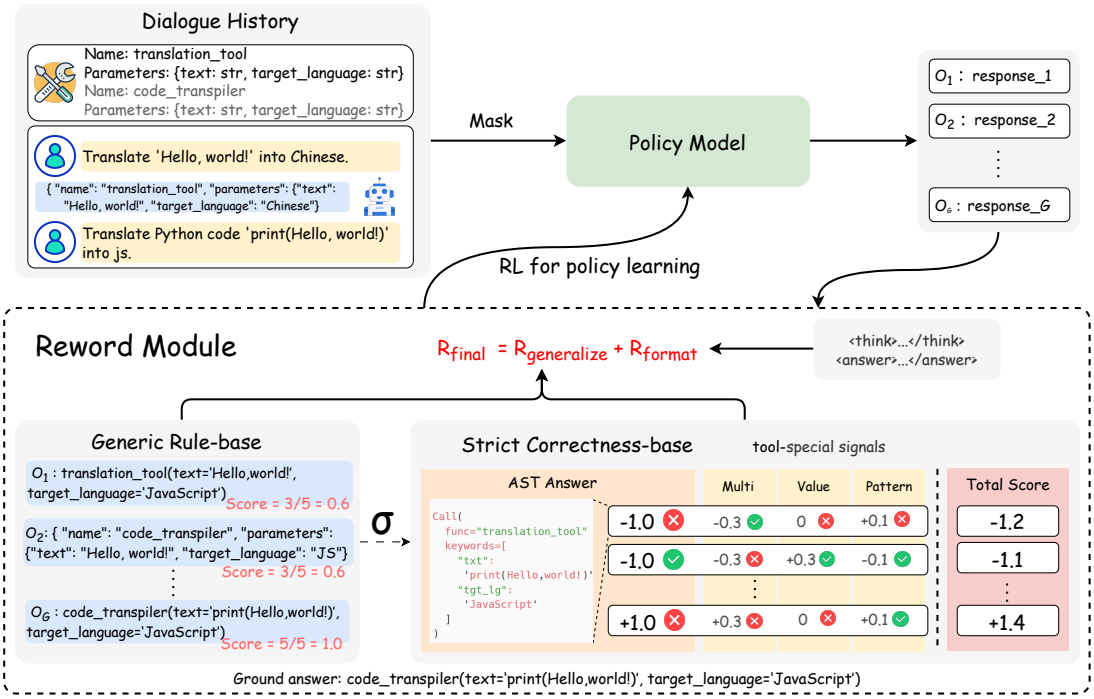

Figure 3: The overall architecture of *GG-GRPO* introduces a dynamic generalization-guided reward design for rule-based RL. It progressively shifts the reward mechanism from a fine-grained generic reward to a strict answer correctness reward.

name and value) between the model's response y and the ground-truth answer y^* by splitting both into tokenized elements using delimiters, e.g., `()[],.: '"=`, forming sets $\mathcal{Y} = \{\text{tokens}(y)\}$ and $\mathcal{Y}^* = \{\text{tokens}(y^*)\}$. The overlap rate is then calculated as:

$$r_{\text{general}} = r_{\text{overlap}} = \frac{|\mathcal{Y} \cap \mathcal{Y}^*|}{|\mathcal{Y}^*|}. \tag{3}$$

This allows the model to receive partial credit for incomplete but semantically relevant responses, encouraging broad exploration of tool-use patterns during low-capability phases.

2. Strict AST-Based Reward. As training progresses, we transition to a strict *Abstract Syntax Tree (AST)-based check* to enforce task-specific tool integration. This stage verifies the structural and semantic correctness of tool invocations (e.g., API argument validity, multi-tool dependency chains) by comparing the generated tool call syntax against a reference AST $\mathcal{T}^*$:

$$r_{\text{ast}} = \begin{cases} 1 & \text{if AST}(y) \equiv \mathcal{T}^*, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

This ensures the model converges to precise, tool-agnostic reasoning that adheres to complex tool specifications.

To further enhance the model's tool-integrated reasoning in context, we incorporate three *tool-specific feedback* signals into the strict reward $\mathcal{R}_{\text{strict}}$: (1) **multi**-tool collaboration: $+0.3$ reward for reinforcing correct collaborative tool usage patterns in multi-step tool chains or dialogs. (2) parameter **value** error: $+0.3$ penalty per invalid parameter value to enforce exploring precise context grounding because it demands high-order reasoning from context (Zeng et al., 2025b; Lin et al., 2024). (3) call **pattern** consistency: ± 0.1 reward/penalty for matching/deviating from the reference format, to accommodate different tool invocation formats (i.e., JSON vs. code).

Switching Trick. The dynamic shift from a general reward r_{general} to a strict reward r_{strict} is governed by a sigmoid-based decay function:

$$\mathcal{R}_{\text{generalize}} = \sigma(t, m) \cdot r_{\text{strict}} + (1 - \sigma(t, m)) \cdot r_{\text{general}}, \tag{5}$$

where $\sigma(t, m) = \frac{1}{1+e^{-\kappa(t-m)}}$ is a sigmoid function with steepness κ, t and m are the current training step and transition midpoint, respectively. This automated transition trick can avoid abrupt reward changes that may destabilize training.

Format Reward. The format reward $\mathcal{R}_{\text{format}} \in \{0, 1\}$ checks whether the model output contains all required special tokens in the correct order (i.e., `<think>...</think><answer>...</answer>`).

Overall, our strategy first nurtures broad generalization capabilities and then refines them into structured tool-use behaviors, effectively addressing the exploration-exploitation dilemma in open-domain tool learning.

4.3 RL Training with Generalization-guided Reward

To train LLMs from *Zero* model through scaling reinforcement learning, we employ Group Relative Policy Optimization (GRPO) (DeepSeek-AI, 2025; Shao et al., 2024), that unlocks the model's reasoning capabilities at test time.

GRPO foregoes the critic model and estimates the baseline from group scores instead. For each question q, GRPO generates G completions $\{o_1, o_2, \ldots, o_G\}$ using $\pi_{\theta_{\text{old}}}$, then optimizes π_θ by maximizing the following objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}\sim\pi_{\theta_{\text{old}}}} \left\{ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \min\left[\rho_{i,t}A_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)A_i\right] \right\} \tag{6}$$

where the importance ratio $\rho_{i,t}$ is defined as:

$$\rho_{i,t} \triangleq \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, \tag{7}$$

here, ϵ is a hyperparameter. Following Yu et al., 2025, we remove the KL divergence regularization from the GRPO objective. And A_i is the advantage computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the completions within each group:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \ldots, r_G\})}{\text{std}(\{r_1, r_2, \ldots, r_G\})}. \tag{8}$$

In the reward design of GRPO, we replace the rule-based accuracy reward function with a generalization-guided reward (GG-GRPO) for more effective and adaptive reward computation. The new reward formula is expressed as follows:

$$r_i = \mathcal{R}_{format}(o_i) + \mathcal{R}_{generalize}(o_i). \tag{9}$$

where, $\mathcal{R}_{format}(o_i)$ denotes format reward of o_i response, $\mathcal{R}_{generalize}(o_i)$ denotes dynamic generalization guided reward.

5 Experiments

In this section, we show the superiority of our method in performance and robustness across various benchmarks, and in-depth analysis to verify the effectiveness of our method.

5.1 Experimental Setup

In the experiment, we employ the Qwen2.5-7B Base and Qwen2.5-32B Base as *Zero* model, and train with GG-GRPO to get our Tool-Zero-7B and Tool-Zero-32B respectively[2]. More details in Appendix A.

Evaluation Dataset. The *BFCL* evaluates the LLMs ability to invoke functions in the real-world by actually triggering the API call and comparing responses, provides a comprehensive dataset comprising 4k+ instances (updating), consisting of Non-live, Live (with user-contributed complex tools avoiding contamination), Multi-turn subset. Other benchmarks, namely *API-Bank* (Li et al., 2023), *Nexus Raven* (Srinivasan et al., 2023), *Tool-Alpaca* (Tang et al., 2023), and *Seal-Tools* (Wu et al., 2024), are elaborated in Appendix A.2.

Baselines (1) *Vanilla Model*: the original model without additional training (e.g., Llama3.1-series, Qwen2.5-series). (2) *SFT Model*s: instruct models fine-tuned on supervised data, to assess whether GRPO training outperforms standard SFT, including ToolACE-8B (trained in ToolACE), xLAM-series (trained in xLAM)(Zhang et al., 2024a), and Hammer-series (trained on xLAM with function mask to enhance generalization) (Lin et al., 2024). (3) *API-based* closed-source models (e.g., GPT-series, Gemini-series). (4) *R1-like Model*: models trained using GRPO with SFT as the RL paradigm, such as QwQ-32B (Team, 2025), Tool-N1 series (single turn tool-use models trained in mixed ToolACE and xLAM data) (Zhang et al., 2025), and ToolRL(trained in subset of mixed ToolACE and xLAM data) (Qian et al., 2025). Instead, our Tool-Zero series trained with pure RL without SFT (i.e., R1-Zero).

5.2 Overall Performance

Results on BFCL. Table 3 shows the evaluation results, covering three subset metrics. We observe that SFT models like ToolACE-8B and xLAM-7b-r perform well on Live (data with the same distribution as training data) due to domain-specific training but exhibit poor generalization in out-of-distribution metrics (e.g., Multi-Turn). In contrast, Tool-Zero series models outperform others across all metrics. For instance, Tool-Zero-7B achieves 13.32 and 19.32 improvement in Live and Multi-Turn, respectively, compared to Qwen2.5-7B-Instruct.

[2]trained with ToolACE dataset in the main experiment

Type	Model	Non-Live	Live	Multi-Turn	**Overall Acc**
♣Vanilla	Llama-3.1-8B-Instruct	84.21	61.08	9.62	50.87
	Qwen2.5-7B-Instruct	86.46	67.44	7.62	53.69
	Qwen2.5-32B-Instruct	85.81	74.23	17.75	59.67
♦SFT	Hammer2.1-7b	<u>88.65</u>	75.11	23.50	61.83
	ToolACE-8B	87.54	78.59	7.75	58.42
	xLAM-7b-r	81.06	75.22	10.00	54.75
♥API-based	GPT-3.5-Turbo-0125	83.94	64.02	19.50	53.91
	GPT-4o-mini-2024-07-18	85.21	74.41	34.12	64.10
	GPT-4o-2024-11-20	88.10	<u>79.83</u>	<u>47.62</u>	<u>72.08</u>
	Gemini-2.0-Flash-001	84.90	79.12	17.88	60.42
	Gemini-2.0-Pro-Exp-02-05	83.94	78.50	20.75	61.55
♠R1-like	DeepSeek-R1	87.35	74.41	12.38	56.89
	QwQ-32B	86.48	75.48	2.12	53.93
	Tool-N1-7B*	89.25	80.38	-	-
	Tool-N1-14B*	90.52	81.42	-	-
	ToolRL-7B	82.21	74.90	18.12	58.38
	Tool-Zero-7B	88.98	80.76	25.93	65.22
	Tool-Zero-32B	**90.76**	**82.43**	**28.18**	**67.12**

Table 3: Comparison on the BFCL-v3. *Overall Acc* denotes the average performance on three subsets. * indicates single-turn tool use models, and multi-turn results are not reported. **Bold** for best performance in R1-like models and <u>underline</u> for best performance in the other types.

Models	BFCL-v3	API-Bank	SealTool	Tool-Alpaca	Nexus Raven	Avg.
♣ Qwen2.5-32B-Instruct	③59.67	③75.87	②93.08	①65.16	②89.12	③76.58
♦ ToolACE-8B	④58.42	⑤61.74	③74.74	⑤58.83	④63.15	④63.38
♦ xLAM-7B-fc	⑤54.75	④72.45	⑤76.90	④59.00	⑤57.50	⑤64.12
♦ Hammer2.1-7b	②61.83	①81.45	①94.94	②64.60	③84.35	②77.43
♥ GPT-4o-2024-11-20	①72.08	②80.52	④90.63	③62.37	①90.19	①79.16
♠ DeepSeek-R1	④56.89	③71.22	④89.97	②65.75	②82.88	③73.34
♠ QwQ-32B	⑤53.93	④70.29	③92.94	④62.29	③63.61	④68.61
♠ ToolRL-7B	③58.38	⑤69.92	⑤89.27	⑤59.36	③67.30	⑤67.34
♠ Tool-Zero-7B	②65.22	②79.85	②94.73	③63.71	③82.74	②77.32
♠ Tool-Zero-32B	①67.12	①81.63	①95.16	①64.38	①85.33	①78.99

Table 4: Comparison on more benchmarks. Rankings within each column are shown with circled numbers, where smaller numbers indicate larger values. Tool-Zero demonstrate better performance across multiple benchmarks consistently.

Pure RL paradigms outperform the SFT-then-RL approach. For instance, among R1-like models, Tool-Zero-7B surpasses DS-R1 by +8.33 and ToolRL-7B by +6.84. This indicates RL better elicits intrinsic reasoning abilities from *Zero* model, whereas SFT merely focuses on mimicking superficial patterns. Notably, compared to SFT models, models trained with GRPO (Tool-N1, ToolRL, Tool-Zero) perform comparably on Live and better on Non-live and multi-turn tasks. These results confirm that the RL paradigm is more effective for enhancing tool-integrated reasoning.

Results on More Benchmarks. Table 4 presents the results. Across different benchmarks, SFT models show inconsistent performance, while GPT-4o performs best. Notably, Hammer2.1-7b exhibits relatively consistent performance, attributed to its function masking techniques. Compared to SFT and R1-like models, Tool-Zero models demonstrate significantly more stable performance, highlighting the robustness of GG-GRPO. These findings indicate that our method generalizes effectively across various tool-use scenarios, offering new avenues for enhancing the tool-integrated reasoning capabilities of LLMs.

5.3 Experimental Analysis

5.3.1 Ablation Study

We conduct an ablation study for GG-GRPO, which comprises the progressive reward strategy (PRS), three tool-specific signals (multi-tool, value error, call pattern), and the tool mask. Using the Vanilla model Qwen2.5-7B-inst, we compare model training via SFT and pure GRPO train-

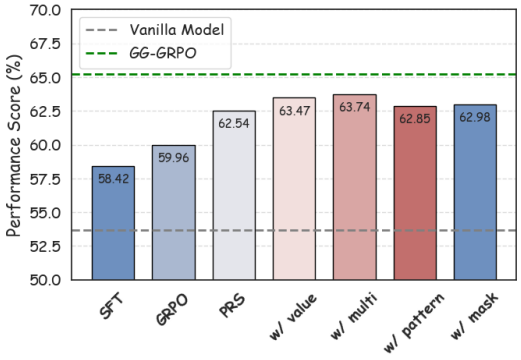

Figure 4: Ablation study results for GG-GRPO on BFCL benchmark overall performance.

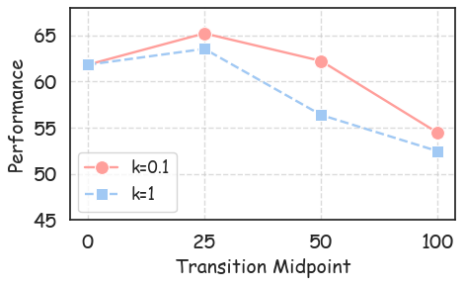

Figure 5: Hyperparameter analysis for progressive reward strategy on BFCL benchmark overall performance.

ing with same training data ToolACE. The results are presented in 4. We observe that GG-GRPO achieved a +5.26 improvement compared to GRPO, and a +6.8 mprovement compared to SFT. Experimental results demonstrate that all components contribute significantly to model performance. Among them, multi-tool and value error signals yield more substantial improvements compared to call pattern signals and the tool mask.

Additionally, we conduct a hyperparameter ablation study on the *progressive reward strategy* by varying two key parameters: transition midpoint $t_m \in \{0, 25, 50, 100\}$ and steepness factor $\kappa \in \{0.1, 1\}$ (controlling reward transition slope). Results (Figure 5) show that a smaller transition midpoint ($t_m = 25$) yields the best performance, while larger values ($t_m \geq 50$) lead to degradation. This aligns with prior observations (Pan et al., 2024; Zhang et al., 2025) that excessive exploration in fine-grained schemes may induce reward hacking and overfitting to superficial cues. Also, a lower steepness factor consistently outperforms, indicating that gradual reward shaping stabilizes training. These findings validate the design choices in GG-GRPO's progressive reward mechanism.

Models	Non-live	Live	Multi-turn
In ToolACE			
w/ qwen2.5-7b	88.98	**80.76**	**25.93**
w/ qwen2.5-32b	90.76	82.43	28.18
w/ qwen2.5-7b-inst	**89.39**	79.39	21.74
w/ qwen2.5-7b-coder	88.94	80.12	24.38
In xLAM			
w/ qwen2.5-7b	**87.15**	**76.93**	**32.38**
w/o MT-Aug.	87.04	74.79	16.18
w/ qwen2.5-7b-inst	85.28	75.32	29.47

Table 5: The result of data & backbones generalizability analysis, `MT-Aug.` typos multi turn augment in Section 4.1.

5.3.2 Training Data & Backbones Generalizability

To further validate the effectiveness of the proposed methods, we investigated the performance of our GG-GRPO across different datasets and backbone language models. As shown in Table 5, the experimental results demonstrate that training with the Base model consistently yields better performance across various training datasets compared to the Instruct model. This indicates that models with stronger instruction-following capabilities do not necessarily bring greater training benefits to tool-augmented models in RL. We attribute this to the Base model's higher plasticity, which more easily elicits intrinsic reasoning abilities. Additionally, when trained on different xLAM datasets, it also achieves consistently strong performance. Furthermore, through ablation experiments on Multi-Turn Augment in xLAM, we observed a significant increase in results from 16.18 to 32.28, highlighting the effectiveness of this augmentation strategy.

6 Conclusion

This study firstly extends a pure rule-based RL paradigm in tool-augmented models. Designing a dynamic generalization-guided reward to tackle the generalization limitations. By fostering intrinsic reasoning through progressive exploration-exploitation strategies, our approach reduces reliance on task-specific data and enhances tool-agnostic adaptability. Across diverse benchmarks, Tool-Zero models outperform SFT and RL-with-SFT baselines. These results validate RL's potential for scalable, autonomous tool learning in LLMs, advancing versatile AI agents for open-domain tasks.

7 Limitaiton

While our study has achieved notable advancements, it is important to acknowledge several limitations that could be addressed in future work. (1) The applicability of the pure RL paradigm across diverse backbone model sizes remains uninvestigated; it may be ineffective for smaller models lacking intrinsic reasoning capacity or extremely large models with distinct optimization dynamics. Further exploration of model size–RL performance relationships is needed to validate generalizability. (2) The progressive reward-switching strategy, though effective for generalization, introduces additional computational costs during the RL training phase, particularly for large models (e.g., Tool-Zero-32B). This limits scalability on resource-constrained hardware without further optimization. We will address these limitations in our future work.

References

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, et al. 2025. Acebench: Who wins the match point in tool learning? *arXiv preprint arXiv:2501.12851*.

Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2024. Octopus: On-device language model for function calling of software apis. *arXiv preprint arXiv:2404.01549*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. 2024. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, et al. 2024. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. 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, et al. 2024b. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025. Ui-r1: Enhancing action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*.

Graziano A Manduzio, Federico A Galatolo, Mario GCA Cimino, Enzo Pasquale Scilingo, and Lorenzo Cominelli. 2024. Improving small-scale large language models function calling for reasoning tasks. *arXiv preprint arXiv:2410.18890*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

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

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. 2024. Feedback loops with language models drive in-context reward hacking. *arXiv preprint arXiv:2402.06627*.

Zhenyu Pan and Han Liu. 2025. Metaspatial: Reinforcing 3d spatial reasoning in vlms for the metaverse. *arXiv preprint arXiv:2503.18470*.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, et al. 2025. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*.

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*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

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

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. 2025. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. 2022. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*.

Linxin Song, Jiale Liu, Jieyu Zhang, Shaokun Zhang, Ao Luo, Shijian Wang, Qingyun Wu, and Chi Wang. 2024. Adaptive in-conversation team building for language model agents. *arXiv preprint arXiv:2405.19425*.

Venkat Krishna Srinivasan, Zhen Dong, Banghua Zhu, Brian Yu, Damon Mosk-Aoyama, Kurt Keutzer, Jiantao Jiao, and Jian Zhang. 2023. Nexusraven: a commercially-permissive language model for function calling. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Mengsong Wu, Tong Zhu, Han Han, Chuanyuan Tan, Xiang Zhang, and Wenliang Chen. 2024. Seal-tools: Self-instruct tool learning dataset for agent tuning and detailed benchmark. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 372–384. Springer.

Shijie Xia, Yiwei Qin, Xuefeng Li, Yan Ma, Run-Ze Fan, Steffi Chern, Haoyang Zou, Fan Zhou, Xiangkun Hu, Jiahe Jin, et al. 2025. Generative ai act ii: Test time scaling drives cognition engineering. *arXiv preprint arXiv:2504.13828*.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Berkeley function calling leaderboard.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Zhicheng Guo, Jingtao Zhan, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2024. Steptool: A step-grained reinforcement learning framework for tool learning in llms. *arXiv preprint arXiv:2410.07745*.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 2025a. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. `https://hkust-nlp.notion.site/simplerl-reason`. Notion Blog.

Yirong Zeng, Xiao Ding, Yuxian Wang, Weiwen Liu, Wu Ning, Yutai Hou, Xu Huang, Bing Qin, and Ting Liu. 2025b. Boosting tool use of large language models via iterative reinforced fine-tuning. *arXiv preprint arXiv:2501.09766*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Awalgaonkar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. 2024a. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*.

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

Shaokun Zhang, Jieyu Zhang, Dujian Ding, Mirian Hipolito Garcia, Ankur Mallick, Daniel Madrigal, Menglin Xia, Victor Rühle, Qingyun Wu, and Chi Wang. 2024b. Ecoact: Economic agent determines when to register what action. *arXiv preprint arXiv:2411.01643*.

A Details of Experimental Setup

A.1 The Implementation Settings

The experiments were executed using the publicly accessible training framework MindSpeed-RL[3], an end-to-end reinforcement learning acceleration framework based on the Ascend ecosystem. Key hyperparameters included: $\kappa = 0.1$, temperature parameter for exploration-exploitation trade-off; Transition midpoint set to 25 (defining the inflection point in reward function scheduling).

For all the tool calls in the dataset, we use a hybrid format combining JSON structure and Python code snippets was adopted to encode various tool call format. For the GG-GRPO (a variant of GRPO) training, model training can be done within 28 hours with 5*8 Ascend 910b NPUs per run with the following hyperparameters:

Category	Hyperparameter
Data Configuration	
Global Batch Size	128
Max Prompt Length	2048
Max Response Length	2048
Optimization	
Learning Rate	5e-7
LR Decay Style	cosine
Mini Batch Size	1024
Tensor Model Parallel Size	4
KL Loss Used	False
ϵ	0.2
Rollout Configuration	
Rollout Name	vllm
GPU Memory Utilization	0.9
Number of Rollouts	4
Temperature	0.8

Table 6: Configuration for GG-GRPO training.

A.2 Benchmark & Metric Details.

The BFCL is an evolving benchmark. For our study, we utilized the version checked out on February 26, 2024. Other benchmarks include: (1) API-Bank with 314 tool-use dialogues and 753 API calls, evaluating known API invocation (L-1) and candidate list retrieval/calling (L-2), we report their average result in evaluation; (2) Nexus Raven API Evaluation offering 318 test examples across 65 APIs for function-calling assessment; (3) Tool-Alpaca's 271 synthetic tool-use instances in 50 categories (100 simulated tests used); (4) Seal-Tools, a recent benchmark with 4,076 auto-generated APIs across life domains. The BFCL assesses models using Abstract Syntax Tree Evaluation and Executable Function Evaluation Accuracy, and the other benchmarks assesses models using Function and Parameter matching F1 score (Lin et al., 2024).

[3]https://gitee.com/ascend/MindSpeed-RL

A.3 System Thinking Template

We adopt a lightweight prompting schema to elicit tool-use capabilities from the LLM, drawing inspiration from prior work (DeepSeek-AI, 2025; Face, 2025). As illustrated in Figure 6, the template explicitly instructs the model to encapsulate intermediate reasoning within `<think>...</think>` tags, followed by the final answer enclosed in `<answer>...</answer>` tags. Tool call specifications are embedded within the answer section using `<tool_call>...</tool_call>` markup. By allowing the model greater freedom in articulating its reasoning process, we aim to enhance generalization across diverse tool integration scenarios. Additionally, this design facilitates seamless adaptation to complex tool-augmented reasoning tasks.

System Prompt for Training

A conversation between User and Assistant, the user asks a question, and the Assistant solves it.
The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.
The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively,
i.e., <think> reasoning process here </think><answer> answer here </answer>.

You are an expert in composing functions, 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.
1. If none of the function can be used, point it out.
2. If the given question lacks the parameters required by the function, also point it out.
3. You should only return the function call in tools call sections.

If you decide to invoke any function(s), MUST use the format:

```
<tool_call>
[func_name1(params_name1=params_value1, ...), func_name2(params)]
</tool_call>
```

Here is a list of functions in JSON format that you can invoke: `{{Tool List}}`

Figure 6: The system think prompt with Python code format for RL Training. The prompt guides the LLM to explicitly separate reasoning process and answer.