# WHY MULTI-STEP TOOL-USE REINFORCEMENT LEARNING COLLAPSES AND HOW SUPERVISORY SIGNALS FIX IT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Tool use enables large language models (LLMs) to perform complex tasks, and recent agentic reinforcement learning (RL) methods show promise for improving these capabilities. However, RL alone often leads to instability or limited gains in tool-use tasks. We identify a recurring failure mode, termed *structural collapse*, in which performance drops sharply as valid tool calls become malformed control-token sequences. We analyze this failure at the token level and find that a small set of control tokens appears repeatedly across trajectories, accumulating reward-weighted training signals. They accumulate greater absolute advantage per token type than ordinary tokens, and their probabilities change much more sharply before collapse. Changing the invocation format recovers some performance, suggesting that part of the tool-use capability remains accessible. We further show that masking the direct policy-gradient (PG) terms at control-token positions can mitigate collapse. Building on these findings, we systematically compare supervisory signals, including off-policy supervision, hint-based guidance, and erroneous trajectory supervision, under synchronous and interleaved training schemes. We also introduce process reflection supervision, which uses teacher-generated error analyses and related examples to address the model's evolving failures. Across the evaluated models, interleaving supervised fine-tuning (SFT) with RL often improves stability, although the most effective supervision strategy varies by model. These improvements do not consistently transfer to format and content out-of-distribution (OOD) evaluation. Further analyses examine learning-rate effects and generalization across settings. Together, our findings identify a concrete failure mode in tool-use RL, provide token-level evidence about its development, and show how supervisory signals affect stability and generalization.

## 1 INTRODUCTION

Tool use is increasingly important for large language models (LLMs), enabling them to act as intelligent agents for complex interactive tasks Li et al. (2023); Mialon et al. (2023); Li et al. (2026); Qu et al. (2025); Yao et al. (2024). However, multi-step structured interactions and diverse tool feedback make tool-use capabilities difficult to improve. Prior work has addressed this challenge through improved interaction frameworks Qin et al. (2024), large-scale synthesis of high-quality trajectories Hao et al. (2025b); Liu et al. (2025b); Prabhakar et al. (2025b), and advanced training methods Liu et al. (2025b); Lu et al. (2025). More recently, advances in agentic RL have positioned reinforcement learning as a promising framework for improving complex agentic interactions Lu et al. (2025); Wang et al. (2025); Mai et al. (2025); Zhang et al. (2025b); Wu et al. (2025).

However, RL-based optimization often suffers from **training instability** and **limited performance gains** in tool-use settings. Figure 1 illustrates a recurring failure mode, which we call *structural collapse*: valid tool invocations deteriorate into malformed or semantically empty control-token sequences. This failure is structurally consequential because a small set of control tokens determines whether a tool call is opened, closed, and terminated correctly. Our diagnostics show that these tokens recur at protocol boundaries across trajectories, concentrating raw cumulative advantage across a few vocabulary entries, and undergo disproportionate probability changes before collapse. Compared with individual ordinary-token types, the control-token group carries high cumulative advan-



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

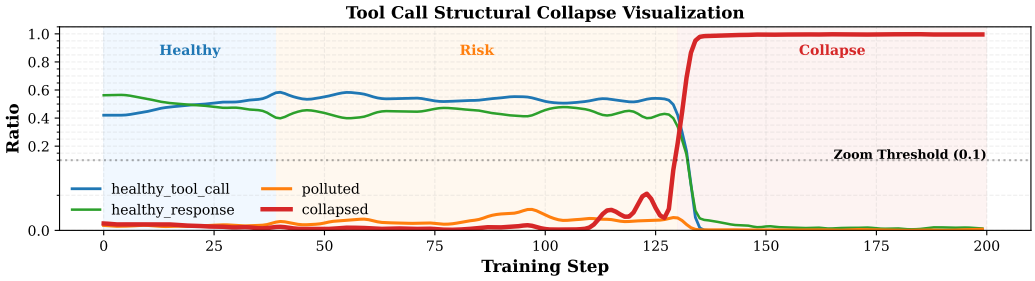



Figure 1: Token-frequency dynamics during RL training of Qwen2.5-1.5B-Instruct. Polluted denotes misplaced or incomplete control tokens; Collapsed denotes a semantically empty output, such as a control-token sequence without an action or response.

tage. In the policy-gradient masking experiments, masked runs exhibit greater training stability than the unmasked run. The logged advantages are measured before masking and can therefore remain high even though the masked positions contribute no direct PG term. These outcomes are consistent with a contribution from direct updates at control positions to training instability.

Supervised fine-tuning (SFT) on high-quality tool-use trajectories can provide a stronger initialization and improve early training stability, motivating its use to preserve structure during long-horizon exploration. At the same time, prior work suggests that SFT may limit generalization relative to RL Chu et al. (2025). We therefore ask: *how should supervisory signals be integrated into RL, and which forms of supervision most effectively balance stability and generalization?*

To answer this question, we study supervisory signals under synchronous and interleaved training, including SFT followed by RL, off-policy supervision, hint-based guidance, erroneous trajectory supervision, and process reflection supervision. The results show that adding supervision to RL does not by itself guarantee stable optimization. Our results show that simply mixing supervision with RL is insufficient: synchronous methods often suffer from distribution mismatch, while interleaved training with supervised fine-tuning (SFT) significantly improves stability but may degrade performance under format- and content-level out-of-distribution (OOD) evaluation. We further analyze key factors affecting stability and generalization, including learning rate sensitivity and cross-setting transfer behavior. Across these studies, we find that supervisory signals play a central role in controlling structural behavior during RL, beyond what scalar rewards alone can achieve.

Our work makes three key contributions: (1) **We characterize structural collapse in multi-turn agentic RL.** Tool-use behavior can deteriorate into malformed or semantically empty control-token sequences, while alternative formats recover some performance. (2) **We analyze token-level dynamics associated with collapse.** Frequently recurring control tokens concentrate cumulative advantage across a small set of vocabulary entries and undergo disproportionate probability changes. Masking supports a contribution from direct updates at these positions. (3) **We compare supervisory signals and training schedules.** We systematically explore six supervisory configurations across two model families and categorize them into synchronous and interleaved paradigms. The interleaved methods show effectiveness and validate the utility of non-tool-call format information.

## 2 Related Works

### 2.1 Tool Learning

Recent advances have extended LLMs with tool-use capabilities Hao et al. (2025a); Prabhakar et al. (2025a); Qin et al. (2024), enabling interaction with external APIs and environments beyond text generation Lu et al. (2025). Through tool invocation, models can perform task execution Li et al. (2023); Trivedi et al. (2024), information retrieval Jin et al. (2025), and planning Xie et al. (2024), improving their ability to solve complex tasks efficiently. Prior works have enhanced tool-use capabilities through reasoning frameworks Qin et al. (2024); Yao et al. (2023), supervised fine-tuning Lu et al. (2025); Zhang et al. (2025d); Li et al. (2025), and reinforcement learning Hu (2025); Shao et al. (2024). Although RL improves tool-invocation performance Qian et al. (2025); Jiang et al. (2025); Xue et al. (2025); Wang et al. (2026), its effectiveness strongly depends on the base model's

prior tool knowledge. Smaller or weakly initialized models often fail to benefit from RL, while high-quality SFT data alone can already provide strong performance. This motivates incorporating supervisory signals into RL to stabilize multi-turn interactions and improve tool-use learning.

### 2.2 Expert Trajectories in Reinforcement Learning

RL improves models through trajectory exploration and reward-based updates Shao et al. (2024); Srivastava & Aggarwal (2025), but can stagnate when sampling quality is poor. Recent works mitigate this by incorporating expert or ground-truth trajectories into RL Fu et al. (2025); Liu et al. (2025a). For instance, LUFFY Yan et al. (2025) replaces part of sampled trajectories with expert ones and reweights low-probability tokens, while ReLIFT Ma et al. (2025) alternates RL with targeted SFT. Other methods use partial correct answers Zhang et al. (2025a;c); Huang et al. (2025); Zhan et al. (2025) or retrieved experiences Chen et al. (2025); Fang et al. (2025). However, these methods are mainly studied in single-turn reasoning tasks (e.g., mathematics), and their effectiveness in multi-turn tool-use settings remains underexplored.

## 3 Method

In this section, we first analyze the *training instability and limited performance gains* observed under pure reinforcement learning settings. By examining model-generated trajectories during training, we identify the underlying causes of these failures. In contrast, introducing supervised fine-tuning before RL substantially improves stability and performance, highlighting the importance of supervisory signals. Motivated by this observation, we further conduct a systematic study of different supervisory signals and integration strategies in agentic RL, analyzing their effects on stability, structural behavior, and generalization.

We represent a multi-turn tool-utilization trajectory as $\tau = (a_1, r_1, \ldots, a_T, r_T)$, where each action $a_t \in \mathcal{A}$ is either a feasible tool invocation or a natural-language response, and each feedback $r_t \in \mathcal{R}$ is an environment response (e.g., a tool output) or user response. At step $t$, the feedback is determined by the current action and preceding trajectory, $r_t = R(a_t \mid a_1, r_1, \ldots, a_{t-1}, r_{t-1})$, while the model samples the next action conditioned on the accumulated interaction history, $a_t \sim P(a_t \mid a_1, r_1, \ldots, a_{t-1}, r_{t-1})$.

### 3.1 Experiment Setup

**Dataset.** We conduct experiments on BFCL-V3 Patil et al. (2025), a multi-turn tool-use benchmark where models must invoke multiple tools across interactive environments to answer user queries and follow-up questions. We focus on four challenging settings: *Base*, *Miss Func*, *Miss Param*, and *Long Context*. For training, we randomly sample 100 instances from each of the first three settings, resulting in 300 questions following previous work Lu et al. (2025), which proves the performance can be improved by small amount of data, while reserving the remaining data for evaluation. Due to excessive context length, *Long Context* is excluded from training. We also augment supervised fine-tuning with converted ToolACE Liu et al. (2025b) data to study the impact of training distribution.

**Model.** For our analysis experiments, we use Qwen2.5-1.5B-Instruct Team et al. (2024) and Qwen3-1.7B Yang et al. (2025) as base models. During the SFT stage, we decompose each multi-turn interaction into multiple single-turn instances and train the model on these isolated steps. In contrast, during RL, we adopt a full multi-turn tool-use setting, where the model must complete the entire interaction trajectory, including all tool invocations, to receive positive feedback. For Qwen3-1.7B, we use the non-thinking variant. Further implementation details are provided in Appendix B.

### 3.2 Problem of catastrophic collapse in the RL

As shown in Figure 2, direct RL on Qwen2.5-1.5B-Instruct often causes **catastrophic collapse**, with sudden reward drops and sustained KL divergence spikes. Qwen3-1.7B exhibits milder instability, but pure RL gains remain limited and may degrade over time. Applying SFT on an additional tool-use dataset before RL stabilizes training: for Qwen2.5-1.5B-Instruct, reward and KL curves evolve smoothly; for Qwen3-1.7B, RL may still occasionally collapse after SFT, but achieves substantially higher performance prior to instability.

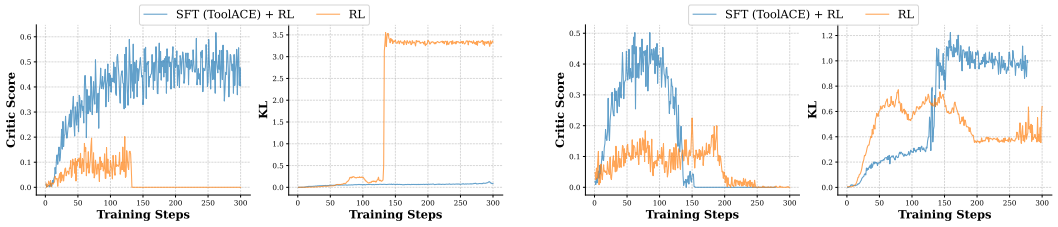



Figure 2: The training dynamics during the training process on the BFCL-V3 dataset. **Left**: Qwen2.5-1.5B-Instruct. **Right:** Qwen3-1.7B.

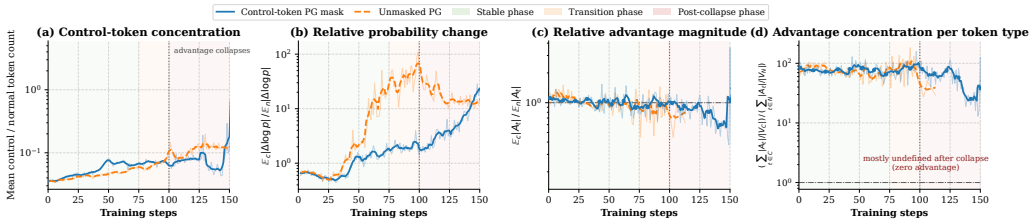



Figure 3: Diagnostics for unmasked and control-token-masked runs. Panels (a)–(d) show token counts, probability movement, per-occurrence advantage, and vocabulary-level advantage concentration. Panels (c)–(d) use pre-mask rollout advantages, so they may remain high although the masked direct PG weight is zero.

### 3.3 Analysis of the reasons of collapse

Training outputs reveal a progressive pattern of structural drift. Under certain prompt formats, control tokens such as <tool_call> and <|im_end|> become increasingly prone to misuse, causing initially valid tool calls to deteriorate into malformed and eventually degenerate outputs. We classify outputs into four structural states: **Healthy Tool Call**, a schema-compliant tool invocation; **Healthy Response**, a coherent natural-language answer without tool-related artifacts; **Text Pollution**, incomplete or misplaced control tokens; and **Collapsed**, a semantically empty sequence such as <tool_call><|im_end|>. As shown in Figure 1, Qwen2.5-1.5B-Instruct initially maintains a clear distinction between tool calls and text responses. As RL progresses, fragmented control-token patterns begin to appear in text responses, and the model eventually defaults to minimal termination sequences. We hypothesize that, without sufficient guidance to sample correct actions, RL disproportionately reinforces a small set of control tokens.

On ACEBench, GRPO scores 24.75 under the alternative format-and-content setting but 0 under content-only OOD, supporting **format sensitivity** and showing that some tool-use performance remains accessible under an alternative format.

These results address distinct questions: format comparisons show sensitivity and partial recovery; pre-collapse probability traces locate abnormal token dynamics; PG masking tests whether direct updates at those positions contribute to instability.

We complement this trajectory-level account with optimization- and probability-level diagnostics in Figure 3. Mean absolute advantage per occurrence is comparable for control and ordinary tokens, while the three control-token types recur frequently at protocol boundaries.

The concentration coincides with disproportionate probability movement before collapse. The occurrence-weighted control-to-normal ratio of $|\Delta \log p|$ reaches high values. Once collapse occurs, the advantages of both control and ordinary tokens approach zero, making the concentration ratio undefined rather than divergent.

More precisely, panel (b) reports the ratio of mean absolute log-probability change, which measures how strongly the update shifts control-token probabilities relative to ordinary-token probabilities. Panel (c) reports the per-occurrence advantage ratio, which measures the typical reward-weighted learning signal associated with a single token occurrence, without accounting for token frequency. Panel (d) instead normalizes accumulated absolute advantage by the number of distinct observed

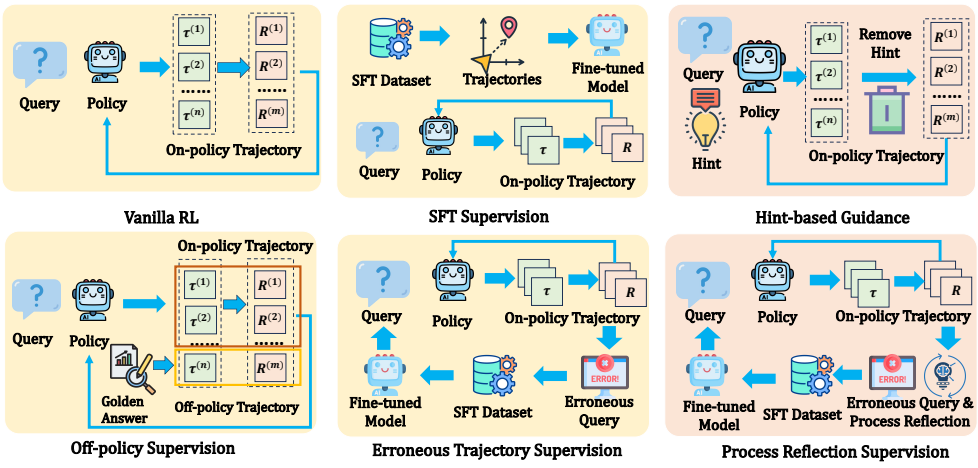



Figure 4: The training frameworks of different methods, where the Hint-based Guidance and Process Reflection Supervision is proposed by our.

tokens, where $N_C$ and $N_N$ denote the numbers of distinct observed control and ordinary token IDs, respectively.

In the masking ablation, $m_t = 0$ removes the direct PG term at control positions although raw rollout advantages remain nonzero. One run remains stable through step 300; the other collapses later than vanilla RL in Figure 7. In the masked run with token-level diagnostics, probability movement and KL differ from the unmasked run over the plotted interval (Figure 3). Together, these results support a contribution from direct control-position updates.

Overall, trajectory diagnostics and PG masking jointly support a contribution from direct updates at control positions to structural instability.

### 3.4 Supervisory Signals for Stabilizing Tool-Use RL

The structural-collapse analysis motivates supervision beyond sparse terminal rewards. Unlike prior works studying isolated supervision strategies, we propose a unified framework to analyze how supervisory signals interact with multi-turn agentic RL. We organize methods into synchronous and interleaved paradigms. We compare SFT-then-RL, off-policy supervision (OPS), hint-based guidance (HBG), and erroneous trajectory supervision (ETS), and further introduce **Process Reflection Supervision (PRS)**, which combines process-level error analysis with synthetic scenario augmentation. Figure 4 summarizes these paradigms; implementation and reward details are in Appendix B.

**Staged SFT Supervision (SFT→RL)** SFT initializes the policy with expert tool-use trajectories before RL. Given $(q, \tau) \in D_{\text{SFT}}$, we optimize

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{|D_{\text{SFT}}|} \sum_{(q,\tau)\in D_{\text{SFT}}} \sum_{t=1}^{|\tau|} \log \pi_\theta(a_t \mid q, \tau_{<t}). \tag{1}$$

This initialization preserves tool-use structure but remains constrained by the coverage of demonstrations.

**Off-policy Supervision (OPS)** Inspired by LUFFY Yan et al. (2025), OPS mixes on-policy rollouts with ground-truth trajectories. For each trajectory $\tau_i$, we normalize its reward over the combined rollout group as $\hat{A}_i = (R(\tau_i) - \text{mean}(G_{\text{on}} \cup G_{\text{off}}))\,/\text{std}(G_{\text{on}} \cup G_{\text{off}})$, where $G_{\text{on}}$ and $G_{\text{off}}$ denote the reward sets of on-policy and off-policy trajectories, respectively. Unlike LUFFY, we apply the same policy ratio $r_{k,t}(\theta) = \pi_\theta(\tau_{k,t} \mid q, \tau_{k,<t})/\pi_{\theta_{\text{old}}}(\tau_{k,t} \mid q, \tau_{k,<t})$ to both trajectory types, yielding

$$\mathcal{J}_{\text{OPS}}(\theta) = \frac{1}{\sum_i |\tau_i|} \sum_{\tau_i \in \mathcal{D}_{\text{on}} \cup \mathcal{D}_{\text{off}}} \sum_{t=1}^{|\tau_i|} \text{CLIP}\Big(r_{k,t}(\theta), \hat{A}_i, \epsilon\Big), \tag{2}$$

**Hint-based Guidance (HBG)** HBG modifies exploration by prepending a task-specific hint $\mathcal{H}_q$ during rollout generation:

$$\tau_{i,t} \sim \pi_\theta(\tau_t \mid q, \tau_{i,<t}, \mathcal{H}_q). \tag{3}$$

The hint is removed when computing the policy update, so optimization is performed under the original query context. HBG therefore changes the explored trajectories without introducing hint dependence at inference time. Rollout log probabilities are conditioned on the hint, whereas update-time old log probabilities are recomputed under the original query context; the policy ratio therefore uses different conditioning contexts.

**Erroneous Trajectory Supervision (ETS)** ETS interleaves RL with corrective SFT on failure cases Ma et al. (2025); Lai et al. (2025).we adopt an interleaved RL and SFT paradigm into the tool-learning tasks, constructing SFT data explicitly from RL failure cases. Starting from an initial policy $\pi_{\theta_0}$, RL collects interaction trajectories, and any query whose sampled trajectories all fail is labeled as erroneous. We then create supervised training data from these failures using the ground-truth solutions to enhance model capability. At iteration $k$, we select queries for which all sampled trajectories fail:

$$\mathcal{E}_k = \bigcup_{q \in Q} \left\{ (q, \tau_{\text{label}}) \mid \forall \tau' \in \mathcal{T}_q,\ R(\tau') = 0 \right\}, \tag{4}$$

where $\mathcal{T}_q$ denotes the sampled trajectories and $\tau_{\text{label}}$ the ground-truth trajectory. Corrective SFT uses

$$\mathcal{L}_{\text{err}}(\theta) = -\frac{1}{|\mathcal{E}_k|} \sum_{(q,\tau) \in \mathcal{E}_k} \sum_{t=1}^{|\tau|} \log \pi_\theta(a_t \mid q, \tau_{<t}). \tag{5}$$

We alternate these corrective updates with RL while gradually increasing the fraction of RL steps following ReLIFT Ma et al. (2025).

**Process Reflection Supervision (PRS)** ETS provides corrective trajectories for failed queries, specifying *what* the correct tool-use behavior should be. Our PRS variant extends this corrective supervision with process-level error analysis and synthetic scenario augmentation. PRS combines process reflection, synthetic scenario augmentation, corrective tool-call examples, and a different trajectory-selection rule. Therefore, the comparison between PRS and ETS evaluates the full supervision package rather than the isolated effect of process reflection.

Unlike ETS, which selects a query only when all its sampled trajectories fail, PRS uses each trajectory. For each success and failed trajectory $\tau$, an auxiliary LLM is given the query, the sampled trajectory, and the corresponding ground-truth trajectory, and generates a reflection $r_\tau^{\text{ref}}$ to analyze user intent, correct behavior, successful actions, error causes, and improvement plans turn by turn. For the failed case, the prompt additionally asks the auxiliary model to generate three to five related scenarios with user requests, typical mistakes, correct tool-call sequences, transferable reasoning, and validation strategies. Thus, PRS includes additional synthetic examples and teacher-generated content beyond the corrective trajectory used by ETS. The failed trajectory reveals *where and how* the policy deviates, while the ground truth provides an executable reference for *what* the correct behavior should be. Their contrast encourages the reflection to identify specific failure causes and extract reusable correction principles rather than generic advice. The ground-truth trajectory is provided to the auxiliary analyzer; the student is trained on the reflection target conditioned on $(q, \tau)$, as shown below. Here, $r_\tau^{\text{ref}}$ denotes the auxiliary model's complete output, including the error diagnosis and the generated related scenarios with their corrective tool-call examples; the loss therefore supervises the full reflection-and-augmentation response.

We construct the reflection dataset as

$$\mathcal{R}_k = \left\{ (q, \tau, r_\tau^{\text{ref}}) \mid \tau \sim \pi_{\theta_k},\ R(\tau) = 0 \right\}. \tag{6}$$

PRS jointly optimizes process-level reflection supervision and the action-level corrective supervision used in ETS:

$$\mathcal{L}_{\text{PRS}}(\theta) = -\frac{1}{|\mathcal{R}_k|} \sum_{(q,\tau,r_\tau^{\text{ref}}) \in \mathcal{R}_k} \log \pi_\theta \left( r_\tau^{\text{ref}} \mid q, \tau \right) + \mathcal{L}_{\text{err}}(\theta), \tag{7}$$

Table 1: Experimental results on Qwen series models. Best results are highlighted in **bold**. Detailed training dynamics of individual metrics are provided in the Appendix D.

| Model | Method | Base | Δ | Miss F. | Δ | Miss P. | Δ | Long C. | Δ | Average | Δ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Qwen2.5-1.5B | Vanilla | 4.0 | - | 5.0 | - | 1.0 | - | 4.0 | - | 3.50 | - |
| | $SFT_{BFCL}$ | 15.0 | (+11.0) | 4.0 | (-1.0) | 6.0 | (+5.0) | 7.0 | (+3.0) | 8.00 | (+4.50) |
| | GRPO | 0.0 | (-4.0) | 0.0 | (-5.0) | 0.0 | (-1.0) | 0.0 | (-4.0) | 0.00 | (-3.50) |
| | $SFT_{BFCL}$ + RL | 21.0 | (+17.0) | 22.0 | (+17.0) | 19.0 | (+18.0) | 7.0 | (+3.0) | 17.25 | (+13.75) |
| | $SFT_{ToolACE}$ + RL | 23.0 | (+19.0) | 23.0 | (+18.0) | 13.0 | (+12.0) | 10.0 | (+6.0) | 17.25 | (+13.75) |
| | OPS | 1.0 | (-3.0) | 3.0 | (-2.0) | 1.0 | (0) | 1.0 | (-3.0) | 1.50 | (-2.00) |
| | HBG | 0.0 | (-4.0) | 0.0 | (-5.0) | 0.0 | (-1.0) | 0.0 | (-4.0) | 0.00 | (-3.50) |
| | ETS | 26.0 | (+22.0) | **25.0** | (+20.0) | 16.0 | (+15.0) | 13.0 | (+9.0) | 20.00 | (+16.50) |
| | PRS | **31.0** | (+27.0) | **25.0** | (+20.0) | **26.0** | (+25.0) | **21.0** | (+17.0) | **25.75** | (+22.25) |
| Qwen2.5-7B | Vanilla | 13.0 | - | 9.0 | - | 12.0 | - | 12.0 | - | 11.50 | - |
| | $SFT_{BFCL}$ | 17.0 | (+4.0) | 9.0 | (0) | 18.0 | (+6.0) | 15.0 | (+3.0) | 14.75 | (+3.25) |
| | GRPO | 0.0 | (-13.0) | 0.0 | (-9.0) | 0.0 | (-12.0) | 0.0 | (-12.0) | 0.00 | (-11.50) |
| | $SFT_{BFCL}$ + RL | 0.0 | (-13.0) | 0.0 | (-9.0) | 0.0 | (-12.0) | 0.0 | (-12.0) | 0.00 | (-11.50) |
| | OPS | 0.0 | (-13.0) | 0.0 | (-9.0) | 0.0 | (-12.0) | 0.0 | (-12.0) | 0.00 | (-11.50) |
| | HBG | 0.0 | (-13.0) | 0.0 | (-9.0) | 0.0 | (-12.0) | 0.0 | (-12.0) | 0.00 | (-11.50) |
| | ETS | 0.0 | (-13.0) | 0.0 | (-9.0) | 0.0 | (-12.0) | 0.0 | (-12.0) | 0.00 | (-11.50) |
| | PRS | **49.0** | (+36.0) | **44.0** | (+35.0) | **31.0** | (+19.0) | **35.0** | (+23.0) | **39.75** | (+28.25) |
| Qwen3-1.7B | Vanilla | 14 | - | 11 | - | 14 | - | 11 | - | 12.50 | - |
| | $SFT_{BFCL}$ | 23 | (+9) | 20 | (+9) | 23 | (+9) | 14 | (+3) | 20.00 | (+7.50) |
| | GRPO | 2 | (-12) | 0 | (-11) | 2 | (-12) | 2 | (-9) | 1.50 | (-11.00) |
| | $SFT_{BFCL}$ + RL | 0 | (-14) | 0 | (-11) | 0 | (-14) | 0 | (-11) | 0.00 | (-12.50) |
| | $SFT_{ToolACE}$ + RL | 0 | (-14) | 0 | (-11) | 0 | (-14) | 0 | (-11) | 0.00 | (-12.50) |
| | OPS | 0 | (-14) | 0 | (-11) | 0 | (-14) | 0 | (-11) | 0.00 | (-12.50) |
| | HBG | 1 | (-13) | 0 | (-11) | 1 | (-13) | 1 | (-10) | 0.75 | (-11.75) |
| | ETS | **26** | (+12) | **27** | (+16) | **19** | (+5) | **21** | (+10) | **23.25** | (+10.75) |
| | PRS | 23 | (+9) | 22 | (+11) | 17 | (+3) | 16 | (+5) | 19.50 | (+7.00) |

Teacher-generated supervision provides a general means of enhancing language-model capabilities through knowledge distillation. PRS incorporates this principle into multi-turn tool-use RL by converting failures of the current policy into error analyses, related scenarios, and corrective tool-use examples. The additional teacher knowledge and expanded supervision are intentional components of the method. The two methods provide different supervision packages: ETS supplies an *action-level correction* by specifying a successful executable trajectory, whereas PRS combines process-level diagnosis with synthetic related scenarios and their corrective tool-use examples. By regenerating this augmented supervision from the current policy during training, PRS adapts its content to the model's evolving weaknesses while retaining RL exploration. The reflection-generation prompt and examples are provided in Appendix F.

# 4 Experiments

## 4.1 Dataset and Models

We train *Qwen2.5-1.5B-Instruct*, *Qwen2.5-7B-Instruct* and *Qwen3-1.7B* on BFCL-V3 and evaluate their performance on the corresponding test sets. To assess out-of-distribution generalization, we further evaluate the trained models on ACEBench. Detailed training configurations are provided in Appendix B.

## 4.2 Main Results

Based on Table 1, both Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct and Qwen3-1.7B show negligible baseline performance, highlighting the difficulty of multi-turn tool-use without targeted training.

For Qwen2.5-1.5B-Instruct, SFT on BFCL raises the performance floor, with subsequent GRPO providing further refinement, while GRPO alone yields no meaningful gains and its stochasticity often causes catastrophic collapse, indicating that prior structural supervision is indispensable for

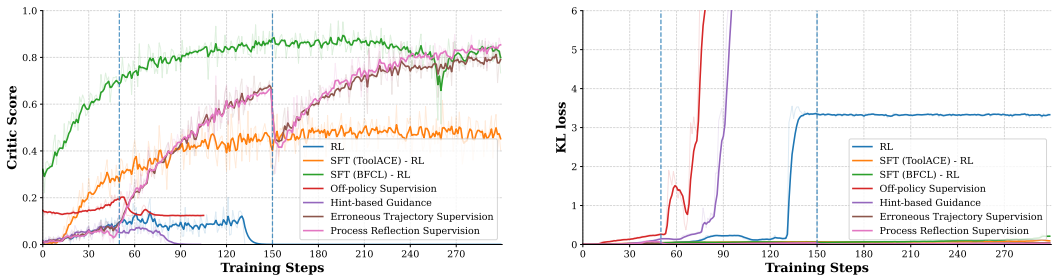



Figure 6: The training dynamics during the training process on the BFCL-V3 dataset of Qwen2.5-1.5B-Instruct.

agentic RL. Among supervisory signals, Off-Policy Supervision and Hint-Based Guidance offer limited benefits. KL divergence during synchronous training is substantially higher than in standard RL (Figure 6), likely due to rapid output distribution shifts when trajectories are not sampled strictly on-policy, destabilizing optimization. This calls for a more granular policy-update and sampling synchronization framework. Process Reflection Supervision achieves the highest average score (25.75), followed by Erroneous Trajectory Supervision; notably, PRS's reasoning-heavy format, reflection process and data enhancement, though diverging from standard tool-calling syntax, provides critical logical scaffolding that enhances tool-use competence. These results suggest interleaved training, decoupling SFT-like structural updates from RL exploration, is fundamentally more stable than synchronous training in Qwen2.5-1.5B.

Qwen2.5-7B shows a sharper separation among the tested methods at their reported final checkpoints: vanilla and SFT retain nonzero performance, while GRPO, SFT+RL, OPS, HBG, and ETS end at zero across the reported settings. PRS retains substantial performance across all four BFCL settings in Table 1. This result shows that model scale alone does not prevent collapse under the tested updates, and that the benefit of supervision is not shared by all interventions. It also makes the PRS result notable. These comparisons show that supervision effects depend on the model and training setup; they do not support a universal ranking of methods.

For Qwen3-1.7B, SFT prior to RL does not yield strong final improvements. As shown in Figure 17, the model initially outperforms direct RL but eventually suffers collapse, illustrating that early gains need not persist to the final checkpoint. The stage-level prompt-format difference described in Appendix C is a possible contributor. The stronger ETS result relative to PRS on this model further indicates that corrective supervision can help without the full reflection-and-augmentation package being consistently better.

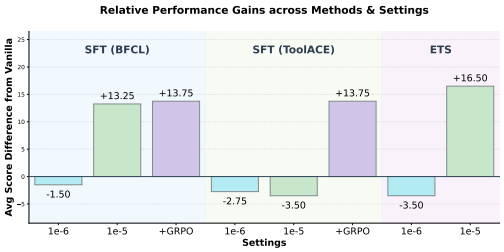



Figure 5: Results on Qwen2.5-1.5B-Instruct under different SFT learning rates. The Y-axis illustrates performance gains and losses relative to the Vanilla baseline (0.0 line), while the X-axis represents various hyperparameter configurations and training stages. The + *GRPO* results apply RL with a fixed learning rate of $1 \times 10^{-6}$ to models initialized from SFT at a learning rate of $1 \times 10^{-5}$.

### 4.3 The Analysis of Learning Rate

Figure 5 examines the effect of the SFT learning rate on Qwen2.5-1.5B-Instruct, comparing $1 \times 10^{-6}$ and $1 \times 10^{-5}$ for supervised updates. The higher SFT learning rate improves the BFCL SFT result in this setting. SFT on ToolACE initially suffers from distribution mismatch but later recovers after RL. For ETS, the higher SFT learning rate also improves the reported metrics, suggesting that the scale of corrective supervised updates matters. The subsequent RL learning rate is held fixed at $1 \times 10^{-6}$ (Appendix B). It suggests that a larger SFT learning rate can provide a stronger initialization or corrective update before and during subsequent RL in our setting.

Table 2: Generalization results of Qwen2.5-1.5B-Instruct on ACEBench. Left: Format and Content OOD. Right: Content OOD. Best results among trained methods (excluding Vanilla) are bold; **Ad.** denotes Adjust, **Sw.** Switch, and **sf.** single function. Green shading marks methods with an isolated SFT stage.

| Method | Format and Content OOD | | | | | Content OOD | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Ad.** | **Sw.** | **sf.** | **Multi Turn** | **Avg** | **Ad.** | **Sw.** | **sf.** | **Multi Turn** | **Avg** |
| Vanilla | 38.0 | 20.0 | 46.0 | 0.0 | 26.00 | 40.0 | 32.0 | 52.0 | 2.5 | 31.6 |
| $SFT_{BFCL}$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.00 | **30.0** | 20.0 | 44.0 | 2.5 | 24.1 |
| GRPO | 32.0 | 24.0 | **43.0** | 0.0 | 24.75 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $SFT_{BFCL}$ + RL | 0.0 | 0.0 | 0.0 | 0.0 | 0.00 | 8.0 | 8.0 | 10.0 | 0.0 | 6.5 |
| OPS | **36.0** | **24.0** | **43.0** | 0.0 | **25.75** | 20.0 | 16.0 | **50.0** | **6.7** | 23.2 |
| HBG | 28.0 | 20.0 | 42.0 | 0.0 | 22.50 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ETS | 10.0 | 0.0 | 25.0 | 0.0 | 8.75 | 26.0 | 22.0 | 36.0 | 2.5 | 21.6 |
| PRS | 16.0 | 8.0 | 24.0 | 0.0 | 12.00 | 26.0 | **30.0** | 43.0 | 4.2 | **25.8** |

### 4.4 The Analysis of Generalization

We evaluate model generalization under out-of-distribution (OOD) settings using the original ACEBench framework, which features unseen tool types and prompt structures. Many prior studies prioritize scenario generalization without format generalization. We report both generalization axes to analyze the role of supervisory signals.

**Defining ID vs. OOD in our setup.** We distinguish content OOD from format OOD. Content OOD evaluates models on tool types and scenarios absent from the in-distribution (ID) training data. Format OOD is instantiated by ACEBench's fixed invocation template, whereas ID evaluation (including training) follows Qwen's native format. Therefore, we observe a general performance drop in OOD scenarios, caused by both factors. Experimental results yield several critical insights:

**Format- and content-sensitive overfitting during SFT.** In Table 2, SFT methods that excel in-distribution suffer sharp degradation under Format and Content OOD in simple single-step settings but not on Content OOD. This supports our claim that standard SFT induces severe format overfitting, tethering the model to training-time syntax.

**RL "collapse" as a format artefact**. Methods that appear unstable during training (e.g., GRPO-only, OPS, HBG) exhibit more stable behavior under Format and Content OOD tests, while Content OOD in same tool format experiments show decline. This confirms that the observed instability is format-specific (token-level probability shifts) rather than a global loss of reasoning or linguistic competence, as hypothesized in Section 3.3. The Multi Turn results further indicate limited generalization during training.

**Generalization trade-off.** Interleaved supervision improves in-distribution stability but does not consistently improve OOD performance. Among the supervised configurations, PRS mitigates part of the degradation in some content-OOD settings, but it remains below the vanilla model in aggregate OOD performance. Since PRS combines process diagnosis, synthetic scenario augmentation, and corrective examples, its results reflect the complete supervision package rather than the effect of reflection alone. This highlights the importance of expanding the model's knowledge through data augmentation and knowledge distillation during training.

## 5 Conclusion

We identify structural collapse in multi-turn tool-use RL, where valid calls deteriorate into malformed or semantically empty control-token sequences. Alternative prompts recover some performance. Token diagnostics and PG masking support a contribution from frequently recurring control-position updates. We systematically study supervisory signal integration, finding interleaved training more stable and effective than synchronous. We also analyze learning rate, SFT data distribution, and generalization, offering a principled framework for stable, effective agentic RL in tool-use settings.

## AI USE STATEMENT

Generative AI tools were used for manuscript polishing and literature search. In addition, an auxiliary language model was used as part of the PRS training pipeline to generate process-level reflections and synthetic related scenarios, as described in Section 3.4 and Appendix F. All AI-assisted content and retrieved references were reviewed and verified by the authors.

## REFERENCES

Silin Chen, Shaoxin Lin, Xiaodong Gu, Yuling Shi, Heng Lian, Longfei Yun, Dong Chen, Weiguo Sun, Lin Cao, and Qianxiang Wang. Swe-exp: Experience-driven software issue resolution. *CoRR*, abs/2507.23361, 2025. doi: 10.48550/ARXIV.2507.23361. URL https://doi.org/10.48550/arXiv.2507.23361.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *CoRR*, abs/2508.06433, 2025. doi: 10.48550/ARXIV.2508.06433. URL https://doi.org/10.48550/arXiv.2508.06433.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*, 2025.

Yupu Hao, Pengfei Cao, Zhuoran Jin, Huanxuan Liao, Yubo Chen, Kang Liu, and Jun Zhao. CITI: enhancing tool utilizing ability in large language models without sacrificing general performance. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 23996–24004. AAAI Press, 2025a. doi: 10.1609/AAAI.V39I22.34573. URL https://doi.org/10.1609/aaai.v39i22.34573.

Yupu Hao, Pengfei Cao, Zhuoran Jin, Huanxuan Liao, Yubo Chen, Kang Liu, and Jun Zhao. Evaluating personalized tool-augmented llms from the perspectives of personalization and proactivity. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 21897–21935. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.acl-long.1064/.

Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models. *CoRR*, abs/2501.03262, 2025. doi: 10.48550/ARXIV.2501.03262. URL https://doi.org/10.48550/arXiv.2501.03262.

Qihan Huang, Long Chan, Jinlong Liu, Wanggui He, Hao Jiang, Mingli Song, Jingyuan Chen, Chang Yao, and Jie Song. Boosting MLLM reasoning with text-debiased hint-grpo. *CoRR*, abs/2503.23905, 2025. doi: 10.48550/ARXIV.2503.23905. URL https://doi.org/10.48550/arXiv.2503.23905.

Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, Tianyu Pang, and Wenhu Chen. Verltool: Towards holistic agentic reinforcement learning with tool use. *CoRR*, abs/2509.01055, 2025. doi: 10.48550/ARXIV.2509.01055. URL https://doi.org/10.48550/arXiv.2509.01055.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents, 2025. URL https://arxiv.org/abs/2508.14040.

Jiachun Li, Zhuoran Jin, Tianyi Men, Yupu Hao, Kejian Zhu, Lingshuai Wang, Dongqi Huang, Longxiang Wang, Shengjia Hua, Lu Wang, et al. Agentic environment engineering for large language models: A survey of environment modeling, synthesis, evaluation, and application. *arXiv preprint arXiv:2606.12191*, 2026.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.187. URL https://doi.org/10.18653/v1/2023.emnlp-main.187.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated RL. *CoRR*, abs/2503.23383, 2025. doi: 10.48550/ARXIV.2503.23383. URL https://doi.org/10.48550/arXiv.2503.23383.

Mingyang Liu, Gabriele Farina, and Asuman E. Ozdaglar. UFT: unifying supervised and reinforcement fine-tuning. *CoRR*, abs/2505.16984, 2025a. doi: 10.48550/ARXIV.2505.16984. URL https://doi.org/10.48550/arXiv.2505.16984.

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, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=8EB8k6DdCU.

Siyuan Lu, Zechuan Wang, Hongxuan Zhang, Qintong Wu, Leilei Gan, Chenyi Zhuang, Jinjie Gu, and Tao Lin. Don't just fine-tune the agent, tune the environment. *CoRR*, abs/2510.10197, 2025. doi: 10.48550/ARXIV.2510.10197. URL https://doi.org/10.48550/arXiv.2510.10197.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, et al. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*, 2025.

Xinji Mai, Haotian Xu, Zhong-Zhi Li, Weinong Wang, Jian Hu, Yingying Zhang, Wenqiang Zhang, et al. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving. *arXiv preprint arXiv:2505.07773*, 2025.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=jh7wH2AzKK.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): from tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *CoRR*, abs/2504.03601, 2025a. doi: 10.48550/ARXIV.2504.03601. URL https://doi.org/10.48550/arXiv.2504.03601.

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

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *CoRR*, abs/2504.13958, 2025. doi: 10.48550/ARXIV.2504.13958. URL https://doi.org/10.48550/arXiv.2504.13958.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: a survey. *Frontiers Comput. Sci.*, 19 (8):198343, 2025. doi: 10.1007/S11704-024-40678-2. URL https://doi.org/10.1007/s11704-024-40678-2.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Saksham Sahai Srivastava and Vaneet Aggarwal. A technical survey of reinforcement learning techniques for large language models. *CoRR*, abs/2507.04136, 2025. doi: 10.48550/ARXIV.2507.04136. URL https://doi.org/10.48550/arXiv.2507.04136.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.850. URL https://doi.org/10.18653/v1/2024.acl-long.850.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, et al. Ragen-2: Reasoning collapse in agentic rl. *arXiv preprint arXiv:2604.06268*, 2026.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. *arXiv preprint arXiv:2502.04644*, 2025.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*, 2024.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *CoRR*, abs/2509.02479, 2025. doi: 10.48550/ARXIV.2509.02479. URL https://doi.org/10.48550/arXiv.2509.02479.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *CoRR*, abs/2504.14945, 2025. doi: 10.48550/ARXIV.2504.14945. URL https://doi.org/10.48550/arXiv.2504.14945.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024. doi: 10.48550/ARXIV.2406.12045. URL https://doi.org/10.48550/arXiv.2406.12045.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. *CoRR*, abs/2510.02245, 2025. doi: 10.48550/ARXIV.2510.02245. URL https://doi.org/10.48550/arXiv.2510.02245.

Feng Zhang, Zezhong Tan, Xinhong Ma, Ziqiang Dong, Xi Leng, Jianfei Zhao, Xin Sun, and Yang Yang. Adhint: Adaptive hints with difficulty priors for reinforcement learning. *arXiv preprint arXiv:2512.13095*, 2025a.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025b.

Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. Stephint: Multi-level stepwise hints enhance reinforcement learning to reason. *arXiv preprint arXiv:2507.02841*, 2025c.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning. *CoRR*, abs/2505.00024, 2025d. doi: 10.48550/ARXIV.2505.00024. URL https://doi.org/10.48550/arXiv.2505.00024.

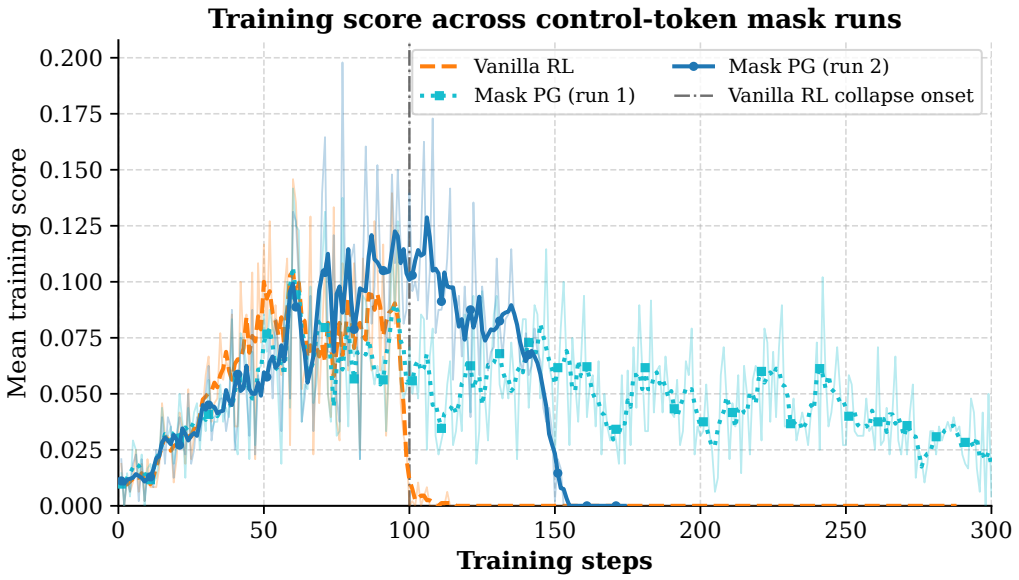



Figure 7: Vanilla RL collapses near step 100. One of two control-token PG-mask runs remains stable through 300 steps; the other collapses later than vanilla RL. These trials support a stabilizing contribution from masking.

## A Analysis

One mask run avoids collapse through step 300; the other collapses later than vanilla RL. Raw advantages are logged before masking and can remain high; PG aggregation uses masked positions, while KL retains the original response mask.

**From accumulated advantage to probability displacement.** For completeness, we give the exact aggregation used by the diagnostics. At a given training step, let $\mathcal{T}_C$ and $\mathcal{T}_N$ denote the sampled occurrences whose token IDs are classified as control and ordinary, respectively. For each occurrence $t$, the instrumentation records its advantage $A_t$ and the log-probability displacement

$$\Delta \log p_t = \log p_{\text{new}}(y_t \mid h_t) - \log p_{\text{old}}(y_t \mid h_t). \tag{8}$$

The occurrence-level means are therefore

$$\mathbb{E}_C[|A|] = \frac{1}{|\mathcal{T}_C|} \sum_{t \in \mathcal{T}_C} |A_t|, \qquad \mathbb{E}_N[|A|] = \frac{1}{|\mathcal{T}_N|} \sum_{t \in \mathcal{T}_N} |A_t|, \tag{9}$$

and analogously

$$\mathbb{E}_C[|\Delta \log p|] = \frac{1}{|\mathcal{T}_C|} \sum_{t \in \mathcal{T}_C} |\Delta \log p_t|, \qquad \mathbb{E}_N[|\Delta \log p|] = \frac{1}{|\mathcal{T}_N|} \sum_{t \in \mathcal{T}_N} |\Delta \log p_t|. \tag{10}$$

Taking the ratio of the two means gives $R_A$ and $R_{\Delta p}$ in the main text. These ratios answer an occurrence-level question: whether a typical control-token occurrence receives a larger advantage or experiences a larger probability displacement than a typical ordinary-token occurrence.

To measure concentration across vocabulary entries, let $V_C$ and $V_N$ be the sets of distinct control and ordinary token IDs observed at that step, with $N_C = |V_C|$ and $N_N = |V_N|$. The total absolute advantage carried by each group is first accumulated over occurrences and then divided by its number of token types:

$$M_C = \frac{1}{N_C} \sum_{t \in \mathcal{T}_C} |A_t|, \qquad M_N = \frac{1}{N_N} \sum_{t \in \mathcal{T}_N} |A_t|. \tag{11}$$

The plotted concentration statistic is $R_{\text{type}} = M_C/M_N$. It can be large even when $R_A \leq 1$, because a small set of control IDs may recur many times while the ordinary-token advantage is spread across a large vocabulary. Conversely, it is not a measure of the control tokens' share of the total advantage over all occurrences. When both group sums approach zero after collapse, $M_C/M_N$ has no meaningful denominator and is reported as undefined rather than interpreted as a post-collapse increase.

For a sampled token occurrence, the unclipped policy-gradient term has the form

$$\ell_t(\theta) = -\rho_t(\theta)A_t, \qquad \rho_t(\theta) = \frac{\pi_\theta(y_t \mid h_t)}{\pi_{\theta_{\text{old}}}(y_t \mid h_t)}. \tag{12}$$

At the behavior policy, where $\rho_t(\theta_{\text{old}}) = 1$, its gradient is

$$\nabla_\theta \ell_t(\theta_{\text{old}}) = -A_t \nabla_\theta \log \pi_{\theta_{\text{old}}}(y_t \mid h_t). \tag{13}$$

Thus, $|A_t|$ scales the magnitude of the direct reward-weighted contribution at position $t$. More exactly,

$$\|\nabla_\theta \ell_t(\theta_{\text{old}})\|_2 = |A_t| \, \|\nabla_\theta \log \pi_{\theta_{\text{old}}}(y_t \mid h_t)\|_2. \tag{14}$$

Consequently, $R_A$ is a proxy for the relative per-occurrence policy-gradient scale when the score-function norms of the two groups are comparable. It is not the gradient norm itself: score-function geometry, PPO clipping, KL terms, and cancellation between signed contributions are not represented by $|A_t|$. The relationship between occurrence-level advantage and vocabulary-level concentration follows from an exact decomposition. Define the mean number of sampled occurrences per observed token ID as

$$F_C = \frac{|\mathcal{T}_C|}{N_C}, \qquad F_N = \frac{|\mathcal{T}_N|}{N_N}. \tag{15}$$

Substituting the definitions of $M_C$ and $M_N$ gives

$$M_C = F_C \, \mathbb{E}_C[|A|], \qquad M_N = F_N \, \mathbb{E}_N[|A|], \tag{16}$$

$$R_{\text{type}} = \frac{F_C}{F_N} R_A. \tag{17}$$

Equation 17 explains the distinct roles of panels (c) and (d). Panel (c) tests whether an individual control occurrence has an unusually large advantage. Panel (d) additionally incorporates how often optimization exposure is repeatedly assigned to each vocabulary ID. Therefore, $R_{\text{type}}$ can be large without $R_A > 1$ when $F_C/F_N$ is large. This is the repeated-exposure concentration hypothesis: ordinary-sized per-occurrence signals are accumulated many times on a few structurally decisive IDs because they recur at protocol boundaries. The accumulated absolute advantage also bounds an idealized sum of direct gradient contributions. For a token-ID group $G$, the triangle inequality gives

$$\left\| \sum_{t \in \mathcal{T}_G} A_t \nabla_\theta \log \pi_\theta(y_t \mid h_t) \right\|_2 \leq \sum_{t \in \mathcal{T}_G} |A_t| \|\nabla_\theta \log \pi_\theta(y_t \mid h_t)\|_2. \tag{18}$$

Hence, $\sum |A_t|$ measures available reward-weighted exposure before directional cancellation, rather than the realized parameter displacement. Normalizing it by the number of observed IDs estimates how densely that exposure is allocated across the vocabulary. Finally, the measured probability displacement supplies an outcome-side diagnostic. For an update $\Delta\theta = \theta_{\text{new}} - \theta_{\text{old}}$, a first-order expansion yields

$$\Delta \log p_t \approx \nabla_\theta \log \pi_{\theta_{\text{old}}}(y_t \mid h_t)^\top \Delta\theta. \tag{19}$$

While $R_A$ and $R_{\text{type}}$ describe the reward-weighted signal made available to the optimizer, $R_{\Delta p}$ measures the resulting movement of the sampled-token probabilities after the complete update. Agreement in their timing is therefore evidence consistent with concentrated exposure producing disproportionate movement. It is not an identification result, because $\Delta\theta$ also contains KL regularization, clipping effects, contributions from other positions, and shared-parameter interactions. The masking intervention sharpens this interpretation. If $m_t = 0$ for control positions and $m_t = 1$ otherwise, the masked reward-weighted objective replaces Equation 13 by

$$\nabla_\theta \ell_t^{\text{mask}} = -m_t A_t \nabla_\theta \log \pi_\theta(y_t \mid h_t). \tag{20}$$

The mask removes this direct PG term, not all probability movement: shared parameters and other loss terms still act on control-token logits. In these runs the masked strings are `<tool_call>`,

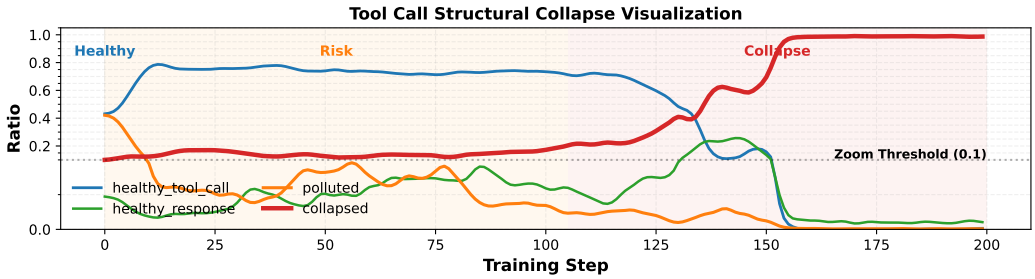



Figure 8: The changes of token appearing frequency in the SFT then RL training process of model Qwen3-1.7B.

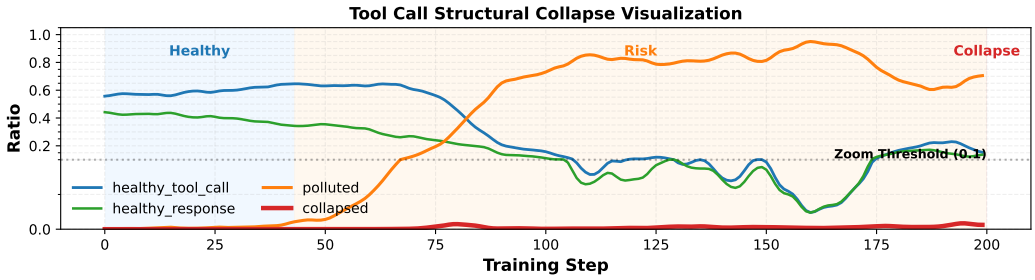



Figure 9: The changes of token appearing frequency in the RL training process of model Qwen3-1.7B.

</tool_call>, and <|im_end|>; all generated occurrences are masked. The diagnostics are collected before masking, hence $A_t$ may remain nonzero although $m_t A_t = 0$ at a control position. The ablation therefore tests the implemented intervention, including its aggregation change.

The instrumentation records per-occurrence advantage and $|\Delta \log p| = |\log p_{\text{new}} - \log p_{\text{old}}|$. Control tokens do not exhibit consistently larger mean absolute advantages per occurrence than ordinary tokens. Instead, repeated occurrences concentrate cumulative absolute advantage across a small set of control-token types, as characterized by $R_{\text{type}}$. Some individual ordinary-token IDs nevertheless accumulate more absolute advantage than the control-token group combined. These statistics indicate concentration across recurring types, rather than uniformly stronger per-occurrence signals. The control-to-ordinary ratio of mean absolute log-probability change increases markedly before collapse. These measurements capture post-update probability movement rather than direct gradient contributions. Masked runs retain raw advantages while removing direct control-position PG terms; shared parameters and other loss terms can still change these probabilities. Masking prevents collapse within the observation window in some runs and delays it in others. These observations are consistent with a stabilizing effect of the implemented intervention.

Table 3: Examples of different output types observed during multi-turn tool-use training.

| Type | Example |
|---|---|
| Healthy Tool Call | <tool_call>{''name'': ''search'', ''arguments'': {...}}</tool_call><\|im_end\|> |
| Healthy Response | The answer can be derived by first querying the database...<\|im_end\|> |
| Polluted Output | The result is ...<tool_call><\|im_end\|> |
| Collapsed Output | <tool_call><\|im_end\|> |

The detailed classification of model output structures:

- Healthy Tool Call: Model output contains a valid tool invocation: both opening <tool_call> and closing </tool_call> tags are present and correctly matched. The

internal action is valid, the structure conforms to the expected JSON schema, and no stray <|im_end|> tokens or partial tags appear within the invocation.

- Healthy Response: Model output does **not** include any tool-call tags (<tool_call>, </tool_call>) but ends with a valid end-of-turn token <|im_end|>. The content is a coherent natural language response without extraneous or misused tool tags.

- Text Polluted: Outputs with partially misused tool-call tags, such as missing closing tags, adding <think>, embedded <|im_end|> inside a tool invocation, or extra tags appended to normal text. Indicates structural corruption where the model mixes dialogue text with tool invocation tags incorrectly, potentially causing protocol violations or parsing errors.

- Collapsed: Outputs consisting solely of the end-of-turn token <|im_end|>, or tool-call sequences reduced to control tokens without semantic content.

The more analysis results are in Figure 8 and Figure 9.

## B Training Details

During Qwen3 RL, the prompt is suffixed with <think>\n\n</think>\n\n. It remains in the update input; mask_observations=True masks prompt loss. Thus sampling and update conditions match, while the SFT targets omit this wrapper.

Table 4: Supervised Fine-Tuning (SFT) configuration.

| **Parameter** | **SFT** |
|---|---|
| Epochs | 3 |
| Train batch size | 128 |
| Learning rate | 1e-5 |
| Max sequence length | 8192 |

Table 5: Reinforcement Learning (RL) configuration.

| **Parameter** | **RL** |
|---|---|
| Total training steps | 300 (Qwen3 interleaved: 200) |
| Batch size | 60 |
| PPO mini-batch size | 20 |
| Learning rate | 1e-6 |
| Max turns per trajectory | 30 |
| KL loss coefficient | 0.0001 |

The SFT and RL configurations are summarized in Tables 4 and 5. OPS uses seven policy trajectories and one ground-truth trajectory; HBG uses six standard and two hint-conditioned trajectories. Interleaved RL rounds are 50/100/150 for Qwen2.5-1.5B and 50/50/100 for Qwen3 (200 steps total). Reported table scores use final checkpoints. All the training process is convergent as 200 steps is enough.

We follow the BFCL reward design in all the reinforcement learning with two evaluation types: **state-based** and **response-based** evaluation. The reward is 1 if both state-based evaluation and response-based evaluation is 1, and it will be 0 if not. State-based evaluation compares the backend state after all function calls with the ground-truth final state, focusing on tasks that modify internal




Figure 10: Qwen3-1.7B training reward for RL runs with versus without the empty `<think></think>` wrapper appended to the rollout prompt; SFT targets omit the wrapper.

states. Response-based evaluation compares the model's execution path with the minimal viable ground-truth function-call path, enabling evaluation of read-only tasks such as querying stock or weather information.

In PRS, we use gpt-5-mini as the auxiliary analyzer to generate textual reflections. The prompt and output example are in Appendix F.

## C THE ANALYSIS OF QWEN3 TRAINING

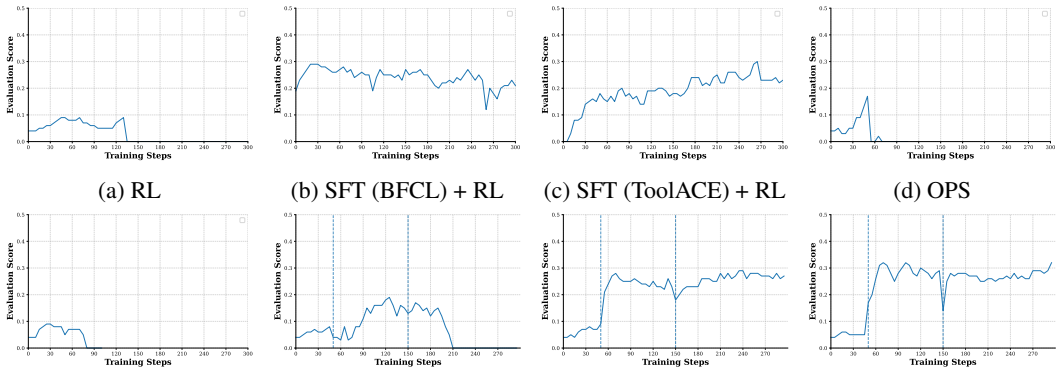
We hypothesize that the stage-level format change may contribute to Qwen3's collapse after SFT-then-RL. Qwen3 uses a thinking-mode protocol, while our SFT targets contain direct tool-call outputs without the empty thinking wrapper prepended to RL prompts. Within RL, the wrapper remains in both sampling and update inputs; only prompt-token loss is masked, so this is not a sampling-update condition mismatch. Figure 10 shows the observed training behavior.

## D DETAILED EVALUATION

Since evaluation is conducted using the final checkpoint, some models may demonstrate strong performance during intermediate training stages but fail to retain these gains at convergence. We report the evaluation results in the training process as follows. We can find the best performance during the training process. The results of Qwen2.5 is shown in Figure 11, Figure 12 and Figure 13. The results of Qwen3 is shown in Figure 14, Figure 15 and Figure 16.

## E TRAINING DYNAMIC

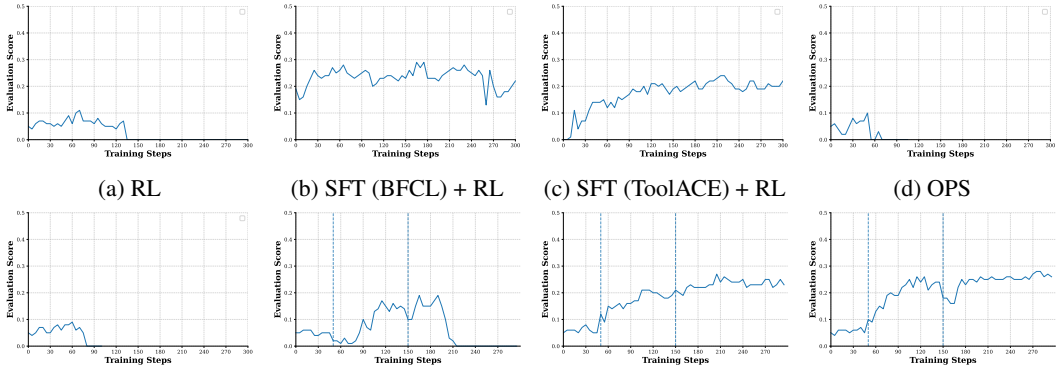
The Training dynamics of Qwen2.5 are shown in Figure 6 and Qwen3 are shown in Figure 17.




(a) RL (b) SFT (BFCL) + RL (c) SFT (ToolACE) + RL (d) OPS

(e) HBG (f) ETS (SFT 1e−6) (g) ETS (SFT 1e−5) (h) PRS

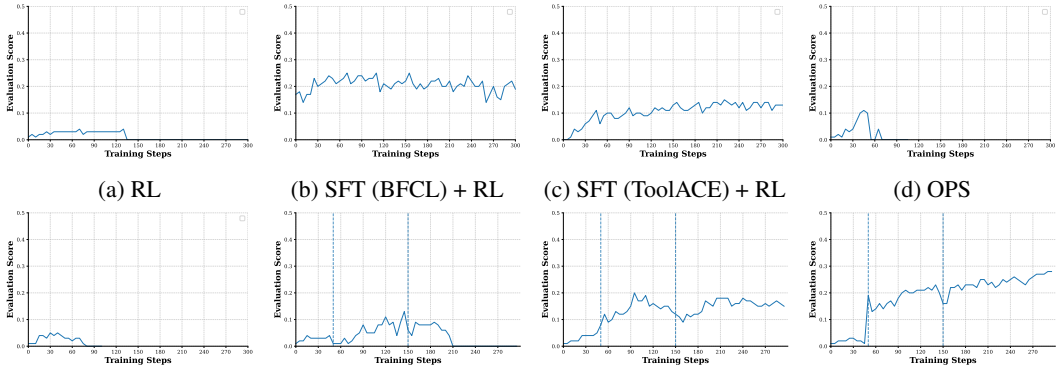
Figure 11: Evaluation results of scenario *Base* under different supervisory signals of Qwen2.5-1.5B-Instruct.




(a) RL (b) SFT (BFCL) + RL (c) SFT (ToolACE) + RL (d) OPS

(e) HBG (f) ETS (SFT 1e−6) (g) ETS (SFT 1e−5) (h) PRS

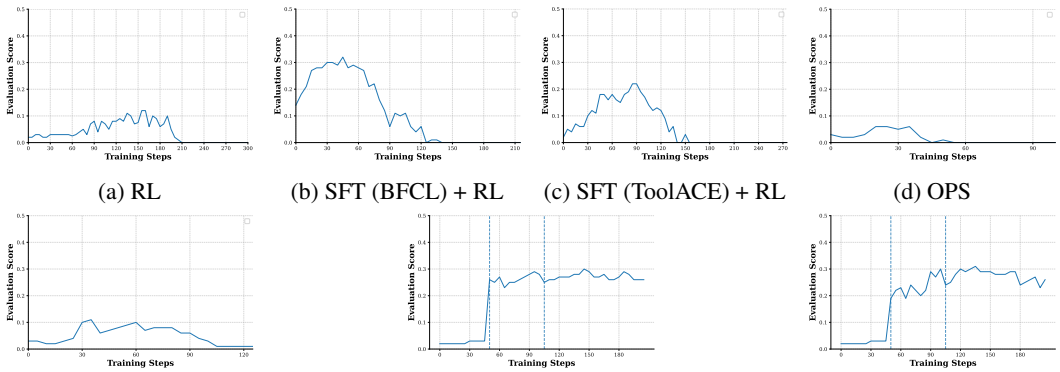
Figure 12: Evaluation results of scenario *Miss Func* under different supervisory signals of Qwen2.5-1.5B-Instruct.




(a) RL (b) SFT (BFCL) + RL (c) SFT (ToolACE) + RL (d) OPS

(e) HBG (f) ETS (SFT 1e−6) (g) ETS (SFT 1e−5) (h) PRS

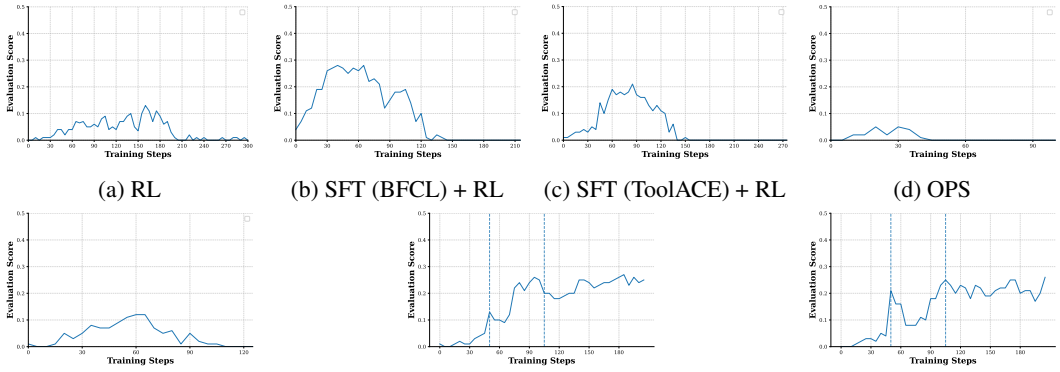
Figure 13: Evaluation results of scenario *Miss Param* under different supervisory signals of Qwen2.5-1.5B-Instruct.




(a) RL (b) SFT (BFCL) + RL (c) SFT (ToolACE) + RL (d) OPS

(e) HBG (f) ETS (g) PRS

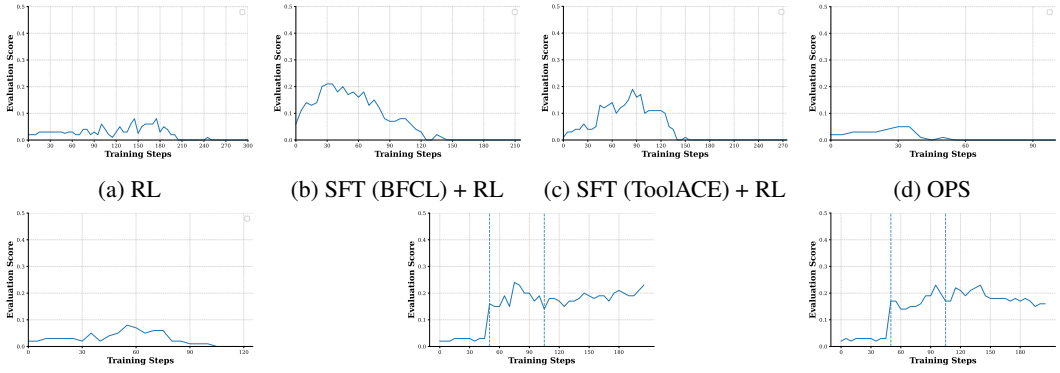
Figure 14: Evaluation results of scenario *Base* under different supervisory signals of Qwen3-1.7B.




(a) RL (b) SFT (BFCL) + RL (c) SFT (ToolACE) + RL (d) OPS

(e) HBG (f) ETS (g) PRS

Figure 15: Evaluation results of scenario *Miss Func* under different supervisory signals of Qwen3-1.7B.




(a) RL (b) SFT (BFCL) + RL (c) SFT (ToolACE) + RL (d) OPS

(e) HBG (f) ETS (g) PRS

Figure 16: Evaluation results of scenario *Miss Param* under different supervisory signals of Qwen3-1.7B.

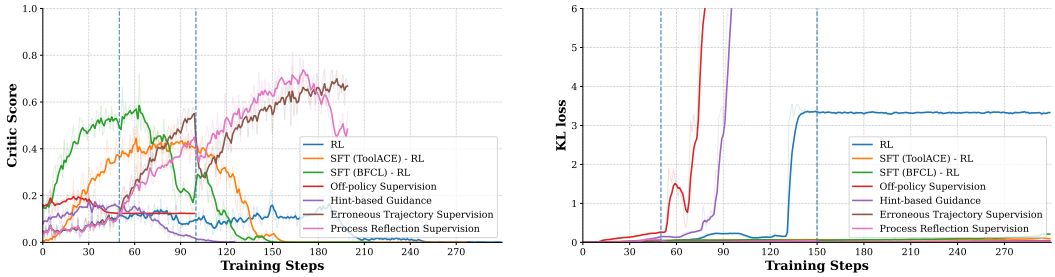



Figure 17: The training dynamics during the training process on the BFCL-V3 dataset of Qwen3-1.7B.

## F PROMPT AND EXAMPLE

The prompt of PRS is in Prompt 1 and the example is in Example 2.

```
You are an expert in intelligent agent tool-use analysis.

Below is the erroneous interaction between a model and tools
(including the ground-truth tool-use sequence for reference). Your
task is to provide a **comprehensive, layered, and actionable
analysis**, and **generalize to similar scenarios** to create
augmentation data.
**Note:** You should write in natural language (not structured
JSON), but every "correct tool call" must strictly follow the raw
<tool_call> ... </tool_call> format.

Inputs to be filled by caller:
- Available Tools: {available_tools}
- Model Interaction Log (Error Turn): {formatted_summary}

Your required outputs:

1. **Core Error Analysis (must be first)**
   - Briefly state the root cause of the error (1-2 lines)
   - Provide 2-4 pieces of evidence from the interaction log to
   support your conclusion
   - Provide one immediate and actionable fix recommendation

2. **Generalization: Generate 3-5 similar scenarios**
   For each scenario, describe in natural paragraphs:
   - One-sentence scenario description (what & why it occurs)
   - The user request and necessary context
   - A typical model mistake that might happen
   - **The correct tool call sequence**
     (each step must use the raw
     `<tool_call>\n{{...}}\n</tool_call>` block, containing exactly
     one JSON object with only `name` and `arguments`)
   - What reasoning procedures can be borrowed from the current
   failed case
   - What aspects differ and require different handling
   - A brief validation strategy for confirming the fix (auto or
   manual)

   The scenarios must cover different error categories:
   - Missing parameters
   - Wrong tool selection
   - Output mismatch
   - Timeout/failure
   - Ambiguous request or multi-intent

3. **Provide one scenario that is similar but requires a different
solution**
   - Explain the similarity (shared reasoning or decision features)
   - Explain clearly why different tool(s) are required
   - Provide correct tool calls in the same `<tool_call>` format
   - Provide rationale

4. **Output style and quality requirements**
   - Natural language, clear paragraph structure
   - No JSON schemas, no markdown code fences, no meta-explanations
   - Tool calls must be valid JSON and executable
   - Provide 1-2 acceptance criteria per scenario for filtering
   low-quality synthetic data
```

```
5. **Example (format reference only, do not repeat unless
needed)**:
<tool_call>
{{"name": "search_database", "arguments": {{"query": "latest model
evaluation", "top_k": 5}}}}
</tool_call>

---

Please begin by analyzing the root cause of the provided error
turn, then generalize to 3--5 similar scenarios (including at
least 1 scenario that requires different handling).
```

Prompt 1: The PRS prompt

```
Turn 1
1) User intent understanding, correct behavior, and success points
- Intent: Price-check then book a business-class SF->LA flight for
Robert Trenton on 2024-11-25 using travel card card_3487 (access
code 1293), and obtain an invoice to verify charges.
- Correct behavior: (a) get flight cost to confirm price, (b) book
the flight with provided card and access token, (c) retrieve the
invoice for verification.
- Success points: All three tools were called in the correct
order. Responses show cost $400, booking_id 3426812 and
transaction_id 45451592, and retrieve_invoice returned a
consistent invoice with cost and booking details. This aligns with
the user's intent and ground truth.

2) Mistake classification and root cause analysis
- Mistakes: None meaningful. Actions, arguments, and sequence
match the ground-truth sequence exactly.
- Root cause: N/A.

3) Improvement plan (corrected reasoning and ideal tool call
sequence)
- Reasoning: Confirm price before booking to avoid unexpected
charges; then book using provided card and access token; finally
retrieve the invoice to verify charges and record transaction
details.
- Ideal tool call sequence (matches ground truth and log):
<tool_call>
{"name": "get_flight_cost", "arguments": {"travel_from": "SFO",
"travel_to": "LAX", "travel_date": "2024-11-25", "travel_class":
"business"}}
</tool_call>
<tool_call>
{"name": "book_flight", "arguments": {"access_token": "1293",
"card_id": "card_3487", "travel_date": "2024-11-25",
"travel_from": "SFO", "travel_to": "LAX", "travel_class":
"business"}}
</tool_call>
<tool_call>
{"name": "retrieve_invoice", "arguments": {"access_token": "1293",
"booking_id": "3426812"}}
</tool_call>

- Alignment: This is identical to the ground truth sequence and
satisfies the user's requirement to book and verify the invoice.
```

```
Turn 2
1) User intent understanding, correct behavior, and success points
- Intent: Post a tweet on Robert Trenton's travel account: "Loved
my flight journey!" with hashtag #TravelDiaries, then retweet it
from his travel account to maximize visibility.
- Correct behavior: Before calling post_tweet/retweet, ensure the
Twitter account is authenticated (credentials or an authenticated
session) and that the correct account is targeted for
posting/retweeting. If credentials or an authenticated account
identifier are missing, request them rather than calling tools.

2) Mistake classification (fine-grained) and root cause analysis
- Mistake classification:
  - Missing Parameter / Precondition Check Failure: The model
  attempted post_tweet without confirming authentication or asking
  for credentials.
  - Incorrect remediation flow: After failed authentication attempt
  using wrong credentials, the model tried retweeting (still
  unauthenticated), producing further errors.
- Root cause:
  - The agent did not treat authentication as a required
  precondition; it attempted actions without validating credentials
  or asking for clarifying information (username, which account to
  use).
  - It also attempted to remediate by guessing credentials
  ("user"/"pass") rather than requesting valid credentials from the
  user.

3) Improvement plan (corrected reasoning and ideal tool call
sequence)
- Corrected reasoning: When a user requests an action that
requires authentication but does not provide credentials or
indicate an authenticated session, the agent must pause and
request the missing authentication information (or ask the user to
confirm which authenticated account to use). No tool calls should
be made until valid credentials or an authenticated session are
provided.
- Action to take now (at this turn): Ask for the necessary
authentication details and account identifier. Do not call any
tools until the user supplies credentials or instructs a
currently-authenticated account to be used.
- No tool calls should be made at this turn. If the user supplies
credentials, the correct sequence (ground truth) would be:
<tool_call>
{"name": "authenticate_twitter", "arguments": {"username": "john",
"password": "john1234"}}
</tool_call>
<tool_call>
{"name": "post_tweet", "arguments": {"content": "Loved my flight
journey!"}}
</tool_call>
<tool_call>
{"name": "retweet", "arguments": {}}
</tool_call>
- How this aligns with ground truth and logic: Ground truth
specifies that because of missing parameters, no tool calls should
be made; instead, the agent should request credentials. The
three-step sequence above is the correct follow-up once
credentials are provided (authenticate -> post -> retweet).

Turn 3
1) User intent understanding, correct behavior, and success points
```

```
- Intent: The user supplied Twitter credentials; authenticate with
those credentials, post the tweet, and retweet it.
- Correct behavior: Authenticate using username/password, post the
requested tweet, then retweet to amplify it.
- Success points: The agent authenticated successfully
(authentication_status: true), posted the tweet ("Loved my flight
journey!"), and retweeted successfully. The returned post id (10)
and retweet success indicate the actions worked and satisfy the
user's request.

2) Mistake classification and root cause analysis
- Mistakes: Minor inconsistency only:
  - The ground-truth post_tweet call used just content (no explicit
  tags). In the log the tweet included tags in earlier attempts;
  the final posted tweet in the log did include tags in earlier
  step but ground truth shows no tags in the tool call. This is a
  minor divergence in arguments, not a functional error-user asked
  for hashtag @TravelDiaries, so including the tag is acceptable.
  - The ground truth retweet call had empty arguments; the agent
  used tweet_id in the log. Either form is acceptable depending on
  API design; using tweet_id is explicit and clear.
- Root cause: Small differences in how the API was used
(presence/absence of tags field, different retweet parameter
shapes). These do not change outcome.

3) Improvement plan (corrected reasoning and ideal tool call
sequence)
- Corrected reasoning: Authenticate with supplied credentials;
once authenticated, post the tweet exactly as requested (include
hashtag /TravelDiaries if the user specified it), capture the
returned tweet id, then retweet that tweet to increase visibility.
Confirm success and return post and retweet statuses to user.
- Ideal tool call sequence (matches ground truth intent; any of
the two retweet argument styles is acceptable depending on
API-here we show the ground-truth calls and the explicit tweet-id
approach):
<tool_call>
{"name": "authenticate_twitter", "arguments": {"username": "john",
"password": "john1234"}}
</tool_call>
<tool_call>
{"name": "post_tweet", "arguments": {"content": "Loved my flight
journey!"}}
</tool_call>
<tool_call>
{"name": "retweet", "arguments": {}}
</tool_call>
- If using the explicit tweet id returned by post_tweet
(recommended for clarity):
<tool_call>
{"name": "retweet", "arguments": {"tweet_id": 10}}
</tool_call>

- Alignment: This aligns with the ground-truth successful
sequence: authenticate -> post_tweet -> retweet. The explicit
tweet_id variant ensures the retweet targets the correct post.

Summary Recommendations (cross-turn)
- Always validate preconditions: check for required credentials or
tokens before calling authentication-protected tools. If missing,
request them.
- Follow a predictable pattern for transactional flows: estimate
cost -> confirm booking -> retrieve invoice.
```

```
- When a user supplies credentials, authenticate then perform
actions; do not guess credentials.
- When API shapes differ (e.g., retweet accepting {} vs tweet_id),
prefer explicit parameters (tweet_id) if available to avoid
ambiguity.
```

Prompt 2: The example of PRS