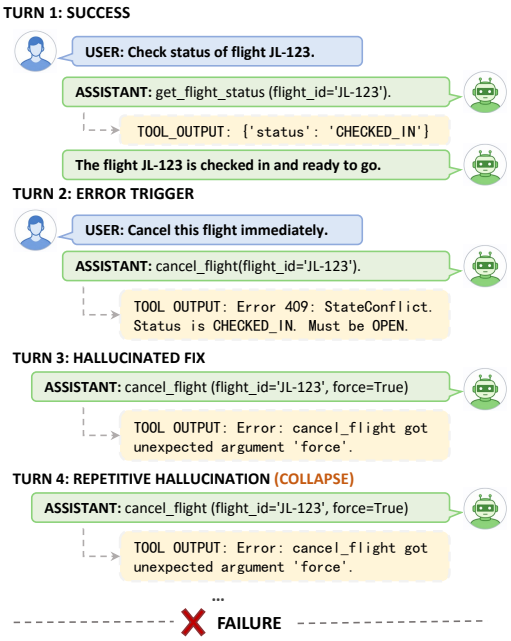
Robust Tool Use via FISSION-GRPO: Learning to Recover from Execution Errors

Zhiwei Zhang[1,3], Fei Zhao[2], Rui Wang[1,3], Zezhong WANG[1], Bin Liang[1,3], Jiakang Wang[2], Yao Hu[2], Shaosheng Cao[2]*, Kam-Fai Wong[1,3]*

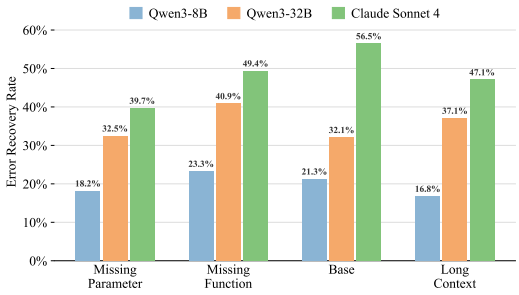
[1]The Chinese University of Hong Kong,
[2]Xiaohongshu Inc.,
[3]MoE Key Laboratory of High Confidence Software Technologies
zhangzhiwei1019@link.cuhk.edu.hk, caoshaosheng@xiaohongshu.com

Abstract

Large language models (LLMs) can call tools effectively, yet they remain brittle in multi-turn execution: after a tool-call error, smaller models often fall into repetitive invalid re-invocations instead of interpreting the feedback and recovering. This failure mode persists because current training paradigms do not explicitly teach models how to recover from execution errors. In particular, standard reinforcement learning (RL) collapses rich failure experience into sparse negative rewards, while pre-collected error-correction datasets become mismatched to the policy's evolving failure modes. To bridge this gap, we propose FISSION-GRPO, a framework that converts execution errors into on-policy corrective supervision within the RL training loop. Our core mechanism *fissions* each failed trajectory into a new training instance by augmenting it with diagnostic feedback from a fine-tuned Error Simulator, then resampling multiple recovery rollouts on-policy. This enables the model to learn from the precise errors it makes during exploration, rather than from static, pre-collected error cases. On BFCL v4 Multi-Turn, FISSION-GRPO improves the error recovery rate of Qwen3-8B by 5.7% absolute and overall accuracy by 4.0% (from 42.75% to 46.75%), outperforming both RL baselines and specialized tool-use agents. The method further generalizes to TAU-Bench and TAU2-Bench, achieving leading results across most settings with gains up to +17.4%.[1]



(a) Typical failure: an API error triggers a hallucinated retry loop.



(b) Error recovery rates on BFCL v4 Multi-Turn across model scales and evaluation subsets.

Figure 1: Error recovery is a key bottleneck for smaller tool-using models in multi-turn execution. (a) shows a representative hallucinated retry loop after an API error, while (b) reports recovery rates on BFCL v4 across model scales.

1 Introduction

Tool-using agents are increasingly moving beyond static text prediction toward interactive decision-making grounded in environment feedback. A growing body of work argues that the next wave of capability gains will come not from scaling human-curated data alone, but from *experience*—data generated by agents interacting with their environments (Silver and Sutton, 2025). Multi-turn tool use is already an instance of this transition: the agent acts via tool calls, observes execution results, and must adapt to changing state and er-

* Corresponding author.

[1]Our code is available at https://github.com/zxzadm/Fission-GRPO.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 40477–40491

ror signals across turns. For such agents to be deployed reliably, especially at smaller scales suitable for low-latency and on-device settings (Belcak et al., 2025; Sharma and Mehta, 2025), they must exhibit *robustness*—the ability to recover from the execution errors that inevitably arise in dynamic, multi-turn environments (Patil et al., 2025).

This robustness requirement exposes a critical gap. In practice, APIs return errors, parameters become invalid, and system states change unexpectedly; a robust agent must interpret such feedback, diagnose the fault, and self-correct (Yao et al., 2022; Shinn et al., 2023). Yet as shown in Figure 1b, smaller models exhibit a pronounced deficiency in *error recovery*—the probability of eventual success conditioned on at least one prior execution error. On BFCL v4 Multi-Turn (Patil et al., 2025), Claude Sonnet 4 exceeds 50% recovery while Qwen3-8B averages only around 20%. Figure 1a illustrates the typical failure mode—hallucinated retries that loop until the conversation collapses.

Current approaches fall short of addressing this challenge. Methods based on **static synthetic datasets** (Liu et al., 2024; Zhang et al., 2025a,b) construct error-correction pairs offline, but the error distribution shifts as the policy improves, making offline error corpora quickly stale—a manifestation of the broader limitation that static, pre-collected data cannot keep pace with an evolving agent (Silver and Sutton, 2025). Meanwhile, **reinforcement learning (RL)** approaches such as GRPO (Shao et al., 2024) treat errors merely as sparse negative rewards. This signals that something went wrong, but offers no guidance on *how* to recover: the gradient discourages the failed action without teaching a corrective alternative. When all sampled rollouts fail, the advantage variance collapses, yielding vanishing gradients that stall learning entirely (Yu et al., 2025; Nan et al., 2025). In essence, existing methods treat errors as outcomes to be *avoided* rather than experiences to be *learned from*.

To bridge this gap, we propose FISSION-GRPO, a framework that transforms execution errors into dense, on-policy corrective experience (Figure 2). In **Stage 1**, we perform standard GRPO exploration, sampling multiple rollouts per query and updating the policy with group-relative advantages. In **Stage 2**, failed rollouts are intercepted and augmented with diagnostic feedback from a learned Error Simulator, constructing corrective contexts of the form [*dialogue*; *failed call*; *feedback*]. In **Stage 3**, these contexts trigger a *fission* update: each error is expanded into G' parallel recovery attempts by resampling new rollouts from the augmented context—analogous to nuclear fission, where a single event induces a chain of reactions, generating dense training signals from a single failure.

The Error Simulator is trained via supervised fine-tuning to produce realistic, context-aware diagnostics resembling runtime error traces, with outputs restricted to non-revealing descriptions (e.g., "parameter `status` expects value `OPEN`") to prevent target leakage. This closed-loop process continuously focuses learning on the model's current error modes, mitigating the distribution mismatch inherent in static error-correction datasets.

We evaluate FISSION-GRPO on BFCL v4 Multi-Turn, TAU-Bench (Yao et al., 2024), and TAU2-Bench (Barres et al., 2025), with consistent improvements across benchmarks and model scales. Our main contributions are:

- **Fission-GRPO Framework.** We propose an RL framework that dynamically converts execution errors into corrective training instances. By resampling from augmented error contexts on-policy, our approach maintains alignment with the model's evolving error distribution.
- **Learned Error Simulator.** We develop a fine-tuned error simulator that generates realistic diagnostic feedback without target leakage, enabling effective recovery training without live API interactions. Cross-domain evaluations on unseen tool schemas and human evaluation (96% non-leakage; Cohen's $\kappa = 0.71$) confirm its reliability and generalizability.
- **Empirical Validation.** FISSION-GRPO achieves state-of-the-art performance on BFCL v4 Multi-Turn across the Qwen3 model family (1.7B, 4B, and 8B). For Qwen3-8B, it improves error recovery by 5.7% absolute and overall accuracy by 4.0% (42.75% $\rightarrow$ 46.75%). The method further generalizes to TAU-Bench and TAU2-Bench with entirely different tool ecosystems, achieving leading results across most settings with gains up to +17.4% on TAU1 Retail.

2 Related Work

2.1 RL for Tool Use

RL has become the standard for aligning LLMs (Schulman et al., 2017; Ouyang et al., 2022). Among recent algorithms, GRPO (Shao et al.,

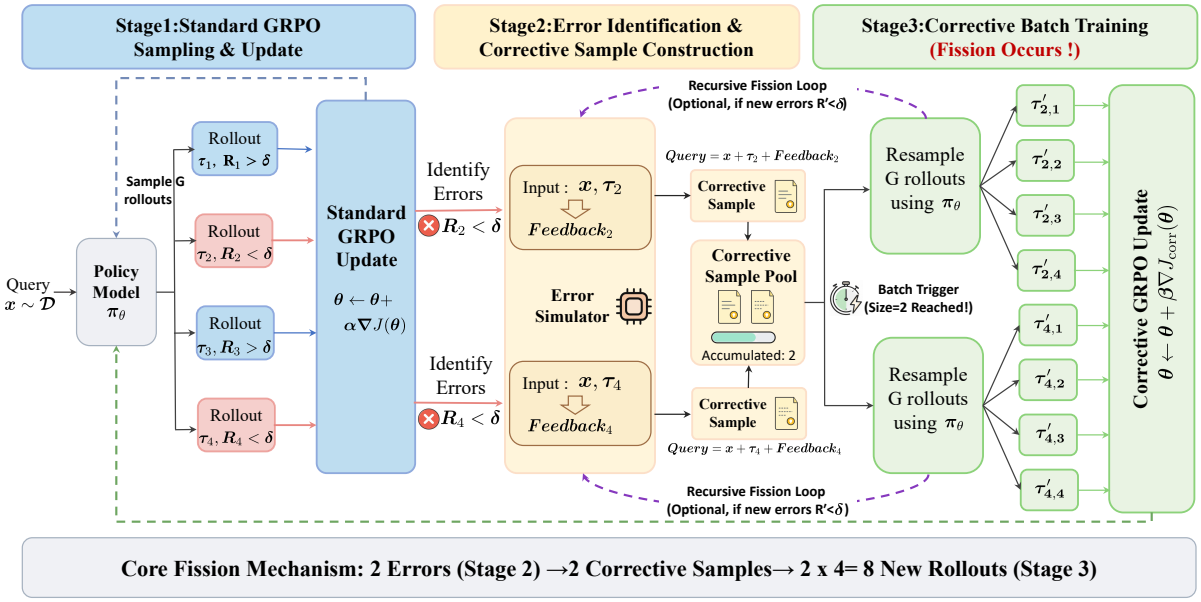


Figure 2: Overview of the FISSION-GRPO Framework. The framework operates in three stages: (1) **Standard Exploration**, utilizing GRPO to optimize policy π_θ on the query distribution $\mathcal{D}$; (2) **Error Identification & Synthesis**, where a simulator $\mathcal{S}_\phi$ generates diagnostic feedback for filtered error trajectories; and (3) **Fission-based Update**, where corrective samples trigger a multiplicative resampling process (factor G') to align the policy with recovery paths.

2024) eliminates the value network by estimating baselines from group averages, making it well-suited to tool-calling tasks with binary or scalar rewards (Guo et al., 2025).

However, GRPO's reliance on intra-group variance creates a failure mode: when a sampled group is homogeneously incorrect, reward variance vanishes and gradients become null—a limitation targeted by DAPO (Yu et al., 2025) and NGRPO (Nan et al., 2025). Even when gradients exist, indiscriminate negative feedback can trigger *Lazy Likelihood Displacement (LLD)* (Deng et al., 2025), suppressing valid reasoning steps merely because they co-occur with failed trajectories.

Existing mitigations—filtering homogeneous batches (DAPO), calibrating advantages (NGRPO), or down-weighting negative gradients (NTHR (Deng et al., 2025))—reshape the *loss landscape* of negative signals but leave the scarcity of positive guidance during exploration unaddressed. Our approach instead actively constructs recovery trajectories via fission, turning zero-reward errors into dense learning signals.

2.2 Robust Tool Use and Error-Driven Synthesis

Research in tool utilization has evolved from single-turn syntactic correctness (Schick et al., 2023; Patil et al., 2024) to reliability across multi-turn workflows (Qin et al., 2023; Yao et al., 2022) and human-level task solving (Wang et al., 2025; Fang et al., 2025b). As tasks grow more complex, recovery from inevitable environment errors (e.g., timeouts, invalid parameters) becomes a defining metric of robustness, as codified in benchmarks like BFCL (Patil et al., 2025) and StableToolBench (Guo et al., 2024).

Recent work addresses this via "diagnosis-and-repair" mechanisms (Su et al., 2025; Huang et al., 2025) or training on diverse error scenarios (Vuddanti et al., 2025). Synthetic correction methods, proven in reasoning domains (Pan et al., 2025; Xu et al., 2025), have been adapted to tool use by frameworks like ToolACE (Liu et al., 2024) and LoopTool (Zhang et al., 2025b) to expand training coverage through model-based synthesis.

Yet these methods rely predominantly on *offline* data construction, creating a temporal mismatch with the policy's evolving on-policy error distribution (Kumar et al., 2024; Zhang et al., 2025b; Fang et al., 2025a; Li et al., 2026). Unlike prior offline synthesis (Pan et al., 2025; Su et al., 2025), our work integrates error simulation directly into the training loop, keeping supervision aligned with current policy limitations.

3 Method

We propose FISSION-GRPO, a framework designed to imbue small language models with robust error recovery capabilities. As illustrated in

Figure 2, our approach operates in a three-stage closed loop: standard GRPO exploration maintains general tool-use competence, while a conditional fission stream intercepts errors and triggers active remedial learning.

3.1 Preliminaries

We formulate tool use as a language generation task. Given a query x and a tool library, a policy π_θ generates a trajectory τ consisting of reasoning thoughts and tool calls. We adopt **GRPO** (Shao et al., 2024) as our optimization backbone. Unlike PPO, GRPO eliminates the need for a value network by estimating the baseline from the group average. For each query x, we sample a group of outputs $\{\tau_i\}_{i=1}^{G}$ and optimize:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{G}\sum_{i=1}^{G}\hat{A}(\tau_i)\cdot\pi_{\text{ratio}}(\tau_i \mid x) - \beta\mathbb{D}_{\text{KL}}\right] \tag{1}$$

where $\hat{A}(\tau_i) = \frac{R(\tau_i)-\mu_R}{\sigma_R+\epsilon}$ is the group-normalized advantage, with μ_R and σ_R being the mean and standard deviation of rewards within the group, and $\pi_{\text{ratio}}(\tau_i \mid x)$ is the clipped probability ratio.

3.2 Reward Design

To guide the policy from syntactic compliance to semantic precision, we design a time-dependent composite reward with normalized aggregation. Each component is scaled by a time-varying weight, and the final reward is the normalized sum:

$$R(\tau,t) = \frac{1}{3}\Big[\, w_{\text{fmt}}(t)\, R_{\text{fmt}}(\tau) + w_{\text{corr}}(t)\, R_{\text{corr}}(\tau) + R_{\text{len}}(\tau)\,\Big] \tag{2}$$

where each weighted term is bounded such that $R(\tau,t)\in[0,2]$ throughout training.

Format Compliance (R_{fmt}). This binary term $R_{\text{fmt}}(\tau)\in\{0,1\}$ enforces structural constraints, ensuring outputs adhere to the required XML/JSON schema. We apply a decaying weight $w_{\text{fmt}}(t)$ that anneals the maximum weighted contribution from 3 to 2, shifting focus from syntax to semantics as training progresses.

Functional Correctness (R_{corr}). This term evaluates alignment between invoked tools and user intent. To accommodate partial matching in complex parameters, we define $R_{\text{corr}}(\tau, y^*)\in[0,1]$ as:

$$R_{\text{corr}}(\tau,y^*) = \alpha\cdot\mathbb{I}(N=N^*) + (1-\alpha)\cdot\frac{1}{|\mathcal{M}|}\sum_{(a,a^*)\in\mathcal{M}}\text{F1}(a,a^*) \tag{3}$$

where $\alpha\in[0,1]$ (set to $\alpha=0.5$ in our experiments) balances function selection against parameter matching, $\mathbb{I}(N=N^*)$ indicates correct function selection, $\mathcal{M}$ denotes matched argument pairs, and F1 measures token-level overlap. The weight $w_{\text{corr}}(t)$ increases monotonically, scaling its maximum weighted contribution from 2 to 3 to prioritize parameter precision in later stages.

Efficiency Regularization (R_{len}). To prevent verbose or degenerate reasoning, we add a length-compliance reward $R_{\text{len}}\in[0,1]$, computed via a piecewise Gaussian function that peaks at the target length and decays on both sides.

3.3 The FISSION-GRPO Framework

As illustrated in Figure 2, FISSION-GRPO operates in a three-stage closed loop. Stage 1 focuses on optimizing **fundamental tool-use capabilities**, while Stages 2 and 3 are dedicated to developing **error recovery skills** through targeted error correction.

3.3.1 Stage 1: Standard Exploration and Update

This stage aims to establish and maintain the model's base performance on tool-calling tasks.

Sampling and Evaluation. Given a query x, we sample a group of trajectories $\{\tau_i\}_{i=1}^{G}$ from the current policy π_θ. We evaluate these rollouts using the composite reward function defined in §3.2, computing format compliance R_{fmt}, functional correctness R_{corr}, and efficiency regularization R_{len}, which are then aggregated into the total reward.

Optimization. We apply the standard GRPO update (Eq. 1) using these trajectories to improve the model's fundamental tool-use capabilities. All sampled trajectories are then forwarded to Stage 2 for diagnostic error analysis and corrective training.

3.3.2 Stage 2: Error Identification and Corrective Sample Construction

Stage 2 converts error traces produced in Stage 1 into actionable corrective instances. Concretely, we apply a two-level filter to isolate erroneous trajectories and then synthesize feedback that can be

appended to the original context for subsequent corrective updates.

Error Identification. We decompose error detection into *format validity* and *functional correctness*. Let R_{fmt} denote whether the tool-call format is valid. If $R_{\text{fmt}}(\tau) = 0$, the trajectory is immediately treated as an error without consulting correctness. Otherwise, we further evaluate correctness with a scalar score R_{corr} and flag the trajectory when it falls below a tunable threshold δ_{corr}:

$$\mathcal{E} = \{\tau_i \mid R_{\text{corr}}(\tau_i) < \delta_{\text{corr}} \vee R_{\text{fmt}}(\tau_i) = 0\} \quad (4)$$

In Fig. 2, we use a simplified illustration (e.g., $R < \delta$) to emphasize the gating effect; this does not contradict Eq. (4).

Hybrid Feedback Synthesis. For effective correction, a scalar penalty is insufficient; we require an explicit diagnostic message f that resembles the runtime system feedback. We adopt a hybrid strategy: (i) for *format errors* ($R_{\text{fmt}} = 0$), we use deterministic error messages (e.g., parser/compiler-style feedback) to explicitly state the violated schema/serialization constraints; (ii) for *semantic errors* ($R_{\text{corr}} < \delta_{\text{corr}}$), we query a learned **Error Simulator** S_ϕ to produce a concise, actionable runtime error string.

The simulator is implemented as a Qwen3-32B model fine-tuned via SFT to emulate runtime environment responses. We construct a training set of approximately 2K instances from error logs, where each instance comprises: (i) the original system prompt and tool specification along with the dialogue state, (ii) the model's failed tool call (τ_{err}), (iii) the ground-truth tool call (τ_{gt}), and (iv) a teacher-written diagnostic error message (via Claude Sonnet 4), after quality filtering. During both training and inference, the simulator consumes $(\text{system} + \text{tools},\ \text{dialogue history},\ \tau_{\text{gt}},\ \tau_{\text{err}})$ and produces a concise feedback string $f \leftarrow S_\phi(x, \tau_{\text{err}}, \tau_{\text{gt}})$, where f is constrained to be a realistic runtime response. We provide the exact prompting template used to query S_ϕ in Appendix A.

Corrective Sample Construction and LIFO Buffering. Given a flagged trajectory τ_{err} with feedback f, we construct a corrective context $x_{\text{corr}} = [x; \tau_{\text{err}}; f]$ by appending the failed attempt and the diagnostic message to the original multi-turn input. We optionally deduplicate corrective instances by hashing the pair (x, τ_{err}), so that distinct diagnostic messages synthesized for the same underlying error are not treated as separate training instances. All corrective samples are stored in a **LIFO** buffer $\mathcal{B}_{\text{corr}}$, so that the most recent errors are consumed first during corrective updates. This design keeps the corrective batch distribution closer to the current policy π_θ, improving the on-policy approximation in multi-turn tool-use training.

3.3.3 Stage 3: Corrective Batch Training

Once the LIFO buffer accumulates sufficient *recent* errors (Batch Trigger), we activate **Fission** to perform targeted remedial updates for recovery.

Multiplicative Resampling. We pop the freshest corrective contexts x_{corr} and, for each of them, sample a "fission group" of G' trajectories conditioned on the same context:

$$\{\tau'_j\}_{j=1}^{G'} \sim \pi_\theta(\cdot \mid x_{\text{corr}}). \quad (5)$$

This turns a single error case into multiple parallel recovery attempts, densifying training signals around the observed error.

More Informative Advantages. Hard queries can yield near-homogeneous outcomes in standard exploration, weakening within-group relative advantages. Conditioning on explicit feedback f typically increases outcome diversity within the fission group, improving the usefulness of advantage estimates for recovery updates. For each corrective context x_{corr}, we compute rewards over the sampled recovery trajectories and normalize them within the fission group: $\hat{A}(\tau'_j) = (R(\tau'_j) - \mu'_R)/(\sigma'_R + \epsilon)$, where μ'_R and σ'_R denote the mean and standard deviation of rewards within the corrective group.

We then optimize the same GRPO-style clipped surrogate objective as in Eq. 1, but over the corrective distribution:

$$\mathcal{J}_{\text{corr}}(\theta) = \mathbb{E}_{x_{\text{corr}}}\Big[\frac{1}{G'}\sum_{j=1}^{G'} \hat{A}(\tau'_j) \cdot \pi_{\text{ratio}}(\tau'_j \mid x_{\text{corr}}) - \beta \mathbb{D}_{\text{KL}}\Big]. \quad (6)$$

This corrective objective preserves the optimization form of standard GRPO while shifting the training distribution toward the policy's current failure modes. As a result, the model is explicitly optimized not only to avoid errors, but also to recover from them under feedback-augmented contexts.

Summary. These three stages form a continuous loop; detailed pseudocode and hyperparameters are provided in Algorithm 1 (Appendix C).

4 Experiments

4.1 Experimental Setup

Data Construction Diverging from prevalent tool-learning paradigms that emphasize extensive scaling of synthetic corpora (e.g., ToolACE (Liu et al., 2024), XLAM (Zhang et al., 2025a)), we prioritize *data quality* and *trajectory correctness*. We implement a three-stage pipeline to construct a compact yet rigorous training set:

(1) **Domain Schema Curation**: We curated a diverse schema library spanning 11 domains (e.g., *Healthcare, Smart Home, Vehicle Control*), prompting Claude Sonnet 4 to generate realistic API definitions grounded in BFCL characteristics.

(2) **Trajectory Synthesis**: Utilizing Claude Sonnet 4, we first synthesized multi-turn user queries based on these schemas, followed by generating full interaction trajectories that fulfill the requests.

(3) **Hierarchical Filtering and Factorization**: To ensure rigorous quality control, we applied a hierarchical protocol. First, raw trajectories underwent a global coherence check via Claude Sonnet 4. Validated trajectories of length K were then factorized into discrete decision instances $\{(h_t, a_t)\}_{t=1}^{K}$, where h_t denotes the cumulative context history. Finally, these decomposed instances underwent a dual-model verification via Qwen3-235B-A22B-Instruct-2507 (Team, 2025) and Kimi K2 (Team et al., 2025). Only samples achieving unanimous consensus were retained, distilling an initial pool of $\sim$2,000 trajectories down to 630 high-quality training instances.

Training Details. All models are trained using the **Verl** framework (Sheng et al., 2024) on a single node with 8×H800 80GB GPUs. For GRPO training, we use a learning rate of 1e-6 with cosine warmup, a batch size of 8, and sample 8 rollouts per query ($G = 8$). The maximum prompt length is set to 12,800 tokens and the maximum response length to 4,096 tokens. We use temperature 0.95 and top-k 50 for sampling. For FISSION-GRPO, we set the correctness threshold $\delta_{\text{corr}} = 1$ for error identification (Eq. 4), determined empirically as a stable threshold across multiple runs.

Benchmarks. We evaluate on three complementary multi-turn tool-use benchmarks, both of which feature interactive error feedback mechanisms that permit the agent to retry after execution errors—directly aligning with our focus on error recovery. **BFCL v4 Multi-Turn** (Patil et al., 2025) stress-tests state tracking and robustness across up to 20 retry attempts per error, enabling fine-grained measurement of how error recovery dynamics translate to overall success. **TAU-Bench** (Yao et al., 2024) and **TAU2-Bench** (Barres et al., 2025) feature genuine multi-turn interactions with LLM-simulated users, employing tool APIs, conversation dynamics, and error distributions substantially different from BFCL. We use GLM-5 as the user simulator and report the pass@1 metric across five settings (Retail, Airline for TAU1; Retail, Airline, Telecom for TAU2).

Baselines. We compare FISSION-GRPO against RL baselines implemented on the Qwen3 series (1.7B/4B/8B), including: (1) **GRPO** (Shao et al., 2024), utilizing group-normalized advantages; (2) **DAPO** (Yu et al., 2025), incorporating dynamic sampling constraints; (3) **Dr.GRPO** (Liu et al., 2025), employing mean-centered estimators to mitigate length bias; and (4) **AWPO** (Lin et al., 2025), which introduces adaptive reward weighting to improve GRPO for tool-use tasks. For broader context, we also report performance of specialized 8B-scale tool agents such as ToolACE (Liu et al., 2024) and BitAgent.

4.2 Main Results

Table 1 presents results on BFCL v4 Multi-Turn, TAU-Bench, and TAU2-Bench. FISSION-GRPO delivers strong and consistent gains across all Qwen3 scales (1.7B, 4B, and 8B) relative to standard GRPO and other post-training baselines. On Qwen3-1.7B, FISSION-GRPO lifts BFCL accuracy from 7.80% to **20.38%** (an absolute gain of 12.58 points). The advantage persists at larger scales: Qwen3-4B and Qwen3-8B reach **40.87%** and **46.75%** BFCL accuracy, while the 8B model further achieves the top TAU-Bench scores of **51.3%** (Retail) and **40.0%** (Airline). FISSION-GRPO also performs particularly well in the *Base* and *Miss Param* categories (e.g., 57.50% on 8B Base and 30.50% on 4B Miss Param), indicating more accurate function and parameter handling. Compared with specialized 8B agents, FISSION-GRPO (Qwen3-8B) surpasses ToolACE-2-8B and BitAgent-8B on BFCL by 9.75 and 9.00 points, respectively, and by a much larger margin on TAU-

Method	BFCL v4 Multi-Turn					TAU-Bench		TAU2-Bench		
	Overall	Base	Miss Func	Miss Param	Long Ctx	Retail	Airline	Retail	Airline	Telecom
Qwen3-1.7B Models										
Base	7.80	10.00	11.00	8.00	2.50	6.1	14.0	7.9	12.0	25.0
GRPO	17.12	22.00	**18.50**	15.50	12.50	7.0	20.0	8.5	12.7	32.5
DAPO	16.00	22.00	17.00	14.00	11.00	**8.7**	18.0	6.7	14.0	28.3
Dr.GRPO	16.12	19.50	17.50	14.50	13.00	7.8	22.0	**9.4**	15.3	32.5
FISSION-GRPO	**20.38**	**29.00**	**18.50**	**16.00**	**18.00**	7.8	**24.0**	8.5	**16.0**	**37.5**
Qwen3-4B Models										
Base	19.37	24.50	19.00	14.50	19.50	19.1	26.0	22.8	20.0	22.5
GRPO	36.38	46.50	34.50	27.50	37.00	20.0	24.0	27.2	26.0	37.5
DAPO	38.25	48.50	36.50	28.00	**40.00**	21.7	**32.0**	28.1	24.0	30.0
Dr.GRPO	34.75	43.00	34.50	27.50	34.00	20.9	22.0	23.7	26.0	35.0
AWPO	38.75	43.00	39.50	**36.50**	36.00	27.0	22.0	**29.8**	28.0	40.0
FISSION-GRPO	**40.87**	**51.50**	**42.50**	30.50	39.00	**37.4**	30.0	29.0	**32.0**	**42.5**
External 8B Agent Models										
ToolACE-2-8B	37.00	47.00	31.00	28.00	42.00	0.9	4.0	8.2	26.7	26.7
BitAgent-8B	37.75	46.50	37.50	24.00	43.00	2.6	6.0	6.1	37.3	6.7
Qwen3-8B Models										
Base	28.75	35.50	37.00	22.50	20.00	35.7	23.2	29.8	28.0	40.0
GRPO	42.75	50.50	41.00	36.00	43.50	39.1	36.0	28.1	26.0	52.5
DAPO	43.12	54.50	44.50	29.00	44.50	45.2	20.0	34.2	26.0	47.5
Dr.GRPO	44.88	55.00	**45.00**	32.50	47.00	47.0	28.0	**39.5**	**34.0**	35.0
AWPO	44.50	53.50	41.50	**40.50**	42.50	43.5	30.0	32.5	26.0	45.0
FISSION-GRPO	**46.75**	**57.50**	43.50	38.00	**48.00**	**51.3**	**40.0**	36.8	32.0	**55.0**

Table 1: Main results on BFCL v4 Multi-Turn, TAU-Bench, and TAU2-Bench. BFCL reports overall and category-level accuracy (%), while TAU-Bench and TAU2-Bench report pass@1 (%) under simulated-user interaction. The best result within each model-size group is highlighted in **bold**; rows shaded in blue denote our proposed FISSION-GRPO.

Bench Retail.

4.3 Error Recovery Analysis

To identify the source of performance gains, we decompose the overall success rate into two components: *One-Shot Success Rate* (success without triggering errors) and *Error Recovery Rate* (success conditioned on the occurrence of an error).

Figure 3 illustrates this breakdown for the Qwen3-8B model. The results clearly indicate that the performance improvement is primarily driven by enhanced error recovery capabilities. FISSION-GRPO yields an average improvement of **5.7%** in Error Recovery Rate across all categories, with particularly substantial gains in *Long Context* (+11.8%) and *Base* (+5.5%) scenarios.

Crucially, this gain does not come at the expense of fundamental capabilities. The One-Shot Success Rate is preserved and even modestly improved by an average of 1.75%, confirming that the fission mechanism provides complementary benefits to both error prevention and error correction.

4.4 Impact of Feedback Quality

To disentangle the contribution of the *Fission* mechanism from the informational gain of the Error Simulator, we conduct an ablation study across three settings: (1) **GRPO**: The standard baseline without explicit recovery training. (2) **Fission-Static**: Applies the fission update but uses a fixed, generic error message for all errors.[2] (3) **Fission-Dynamic**: Our full method using the Error Simulator for context-aware feedback.

Results in Table 2 show a clear progression from GRPO to Fission-Static to Fission-Dynamic. First, Fission-Static consistently outperforms GRPO. Even with uninformative feedback, resampling recovery attempts from failed contexts encourages

[2]The static prompt is: "ERROR: Function call failed. Please verify your output format, function name, required parameters, and parameter values are correct."

Figure 3: Performance decomposition on BFCL v4 Multi-Turn (Qwen3-8B).

Method	Avg.	Base	M.Func	M.Param	Long
Qwen3-1.7B					
GRPO	17.12	22.00	18.50	15.50	12.50
Static	17.75	23.50	**21.00**	15.50	11.00
Dynamic	**20.38**	**29.00**	18.50	**16.00**	**18.00**
Qwen3-4B					
GRPO	36.38	46.50	34.50	27.50	37.00
Static	37.25	50.50	34.00	28.50	36.00
Dynamic	**40.87**	**51.50**	**42.50**	**30.50**	**39.00**
Qwen3-8B					
GRPO	42.75	50.50	41.00	36.00	43.50
Static	44.00	53.50	43.00	35.50	44.00
Dynamic	**46.75**	**57.50**	**43.50**	**38.00**	**48.00**

Table 2: Ablation on feedback quality. *Static* denotes Fission training with generic error prompts; *Dynamic* uses our simulated feedback. *Avg.* denotes overall accuracy; *M.Func* and *M.Param* denote missing function and missing parameter errors, respectively. The best result in each panel is in **bold**.

Dimension	In-domain	Out-of-domain	Δ
Localization	4.21	3.97	−0.24
Actionability	4.26	4.12	−0.14
Non-leakage	4.88	4.83	−0.05

Table 3: Error Simulator cross-domain evaluation (1–5 scale). The simulator maintains high quality on unseen domains with minimal degradation.

the model to refine its internal state tracking (e.g., +1.25% on Qwen3-8B Overall Acc), validating that the *structural* intervention of the fission mechanism is inherently valuable. Since Fission-Static uses a fixed generic prompt containing zero teacher-derived information, its gains over GRPO indicate that the fission mechanism itself contributes meaningful learning benefits, rather than merely inheriting information from larger models. Second, Fission-Dynamic yields substantial additional gains, especially at 4B and 8B. The gap between Static and Dynamic (e.g., +3.62 points on Qwen3-4B) underscores the necessity of precise supervision. Generic signals fail to guide the model through complex errors, whereas simulated feedback more effectively guides learning toward correcting specific semantic errors, particularly in the *Miss Param* and *Long Context* subsets.

4.5 Error Simulator Analysis

A key concern is whether the Error Simulator generalizes beyond its training domains and whether its outputs inadvertently leak ground-truth information. We evaluate both aspects through automated and human assessments across two test sets: (i) an **in-domain** set of 200 held-out error trajectories from the original training domains (excluded from simulator training), and (ii) an **out-of-domain** set of 200 real rollout failures collected from live inference in two new domains—e-commerce order management and calendar scheduling—with no additional fine-tuning of the simulator. We evaluate along three dimensions using Claude Sonnet 4 as an LLM judge (1–5 scale): *Localization* (does the feedback pinpoint the fault?), *Actionability* (can the student attempt a correction?), and *Non-leakage* (does it avoid exposing ground-truth details?).

As shown in Table 3, the cross-domain drop is at most 0.24 points. Non-leakage remains near-ceiling in both in-domain and out-of-domain settings ($4.88 \rightarrow 4.83$), suggesting that the simulator does not degenerate into reproducing ground-truth calls when faced with unfamiliar schemas. To further validate this metric, two independent annotators rated 100 randomly sampled outputs, of which 96% were judged strictly non-leaking, with Cohen's $\kappa = 0.71$ (substantial agreement), broadly consistent with the automated scores.

4.6 Trigger Frequency and Compute Efficiency

We study how the minimum trigger interval N (in global steps) affects performance. N limits cor-

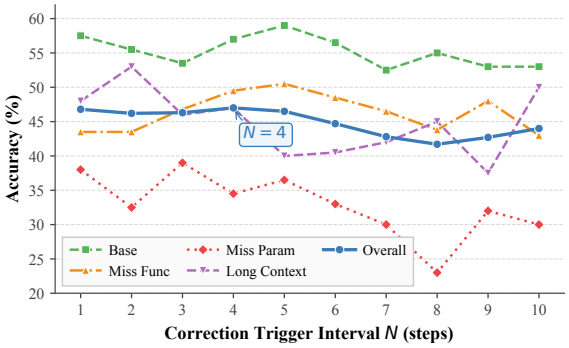


Figure 4: Multi-turn performance across different correction trigger intervals (N) on BFCL v4 Multi-Turn.

N	Method	Avg.	Base	M.F	M.P	Long
3	GRPO (432 upd.)	41.75	56.50	40.00	29.50	41.00
	Fission (432 upd.)	**46.75**	**57.50**	**43.50**	**38.00**	**48.00**
6	GRPO (312 upd.)	43.75	**53.00**	44.00	31.50	46.50
	Fission (312 upd.)	**46.12**	**53.00**	**48.50**	**36.00**	**47.00**

Table 4: Compute-matched comparison on BFCL v4 Multi-Turn (Qwen3-8B). Both methods use identical total update steps.

rection updates to occur at most once every N global steps; we fix the training budget to 234 standard updates and vary only N, using a LIFO strategy to prioritize the most recent errors. Figure 4 shows that performance remains stable for small-to-moderate N but degrades noticeably as corrections become sparse. The drop is most pronounced on Miss Param and Long Context, indicating that parameter-level errors and long-context interactions particularly benefit from timely correction. This stability over a range of N values highlights a practical trade-off: correction does not need to be extremely frequent to remain effective, but overly sparse schedules allow error patterns to accumulate unchecked.

Compute-Matched Comparison. To verify that fission gains reflect more effective budget allocation rather than additional compute, we set $G' = G$ so that each fission update has the same cost as a standard GRPO update, then compare both methods at the same total number of update steps under two trigger settings.

As shown in Table 4, FISSION-GRPO consistently outperforms GRPO under matched compute, with gains concentrated on Miss Param (+8.50 / +4.50) and Miss Func (+3.50 / +4.50)—precisely the error-recovery-heavy subsets. This confirms that fission reallocates the training budget toward the policy's failure modes rather than simply inflating compute. Representative training curves for GRPO and FISSION-GRPO are provided in Appendix B; both exhibit stable reward dynamics with no evidence of collapse or divergence.

4.7 Case Study: Error Recovery Behaviors

To qualitatively illustrate the robustness improvements, we compare three Qwen3-8B variants (Base, GRPO, FISSION-GRPO) on a representative multi-turn file manipulation task (from BFCL v4 Multi-Turn Base) requiring state tracking across directory changes and file moves (full logs in Appendix D).

We observe three distinct error-recovery patterns. The **Base** model exhibits *collapse*: it fails to update internal state after partial command success, entering repetitive invalid retries until conversation breakdown. **GRPO** shows *hallucination*: it recognizes errors but lacks grounding—when a file path becomes invalid, it invents non-existent parameters (e.g., a `path` argument for `ls`) rather than verifying the actual state. In contrast, **FISSION-GRPO** demonstrates *active diagnosis*: it employs a diagnose-then-correct strategy, deploying verification tools (e.g., `find`) to resolve state uncertainty before reattempting the task. This comparison shows that FISSION-GRPO transforms error signals into active diagnostic capabilities rather than brittle heuristics.

5 Conclusion

We presented FISSION-GRPO, a framework that transforms execution errors into on-policy corrective supervision for multi-turn tool use. By intercepting failures, augmenting them with simulated feedback, and resampling recovery attempts, our approach enables smaller models to learn robust self-correction rather than collapsing into repetitive loops. On BFCL v4 Multi-Turn, FISSION-GRPO improves Qwen3-8B by 5.7% in error recovery and 4.0% in overall accuracy, while also showing consistent gains on TAU-Bench and TAU2-Bench. More broadly, the fission paradigm may extend to other iterative refinement domains such as code debugging and mathematical reasoning.

Limitations

Our work has several limitations that suggest directions for future research.

Evaluation Scope. We evaluate FISSION-GRPO on the BFCL v4 Multi-Turn benchmark and TAU-

Bench / TAU2-Bench, both of which feature interactive error feedback mechanisms with retry attempts. While these benchmarks cover diverse tool APIs and error dynamics, our evaluation remains within the domain of tool-calling agents. Extending to other settings with error-retry dynamics (e.g., interactive code debugging or web navigation with fallback) is a promising direction for future work.

Computational Overhead. The fission mechanism introduces additional computational cost by resampling G' rollouts for each intercepted error. Our compute-matched experiments (§4.6) show that gains persist under identical update budgets, and a configurable trigger interval N allows trading off correction frequency against training efficiency. Nonetheless, scaling to very large models or environments with expensive real API calls could amplify absolute costs, and further optimization of fission scheduling remains an open direction.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. This work is partially supported by Hong Kong RGC GRF No. 14206324.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*.

Wenlong Deng, Yi Ren, Muchen Li, Danica J Sutherland, Xiaoxiao Li, and Christos Thrampoulidis. 2025. On the effect of negative gradient in group relative deep reinforcement optimization. *arXiv preprint arXiv:2505.18830*.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. 2025a. Webevolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8970–8986.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, Hongming Zhang, Haitao Mi, and Dong Yu. 2025b. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *Preprint*, arXiv:2508.00414.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *ACL (Findings)*.

Shiting Huang, Zhen Fang, Zehui Chen, Siyu Yuan, Junjie Ye, Yu Zeng, Lin Chen, Qi Mao, and Feng Zhao. 2025. Critictool: Evaluating self-critique capabilities of large language models in tool-calling error scenarios. *arXiv preprint arXiv:2506.13977*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and 1 others. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. 2026. Verified critical step optimization for LLM agents. *CoRR*, abs/2602.03412.

Zihan Lin, Xiaohan Wang, Hexiong Yang, Jiajun Chai, Jie Cao, Guojun Yin, Wei Lin, and Ran He. 2025. Awpo: Enhancing tool-use of large language models through adaptive integration of reasoning rewards. *arXiv preprint arXiv:2512.19126*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and 1 others. 2025. Ngrpo: Negative-enhanced group relative policy optimization. *arXiv preprint arXiv:2509.18851*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H Vicky Zhao, Conghui He, and Lijun Wu. 2025. Lemma: Learning from errors for mathematical advancement in llms. *arXiv preprint arXiv:2503.17439*.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

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

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

Raghav Sharma and Manan Mehta. 2025. Small language models for agentic systems: A survey of architectures, capabilities, and deployment trade offs. *arXiv preprint arXiv:2510.03847*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

David Silver and Richard S Sutton. 2025. Welcome to the era of experience. *Google AI*, 1:11.

Junhao Su, Yuanliang Wan, Junwei Yang, Hengyu Shi, Tianyang Han, Junfeng Luo, and Yurui Qiu. 2025. Failure makes the agent stronger: Enhancing accuracy through structured reflection for reliable tool interactions. *arXiv preprint arXiv:2509.18847*.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Sri Vatsa Vuddanti, Aarav Shah, Satwik Kumar Chittiprolu, Tony Song, Sunishchal Dev, Kevin Zhu, and Maheep Chaudhary. 2025. Paladin: Self-correcting language model agents to cure tool-failure cases. *arXiv preprint arXiv:2509.25238*.

Rui Wang, Ce Zhang, Jun-Yu Ma, Jianshu Zhang, Hongru Wang, Yi Chen, Boyang Xue, Tianqing Fang, Zhisong Zhang, Hongming Zhang, Haitao Mi, Dong Yu, and Kam-Fai Wong. 2025. Explore to evolve: Scaling evolved aggregation logic via proactive online exploration for deep research agents. *Preprint*, arXiv:2510.14438.

Kaishuai Xu, Tiezheng Yu, Wenjun Hou, Yi Cheng, Chak Tou Leong, Liangyou Li, Xin Jiang, Lifeng Shang, Qun Liu, and Wenjie Li. 2025. Subtle errors in reasoning: Preference learning via error-injected self-editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31184–31203.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Quoc Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, and 1 others. 2025a. xlam: A family of large action models to empower ai agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11583–11597.

Kangning Zhang, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weiwen Liu, Weinan Zhang, and Yong Yu. 2025b. Looptool: Closing the data-training loop for robust llm tool calls. *arXiv preprint arXiv:2511.09148*.

A Prompt Template for the Error Simulator

To improve reproducibility, we provide the prompting template used to query the error simulator S_ϕ. We use a two-message chat format: a system prompt that specifies the simulator role and

output constraints, followed by a user prompt that injects the original context, ground-truth tool calls, and the model's failed attempt.

B Representative Training Curves

To complement the main results, we provide representative training reward curves for GRPO and FISSION-GRPO with Qwen3-8B in Figure 5. Although step-level rewards are noisy, the smoothed trajectories exhibit similar macro-level dynamics: both methods improve rapidly during early training and remain stable thereafter, with no evidence of reward collapse or divergence. These curves further suggest that the gains of FISSION-GRPO are achieved without introducing optimization instability.

C Training Algorithm Details

Algorithm 1 outlines the detailed execution flow of the FISSION-GRPO framework. The process alternates between standard exploration (to maintain general capability and mine errors) and fission-based updates (to learn specific recovery strategies).

D Extended Case Study Analysis

In this section, we provide a detailed breakdown of the case study referenced in Section 4.7. Figure 7 visualizes the trajectories of Qwen3-8B under three training conditions on a multi-turn file manipulation task (Sample ID: `multi_turn_base_1`).

Scenario Overview. The user requests to verify the current directory, move a `log.txt` file into a new `archive` folder, and then search for a keyword within that file. The key challenge arises in Turn 2: `mkdir archive` fails (directory already exists), but `cd workspace` and `mv log.txt` succeed. This partial-success state requires careful tracking—in Turn 3, since the file was moved to `archive`, a direct `grep` will fail, requiring the agent to locate the file first.

Figure 5: Representative training reward curves for GRPO and FISSION-GRPO. Although step-level rewards are noisy, the smoothed trajectories remain stable throughout training, with rapid early improvement and no evidence of collapse or divergence.

Qwen3-8B (Base) — State Awareness Collapse. The Base model correctly issues the initial batch command `[cd, mkdir, mv]`. However, it fails to update its internal state to reflect that it is already inside `workspace` after the successful `cd`. When attempting to handle the `mkdir` error, it redundantly retries `cd workspace`, which fails ("No such directory" within the current directory). Confused by this feedback, it spirals into a loop of invalid operations, ultimately failing to realize the file was already moved.

Qwen3-8B + GRPO — Latent State Mismatch & Hallucination. The GRPO model succeeds in Turn 2 (the file is moved), but fails to track the consequence—specifically, that `log.txt` is no longer in the current directory but in the `archive` subdirectory. This latent state mismatch surfaces in Turn 3: it first tries `grep("log.txt")` (fails), then attempts a heuristic guess `grep("archive/log.txt")` (also fails). Lacking a grounded fallback strategy, it resorts to hallucination, inventing a non-existent `path` parameter for `ls`.

Qwen3-8B + FISSION-GRPO — Active Diagnosis. Our model handles the Turn 2 state transition correctly. More importantly, in Turn 3, when faced with the same "No such file" error, it demonstrates a superior recovery mechanism: instead of guessing, it deploys `find(name="log.txt", path="workspace")` to empirically verify the file's location. Using the confirmed path, it performs a precise state update via `cd(folder="archive")`, then executes `grep` successfully. This confirms that FISSION-GRPO learns to bridge state gaps through active diagnosis rather than relying on fragile internal memory or hallucinated corrections.

Algorithm 1 Detailed Training Procedure of FISSION-GRPO

Require: Policy π_θ, Reference Policy π_{ref}, Error Simulator $\mathcal{S}_\phi$
Require: Training dataset $\mathcal{D}$
Require: Hyperparameters: Learning rate η, KL coefficient β, Clip ratio ϵ
Require: Group sizes: G (Exploration), G' (Fission/Correction)
Require: Thresholds: Buffer trigger B_{trig}, Success score $R_{\text{thresh}} = 1.0$
Initialize Corrective Sample Pool $\mathcal{B} \leftarrow \emptyset$ ▷ Implemented as LIFO Stack
for iteration $k = 1, \dots, K$ **do**
  // STAGE 1: STANDARD EXPLORATION & MINING
  Sample batch of user queries $x \sim \mathcal{D}$
  Generate exploration group $\{\tau_i\}_{i=1}^{G} \sim \pi_\theta(\cdot|x)$
  Compute rewards for each trajectory: $r_i \leftarrow R(\tau_i, t)$ ▷ Eq. 2, simplified in pseudocode
  Compute GRPO Advantages:
    $\mu_R \leftarrow \frac{1}{G}\sum r_i, \quad \sigma_R \leftarrow \text{Std}(r_i)$
    $\hat{A}_i \leftarrow \frac{r_i - \mu_R}{\sigma_R + \epsilon}$
  Update Policy (Standard):
    $\mathcal{L}_{\text{GRPO}} \leftarrow \frac{1}{G}\sum_{i=1}^{G}\left[\min(\rho_i \hat{A}_i, \text{clip}(\rho_i, 1 \pm \epsilon)\hat{A}_i) - \beta \mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right]$
    $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{GRPO}}$
  // STAGE 2: SYNTHESIS & ACCUMULATION
  Identify error set $\mathcal{E} = \{\tau_i \mid R_{\text{corr}}(\tau_i) < R_{\text{thresh}} \vee R_{\text{fmt}}(\tau_i) = 0\}$
  for each error trajectory $\tau_{\text{err}} \in \mathcal{E}$ **do**
    if $R_{\text{fmt}}(\tau_{\text{err}}) == 0$ **then**
      $f \leftarrow \text{GetFormatError}(\tau_{\text{err}})$
    else
      $f \leftarrow \mathcal{S}_\phi(x, \tau_{\text{err}}, \tau_{\text{gt}})$ ▷ Generate diagnostic feedback
    end if
    Construct corrective context $x_{\text{corr}} \leftarrow [x; \tau_{\text{err}}; f]$
    Compute Deduplication Key $k \leftarrow \text{Hash}(x, \tau_{\text{err}})$
    if $k \notin \text{Keys}(\mathcal{B})$ **then**
      $\text{Push}(x_{\text{corr}}) \rightarrow \mathcal{B}$ ▷ LIFO Push
    end if
  end for
  // STAGE 3: FISSION-BASED REMEDIAL UPDATE
  if $|\mathcal{B}| \geq B_{\text{trig}}$ **then**
    $X_{\text{batch}} \leftarrow \text{Pop}(B_{\text{trig}})$ items from top of $\mathcal{B}$ ▷ LIFO: Fetch freshest errors
    Initialize batch loss $\mathcal{L}_{\text{total}} \leftarrow 0$
    for each corrective context $x_{\text{corr}} \in X_{\text{batch}}$ **do**
      Fission Resampling:
        Generate recovery group $\{\tau'_j\}_{j=1}^{G'} \sim \pi_\theta(\cdot|x_{\text{corr}})$
        Compute rewards $\{r'_j\}$ for recovery attempts
      Compute Corrective Advantages:
        $\mu'_R \leftarrow \frac{1}{G'}\sum r'_j, \quad \sigma'_R \leftarrow \text{Std}(r'_j)$
        $\hat{A}'_j \leftarrow \frac{r'_j - \mu'_R}{\sigma'_R + \epsilon}$ ▷ Variance restored via Fission
      Accumulate Gradients:
        $\mathcal{L}_{\text{corr}} \leftarrow \frac{1}{G'}\sum_{j=1}^{G'}\left[\min(\rho'_j \hat{A}'_j, \dots) - \beta \mathbb{D}_{\text{KL}}\right]$
        $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{corr}}$
    end for
    $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{total}}$ ▷ Apply corrective update
  end if
end for

Prompt Templates

Prompt 1: System prompt for querying the error simulator.

```
You are a Runtime Environment Simulator for an AI Agent.
Your role is to act as the API Server or Operating System that executes tool calls.

IMPORTANT CONTEXT:
You are receiving a tool call from an Agent that has ALREADY FAILED validation or logic checks against the Ground Truth.
Your task is NOT to judge correctness. Your task is to generate the specific ERROR MESSAGE that the system would return
      to the Agent.

GOAL:
Generate a short, realistic, and actionable error message (starting with "ERROR: ")
that will help the Agent understand why its call failed compared to the expected Ground Truth.

PRIORITY ERROR CATEGORIES:
- Dependency & Sequence Violations
- Parameter Hallucination
- Schema & Parameter Errors
- Business Logic Errors

EVALUATION LOGIC:
- Reference the Ground Truth: ground_truth is usually correct and serves as the primary standard.
- Verify Context: cross-check the Agent output against the User Request in the Original Context.
- Ambiguity Rule: if the Agent output differs from ground_truth but is still plausible, note missing validation.

OUTPUT RULES:
- Start with "ERROR: " (case-sensitive)
- Be specific: mention actual parameter names/values from the failed attempt
- Sound like a system/API response
- Keep it concise (1--2 sentences)
- Return ONLY the error string (no JSON, no markdown, no extra explanation)

ERROR MESSAGE EXAMPLES (Real error style):
<<ERROR_EXAMPLES_SNIPPET>>

CRITICAL REMINDERS:
- Do NOT output JSON like {"error": "..."}; output plain text only
- Do NOT add any preamble; only the "ERROR: ..." line
- Do NOT fabricate placeholders unless they appear in the failed attempt
- The error should be what the runtime system returns, not an analysis
```

Prompt 2: User prompt template (simulation input).

```
## Simulation Task
The Agent attempted to execute a tool call, but it was INCORRECT compared to the Ground Truth.
Generate the system error message triggered by the Agent's specific mistake.

1) Original Context (what the Agent saw)

[System instructions & tools]
<<SYSTEM_AND_TOOLS>>

[Dialogue history before this attempt (non-system turns)]
<<DIALOGUE_HISTORY>>

2) Execution Comparison

[Ground-truth tool call(s)]
<<GROUND_TRUTH_TOOL_CALLS>>

[Failed tool call(s) extracted from the model output]
<<FAILED_TOOL_CALLS>>

3) Instruction

Compare the failed attempt against the ground truth under the given context.
Identify the first critical failure and generate the runtime error.

Output:
Return ONLY one error string starting with "ERROR:".
```

Figure 6: Two-message prompting format used to query the error simulator S_ϕ.

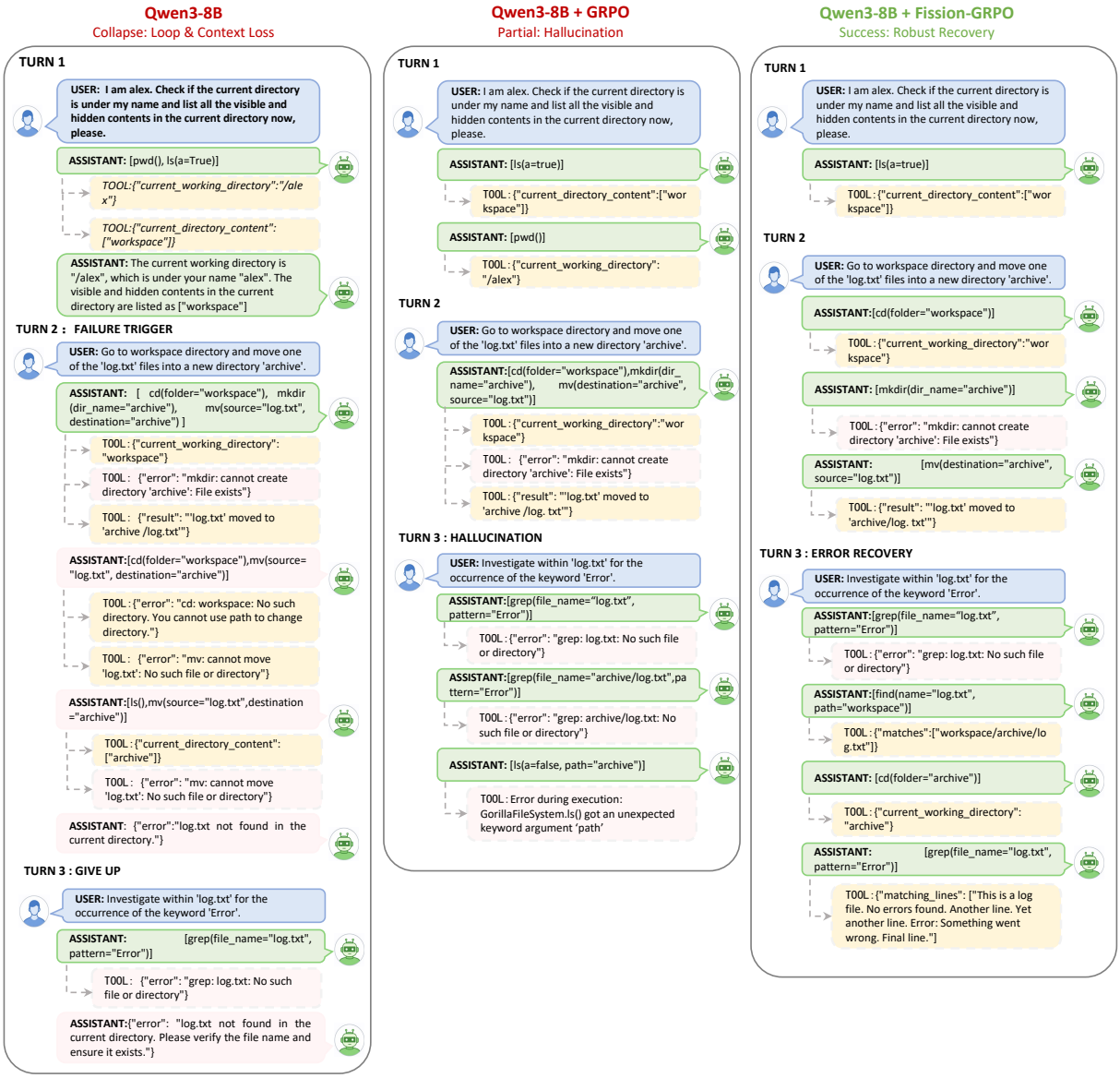


Figure 7: Detailed visualization of Multi-turn Error Recovery. Comparisons of trajectories generated by Qwen3-8B under different training regimes. The Base model collapses due to immediate state loss; the GRPO model suffers from latent state mismatch leading to hallucination in later turns; FISSION-GRPO overcomes this by employing diagnostic tools (`find`) to actively resolve state uncertainties.