Procedure-Aware Reinforcement Learning for Tool-Augmented Large Language Models

Rewarding *how* tools are called, not only *what* is called

Qinglong Zheng* **Jiajun Fan*** **Chaoran Cheng** **Ge Liu**

University of Illinois Urbana-Champaign

COLM 2026

*Equal contribution

Tool-use training: from SFT to RL with verifiable rewards

- **SFT** on large synthetic trajectories: strong baseline, but costly labels and weak generalization to unseen tools.
- **RLVR** (e.g. ToolRL): optimize verifiable rewards directly, and surpass SFT.
- Current rewards match **tool name, parameter keys, parameter values** with greedy matching between predicted and reference calls.

> **The blind spot:** the matching ignores execution order. Tool use is scored as a **bag of actions**.

Why order matters

In stateful environments, the sequence of calls decides the resulting system state.

Persistent multi-turn failures:

- disordered execution sequences
- parameter mismatches that break API calls
- reasoning and execution that disagree

Hypothesis. Enforcing sequential validity at the turn level is a prerequisite for robust multi-turn task completion.

Same two calls, same correctness reward

Reference: `cd` → `touch`

	R_{trace}	R_{state}	R_{proc}
`touch` → `cd`	0.41	0	**0.21**
`cd` → `touch`	1.00	1	**1.00**

Illustrative example; scores computed with the paper's Eq. 7–9.

Declarative vs. procedural learning

Declarative

Fit the distribution of *what* correct tool calls look like.

Outcome reward: one coarse signal per episode, little guidance for credit assignment.

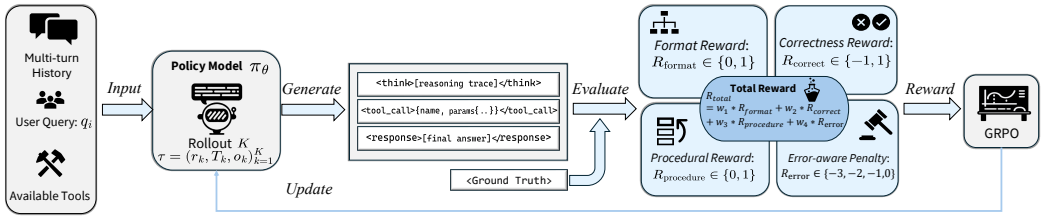
Procedural

Learn *how* to reach the goal in the correct sequence.

Process reward: evaluate intermediate steps, here with rules instead of human step labels.

Four pillars we reward: **decision accuracy**, **selection logic**, **sequential validity**, **component precision**.

Framework overview



- Policy samples a group of rollouts in a structured XML format: `think`, `tool_call`, `response`.
- Each rollout is scored against the reference tool calls by four reward components.
- GRPO with KL regularization updates the policy.

Paper Fig. 1. Reward ranges in the figure are abbreviated.

Reward decomposition

$$R_{\text{total}} = R_{\text{format}} + 3\,R_{\text{correct}} + 2\,R_{\text{proc}} + R_{\text{error}} \qquad R_{\text{total}} \in [-6, 6]$$

Format $\{0, 1\}$
Required XML tags present and in order, strict tag boundaries.

Correctness $[-1, 1]$
Tool name, parameter keys, parameter values. Order-agnostic.

Procedural (new) $[0, 1]$
Is the invocation order right, and does it end in the right state?

Error-aware penalty (new)
$\{-3, \ldots, 0\}$
Structural errors: tools, parameter schema, types.

Procedural reward, part 1: trace via NDCG

Reference tool sequence G, predicted sequence P (tool names only, so ordering is separated from parameter correctness).

$$R_{\text{proc}} = 0.5\, R_{\text{trace}} + 0.5\, R_{\text{state}}$$

$$\text{rel}(i) = \frac{1}{1 + |i - j^*|}, \qquad R_{\text{trace}} = \frac{\text{DCG@}p}{\text{IDCG@}p}, \qquad \text{DCG@}p = \sum_{i=1}^{p} \frac{2^{\text{rel}(i)} - 1}{\log_2(i+1)}$$

- j^*: closest matching position in G for the tool predicted at position i; $p = |G|$.
- Graded credit for partial orderings: optimal, suboptimal and invalid paths get different scores.
- Position-sensitive: early mistakes cost more.

Procedural reward, part 2: state via suffix matching

$$R_{\text{state}} = \mathbb{I}\big[\text{suffix}(G, P)\big] \cdot \mathbb{I}\big[\text{unique}(G, P)\big] \in \{0, 1\}$$

- Strict trajectory matching punishes valid detours, redundant but reasonable calls, and recovery steps.
- R_{state} only asks: does the prediction **end** with the reference sequence, with no earlier occurrence?
- Exploratory paths that converge to the correct execution still get credit.

Example with $G = [\texttt{cd}, \texttt{touch}]$: $P = [\texttt{ls}, \texttt{cd}, \texttt{touch}] \Rightarrow R_{\text{state}} = 1$
$P = [\texttt{touch}, \texttt{cd}] \Rightarrow R_{\text{state}} = 0$

R_{state} is a sequence-level proxy on tool names; it does not query the environment state.

Error-aware penalty and optimization

Error-aware penalty

$$R_{\text{error}} = P_{\text{func}} + P_{\text{param}} + P_{\text{type}}$$

Each term is -1 if triggered, else 0:

- P_{func}: hallucinated or missing tools
- P_{param}: parameter-schema violations
- P_{type}: type mismatches (string vs. integer)

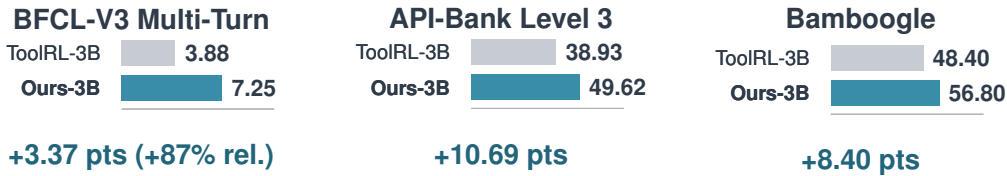
Makes rollouts within a group easier to tell apart.

GRPO with KL regularization

$$A_i = \frac{R_i - \mu_Q}{\sigma_Q + \epsilon}$$

Qwen2.5-Instruct 1.5B and 3B
ToolRL 4k data (3,920 samples)
8 rollouts per prompt, 15 epochs
KL coefficient $\beta = 0.001$
2× NVIDIA A6000

Results: Ours-3B vs. ToolRL-3B



- Largest gains where sequencing matters: multi-turn and sequential tool coordination.
- Bamboogle is outside the training distribution: the benefit transfers to multi-hop search.

Paper Tables 1–3. BFCL: mean over runs (std 0.88 / 0.75). Bamboogle: std 2.45 / 2.19 over 3 search runs.

BFCL-V3 in context

Model	Non-Live	Live	Multi-Turn	Avg. Acc
Ours, Qwen2.5-3B	**83.29**	**69.13**	**7.25**	**65.67**
ToolRL, Qwen2.5-3B	79.52	67.88	3.88	63.07
Raw Qwen2.5-3B	75.52	62.18	1.00	57.31
Ours, Qwen2.5-1.5B	76.48	63.27	3.50	58.48
ToolRL, Qwen2.5-1.5B	78.40	67.06	1.25	57.08
Gemma-3-12b-it	79.44	74.24	5.75	64.69
GPT-4.1-mini	84.60	74.76	2.50	64.65
Gemini-2.5-Flash-Lite	83.90	54.85	7.63	57.94
GPT-5-mini	68.04	62.55	5.50	57.11
Claude-Haiku-4.5	55.42	52.48	1.75	47.26

Paper Table 1. Avg. Acc: mean of five sub-categories. Larger-scale baselines: official BFCL leaderboard scores, each under its own protocol (context, not a matched comparison).

Case study: BFCL-V3 multi-turn file system

ToolRL-3B **fails**

“There is no Documents dir, so we need to create one first.”

`mkdir(Documents)`, `touch(summary.txt)` **ERR** directory already exists; file lands in root

`touch(summary.txt)` **ERR** file already exists

Gives up; later turns fail on state mismatch.

Ours-3B **succeeds**

“We need to first navigate to the Documents directory and then create the file.”

`cd(Documents)`, `touch(summary.txt)` **OK**

`echo("quantum computing", ...)` **OK**

`wc(summary.txt, mode="w")` **OK** count = 2

Paper Table 4.

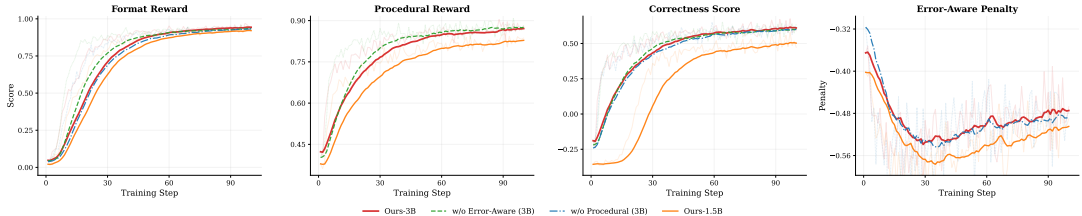
Ablation: the two new signals are complementary

Variant (Qwen2.5-3B)	Non-Live	Live	Multi-Turn	Avg. Acc
Full model	**83.88**	**70.62**	**7.25**	**53.92**
w/o error-aware penalty	81.55	67.69	5.50	51.58
w/o procedural reward	82.27	68.18	4.50	51.65
ToolRL-3B	80.45	67.87	3.88	50.73

- **Procedural reward** teaches *when and in what order*: removing it gives the largest Multi-Turn drop ($7.25 \rightarrow 4.50$).
- **Error-aware penalty** teaches *how to call correctly*: removing it hurts Non-Live and Live most.
- All deltas significant under a paired bootstrap test ($p < 10^{-7}$).

Paper Table 5 (same-checkpoint protocol; Avg. Acc here is the mean of three sub-categories).

Training dynamics



- Format compliance saturates near 1.0 for every configuration.
- Procedural reward converges steadily: about 0.88 for 3B and 0.83 for 1.5B.
- The error penalty deepens during early exploration, then recovers; 3B recovers faster.

Paper Fig. 4, first 100 GRPO steps.

Does it hold beyond 3B, and beyond RL-vs-RL?

Scaling to 7B (BFCL-V3)

	ToolRL-7B	Ours-7B
Multi-Turn	15.13	**17.29**
Live	74.96	**75.91**
Irrelevance det.	76.29	**79.29**
Avg. Acc	67.09	**68.30**

The gain is not just compensation for a weak small model.

SFT vs. RL on identical 4k data (3B)

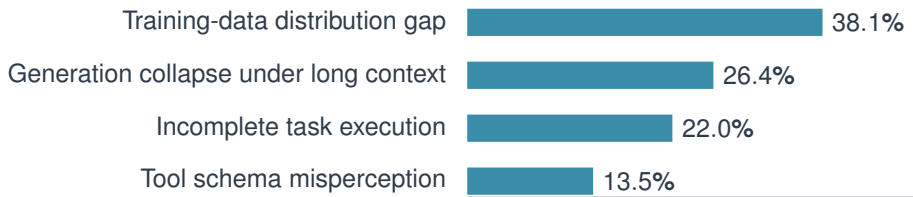
	Multi-Turn	Avg. Acc
SFT	1.92	55.31
Correctness RL (ToolRL)	3.88	63.07
Procedure-aware RL	**7.25**	**65.67**

Same annotations for all three: the gain comes from the reward design.

Paper App. A.3 (Table 8), A.4 (Table 9). Also ahead of a matched ToolZero reproduction (App. A.6).

What still fails: multi-turn error analysis

851 error instances over failed BFCL-V3 multi-turn cases (Ours-3B):



- 94.2% of training samples are single-turn; none cover interactive file-system operations.
- Absolute Multi-Turn accuracy is low for everyone (GPT-4.1-mini 2.50, GPT-5-mini 5.50).

Paper Sec. 4.5, App. A.2.

Takeaways

> **For sequential decision-making, the *how* of execution is as critical as the *what*.**

- Correctness-only RL rewards are order-agnostic; we add a rule-based **procedural reward** (NDCG trace + suffix state) and an **error-aware penalty**.
- Consistent gains over ToolRL on BFCL-V3, API-Bank and Bamboogle, with the largest on multi-turn (3.88 $\rightarrow$ 7.25).
- Both signals matter, and the benefit persists at 7B.
- **Limitations:** needs reference tool sequences; multi-turn accuracy remains low in absolute terms; main study at 1.5B to 7B.

Paper: `openreview.net/forum?id=d4wrBuJ4xo` qzheng19@illinois.edu jiajunf3@illinois.edu