ToolVerse: Unlocking Massive Environments and Long Horizons Tasks for Agentic Reinforcement Learning

Anonymous Authors[1]

Abstract

While LLM agents demonstrate strong reasoning abilities in compact and well-defined scenarios, they struggle to maintain robustness and effectiveness when faced with large-scale, diverse, and dynamic real-world environments that demand seamless tool integration. To address this gap, we introduce **ToolVerse**, a comprehensive framework that scales up agentic RL environments and enables agents to perform complex long-horizon reasoning in Tool-Integrated Reasoning (TIR) tasks. First, ToolVerse automatically builds the massive executable agent training environments from nearly 400 real-world Model Context Protocols (MCPs) that contain about 4500 tools. Second, we propose a task design strategy based on a tool dependency graph, utilizing Dynamic Unlocking Sampling Algorithm to generate long-horizon tasks, and produce **GUST** (**G**raph **U**nlocking **S**ampling **T**asks) dataset. Third, to alleviate the credit assigment problem in long-horizon agentic RL, we propose a fine-grained Turn-Aware Relative Advantage algorithm. We conduct extensive Agentic RL training using ToolVerse and evaluate our framework on serveral agentic benchmarks. Experimental results demonstrate that our framework significantly strengthens LLMs' capabilities in long-horizon tool use, achieving a marked performance boost and showcasing robust reasoning within dynamic environments.

1. Introduction

Large Language Models (LLMs) have demonstrated considerable promise as autonomous agents, capable of interacting with tools and environments to accomplish complex tasks (Wu et al., 2025; Ye et al., 2025; Prabhakar et al., 2025). When combined with Agentic Reinforcement Learning, LLMs can develop the ability to reason over long horizons and make sequential decisions (Shang et al., 2025; Qian et al., 2025; Dong et al., 2025; Li et al., 2025a; Cao et al., 2025). By integrating tool usage with reasoning and adapting actions based on intermediate observations, LLMs can plan over extended time frames, positioning them as powerful candidates for autonomous agent development in dynamic, real-world environments.

However, as shown in Figure 1, developing agentic systems presents three primary challenges. First, existing Agent RL environments are typically limited to interactions with a single tool or a small set of tools, such as search engines or code interpreters (Mai et al., 2025; Li et al., 2025c; Qian et al., 2025; Xue et al., 2025). While effective within narrow domains, these environments lack the complexity required for long-horizon, multi-turn tool integration. Second, designing tasks that involve multi-turn, long-horizon reasoning across multiple integrated tools is inherently difficult (Ye et al., 2025). These tasks demand that each action be informed by prior steps, requiring sophisticated planning and reasoning over extended sequences of interactions. Third, precise credit assignment is difficult in multi-turn agentic RL. Sparse terminal rewards fail to provide meaningful advantage estimates for individual actions within a long-horizon trajectory. This lack of granular feedback leads to high variance in policy gradients, resulting in poor sample efficiency or even training instability.

In this paper, we introduce **ToolVerse**, a comprehensive framework designed to address the limitations of current Agentic RL training environments and to support more complex agent training. **(1) Scaling Executable Agentic RL Environments**: To overcome the lack of diverse, general-purpose environments, we scale the training landscape by automating the generation of executable MCP tools. This expands the range of tools available for interaction, facilitating training in more realistic, complex scenarios. **(2) Graph-Guided Long-Horizon Task Composition**: To address the challenge of task complexity, we propose a task design strategy based on a tool dependency graph and a dynamic unlocking sampling algorithm, producing **GUST**

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. We posit the scalability of **Agentic RL** is primarily limited by three dimensions: **Scope** (environment diversity), **Tool Use Complexity** (reasoning depth), and **Credit Assignment** (training signal granularity). ToolVerse bridges the gap across these aspects.

Method	Tool Scope (Env Diversity)	Tool Use Complexity	Credit Assignment
ZeroTIR	Single tool (Code)	Math Reasoning	Outcome (Sparse)
SimpleTIR	Single tool (Code)	Math Reasoning	Outcome (Sparse)
AGENTFLOW	Search + Code	QA/Agentic/Math Reasoning	Outcome (Sparse)
ToolRL	Three Benchmarks' tools	Tool use Reasoning	Outcome (Dense)
SALT	Three Benchmarks' tools	Tool use Reasoning	Outcome (Sparse)
FTRL	Many General tools	QA/Tool Use Reasoning	Outcome (Sparse)
ToolVerse (Ours)	Many General tools	**Long-horizon Tool Reasoning**	**Turn-Aware (Dense)**

dataset. This approach generates task subsets that require integrated reasoning across multiple steps, with outputs from earlier tasks feeding into subsequent ones, thereby enabling agents to develop sophisticated tool use capabilities over extended interaction sequences. **(3) Turn-Aware Relative Advantage Estimation**: To enhance credit assignment during trajectory evaluation, we introduce turn-aware reward and advantage estimation. This allows the model to learn from training with more precise feedback, enabling it to develop a deeper understanding from the task performance at each step.

Our contributions can be summarized as follows:

- We design an automated pipeline to extend the executable MCP environments and significantly scale up the diversity and robustness of agentic RL training scenarios. This pipeline transitions agentic RL from narrow, constrained domains to large-scale, real-world tool-integrated reasoning scenarios.
- We propose a novel task generation paradigm leveraging tool dependency graphs and **Dynamic Unlocking Sampling Algorithm** to design coherent, long-horizon, and multi-turn reasoning tasks. This yields our **GUST** dataset, a high-quality resource for Agentic RL training that significantly bolsters LLMs' tool-integrated reasoning capabilities.
- We introduce a fine-grained Turn-Aware Relative Advantage estimation method to mitigate sparse rewards and high variance in agentic RL, enabling precise credit assignment across long-horizon trajectories. Extensive experiments across several agentic benchmarks demonstrate that our framework substantially enhances training stability and strengthens the tool-integrated reasoning performance of LLMs.

2. Relate Work

2.1. Tool-integrated reasoning in LLM Agents

Recent advancements in Agentic RL have highlighted the significance of integrating tool usage within large language models (LLMs) to enhance their ability to perform complex, multi-step reasoning. For instance, TORL (Li et al., 2025a) and rStar2-Agent (Shang et al., 2025) leverage reinforcement learning to train LLMs in utilizing code interpreters, treating code execution as a reliable feedback mechanism. Similarly, ZeroTIR (Mai et al., 2025) investigates scaling laws for agentic RL, revealing that tool-use abilities can emerge through outcome-based rewards, with performance linked to the computational resources available. To maintain stability in multi-turn training, SimpleTIR (Xue et al., 2025) filters out "void" trajectories—sequences that lack meaningful tool use or outputs. Additionally, multi-tool collaboration has been explored by works like ToolStar(Dong et al., 2025) , which enable agents to coordinate between search engines and code interpreters. ToolRL (Qian et al., 2025) focuses on reward design to optimize selection strategies across multiple tools. However, these efforts are predominantly constrained to narrow toolsets, such as code interpreters and search engines, and typically involve fixed interaction patterns. This limitation hinders the development of generalized capabilities for interacting with diverse, structured APIs in real-world environments.

In contrast, ToolVerse significantly expands the scope of tool integration by offering a scalable, dynamic framework that supports the integration of a wide variety of tools (nearly 4500) within real-world, complex training environments. Our framework's ability to handle long-horizon reasoning and multi-turn task design, combined with dynamic tool usage, sets it apart from previous efforts that are limited by static toolsets and simpler problem domains.

2.2. Training environment for Agentic RL

The progress of Agentic RL is fundamentally dependent on the fidelity and scalability of its training environments. Several studies (Li et al., 2025b; Castellani et al., 2025; Xiang et al., 2023) utilize LLMs' reasoning abilities and background knowledge to emulate environments. While this approach eliminates the necessity of constructing actual environments, it often suffers from issues such as hallucina-

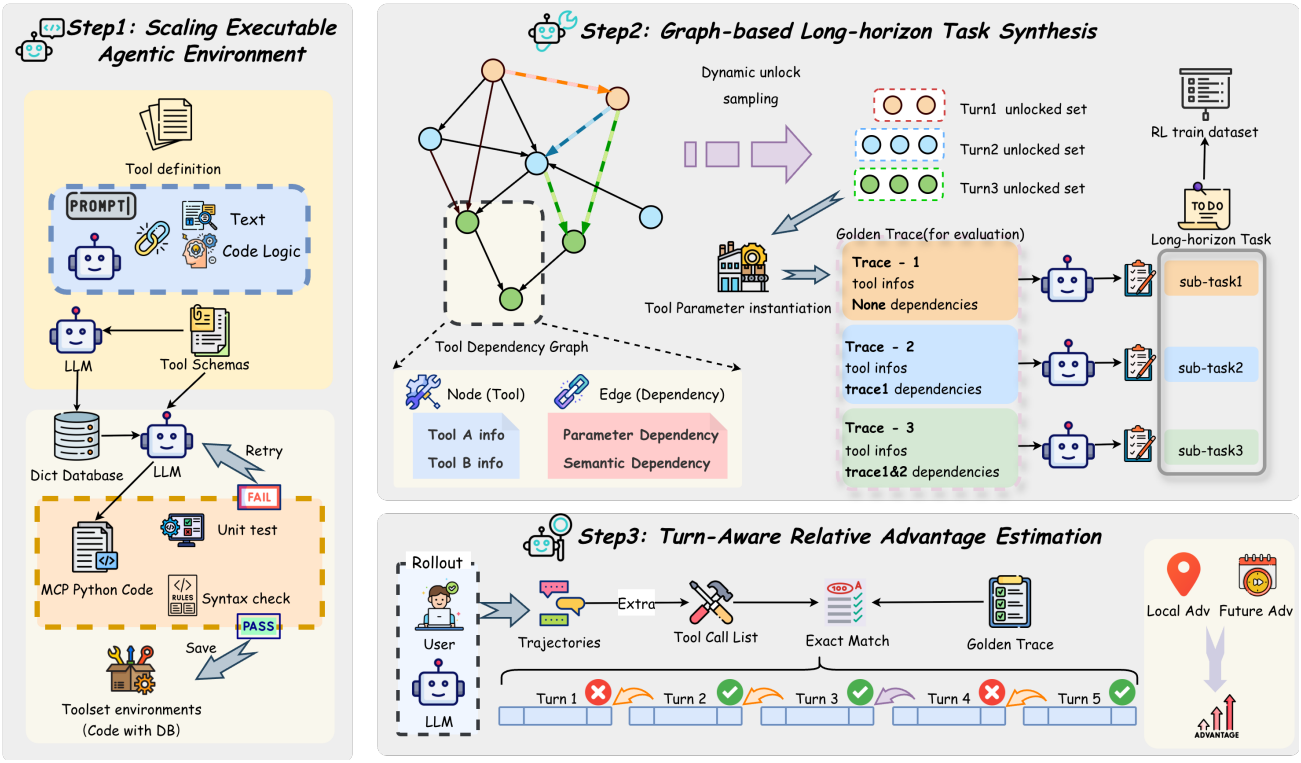

Figure 1. Overview of the **ToolVerse** framework. ***Step1: Scaling Executable Agentic Environment***: We automate the conversion of raw tool definitions into executable MCP environments. ***Step2: Graph-based Long-horizon Task Synthesis***: A tool dependency graph is constructed to model logical data flows. We employ a dynamic unlocking sampling algorithm to synthesize long-horizon tasks. ***Step3: Turn-Aware Relative Advantage Estimation***: We propose turn-aware advantage algorithms based on the characteristics of the task.

tions, inconsistency, limited transparency, and inadequate management of persistent states. Recent methods such as TaskCraft (Shi et al., 2025) have automated the generation of scalable tasks to address data scarcity, while offline-database methods (Fang et al., 2025; Ye et al., 2025) utilize static captures of real-world data to preserve simulation fidelity.

Nevertheless, a critical gap persists: a framework capable of enabling scalable, executable, and dynamic tool-integrated training environments while maintaining the fidelity of ground-truth feedback for RL training. ToolVerse fills this gap by automating the generation of executable MCP environments, facilitating the creation of diverse, real-world training scenarios that are both scalable and grounded in verifiable feedback. This dynamic environment supports more realistic agentic RL training, avoiding the bottlenecks of previous static or simulation-based environments.

3. Method

In this section, we delineate the **ToolVerse** framework. We first introduce the automated synthesis pipeline for scaling executable environments (Sec. 3.1). We then formalize the graph-guided generation mechanism for long-horizon task curricula (Sec. 3.2). Finally, we use a Turn-Aware Relative Advantage Estimation algorithm (Sec. 3.3) tailored for long-horizon reasoning, addressing the credit assignment limitations in standard group-based reinforcement learning.

3.1. Scaling Executable Agentic Environments

To construct agentic environments across diverse domains, we propose an automated closed-loop synthesis pipeline. Starting with a repository of raw toolsets $\mathcal{S}^{\text{raw}} = \{S_1, \ldots, S_N\}$, where each S_i targets a specific functional domain, our objective is to transform these static definitions into robust executable environments.

First, we perform *Schema Refactoring* to refine each raw schema S_i into a structured form $\hat{S}_i$. This step aligns function signatures with intended functionality and eliminates textual noise, simplifying logic to facilitate LLM-based Python code generation. Next, we construct a domain-specific dictionary database $\mathcal{D}_i$ for each tool. This database models essential state variables (e.g., inventory, profiles) to enable stateful environmental interactions.

Given $\hat{S}_i$ and $\mathcal{D}_i$, we generate executable functions $\mathcal{F}_i = \{f_1, \ldots, f_m\}$ using the MCP tools library. These functions implement the logic to query or update $\mathcal{D}_i$, establishing a dynamic, operable environment.

Algorithm 1 Dynamic Unlocking Sampling

Input: Tool Dependency Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Batch Size N, Multi-turn Trajectory $\mathcal{T}$
Initialize:
Compute in-degrees: $D[v] \leftarrow \text{InDegree}(v), \forall v \in \mathcal{V}$
Initialize ready queue: $\mathcal{Q} \leftarrow \{v \in \mathcal{V} \mid D[v] = 0\}$
Initialize trajectory: $\mathcal{T} \leftarrow \emptyset$
{*Start Dynamic Sampling Loop*}
while $\mathcal{Q} \neq \emptyset$ **do**
  $k \leftarrow \min(|\mathcal{Q}|, N)$
  $\mathcal{S}_t \leftarrow \text{Sample}(\mathcal{Q}, k)$ {Select next sub-tasks}
  $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \mathcal{S}_t$
  Append $\mathcal{S}_t$ to $\mathcal{T}$
  {*Unlock Dependencies for each selected task*}
  for each node u in $\mathcal{S}_t$ **do**
    for each successor v of u in $\mathcal{G}$ **do**
      $D[v] \leftarrow D[v] - 1$
      if $D[v] = 0$ **then**
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{v\}$ {Add v to ready queue}
      end if
    end for
  end for
end while

Finally, to ensure robustness, we automatically generate unit tests $\mathcal{T}_i$. We retain only toolsets that pass all syntax and unit tests ($\mathbb{V}(\mathcal{F}_i, \mathcal{T}_i) = \text{True}$). These environments are integrated into the Verl framework (Sheng et al., 2024) to provide a diverse, high-fidelity action space for RL training.

3.2. Graph-based Long-horizon Task Synthesis

To facilitate long-horizon reasoning, we move beyond random walk-based task generation and propose a structured, graph-theoretic approach to task synthesis.

3.2.1. Tool Dependency Graph Construction

We define a tool dependency graph (TDG) for each scenario $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes represent tools $\mathcal{V}$ and edges $\mathcal{E}$ capture the dependencies between them. The dependency structure is inferred with the help of an LLM based on two key principles: (1) an edge is established from tool T_A to tool T_B if the output of T_A is required as an input for T_B; (2) an edge is drawn from T_A to B if T_B can only be invoked after T_A in the logical sequence of the scenario. The resulting graph, $\mathcal{G}$, serves as the foundational structure for reasoning, where paths represent valid sequences of tool invocations with contextual dependencies.

3.2.2. Dynamic Unlocking Sampling

Causal Grounding via Topological Constraints. To generate multi-turn trajectories with logical consistency, we introduce the Dynamic Unlocking Sampling (DUS) algorithm. The algorithm maintains a ready queue $\mathcal{Q} \subseteq \mathcal{V}$, consisting of tools with zero in-degree, ensuring that tasks with high dependency are "locked" until their prerequisites are completed. This topological constraint prevents causal errors in reasoning chains.

Iterative Unlocking Mechanism. The process starts with a pool of root nodes $\mathcal{Q} = \{v \in \mathcal{V} \mid d_{\text{in}}(v) = 0\}$, typically comprising basic operations. At each step t, we sample a subset $S_t \subseteq \mathcal{Q}$ to form a trajectory stage. The batch size is determined by $k = \min(|\mathcal{Q}|, N)$. After executing S_t, the algorithm updates the in-degree for all successor nodes: $d_{\text{in}}(v) \leftarrow d_{\text{in}}(v) - 1$. A node v is then added to $\mathcal{Q}$ when $d_{\text{in}}(v) = 0$, signaling that all dependencies are satisfied.

Emergence of Long-Horizon Logic. This topological progression induces a curriculum of increasing task complexity within the final trajectory $T = [S_1, S_2, \ldots, S_m]$. Simple queries necessarily precede more complex operations, leading to the natural emergence of complex reasoning at later stages. The resulting trajectories exhibit a coherent cognitive flow, providing high-fidelity training signals for long-horizon LLM agents.

3.2.3. Inverse Context Reconstruction

In the previous step, the tool invocation sequence was sampled. In this step, we follow the tool dependency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to instantiate these tools within the environment. Specifically, for each tool $T \in \mathcal{V}$, we retrieve the appropriate parameters from the database in a forward manner while preserving context dependencies. The input parameters for tools in subsequent rounds are determined by the outputs of earlier tool calls, as dictated by the edges in the tool dependency graph. We leverage large language models to assist in this process. The resulting tool invocation sequence, which correctly completes the task, is referred to as the **Golden Trace**.

3.3. Turn-Aware Relative Advantage Estimation

Standard Group Relative Policy Optimization (GRPO)(Shao et al., 2024) normalizes rewards across the entire trajectory, assigning a single scalar advantage to all tokens in a response. However, in long-horizon tool use, this coarse-grained feedback fails to distinguish between correct intermediate steps and fatal errors in later stages. Leveraging the structured nature of our tasks—where the "Golden Trace" provides ground truth for each turn—we propose a fine-grained Turn-Aware Relative Advantage mechanism.

We decompose the turn-level advantage into two components: *Local* (immediate correctness) and *Future* (downstream impact). For a group of K outputs in a T-turn task, we define the total advantage $A_{i,t}^{\text{total}}$ at turn t for the i-th agent

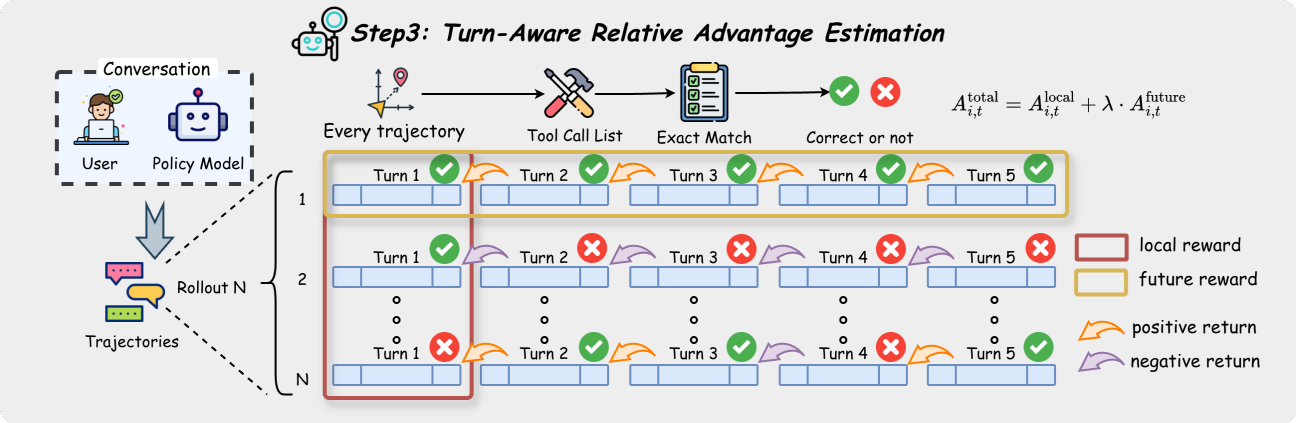

Figure 2. Overview of the **Turn-Aware Relative Advantage Estimation**. For each turn in the multi-turn multi-step tool usage trajectory, we perform rule-based validation to ensure its validity. During the relative advantage computation, we calculate **the normalized advantage for each turn** based on the group distribution at that turn, and then propagate this advantage uniformly to all tokens within the corresponding turn.

as follows.

Binary Reward. Given the optimal action g_t provided by the golden trace for each turn, we define a strict binary reward $r_{i,t}$ for the i-th agent's action $a_{i,t}$:

$$r_{i,t} = \begin{cases} 1.0 & \text{if } a_{i,t} \text{ matches } g_t \\ 0.0 & \text{otherwise} \end{cases} \tag{1}$$

where $\equiv$ denotes functional equivalence.

Local Advantage (A^{local}). The local advantage measures the agent's immediate performance relative to the group at turn t. We compute the mean μ_t^{local} and standard deviation σ_t^{local} of the rewards $r_{1,t}, \ldots, r_{K,t}$ within the group. The standardized local advantage is:

$$A_{i,t}^{\text{local}} = \frac{r_{i,t} - \mu_t^{\text{local}}}{\sigma_t^{\text{local}} + \epsilon} \tag{2}$$

This term incentivizes agents to outperform others; if the agent succeeds ($r_{i,t} = 1$) while others fail, $A_{i,t}^{\text{local}}$ becomes significantly positive.

Gated Future Advantage (A^{future}). To account for long-horizon reasoning, we assess the future value flow. However, in rigid tool environments, a mistake in the current step often invalidates subsequent tasks. To mitigate this, we introduce a Consistency Gate $\delta_{i,t}$ (typically $\delta_{i,t} = r_{i,t}$), ensuring that future rewards are only credited if the current step is valid. The future value $V_{i,t}$ is defined as the discounted sum of future rewards:

$$V_{i,t} = \delta_{i,t} \cdot \sum_{k=0}^{\infty} \gamma^k r_{i,t+k+1} \tag{3}$$

Similar to the local term, we normalize $V_{i,t}$ against the group statistics $(\mu_t^{\text{future}}, \sigma_t^{\text{future}})$ to obtain the relative future advantage $A_{i,t}^{\text{future}}$:

$$A_{i,t}^{\text{future}} = \frac{V_{i,t} - \mu_t^{\text{future}}}{\sigma_t^{\text{future}} + \epsilon} \tag{4}$$

This term encourages the agent to select actions that satisfy the current constraint while enabling future success.

Total Advantage Fusion. The total advantage used for policy updates is a weighted fusion of the local and future components:

$$A_{i,t}^{\text{total}} = A_{i,t}^{\text{local}} + \lambda \cdot A_{i,t}^{\text{future}} \tag{5}$$

where $\lambda \in [0.5, 1.0]$ is a hyperparameter controlling the emphasis on long-horizon consistency. By optimizing this decomposed advantage, our method mitigates the credit assignment problem in multi-turn tool interactions, offering denser and more precise supervision than trajectory-level baselines. We also demonstrated the effectiveness of the method in terms of formulas in Appendix A.

In summary, we decompose the trajectory into distinct turns. For each turn, validity is first confirmed via rigid rule-based checks. We then apply a turn-aware group relative advantage estimation: the advantage for the k-th turn is normalized against the distribution of outcomes at turn k across the generated group. Finally, this turn-specific advantage values are assigned to all token positions belonging to that turn.

4. GUST Dataset

In this section, we detail the construction of the executable environments and the automated pipeline employed to synthesize, filter, and curate the **GUST** dataset.

4.1. Environments Analysis

To build a diverse and realistic foundation for agent training, we sourced a vast collection of JSON tool definitions from both open-source and proprietary repositories, covering a wide range of real-world scenarios. As detailed in Section 3.1, we developed an automated pipeline to convert these static definitions into executable MCP tools. Only toolsets that successfully formed a closed loop—passing all syntax validations and unit tests—were retained. This process resulted in over 400 executable tool environments. To ensure the environments possess appropriate complexity for reasoning training, we strictly filtered for toolsets containing between 5 and 20 tools. Figure 3 illustrates the semantic breadth of the environment.We will refactor any toolkit environments that fail to generate tasks successfully in the future.

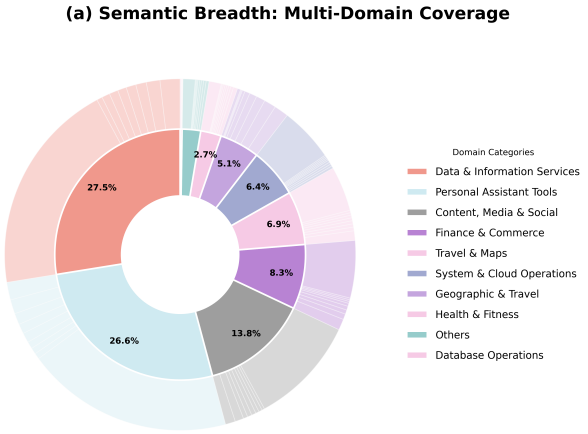

Figure 3. Semantic Breadth: The distribution of tools spans eight macro-domains and numerous specific entities, illustrating the environment's coverage of real-world scenarios.

4.2. Dataset Analysis

Based on the constructed toolsets, we synthesized high-quality training tasks and golden traces using the methods described in Section 3.2 and 3.3. For each toolset, we generated three distinct dependency graphs to cover different logical flows. From each graph, we sampled five tasks using our dynamic unlocking algorithm. To ensure the validity of these synthesized tasks, we implemented a rigorous verification pipeline using **LangGraph** (LangGraph, 2025) as the agent execution framework. We applied a Pass@8 filtering strategy: for each candidate task, a teacher agent (e.g., GPT-4.1 (GPT-4.1, 2025), Deepseek-V3.2 (DeepSeek-AI et al., 2025)) attempts to solve it up to 8 times (we chose the Qwen3-32B (Yang et al., 2025) model, which has performance and data distribution similar to the policy model). We calculate the reward for each sampled trajectory using the criteria established in Section 3.3. A task is deemed valid and retained only if at least one trajectory achieves a trace_score is one; otherwise, the task is discarded. Finally, to maintain dataset diversity and prevent overfitting to specific scenarios, we retained a maximum of ten distinct tasks per toolset environment. Detailed processing results are shown in Figure 4. Table 3 summarizes the statistics of the final GUST dataset and toolset.

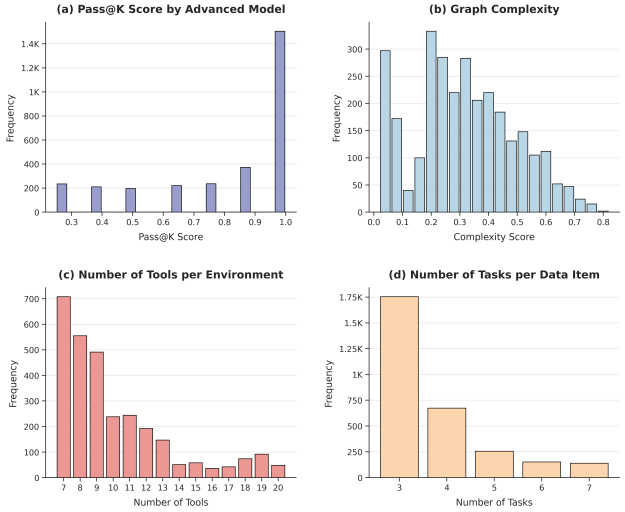

Figure 4. Statistical distribution of the GUST dataset across four metrics: (a) **Pass@K Score by Advanced Model**, showing a high concentration of successful task completions; (b) **Graph Complexity**, indicating a diverse range of logic flow difficulty; (c) **Number of Tools per Environment**, with most environments containing between 7 and 13 tools; and (d) **Number of Tasks per Data Item**, revealing that the data consist of 3 to 7 tasks.

4.3. Case Study

As show in Figure 5 , in our designed long-horizon tool use task, the agent faces the challenge of temporal dependency resolution. The model must not only plan the trajectory of tool invocations but also maintain a coherent internal state across multiple turns. As illustrated, the agent successfully identifies missing information (e.g., `budget constraints`), solicits it from the user, and synthesizes this new constraint with previously retrieved data (e.g., the specific date and location from the `Search_Event` output) to execute the final `Book_Hotel` action accurately.

5. Experiments

5.1. Experiment Setup

Training. We perform RL on the Qwen2.5-14B-Instruct (Qwen et al., 2025), Qwen3-4B, and Qwen3-8B models (Yang et al., 2025) using Group Relative Policy Optimization (GRPO) (Shao et al., 2024). A high-fidelity user simulator, DeepSeek-V3.2 (DeepSeek-AI et al., 2025), is deployed to interact dynamically with the agent. We train using the custom-built GUST dataset and environment, where per environment acts as a MCP toolset. Modifications to the

Table 2. Main results on agentic tool-using benchmarks, demonstrating that ToolVerse consistently achieves significant performance gains across all model scales and evaluation environments, with the TARA algorithm achieving the best overall results.

Model	BFCL Multi-Turn					τ^2-Bench				ACEBench-Agent		
	Base	Miss-Func	Miss-Param	Long-Context	Overall	Airline	Retail	Telecom	Overall	Multi-Step	Multi-Turn	Overall
Advanced Models												
DeepSeek-V3.2	41.50	39.50	33.50	35.00	37.39	63.8	81.1	96.2	80.37	92.50	68.75	80.62
GPT-4.1	47.50	32.50	32.50	43.00	38.88	56.0	74.0	34.0	54.67	95.00	70.83	82.92
Kimi-K2-Instruct	62.00	41.00	44.50	55.00	50.63	56.5	70.6	65.8	64.30	85.00	73.33	79.17
Qwen3-4B (Thinking)	32.00	20.00	24.00	25.50	25.38	19.50	30.48	13.20	21.06	35.91	46.66	41.28
+ ToolVerse (w/ GRPO)	37.00	30.00	25.00	22.00	**28.50**$_{+3.12}$	22.00	39.50	15.80	**25.77**$_{+4.71}$	40.00	56.67	**48.34**$_{+7.06}$
+ ToolVerse (w/ TARA)	36.00	31.50	22.00	23.50	**28.25**$_{+2.87}$	20.00	38.60	21.90	**26.83**$_{+5.77}$	50.00	60.00	**55.00**$_{+13.72}$
Qwen3-8B (Thinking)	32.00	33.50	22.00	28.00	28.88	22.00	43.20	18.40	27.87	53.44	39.58	46.51
+ ToolVerse (w/ GRPO)	41.50	38.00	31.00	30.50	**35.25**$_{+6.37}$	28.00	43.00	19.30	**30.10**$_{+2.23}$	60.00	53.33	**56.66**$_{+10.15}$
+ ToolVerse (w/ TARA)	51.50	38.50	28.50	31.50	**37.50**$_{+8.62}$	26.00	50.00	21.10	**32.37**$_{+4.50}$	60.00	63.33	**61.66**$_{+15.15}$
Qwen2.5-14B-Instruct	22.50	17.50	16.00	15.50	17.88	17.00	37.73	18.42	24.38	40.00	55.56	47.78
+ ToolVerse (w/ GRPO)	34.50	20.00	18.50	17.00	**22.50**$_{+4.62}$	17.50	47.60	25.40	**30.17**$_{+5.79}$	45.00	56.67	**50.84**$_{+3.06}$
+ ToolVerse (w/ TARA)	40.50	16.50	18.00	21.50	**24.12**$_{+6.24}$	20.00	45.60	31.60	**32.40**$_{+8.02}$	60.00	63.33	**61.66**$_{+13.88}$

Table 3. Statistics of the constructed GUST dataset.

Statistic	Count
Total Environments	422
Total Executable Tools	4438
Total Data Items	2987
Avg. Tasks per data item	3.8
Avg. Tools per environment	12.2

Verl framework (Sheng et al., 2024) are made to seamlessly integrate our MCP tool environment during training. Our experiments are conducted on a cluster of 32 NVIDIA A100 GPUs. The training configuration employs a global batch size of 128. Detailed hyperparameters are documented in Appendix B.

Evaluation. We evaluate the effectiveness of TOOLVERSE in enhancing long-horizon reasoning and tool-use capabilities across three widely-adopted multi-turn benchmarks: (1)BFCL-v3 Multi-Turn (Patil et al., 2025), which involves Python-based API interactions across four specialized categories, including *Base*, *Miss Param*, *Miss Func*, and *Long Context*; (2)τ^2-Bench (Barres et al., 2025), which assesses complex user-agent interactions in real-world domains such as Airline, Retail, and Telecom; and (3)ACEBench-Agent (Chen et al., 2025), which focuses on multi-step and multi-turn reasoning in dynamic environments. For τ^2-Bench and ACEBench-Agent, which require a conversational user, we employ `DeepSeek-V3.2`(DeepSeek-AI et al., 2025) as the user simulator. Notably, since the original ACEBench uses a `[func_name(param)]` prompt format by default, we modify the official implementation to support the LLMs' native function-calling interface to ensure consistency. To ensure the reliability and stability of our experimental results, we conduct four independent runs for each evaluation and report the average@4 scores.

5.2. Main Results

ToolVerse consistently delivers universal performance gains across diverse benchmarks and model architectures. Experimental results in Table 2 demonstrate that ToolVerse consistently yields significant improvements across all three benchmarks and all evaluated model scales. Notably, this framework proves highly effective for both "Thinking" models and "Non-thinking" models; for instance, Qwen3-8B (Thinking) achieves a remarkable gain of +15.15 on ACEBench-Agent, while the non-thinking Qwen2.5-14B-Instruct also sees a substantial +13.88 increase on the same benchmark. This universal success across various testing dimensions—ranging from missing parameter detection in BFCL to complex domain-specific reasoning in τ^2-Bench—validates that ToolVerse provides a robust and model-agnostic training paradigm for enhancing agentic tool-use capabilities.

Our Turn-Aware Relative Advantage algorithm provides substantial gains over the naive GRPO baseline. To isolate the impact of our proposed reward decomposition, we conduct an ablation study by comparing ToolVerse trained with naive GRPO (w/ GRPO) against our full TARA algorithm (w/ TARA) in Table 2. The results consistently show that TARA yields superior performance across all benchmarks. For instance, on ACEBench-Agent, while naive GRPO improves the Qwen3-8B model to 56.66%, TARA further elevates it to 61.66%, representing a significant additional boost. This performance gap is even more pronounced in the multi-turn and multi-step subtasks, where the credit assignment problem is most acute. By replacing the coarse

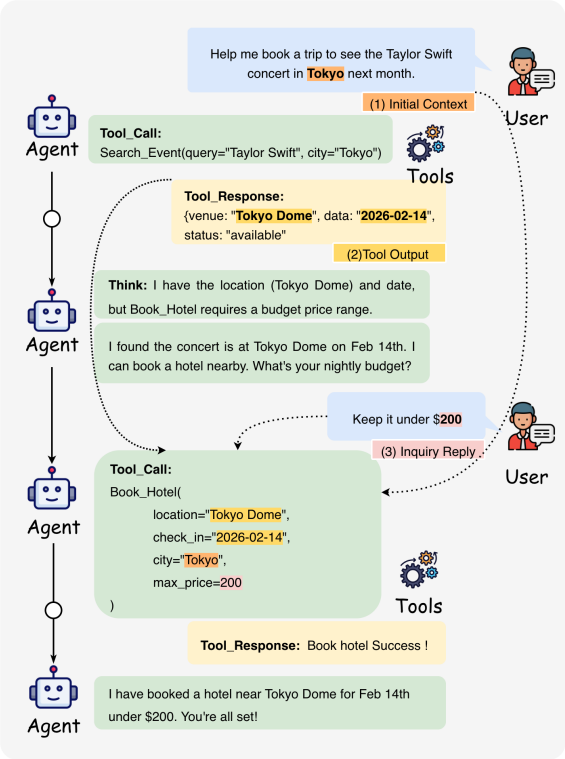

Figure 5. Long-horizon Multi-turn Tool use case Study. The agent demonstrates multi-turn contextual reasoning by inferring user intent, formulating a sequential execution plan, and dynamically resolving tool parameters (e.g., venue location and date) from prior dialogue history to complete the booking task.

trajectory-level advantage in GRPO with fine-grained, turn-level reinforcement signals, TARA effectively identifies and reinforces correct intermediate tool-calling logic, leading to a more robust and precise policy in complex, long-horizon environments.

5.3. Training Dynamics

The training curves in Figure 6 further substantiate the quantitative results. In the Trace Score plot, the Qwen3-8B model with TARA shows a clear and continuous improvement over training steps, with a sharper increase compared to the Naive GRPO approach. The Val Score plot similarly indicates stronger convergence for the TARA-enhanced model, reaching higher validation scores in fewer training steps. This observation suggests that TARA not only accelerates the learning process but also improves the stability of the learning trajectory, leading to faster convergence and better overall performance during the training process.

6. Conclusion

In this work, we presented **ToolVerse**, a framework that advances agentic reinforcement learning by scaling executable

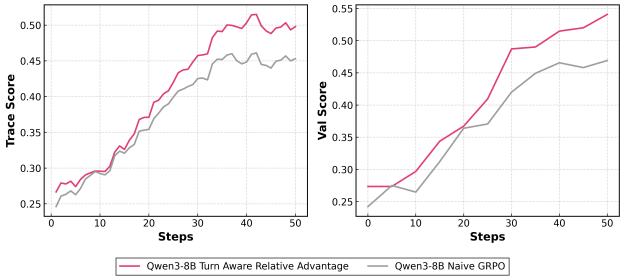

Figure 6. **Training Dynamics Comparison: TARA vs. Naive GRPO.** We report the Trace Score (left) and Validation Score (right) for Qwen3-8B throughout the training process. The TARA-enhanced model (pink) outperforms the Naive GRPO model (gray), with a more consistent and faster improvement in both scores across training steps. The Trace Score evaluates binary task completion, reflecting whether the agent successfully fulfills the final goal by executing every step in the trajectory correctly.

environments and synthesizing long-horizon tasks via the graph-based dynamic unlocking sampling algorithm. We have organized these tools and data into the GUST dataset. By integrating these diverse scenarios with a novel turn-aware credit assignment policy, our approach effectively addresses the challenges of sparse rewards and multi-step reasoning. We proved the correctness and effectiveness of the turn-aware advantage by comparing the training curves with the form of the formula derivation. Empirical results demonstrate that ToolVerse achieves advanced performance in complex tool-use benchmarks. In the future, we will continue to expand more advanced task generation methods in both simulated and real-world environments, and we will also continue to explore more intensive and naive credit assignment strategies.

Limitation

Despite the significant advancements enabled by ToolVerse in scaling agentic RL environments and improving long-horizon tool-integrated reasoning, our work has several limitations that warrant further investigation. First, although our dynamic task generation strategy leverages tool dependency graphs for coherent multi-step reasoning tasks, it relies on predefined dependencies within available protocols. This could limit the diversity of emergent behaviors compared to environments where novel tool interactions can be discovered autonomously. Second, our Turn-Aware Relative Advantage algorithm improves credit assignment granularity but still depends on reward signals defined at each turn; in highly sparse or ambiguous reward settings, the approach may struggle to provide sufficient feedback for optimal policy learning.

References

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Cao, S., Li, D., Zhao, F., Yuan, S., Hegde, S. R., Chen, C., Ruan, C., Griggs, T., Liu, S., Tang, E., Liaw, R., Moritz, P., Zaharia, M., Gonzalez, J. E., and Stoica, I. Skyrl-agent: Efficient rl training for multi-turn llm agent, 2025. URL https://arxiv.org/abs/2511.16108.

Castellani, T., Ye, N., Mittal, D., Yen, T., and Namkoong, H. Synthtools: A framework for scaling synthetic tools for agent development. *arXiv preprint arXiv:2511.09572*, 2025.

Chen, C., Hao, X., Liu, W., Huang, X., Zeng, X., Yu, S., Li, D., Wang, S., Gan, W., Huang, Y., Liu, W., Wang, X., Lian, D., Yin, B., Wang, Y., and Liu, W. Acebench: Who wins the match point in tool usage?, 2025. URL https://arxiv.org/abs/2501.12851.

DeepSeek-AI, Liu, A., Mei, A., Lin, B., Xue, B., Wang, B., Xu, B., Wu, B., Zhang, B., Lin, C., Dong, C., Lu, C., Zhao, C., Deng, C., Xu, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Li, E., Zhou, F., Lin, F., Dai, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Li, H., Liang, H., Wei, H., Zhang, H., Luo, H., Ji, H., Ding, H., Tang, H., Cao, H., Gao, H., Qu, H., Zeng, H., Huang, J., Li, J., Xu, J., Hu, J., Chen, J., Xiang, J., Yuan, J., Cheng, J., Zhu, J., Ran, J., Jiang, J., Qiu, J., Li, J., Song, J., Dong, K., Gao, K., Guan, K., Huang, K., Zhou, K., Huang, K., Yu, K., Wang, L., Zhang, L., Wang, L., Zhao, L., Yin, L., Guo, L., Luo, L., Ma, L., Wang, L., Zhang, L., Di, M. S., Xu, M. Y., Zhang, M., Zhang, M., Tang, M., Zhou, M., Huang, P., Cong, P., Wang, P., Wang, Q., Zhu, Q., Li, Q., Chen, Q., Du, Q., Xu, R., Ge, R., Zhang, R., Pan, R., Wang, R., Yin, R., Xu, R., Shen, R., Zhang, R., Liu, S. H., Lu, S., Zhou, S., Chen, S., Cai, S., Chen, S., Hu, S., Liu, S., Hu, S., Ma, S., Wang, S., Yu, S., Zhou, S., Pan, S., Zhou, S., Ni, T., Yun, T., Pei, T., Ye, T., Yue, T., Zeng, W., Liu, W., Liang, W., Pang, W., Luo, W., Gao, W., Zhang, W., Gao, X., Wang, X., Bi, X., Liu, X., Wang, X., Chen, X., Zhang, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Li, X., Yang, X., Li, X., Chen, X., Su, X., Pan, X., Lin, X., Fu, X., Wang, Y. Q., Zhang, Y., Xu, Y., Ma, Y., Li, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Qian, Y., Yu, Y., Zhang, Y., Ding, Y., Shi, Y., Xiong, Y., He, Y., Zhou, Y., Zhong, Y., Piao, Y., Wang, Y., Chen, Y., Tan, Y., Wei, Y., Ma, Y., Liu, Y., Yang, Y., Guo, Y., Wu, Y., Wu, Y., Cheng, Y., Ou, Y., Xu, Y., Wang, Y., Gong, Y., Wu, Y., Zou, Y., Li, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Zhao, Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Huang, Z., Wu, Z., Li, Z., Zhang, Z., Xu, Z., Wang, Z., Gu, Z., Zhu, Z., Li, Z., Zhang, Z., Xie, Z., Gao, Z., Pan, Z., Yao, Z., Feng, B., Li, H., Cai, J. L., Ni, J., Xu, L., Li, M., Tian, N., Chen, R. J., Jin, R. L., Li, S. S., Zhou, S., Sun, T., Li, X. Q., Jin, X., Shen, X., Chen, X., Song, X., Zhou, X., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Huang, Z., Xu, Z., Zhang, Z., Ji, D., Liang, J., Guo, J., Chen, J., Xia, L., Wang, M., Li, M., Zhang, P., Chen, R., Sun, S., Wu, S., Ye, S., Wang, T., Xiao, W. L., An, W., Wang, X., Sun, X., Wang, X., Tang, Y., Zha, Y., Zhang, Z., Ju, Z., Zhang, Z., and Qu, Z. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Dong, G., Chen, Y., Li, X., Jin, J., Qian, H., Zhu, Y., Mao, H., Zhou, G., Dou, Z., and Wen, J.-R. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.16410.

Fang, R., Cai, S., Li, B., Wu, J., Li, G., Yin, W., Wang, X., Wang, X., Su, L., Zhang, Z., Wu, S., Tao, Z., Jiang, Y., Xie, P., Huang, F., and Zhou, J. Towards general agentic intelligence via environment scaling, 2025. URL https://arxiv.org/abs/2509.13311.

GPT-4.1. Introducing gpt-4.1 in the api, 2025. URL https://openai.com/index/gpt-4-1/.

LangGraph. Langgraph overview, 2025. URL https://docs.langchain.com/oss/python/langgraph/overview.

Li, X., Zou, H., and Liu, P. Torl: Scaling tool-integrated rl, 2025a. URL https://arxiv.org/abs/2503.23383.

Li, Y., Inan, H. A., Yue, X., Chen, W.-N., Wutschitz, L., Kulkarni, J., Poovendran, R., Sim, R., and Rajmohan, S. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025b.

Li, Z., Zhang, H., Han, S., Liu, S., Xie, J., Zhang, Y., Choi, Y., Zou, J., and Lu, P. In-the-flow agentic system optimization for effective planning and tool use, 2025c. URL https://arxiv.org/abs/2510.05592.

Mai, X., Xu, H., Li, Z.-Z., W, X., Wang, W., Hu, J., Zhang, Y., and Zhang, W. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving, 2025. URL https://arxiv.org/abs/2505.07773.

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation

of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Prabhakar, A., Liu, Z., Zhu, M., Zhang, J., Awalgaonkar, T., Wang, S., Liu, Z., Chen, H., Hoang, T., Niebles, J. C., Heinecke, S., Yao, W., Wang, H., Savarese, S., and Xiong, C. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay, 2025. URL https://arxiv.org/abs/2504.03601.

Qian, C., Acikgoz, E. C., He, Q., Wang, H., Chen, X., Hakkani-Tür, D., Tur, G., and Ji, H. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Shang, N., Liu, Y., Zhu, Y., Zhang, L. L., Xu, W., Guan, X., Zhang, B., Dong, B., Zhou, X., Zhang, B., Xin, Y., Miao, Z., Li, S., Yang, F., and Yang, M. rstar2-agent: Agentic reasoning technical report, 2025. URL https://arxiv.org/abs/2508.20722.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Shi, D., Cao, J., Chen, Q., Sun, W., Li, W., Lu, H., Dong, F., Qin, T., Zhu, K., Liu, M., Yang, J., Zhang, G., Liu, J., Zhang, C., Wang, J., Jiang, Y. E., and Zhou, W. Taskcraft: Automated generation of agentic tasks, 2025. URL https://arxiv.org/abs/2506.10055.

Wu, J., Li, B., Fang, R., Yin, W., Zhang, L., Tao, Z., Zhang, D., Xi, Z., Fu, G., Jiang, Y., Xie, P., Huang, F., and Zhou, J. Webdancer: Towards autonomous information seeking agency, 2025. URL https://arxiv.org/abs/2505.22648.

Xiang, J., Tao, T., Gu, Y., Shu, T., Wang, Z., Yang, Z., and Hu, Z. Language models meet world models: Embodied experiences enhance language models. *Advances in neural information processing systems*, 36:75392–75412, 2023.

Xue, Z., Zheng, L., Liu, Q., Li, Y., Zheng, X., Ma, Z., and An, B. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning, 2025. URL https://arxiv.org/abs/2509.02479.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Ye, J., Jiang, C., Du, Z., Xu, Y., Yao, X., Xi, Z., Fan, X., Zhang, Q., Gui, T., Huang, X., and Chen, J. Feedback-driven tool-use improvements in large language models via automated build environments, 2025. URL https://arxiv.org/abs/2508.08791.

A. Theoretical Analysis of Turn-Aware Relative Advantage

In this section, we provide a theoretical justification for the Turn-Aware Relative Advantage mechanism proposed in Section 3.3. We focus on two key properties: (1) variance reduction in the advantage estimation, and (2) the suppression of false positive credits (distractors) via the consistency gate.

A.1. Preliminaries and Definitions

Let $\tau = (s_1, a_1, \ldots, s_T, a_T)$ denote a trajectory of length T. In the standard Group Relative Policy Optimization (GRPO), the advantage for an action a_t is typically estimated using the cumulative return of the entire trajectory or the return-to-go. Let G_t^{std} denote the standard return formulation used for advantage estimation at step t:

$$G_t^{\text{std}} = r(s_t, a_t) + \sum_{k=1}^{T-t} \gamma^k r(s_{t+k}, a_{t+k}) \tag{6}$$

where the first term is the immediate reward and the summation represents the future return $R_{t+1:T}$.

In our proposed method, the advantage is derived from a fused value estimate G_t^{ours}. Based on the definitions of Local Advantage and Gated Future Advantage, the underlying value estimate can be expressed as:

$$G_t^{\text{ours}} = r_{i,t} + \lambda \cdot V_{i,t} = r_{i,t} + \lambda \cdot \delta_{i,t} \cdot \sum_{k=1}^{T-t} \gamma^k r_{i,t+k} \tag{7}$$

Here, $r_{i,t}$ is the deterministic binary reward based on the golden trace, $\lambda \in [0.5, 1.0]$ is the fusion weight, and $\delta_{i,t} \in \{0, 1\}$ is the consistency gate.

A.2. Variance Reduction

A primary challenge in long-horizon reasoning is the high variance of future returns, which introduces noise into the gradient estimation for earlier steps.

Assumption A.1. We assume that the immediate reward $r_{i,t}$ is deterministic given the state s_t and action a_i, t (i.e., strict matching with the golden trace). The future return $R_{t+1:T} = \sum_{k=1}^{T-t} \gamma^k r_{i,t+k}$ is a random variable with variance σ^2_{future} due to the stochasticity of the policy and environment in subsequent steps.

Theorem A.2 (Variance Reduction)**.** *For any valid reasoning step where $r_{i,t} = 1$ (and thus $\delta_{i,t} = 1$), the variance of the Turn-Aware value estimate is strictly lower than that of the standard return-to-go estimate, provided that the fusion hyperparameter satisfies $\lambda < 1$.*

Proof. Consider the variance of the standard estimator G_t^{std}:

$$\text{Var}(G_t^{\text{std}}) = \text{Var}(r_{i,t} + R_{t+1:T}) \tag{8}$$

$$= \text{Var}(r_{i,t}) + \text{Var}(R_{t+1:T}) + 2\text{Cov}(r_{i,t}, R_{t+1:T}) \tag{9}$$

Since $r_{i,t}$ is deterministic given the current step match, $\text{Var}(r_{i,t}) = 0$ and the covariance term vanishes. Thus:

$$\text{Var}(G_t^{\text{std}}) = \text{Var}(R_{t+1:T}) = \sigma^2_{\text{future}} \tag{10}$$

Now consider our proposed estimator G_t^{ours} given a valid current step ($\delta_{i,t} = 1$):

$$\text{Var}(G_t^{\text{ours}}) = \text{Var}(r_{i,t} + \lambda \cdot 1 \cdot R_{t+1:T}) \tag{11}$$

$$= \lambda^2 \text{Var}(R_{t+1:T}) \tag{12}$$

$$= \lambda^2 \sigma^2_{\text{future}} \tag{13}$$

Since $\lambda < 1$, it follows that $\lambda^2 < 1$, and therefore:

$$\text{Var}(G_t^{\text{ours}}) < \text{Var}(G_t^{\text{std}}) \tag{14}$$

Furthermore, for an invalid step ($r_{i,t} = 0$), the gate $\delta_{i,t} = 0$ forces $V_{i,t} = 0$, resulting in $\text{Var}(G_t^{\text{ours}}) = 0$. In both cases, our method significantly reduces the variance introduced by downstream uncertainty. □

A.3. Consistency Gating and Distractor Suppression

In tool-use environments, an agent may select an incorrect action (a "distractor") but still achieve a positive outcome later due to spurious correlations or environment tolerance. Standard RL reinforces such distractors.

Definition A.3 (Distractor Action)**.** An action $a_{i,t}$ is defined as a distractor if it deviates from the golden trace (i.e., $r_{i,t} = 0$), yet the subsequent trajectory yields a positive future return $R_{t+1:T} > 0$.

Theorem A.4 (Distractor Suppression)**.** *Under the Turn-Aware Relative Advantage mechanism, the total advantage $A_{i,t}^{total}$ for any distractor action is guaranteed to be non-positive, regardless of the magnitude of future returns.*

Proof. Let $a_{i,t}$ be a distractor action. By definition, $r_{i,t} = 0$.

First, we analyze the **Local Advantage** $A_{i,t}^{\text{local}}$. The standardized advantage is calculated as:

$$A_{i,t}^{\text{local}} = \frac{r_{i,t} - \mu_t^{\text{local}}}{\sigma_t^{\text{local}} + \epsilon} = \frac{0 - \mu_t^{\text{local}}}{\sigma_t^{\text{local}} + \epsilon} \tag{15}$$

Assuming the group size K is sufficiently large and the task is solvable, at least one sampled trajectory (or the theoretical optimal) will match the trace, implying the group mean reward $\mu_t^{\text{local}} > 0$. Consequently, $A_{i,t}^{\text{local}} < 0$.

Second, we analyze the **Future Advantage** $A_{i,t}^{\text{future}}$. The consistency gate is defined as $\delta_{i,t} = r_{i,t}$. Since $r_{i,t} = 0$, the gate closes:

$$V_{i,t} = \delta_{i,t} \cdot \sum_{k=0}^{\infty} \gamma^k r_{i,t+k+1} = 0 \cdot R_{t+1:T} = 0 \tag{16}$$

Even if the future return $R_{t+1:T}$ is high (lucky guess), it is blocked. The relative future advantage becomes:

$$A_{i,t}^{\text{future}} = \frac{0 - \mu_t^{\text{future}}}{\sigma_t^{\text{future}} + \epsilon} \tag{17}$$

If the group generally performs well ($\mu_t^{\text{future}} > 0$), this term is negative. If the group performs poorly ($\mu_t^{\text{future}} \approx 0$), this term is zero.

Finally, the **Total Advantage** is:

$$A_{i,t}^{\text{total}} = A_{i,t}^{\text{local}} + \lambda A_{i,t}^{\text{future}} \tag{18}$$

Since $A_{i,t}^{\text{local}}$ is strictly negative and $\lambda A_{i,t}^{\text{future}}$ is non-positive, the total advantage $A_{i,t}^{\text{total}}$ is strictly negative. Thus, the probability of the distractor action $\pi(a_{i,t}|s_t)$ will be suppressed, eliminating the credit assignment error found in baseline methods. □

B. Implementation Details

To ensure the reproducibility of our experiments, we provide the specific hyperparameter configurations used during the training phase. The model is optimized using a learning rate of 1×10^{-6} with a mini-batch size of 32. To manage sequence complexity, we set the maximum prompt length to $8,192$ tokens and the total response length to $24,000$ tokens, while limiting each individual turn's response to $1,024$ tokens. Data preprocessing includes shuffling and the filtering of overlong prompts to maintain training stability. Notably, the KL divergence loss is disabled in this setup ($kl\_loss = \text{False}$). For our proposed advantage mixing mechanism, we employ a mixing weight $\lambda_{mix} = 0.5$ for future advantages and a discount factor $\gamma = 0.5$ to balance immediate and long-term rewards.

Table 4. Summary of Experimental Hyperparameters

Hyperparameter	**Value**
Learning Rate	1×10^{-6}
Mini-batch Size	32
Max Prompt Length	$8,192$
Max Response Length	$24,000$
Single-turn Response Length	$1,024$
KL Loss (kl_loss)	False
Mixing Weight (λ_{mix})	0.5
Discount Factor (γ)	0.5
Data Shuffling	True
Filter Overlong Prompts	True

C. Prompts

C.1. Prompt Template for Task Generation

Prompt: Table 6 demonstrates the prompt template used to synthesize the user query and multi-turn dialogue based on the generated tool trajectory.

Prompt: Long-horizon Task Synthesis Prompt Template

Role: You are an expert AI dataset synthesizer specializing in constructing long-horizon reasoning tasks.

Input Data:

- **Tool Dependency Graph:** A logical graph defining the causal links between tools.
- **Golden Trace:** A sequence of executed tool calls: $\mathcal{T} = [t_1, t_2, \ldots, t_n]$, including input arguments and return values.

Task: Based on the provided *Golden Trace* and dependency logic, generate a natural language **User Query** and a **Multi-turn Dialogue History** that would naturally lead to this specific sequence of tool executions.

Requirements:

1. **Implicit Dependencies:** The user query should not explicitly list every step. Instead, it should state a high-level goal that implicitly requires the agent to figure out the dependencies (e.g., "Book me a flight" implies checking availability first).
2. **Information Gap:** Ensure the dialogue reflects a realistic interaction where the agent might need to ask for clarification if parameters were missing in the early stages, matching the structure of the Golden Trace.
3. **Consistency:** The generated user intent must strictly align with the final output of the last tool in the trace.

Input Trace:
{{golden_trace_json}}

Output Format:
Return the result in JSON format containing "user_query", "dialogue_history", and "reasoning_chain".

Table 5. A prompt template used to generate tasks from an instantiated tool "golden trace".

Prompt: Reproduce Tool Defination Prompt Template

You are now a JSON Tool Refactoring Assistant. Your core responsibility is to refactor tool definitions based on a given seed, following specified steps to output tool descriptions in JSON format that comply with the OpenAI Schema specification.

Core Analysis Steps (Step 1):

- Comprehensive Deconstruction: Understand all input tool information, including tool names, descriptions, parameters (name, type, description), and required fields.
- Business Scenario Analysis: Clarify the core business problems these tools solve and the applicable business domains.
- Compatibility Assessment: Judge whether the current tool definitions are suitable for implementation via Python functions with a dictionary database. Analyze issues in naming, parameter design, and functional descriptions (e.g., redundancy, parameter-function mismatch, or inability to implement independently).

Refactoring Requirements (Step 2): Refactor the names, descriptions, and parameters based on Step 1:

1. Functional Fitness: Each refactored tool must be independently implementable by a single Python function with clear logic and no redundant cross-function dependencies.
2. OpenAI Schema Refinement:
   - Name: Concise and precise, reflecting the core function (lowercase + underscores).
   - Description: Clear and complete, explaining the purpose and scope within the business scenario.
   - Parameters: Optimize names and descriptions to ensure clarity; limit to a maximum of 4 parameters using standard Python types (int, float, str, list/array, dict).
   - Call Correlation: Build business logic-driven constraints for the toolset. Define dependencies where the input of one tool correlates to the output of another, supporting multi-turn reasoning.
   - Response Field: Add a mandatory "response" field containing at least "status" (success/error), "message", and other function-related output keys.

Output Requirements:

- Scenario Description: Describe the analyzed result in one paragraph wrapped in `<description></description>` tags.
- Tool Output: A strict JSON array of objects with "type": "function" and "function" keys (containing name, description, parameters, and the custom response field).
- Language: The entire output must be in English.

Input Tool Definition:

{{`tools_defination`}}

Table 6. The prompt template used for reproduce tool defination.