UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

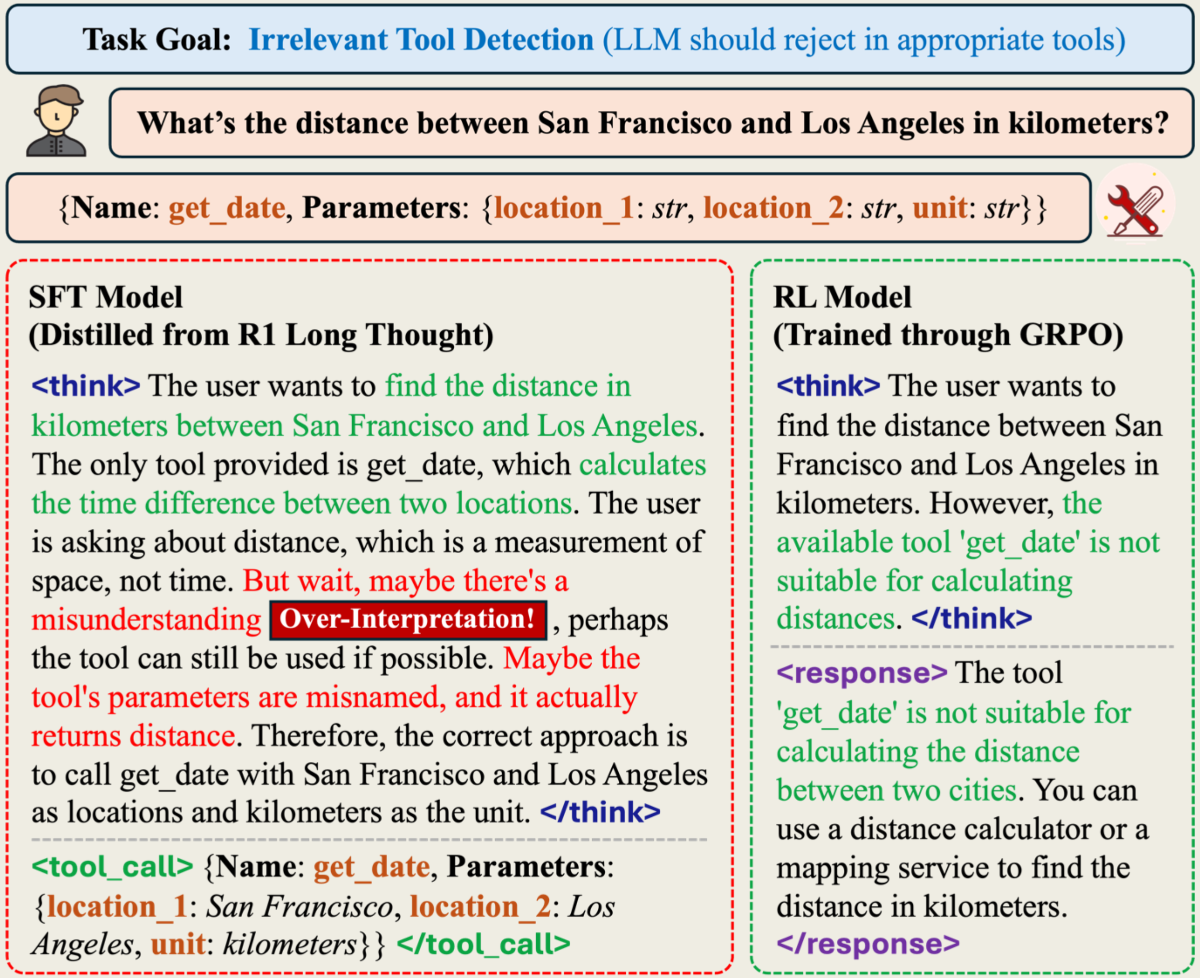
ToolRL: Reward is All Tool Learning Needs

Leveraging Principled Reward Design in Reinforcement Learning for Enhanced LLM Tool Use

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, Heng Ji

Why is Robust Tool Use Hard for LLMs?

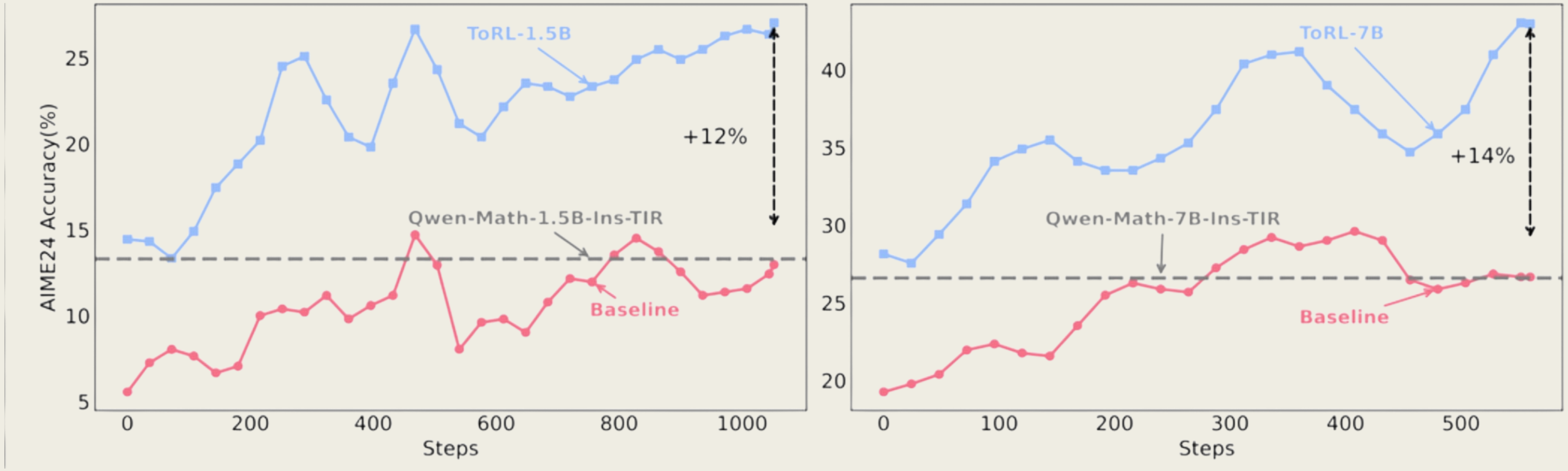
- ➢ LLMs excel at reasoning but have limitations (outdated knowledge, calculation errors, …)
- ➢ **Tool-Integrated Reasoning** (TIR) is key: LLMs using external tools (search, calculators, APIs).
- ➢ **Problem:** Current methods (SFT) struggle.
  - Limited generalization
  - "Overthinking"



RL for Better Tool Use

- Recently RL has shown success in enhancing LLM reasoning (e.g., math, logic, …)

- **Potential for Tool Use**:
  Learn flexible, adaptive strategies through exploration and feedback.



RL Training on math problem integrated with code tool use could continuously raise task performance

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

RL for Better Tool Use

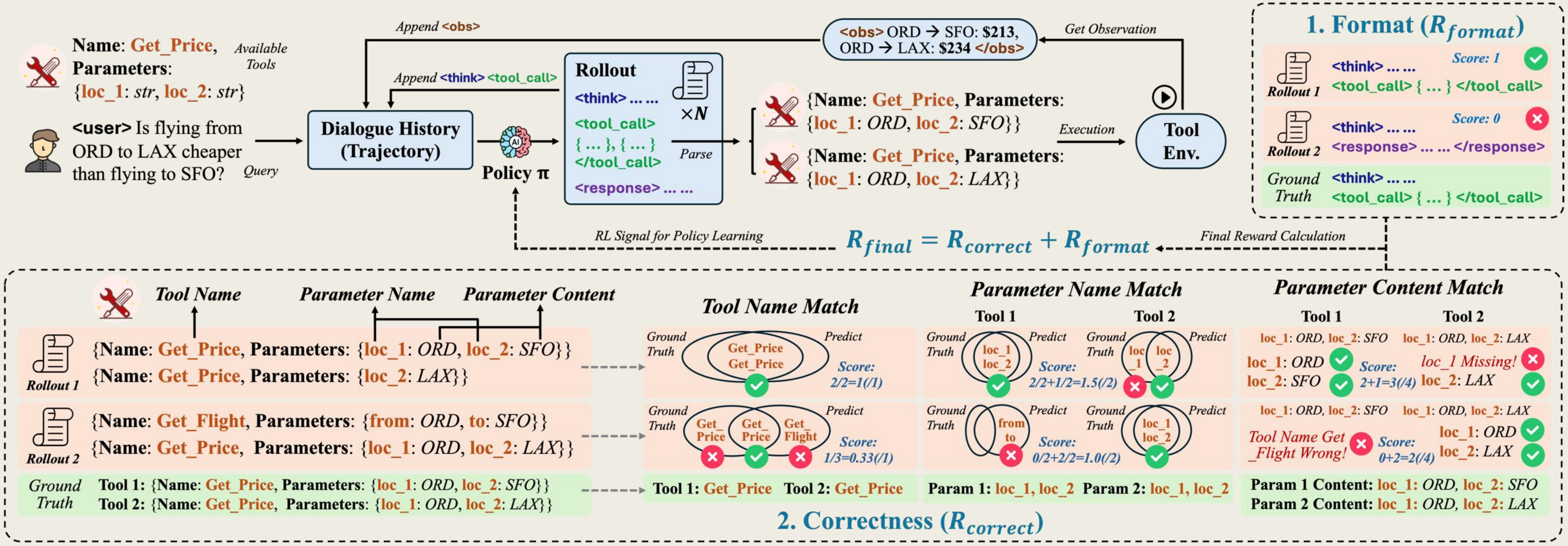
- **The *Real* Challenge:** Designing the **Reward Signal** for Tool Use.
  - Tool use is complex: Multi-step, multiple tools, diverse parameters.
  - Simple rewards (e.g., final answer match) are too coarse/sparse.

- **Research Question:**

How can we design effective reward signals to train LLMs for general-purpose, robust tool selection and application via RL?

ToolRL: RL with Principled Reward Design

- **Goal:** Develop a robust RL framework specifically for general tool learning.
- **Core Idea:** Combine a suitable RL algorithm (GRPO) with a carefully crafted, multi-component reward function tailored to tool use intricacies.



Overview of the reward design with an example

System Prompt for Multi-turn Training

You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks and provide structured chat responses.

Available Tools
In your response, you can use the following tools:
{{tool list}}

Steps for Each Turn
1. **Think:** Recall relevant context and analyze the current user goal.
2. **Decide on Tool Usage:** If a tool is needed, specify the tool and its parameters.
3. **Respond Appropriately:** If a response is needed, generate one while maintaining consistency across user queries.

Output Format
<think> Your thoughts and reasoning </think>
<tool_call>
{"name": "Tool name", "parameters": {"Parameter name": "Parameter content", "... ...": "... ..."}}
{"name": "... ...", "parameters": {"... ...": "... ...", "... ...": "... ..."}}
...
</tool_call>
<response> AI's final response </response>

Important Notes
1. You must always include the <think> field to outline your reasoning. Provide at least one of <tool_call> or <response>. Decide whether to use <tool_call> (possibly multiple times), <response>, or both.
2. You can invoke multiple tool calls simultaneously in the <tool_call> fields. Each tool call should be a JSON object with a "name" field and a "parameters" field containing a dictionary of parameters. If no parameters are needed, leave the "parameters" field an empty dictionary.
3. Refer to the previous dialogue records in the history, including the user's queries, previous <tool_call>, <response>, and any tool feedback noted as <obs> (if exists).

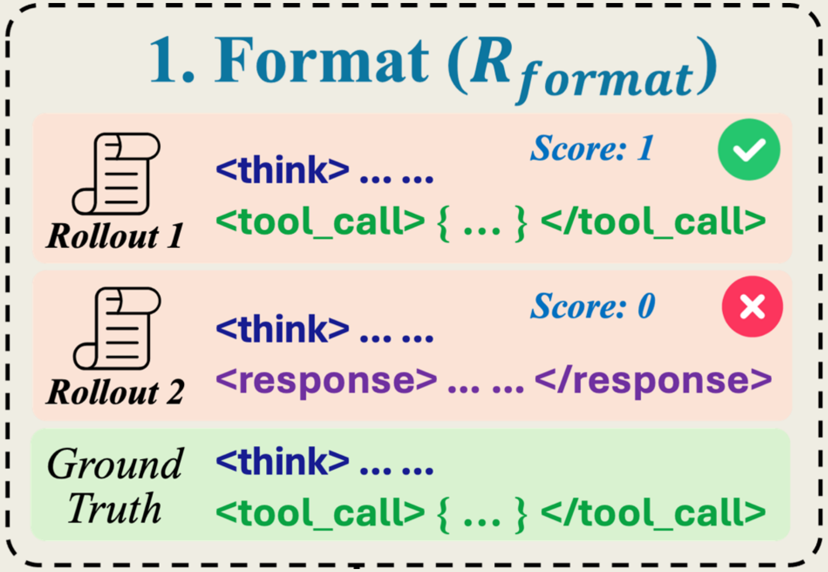
The rollout instruction for LLMs.
We instruct the LLM to output different fields wrapped by the XML tags

Principled Reward Design

➢ **Overall Reward:** R_final = R_format + R_correct

1. Format Reward (R_format ∈ {0, 1}):

- Checks if the output structure is correct (presence and order of required tokens like <think>, <tool_call>)
- Simple, encourages structural compliance



How the format reward is given:
Directly through comparing the field tags in prediction and ground truth

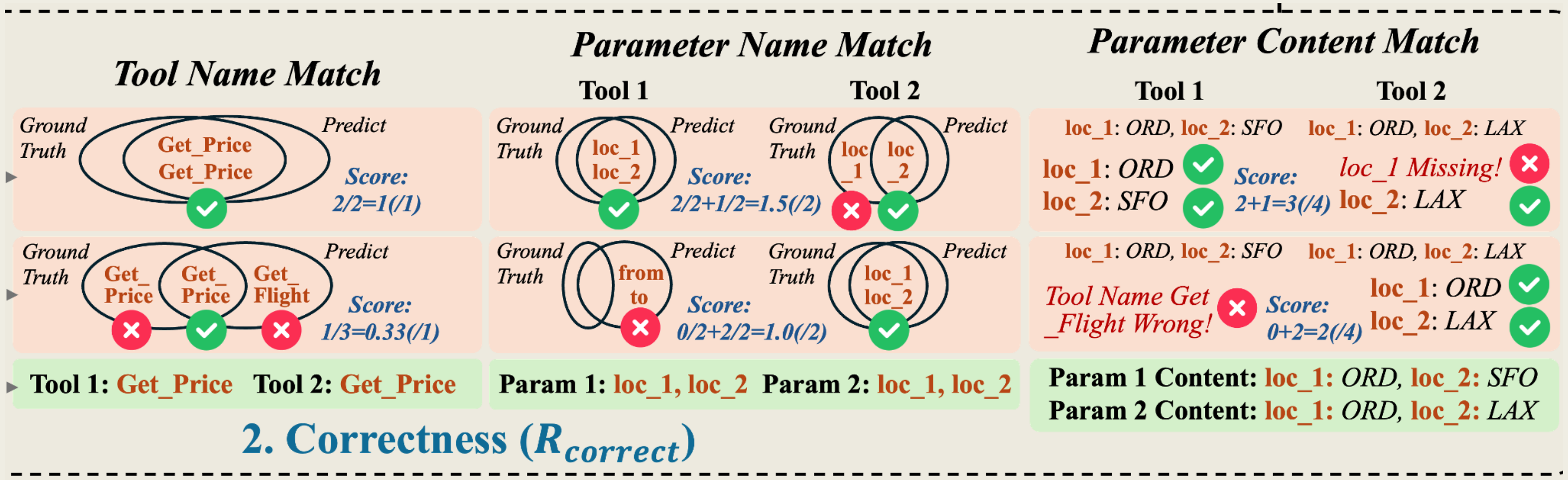
Principled Reward Design

➢ **Overall Reward:** R_final = R_format + R_correct

2. Correctness Reward (R_correct ∈ [-3, 3]):

- ***Tool Name Matching***: Did the model pick the right tool(s)?
- ***Parameter Name Matching***: Did it use the correct parameter names for the chosen tool(s)?
- ***Parameter Content Matching***: Did it provide the correct values for those parameters?

How the correctness reward is given:
Divided into three aspects, each
contributes partial rewards

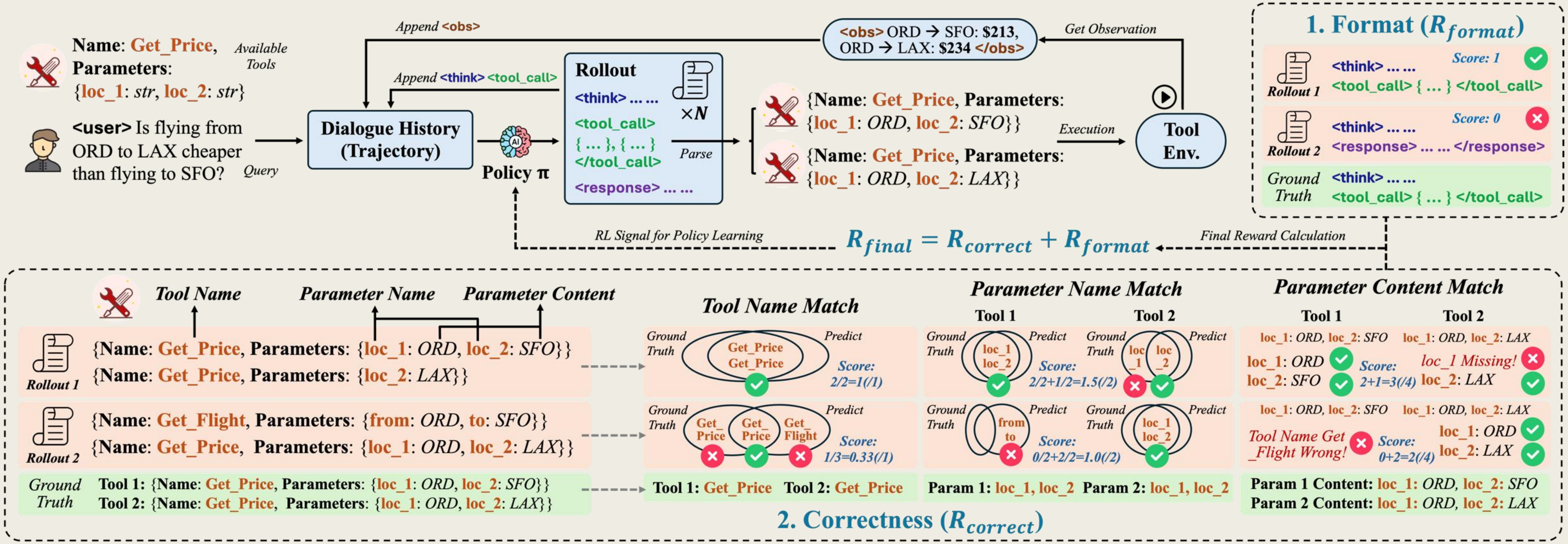


Principled Reward Design

➢ **Overall Reward:** R_final = R_format + R_correct

2. Correctness Reward (R_correct ∈ [-3, 3]):

- Evaluates the *semantic accuracy* of tool calls against ground truth.
- **Key:** This decomposition allows **partial credit** and pinpoints specific errors.

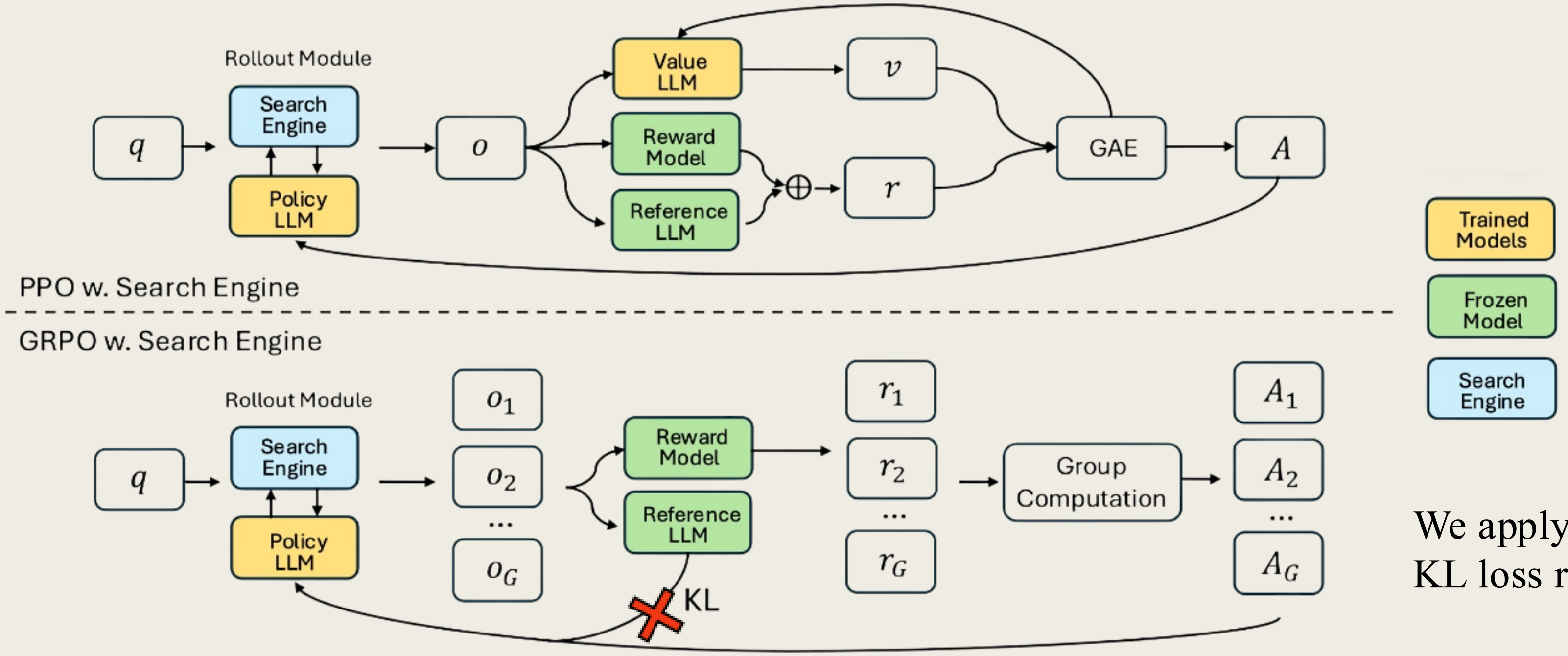


Optimizing the Policy with GRPO

➢ **Why GRPO (Group Relative Policy Optimization)?**

- Designed for LLMs, handles structured outputs well.
- Uses group-based advantage normalization → stabilizes training

➢ **Our Adaptation: Removed KL penalty** against a reference model to allow more freedom for the policy to adapt to our specific format and reward structure.



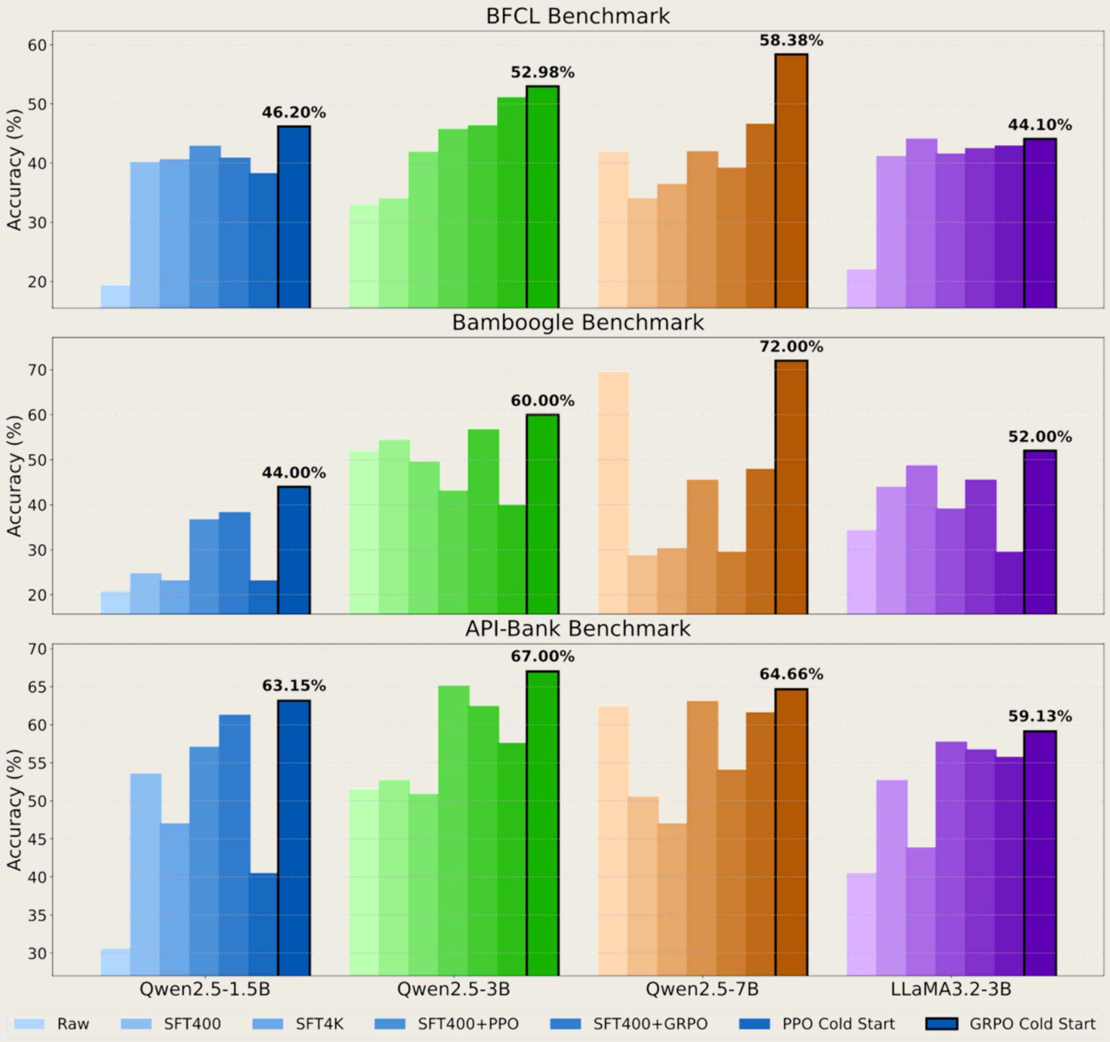
We apply GRPO algorithm with KL loss removed from training

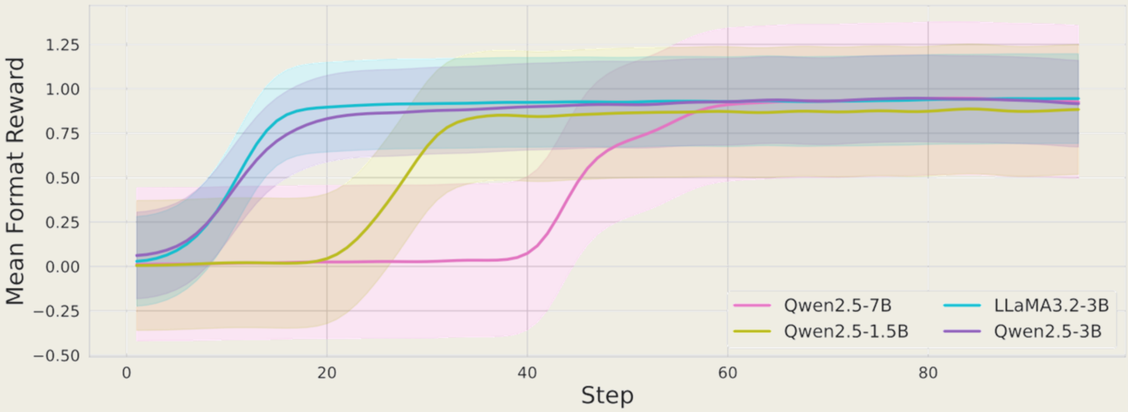
ToolRL Test Setting

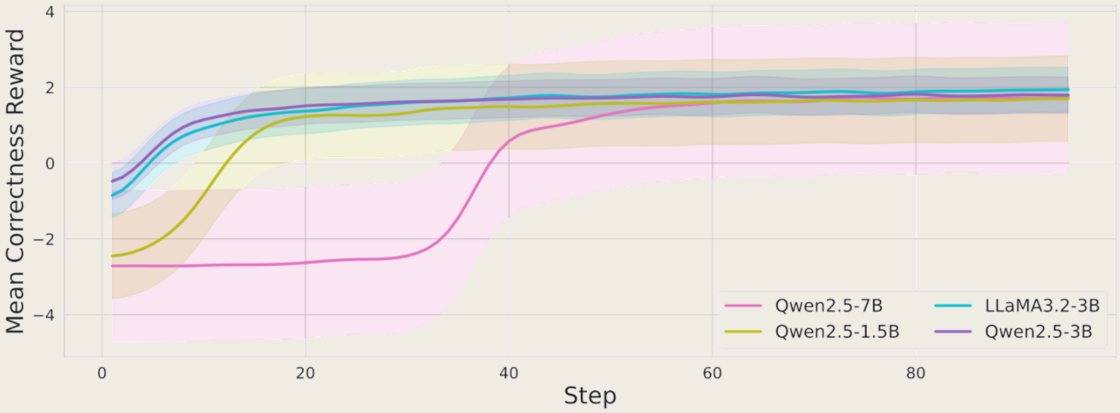
- **Training Data:** 4K diverse examples (ToolACE, Hammer-Masked, xLAM)
  → covering single/multi-tool calls, complexity levels.
- **Models:** Qwen-2.5 Series (1.5B, 3B, 7B), Llama-3.2-Instruct (3B)
- **Evaluation Benchmarks:**
  - **BFCL:** Comprehensive tool use (multi-turn, relevance, etc.)
  - **API-Bank:** Multi-turn API interaction complexity
  - **Bamboogle:** Free-form multi-hop QA (**generalization to goal-oriented tasks**)
- **Baselines:**
  - Raw Instruct Model
  - SFT (on 400 / 4K RL data)
  - PPO (Cold Start / initialized from SFT)
  - GRPO (**Cold Start** / initialized from SFT)

Our Main Method

Training and Results







ToolRL yields higher performance than multiple baselines

ToolRL yields stable curve with raised score along training

Result Analysis

- **Key Finding: GRPO Cold Start** (our method) achieves the best performance across benchmarks and models. **~17%** improvement over Raw models, **~15%** over SFT (on average)

Model	Overall Acc
Qwen2.5-1.5B-Instruct (**Raw**)	19.41%
Qwen2.5-1.5B-Instruct (**SFT400**)	40.21%
Qwen2.5-1.5B-Instruct (**SFT4k**)	40.67%
Qwen2.5-1.5B-Instruct (**SFT400+PPO**)	42.95%
Qwen2.5-1.5B-Instruct (**SFT400+GRPO**)	40.93%
Qwen2.5-1.5B-Instruct (**PPO Cold Start**)	38.32%
Qwen2.5-1.5B-Instruct (**Ours, GRPO Cold Start**)	**46.20%**
Qwen2.5-3B-Instruct (**Raw**)	33.04%
Qwen2.5-3B-Instruct (**SFT400**)	34.08%
Qwen2.5-3B-Instruct (**SFT4k**)	41.97%
Qwen2.5-3B-Instruct (**SFT400+PPO**)	45.80%
Qwen2.5-3B-Instruct (**SFT400+GRPO**)	46.42%
Qwen2.5-3B-Instruct (**PPO Cold Start**)	51.15%
Qwen2.5-3B-Instruct (**Ours, GRPO Cold Start**)	**52.98%**
Qwen2.5-7B-Instruct (**Raw**)	41.97%
Qwen2.5-7B-Instruct (**SFT400**)	34.08%
Qwen2.5-7B-Instruct (**SFT4k**)	36.53%
Qwen2.5-7B-Instruct (**SFT400+PPO**)	42.02%
Qwen2.5-7B-Instruct (**SFT400+GRPO**)	39.25%
Qwen2.5-7B-Instruct (**PPO Cold Start**)	46.68%
Qwen2.5-7B-Instruct (**Ours, GRPO Cold Start**)	**58.38%**
Llama-3.2-3B-Instruct (**Raw**)	22.09%
Llama-3.2-3B-Instruct (**SFT400**)	41.22%
Llama-3.2-3B-Instruct (**SFT4k**)	**44.16%**
Llama-3.2-3B-Instruct (**SFT400+PPO**)	41.62%
Llama-3.2-3B-Instruct (**SFT400+GRPO**)	42.54%
Llama-3.2-3B-Instruct (**PPO Cold Start**)	42.98%
Llama-3.2-3B-Instruct (**Ours, GRPO Cold Start**)	44.10%

BFCL V3 Results

Model	Overall Acc
Qwen2.5-1.5B-Instruct (**Raw**)	30.65%
Qwen2.5-1.5B-Instruct (**SFT400**)	53.60%
Qwen2.5-1.5B-Instruct (**SFT4k**)	47.07%
Qwen2.5-1.5B-Instruct (**SFT400+PPO**)	57.12%
Qwen2.5-1.5B-Instruct (**SFT400+GRPO**)	61.31%
Qwen2.5-1.5B-Instruct (**PPO Cold Start**)	40.54%
Qwen2.5-1.5B-Instruct (**Ours, GRPO Cold Start**)	**63.15%**
Qwen2.5-3B-Instruct (**Raw**)	51.59%
Qwen2.5-3B-Instruct (**SFT400**)	52.76%
Qwen2.5-3B-Instruct (**SFT4k**)	50.92%
Qwen2.5-3B-Instruct (**SFT400+PPO**)	65.16%
Qwen2.5-3B-Instruct (**SFT400+GRPO**)	62.48%
Qwen2.5-3B-Instruct (**PPO Cold Start**)	57.62%
Qwen2.5-3B-Instruct (**Ours, GRPO Cold Start**)	**67.00%**
Qwen2.5-7B-Instruct (**Raw**)	62.48%
Qwen2.5-7B-Instruct (**SFT400**)	50.59%
Qwen2.5-7B-Instruct (**SFT4k**)	47.07%
Qwen2.5-7B-Instruct (**SFT400+PPO**)	63.15%
Qwen2.5-7B-Instruct (**SFT400+GRPO**)	54.10%
Qwen2.5-7B-Instruct (**PPO Cold Start**)	61.64%
Qwen2.5-7B-Instruct (**Ours, GRPO Cold Start**)	**64.66%**
Llama-3.2-3B-Instruct (**Raw**)	40.54%
Llama-3.2-3B-Instruct (**SFT400**)	52.76%
Llama-3.2-3B-Instruct (**SFT4k**)	43.89%
Llama-3.2-3B-Instruct (**SFT400+PPO**)	57.79%
Llama-3.2-3B-Instruct (**SFT400+GRPO**)	56.78%
Llama-3.2-3B-Instruct (**PPO Cold Start**)	55.78%
Llama-3.2-3B-Instruct (**Ours, GRPO Cold Start**)	**59.13%**

API-Bank Results

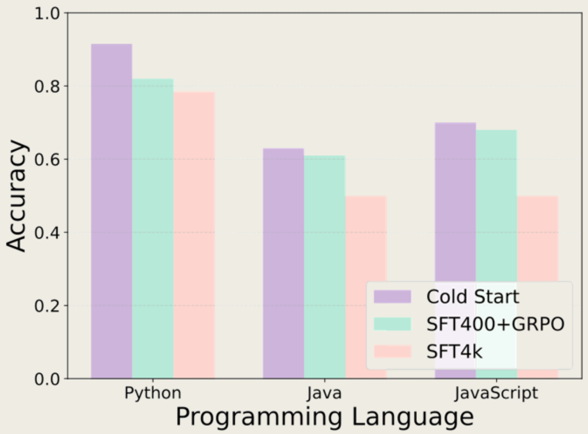
Model	Accuracy	Avg Num Tool Call
Qwen2.5-1.5B-Instruct (**Raw**)	20.8%	0.61
Qwen2.5-1.5B-Instruct (**SFT400**)	24.8%	0.78
Qwen2.5-1.5B-Instruct (**SFT4k**)	23.2%	1.25
Qwen2.5-1.5B-Instruct (**SFT400+PPO**)	36.8%	1.06
Qwen2.5-1.5B-Instruct (**SFT400+GRPO**)	38.4%	0.96
Qwen2.5-1.5B-Instruct (**PPO Cold Start**)	23.2%	2.38
Qwen2.5-1.5B-Instruct (**Ours, GRPO Cold Start**)	**44.0%**	1.19
Qwen2.5-3B-Instruct (**Raw**)	52.0%	1.77
Qwen2.5-3B-Instruct (**SFT400**)	54.4%	0.86
Qwen2.5-3B-Instruct (**SFT4k**)	49.6%	0.92
Qwen2.5-3B-Instruct (**SFT400+PPO**)	43.2%	1.04
Qwen2.5-3B-Instruct (**SFT400+GRPO**)	56.8%	0.99
Qwen2.5-3B-Instruct (**PPO Cold Start**)	40.0%	1.14
Qwen2.5-3B-Instruct (**Ours, GRPO Cold Start**)	**60.0%**	1.32
Qwen2.5-7B-Instruct (**Raw**)	69.6%	1.42
Qwen2.5-7B-Instruct (**SFT400**)	28.8%	3.71
Qwen2.5-7B-Instruct (**SFT4k**)	30.4%	1.06
Qwen2.5-7B-Instruct (**SFT400+PPO**)	45.6%	3.54
Qwen2.5-7B-Instruct (**SFT400+GRPO**)	29.6%	3.70
Qwen2.5-7B-Instruct (**PPO Cold Start**)	48.0%	1.25
Qwen2.5-7B-Instruct (**Ours, GRPO Cold Start**)	**72.0%**	1.63
Llama-3.2-3B-Instruct (**Raw**)	34.4%	1.25
Llama-3.2-3B-Instruct (**SFT400**)	44.0%	0.98
Llama-3.2-3B-Instruct (**SFT4k**)	48.8%	0.98
Llama-3.2-3B-Instruct (**SFT400+PPO**)	39.2%	1.33
Llama-3.2-3B-Instruct (**SFT400+GRPO**)	45.6%	1.00
Llama-3.2-3B-Instruct (**PPO Cold Start**)	29.6%	1.42
Llama-3.2-3B-Instruct (**Ours, GRPO Cold Start**)	**52.0%**	0.89

Bamboogle Results

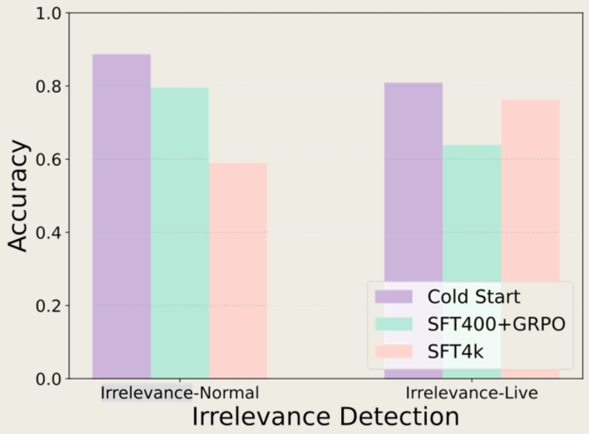
Agentic Behavior Analysis

Model	Accuracy	Avg Num Tool Call
Qwen2.5-1.5B-Instruct (**Raw**)	20.8%	0.61
Qwen2.5-1.5B-Instruct (**SFT400**)	24.8%	0.78
Qwen2.5-1.5B-Instruct (**SFT4k**)	23.2%	1.25
Qwen2.5-1.5B-Instruct (**SFT400+PPO**)	36.8%	1.06
Qwen2.5-1.5B-Instruct (**SFT400+GRPO**)	38.4%	0.96
Qwen2.5-1.5B-Instruct (**PPO Cold Start**)	23.2%	2.38
Qwen2.5-1.5B-Instruct (**Ours, GRPO Cold Start**)	**44.0%**	1.19
Qwen2.5-3B-Instruct (**Raw**)	52.0%	1.77
Qwen2.5-3B-Instruct (**SFT400**)	54.4%	0.86
Qwen2.5-3B-Instruct (**SFT4k**)	49.6%	0.92
Qwen2.5-3B-Instruct (**SFT400+PPO**)	43.2%	1.04
Qwen2.5-3B-Instruct (**SFT400+GRPO**)	56.8%	0.99
Qwen2.5-3B-Instruct (**PPO Cold Start**)	40.0%	1.14
Qwen2.5-3B-Instruct (**Ours, GRPO Cold Start**)	**60.0%**	1.32
Qwen2.5-7B-Instruct (**Raw**)	69.6%	1.42
Qwen2.5-7B-Instruct (**SFT400**)	28.8%	3.71
Qwen2.5-7B-Instruct (**SFT4k**)	30.4%	1.06
Qwen2.5-7B-Instruct (**SFT400+PPO**)	45.6%	3.54
Qwen2.5-7B-Instruct (**SFT400+GRPO**)	29.6%	3.70
Qwen2.5-7B-Instruct (**PPO Cold Start**)	48.0%	1.25
Qwen2.5-7B-Instruct (**Ours, GRPO Cold Start**)	**72.0%**	1.63
Llama-3.2-3B-Instruct (**Raw**)	34.4%	1.25
Llama-3.2-3B-Instruct (**SFT400**)	44.0%	0.98
Llama-3.2-3B-Instruct (**SFT4k**)	48.8%	0.98
Llama-3.2-3B-Instruct (**SFT400+PPO**)	39.2%	1.33
Llama-3.2-3B-Instruct (**SFT400+GRPO**)	45.6%	1.00
Llama-3.2-3B-Instruct (**PPO Cold Start**)	29.6%	1.42
Llama-3.2-3B-Instruct (**Ours, GRPO Cold Start**)	**52.0%**	0.89

- **Free-form QA (Bamboogle):**
  Achieves high accuracy without excessive tool calls, demonstrating **effective and efficient tool use** when needed



(a) Unfamiliar Scenario



(b) Unfamiliar Goal

- **Unfamiliar Scenarios/Goals:**
  ToolRL generalizes well to unseen **programming languages** and novel task goals (**relevance detection**)

Agentic Behavior Analysis

- **Qualitative Examples:**
  - **Proactive Rejection:** Correctly identifies and rejects irrelevant
  - **Clarification:** Asks for missing information instead of hallucinating or misusing tools
  - **Metacognition:** Shows signs of understanding tool capabilities and limitations

---

User: I would like to buy a movie ticket in San Jose at 11 o'clock in the night.
Tools Available:
1. `Movies_1_BuyMovieTickets(movie_name, number_of_tickets, ...)`
→ Buy tickets for a movie at a specific time and location.
2. `Movies_1_FindMovies(location, ...)`
→ Search movies by location, genre, or format.
3. `Movies_1_GetTimesForMovie(movie_name, location, ...)`
→ Get showtimes for a movie on a specific date and location.

<think> The user wants to buy a movie ticket in San Jose at 11 PM. The available tool, `Movies_1_BuyMovieTickets`, allows us to purchase movie tickets with the required details such as movie name, number of tickets, show date, location, show time, and show type. We need to find the movie name and the show date. However, the user did not specify the movie name or the show date. We can infer the show date from the current date, but we need to know the movie name to proceed with purchase. </think>

<response> To assist in purchasing a movie ticket, I need the name of the movie and the specific date of the show. Could you please provide this information? </response>

User: What is the slope of the line which is perpendicular to the line with the equation y = 3x + 2?
Tools Available:
1. `find_critical_points(function, variable, range=[0.0, 3.4])`
→ Find critical points of a function with respect to a variable, optionally within a range.

<think> The problem is asking for the slope of a line that is perpendicular to the line with the equation y = 3x + 2. The slope of the given line is 3. The slope of a line that is perpendicular to another line is the negative reciprocal of the original line's slope. So, the slope of the line perpendicular to y = 3x + 2 would be -1/3. The given tool `find_critical_points` is not relevant to finding the slope of a line and cannot be used to solve this problem. </think>

<response> The slope of the line that is perpendicular to the line with the equation y = 3x + 2 is -1/3. </response>

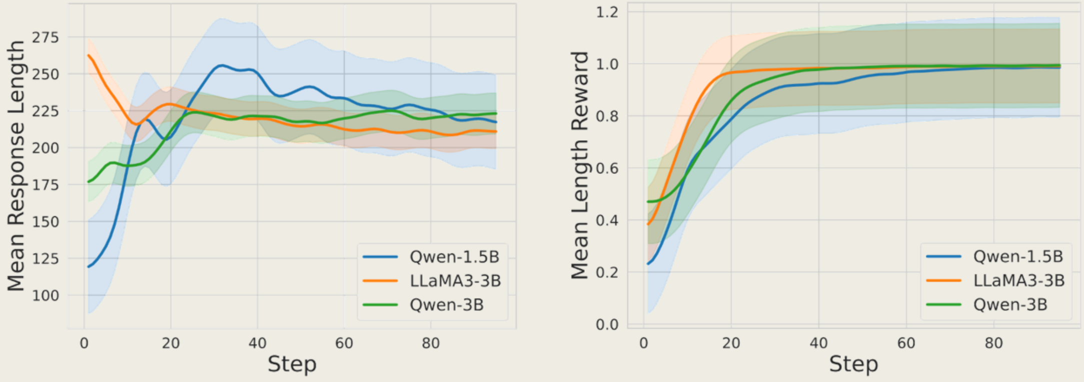
Deep Reward Design Analysis

- To understand *why* our proposed reward design is effective, we perform ablation studies by varying different aspects of the reward:
- Key Dimensions Investigated:
  - **Length Reward:**
    - Does encouraging **longer reasoning** (<think> block) help?
  - **Reward Scale & Dynamics:**
    - How important is the **relative weighting** between Format and Correctness, and should this weighting change over time?
  - **Reward Granularity:**
    - How **detailed** does the Correctness reward need to be (evaluating tool name, parameter names, parameter values separately vs. combined)?

Reward Design Analysis: Length

- **Setup:** Introduced an additional reward component proportional to the length of the model's <think> block (**w/ Length Reward**).
- Gradually increase this length reward w.r.t. training step (**Dynamic**)

Model	Overall Acc
Qwen2.5-1.5B-Instruct (**Original**)	**46.20%**
Qwen2.5-1.5B-Instruct (**w/ Length Reward**)	33.23%
Qwen2.5-1.5B-Instruct (**Dynamic**)	28.51%
Qwen2.5-3B-Instruct (**Original**)	**52.98%**
Qwen2.5-3B-Instruct (**w/ Length reward**)	48.89%
Qwen2.5-3B-Instruct (**Dynamic**)	48.24%
Llama-3.2-3B-Instruct (**Original**)	44.10%
Llama-3.2-3B-Instruct (**w/ Length reward**)	**44.98%**
Llama-3.2-3B-Instruct (**Dynamic**)	43.15%



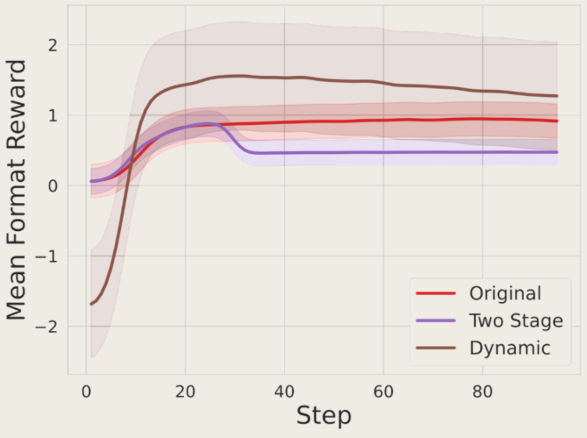
(a) Response Length (b) Length Reward

Takeaway 1: While length rewards **encourage longer reasoning traces**, they do not consistently improve task performance and may even harm it in smaller models, highlighting that **longer reasoning is not inherently better** for tool use tasks.

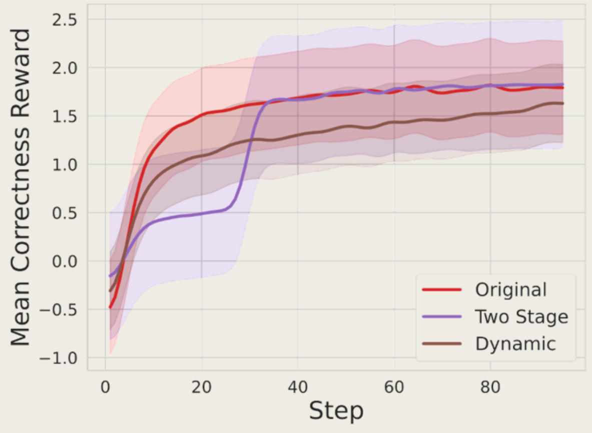
Reward Design Analysis: Scale

- **Setup:** Equalizing the max reward for Format & Correctness (**Equal Max**)
- Switch Format & Correctness max reward abruptly at threshold step (**Two Stage**)
- Gradually change Format & Correctness max reward (**Dynamic**)

Model	Overall Acc
Qwen2.5-1.5B-Instruct (**Original**)	**46.20%**
Qwen2.5-1.5B-Instruct (**Equal max**)	39.47%
Qwen2.5-1.5B-Instruct (**Two stage**)	38.85%
Qwen2.5-1.5B-Instruct (**Dynamic**)	45.71%
Qwen2.5-3B-Instruct (**Original**)	52.98%
Qwen2.5-3B-Instruct (**Equal max**)	51.76%
Qwen2.5-3B-Instruct (**Two stage**)	50.66%
Qwen2.5-3B-Instruct (**Dynamic**)	**53.81%**
Llama-3.2-3B-Instruct (**Original**)	44.10%
Llama-3.2-3B-Instruct (**Equal max**)	42.47%
Llama-3.2-3B-Instruct (**Two stage**)	41.33%
Llama-3.2-3B-Instruct (**Dynamic**)	**46.85%**



(a) Format Reward



(b) Correctness Reward

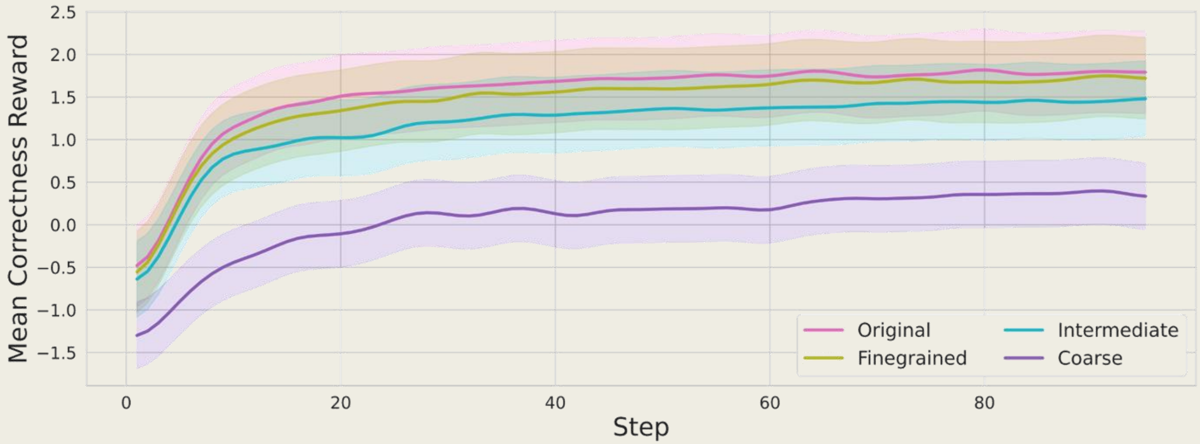
Takeaway 2: Gradually adjusting reward scales during training (starting with format, then smoothly to correctness) better supports learning and generalization than static scales or abrupt changes.

Reward Design Analysis: Granularity

➢ **Setup:** Compared our original reward, we further design three versions:

- Make all name matching no partial rewards (**Finegrained**)
- Combined parameter name + value matching (**Intermediate**)
- Required an exact match for the entire set of tool calls (**Coarse**)

Model	Overall Acc
Qwen2.5-1.5B-Instruct (**Original**)	**46.20%**
Qwen2.5-1.5B-Instruct (**Finegrained**)	40.71%
Qwen2.5-1.5B-Instruct (**Intermediate**)	37.65%
Qwen2.5-1.5B-Instruct (**Coarse**)	36.72%
Qwen2.5-3B-Instruct (**Original**)	**52.98%**
Qwen2.5-3B-Instruct (**Finegrained**)	52.06%
Qwen2.5-3B-Instruct (**Intermediate**)	51.36%
Qwen2.5-3B-Instruct (**Coarse**)	51.40%
Llama-3.2-3B-Instruct (**Original**)	**44.10%**
Llama-3.2-3B-Instruct (**Finegrained**)	39.82%
Llama-3.2-3B-Instruct (**Intermediate**)	38.62%
Llama-3.2-3B-Instruct (**Coarse**)	35.95%



Takeaway 3: Finegrained reward decomposition provides richer learning signals, highlighting its role in **enabling more effective training** compared to coarse reward formulations, which can impede progress and degrade final performance.

Reward Is All Tool Learning Needs

- **Problem:** **SFT struggles** with robust, generalizable tool use for LLMs.
- **Solution:** ToolRL framework using RL (GRPO) with a principled, fine-grained reward design focusing on both **format and decomposed correctness**.
- **Key Insight:** **Thoughtful reward engineering** (granularity, scaling, type) is paramount for unlocking the potential of RL for complex LLM tool learning.
- **Impact:** Provides the first systematic study and a practical roadmap for designing **effective rewards** for training **capable LLM agents**.

Reward Is All Tool Learning Needs

- **Arxiv Paper Link:**
  - https://arxiv.org/pdf/2504.13958

- **Github Link:**
  - https://github.com/qiancheng0/ToolRL

Paper **Code**

arXiv:2504.13958v1 [cs.LG] 16 Apr 2025

ToolRL: Reward is All Tool Learning Needs

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, Heng Ji
University of Illinois Urbana-Champaign
{chengq9, hengji}@illinois.edu

Abstract

Current Large Language Models (LLMs) often undergo supervised fine-tuning (SFT) to acquire tool use capabilities. However, SFT struggles to generalize to unfamiliar or complex tool use scenarios. Recent advancements in reinforcement learning (RL), particularly with R1-like models, have demonstrated promising reasoning and generalization abilities. Yet, reward design for tool use presents unique challenges: multiple tools may be invoked with diverse parameters, and coarse-grained reward signals, such as answer matching, fail to offer the fine-grained feedback required for effective learning. In this work, we present the first comprehensive study on reward design for tool selection and application tasks within the RL paradigm. We systematically explore a wide range of reward strategies, analyzing their types, scales, granularity, and temporal dynamics. Building on these insights, we propose a principled reward design tailored for tool use tasks and apply it to train LLMs using Group Relative Policy Optimization (GRPO). Empirical evaluations across diverse benchmarks demonstrate that our approach yields robust, scalable, and stable training, achieving a 17% improvement over base models and a 15% gain over SFT models. These results highlight the critical role of thoughtful reward design in enhancing the tool use capabilities and generalization performance of LLMs. All the code are released to facilitate future research.[1]

1 Introduction

Recent advances in Large Language Models (LLMs) have showcased remarkable capabilities in complex reasoning tasks (Kumar et al., 2025). Among the techniques that have significantly contributed to this progress, Reinforcement Learning (RL) has emerged as a powerful paradigm, enabling LLMs to develop emergent capabilities such as self-reflection, self-correction, and long-horizon planning (Guo et al., 2025; Team et al., 2025). These capabilities have been instrumental in the success of models like o1 and R1, particularly in mathematical and logical reasoning domains (Qin et al., 2024a; Huang et al., 2024; Li et al., 2025b; Kang et al., 2025).

Beyond traditional reasoning tasks, an increasingly important area is **Tool-Integrated Reasoning** (TIR). TIR involves LLMs interacting with external tools, such as search engines (Jin et al., 2025; Zheng et al., 2025), calculators (Chen et al., 2023b; Qin et al., 2023), or code interpreters (Gou et al., 2023; Liao et al., 2024), in a multi-step, feedback-driven loop to arrive at solutions. TIR is particularly important because it addresses core limitations of LLMs, such as outdated knowledge, calculation inaccuracy, and shallow reasoning. By integrating external tools that offer real-time access and specialized capabilities, TIR enables models to tackle complex tasks in a more grounded and goal-directed way.

Unlike textual reasoning, which primarily involves deduction and inference from static text,

Figure 1: SFT on distilled deep-thinking trajectories suffers from overthinking and limited generalization.

[1] Data and codes released at https://github.com/qiancheng0/ToolRL

Thanks for Listening