# SLCA-GRPO: RESOLVING CROSS-SEGMENT CREDIT MISATTRIBUTION IN TOOL-CALLING RL

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

## ABSTRACT

Tool-calling agents produce heterogeneous outputs, interleaving structured tool invocations with user-facing natural language summaries. This output heterogeneity presents a structural failure mode in standard on-policy Reinforcement Learning (RL): algorithms like GRPO indiscriminately broadcast a homogeneous trajectory-level scalar advantage to all tokens. Consequently, gradient noise from summary generation leaks into tool-decision tokens, causing cross-segment credit misattribution and brittle optimization. In this work, we propose **SLCA-GRPO**, a framework incorporating Segment-Locked Credit Assignment (SLCA). To enable scalable exploration without costly real APIs and stable training, we first construct the Schema-Guided LLM Simulator (SGLS) as foundational training infrastructure. Building on this, SLCA decouples advantage estimation at the *structural segment level* within a single group of rollouts, without requiring additional rollouts from intermediate states. Supported by Hierarchical Rewards (HierR), SLCA routes execution advantages to tool tokens and preference advantages to summary tokens, eliminating advantage contamination (the dominant cross-segment credit misattribution channel) within each policy update. On a 7B backbone, SLCA-GRPO accelerates convergence and outperforms standard GRPO, ToolPO, and RLTR by +2.53 pp on in-domain evaluation, +1.36 pp on the Berkeley Function-Calling Leaderboard (BFCL), and +9.15 pp on $\tau^2$-Bench under the same training budgets, achieving higher accuracy with reduced tool redundancy and costs.

## 1 INTRODUCTION

The evolution of Large Language Models (LLMs) from passive chatbots to active agents hinges on their ability to plan and execute actions via external tools (Qin et al., 2024; Schick et al., 2023; Yao et al., 2023; Patil et al., 2024). Although Supervised Fine-Tuning (SFT) establishes a strong initialization, it depends on imitating fixed patterns and is fragile against the large, changing namespaces of ecosystems such as the Model Context Protocol (MCP) (Hou et al., 2026). To transcend mimicry, post-training via on-policy Reinforcement Learning (RL) is essential (Ouyang et al., 2022; Schulman et al., 2017; Yu et al., 2025). Tool-calling resembles reasoning tasks in that trajectories can contain intermediate work before a final answer, but it adds externally verifiable boundaries: structured API calls, environment-injected observations, and schema constraints. These boundaries naturally decompose a rollout into $y = \left[y_{\text{tool}} \oplus y_{\text{sum}}\right]$, where $y_{\text{tool}}$ encodes the operational trajectory (reasoning traces interleaved with structured tool calls), and $y_{\text{sum}}$ constitutes the user-facing natural language response.

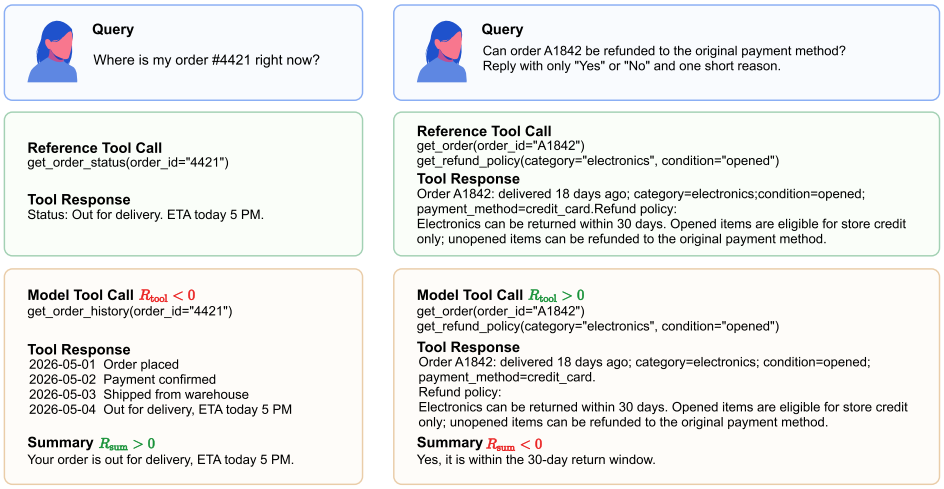
Applying standard RL algorithms to this two-segment structure reveals a fundamental mismatch. Dominant approaches such as Group Relative Policy Optimization (GRPO) (Guo et al., 2025a) broadcast a unified trajectory-level advantage to all tokens; even Process Reward Models (PRMs) aggregate signals ($R_{\text{total}} = R_{\text{tool}} + R_{\text{sum}}$) before advantage estimation. We argue that this **Global Signal Conflation** is a structural failure mode: summary-reward variation can enter the tool-token advantage, producing **Cross-Segment Credit Misattribution**. In sign-conflict regimes, a failed tool call can be reinforced when followed by a correct summary, while a correct tool trajectory can be penalized when followed by an incorrect summary. Existing mitigations do not close this pathway: temporal methods (VinePPO (Kazemnejad et al., 2025), GiGPO (Feng et al., 2025), SPO (Guo et al., 2025b)) address step-wise credit while retaining $R_{\text{total}}$; ToolPO (Li et al., 2026b) adds local tool



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

rewards but still lets summary-dependent noise reach tool tokens; and RLTR (Li et al., 2025) separates planner and summarizer, abandoning a unified backbone.

We verify this failure through diagnostic cases: standard GRPO assigns a single aggregated advantage to both tool and summary tokens, so a correct summary can mask an inefficient or wrong tool trajectory, and an incorrect summary can suppress otherwise correct tool use. Gradient diagnostics show instability spikes while tool/summary gradient directions remain near-orthogonal, consistent with advantage contamination as a source of the observed instability (App. B). Two questions therefore arise: *(1)* can cross-segment advantage contamination be structurally eliminated within a single unified policy, and *(2)* does eliminating it translate into measurable gains beyond what additive augmentation delivers?




**Diagnostic cases.** Unified reward aggregation can misassign credit across tool and summary segments. Left: an unnecessary tool call is paired with a correct delivery summary. Right: correct tool evidence is paired with an incorrect refund summary. Under standard GRPO, the mixed advantage can reward bad tool use or penalize good tool use.

To resolve this, we propose **SLCA-GRPO** (Figure 1). Its core, Segment-Locked Credit Assignment (SLCA), routes tool-side advantages ($\hat{A}_{\text{tool}}$) exclusively to $y_{\text{tool}}$ tokens and summary-side advantages ($\hat{A}_{\text{sum}}$) exclusively to $y_{\text{sum}}$ tokens, blocking the defined summary-to-tool advantage path *before* the backward pass, at zero additional rollout cost and within a single unified policy (§3.3). This routing is a deliberate bias–variance trade-off: SLCA does not claim that tool calls are causally irrelevant to final answers; rather, it assumes that a dense execution reward is the lower-variance and sufficiently informative signal for updating tool-decision tokens, while summary rewards train the articulation segment. Crucially, SLCA operates on a *structural* axis (execution vs. articulation) that is orthogonal to the *temporal* axis addressed by VinePPO / SPO / GiGPO: it therefore *composes with* rather than replaces these methods, and can be combined with temporal credit-assignment within each segment. A Schema-Guided LLM Simulator (SGLS, §3.4) and Hierarchical Rewards (HierR, §3.2) supply the infrastructure that makes this routing practical at scale.

Across three backbones (Qwen2.5-3B/7B-Instruct, Qwen3-8B-Base) and three benchmarks, SLCA-GRPO has higher reported means than matched GRPO on the main comparisons. On the 7B backbone, the matched mean gaps are +2.53 pp on in-domain Toucan, +1.36 pp on BFCL, and +9.15 pp on $\tau^2$-Bench. The corresponding Toucan gaps are +2.35 pp (3B) and +2.05 pp (8B); complete multi-run results and method-specific baseline protocols are reported in the experiments and appendix. Our contributions are threefold:

- We identify **Cross-Segment Credit Misattribution** as a structural failure mode of on-policy RL for tool-calling agents, arising from advantage contamination rather than gradient-direction conflict.
- We propose **SLCA-GRPO**, an intra-trajectory estimator that separately normalizes and routes segment advantages without additional rollouts, complemented by the SGLS simulator and HierR rewards.

- We evaluate SLCA-GRPO across three scales and three benchmarks, with ablations (w/o SLCA / SGLS / HierR) and a reward-protocol sensitivity experiment.

## 2 Related Work

### 2.1 Tool-Calling Post-Training

Tool-calling agents are commonly initialized with SFT on annotated tool-use trajectories (Schick et al., 2023; Qin et al., 2024), often followed by preference- or verification-driven optimization (Ouyang et al., 2022; Christiano et al., 2017; Stiennon et al., 2020; Rafailov et al., 2023). However, SFT can be brittle under rule variants and OOD shifts due to memorization and exposure bias (Chu et al., 2025; Bengio et al., 2015; Ross et al., 2011; Wei et al., 2025b; Lightman et al., 2024; Guo et al., 2025a). As tool namespaces scale, executable benchmarks such as ToolBench, API-Bank, and BFCL make post-training robustness increasingly central (Qin et al., 2024; Li et al., 2023; Patil et al., 2025). We target this post-training setting, where tool trajectories and final summaries form heterogeneous segments but standard end-to-end objectives still broadcast a single advantage to all tokens. Scalable tool-agent training also relies on simulated or emulated tool environments, since live API interaction can be costly, unstable, or unavailable at RL scale (Guo et al., 2024; Ruan et al., 2024; Li et al., 2026b). Our SGLS follows this line but uses schemas as a control plane so that the simulator supplies segment-specific feedback for credit-assignment analysis.

### 2.2 Credit Assignment for Agentic RL

On-policy RL for tool use inherits credit-assignment issues from sparse trajectory-level optimization (Williams, 1992; Schulman et al., 2015; 2017; Guo et al., 2025a). Process supervision, PRMs, and reward-centric tool RL provide denser feedback (Lightman et al., 2024; Uesato et al., 2022; Qian et al., 2025; Lin et al., 2025), but dense rewards do not by themselves prevent aggregation before advantage estimation. Existing credit-assignment methods mainly operate along a *temporal* axis (Kazemnejad et al., 2025; Feng et al., 2025; Guo et al., 2025b) or use token-level refinements, while ToolPO (Li et al., 2026b) adds local tool rewards and RLTR (Li et al., 2025) separates planner and summarizer into a pipeline. SLCA instead decouples advantage estimation along a *structural* axis within the same sampled rollout group: it routes independently normalized tool and summary advantages to their corresponding token segments without intermediate-state rollouts. This structural decomposition targets a different axis from temporal credit assignment; we do not evaluate a combined method here.

## 3 Methodology

**Core Design** SLCA-GRPO turns the diagnosis above into three design requirements: expose structural token segments, obtain segment-specific feedback, and block cross-segment advantages before optimization. As summarized in Figure 1, the framework implements these requirements through mask-based segmentation, HierR segment returns, SLCA routing, SGLS rollouts, and a unified PPO-style objective, in contrast to post-hoc gradient-projection methods (Yu et al., 2020); a schematic comparison is provided in App. 4.

### 3.1 Problem Formulation & Segment Decomposition

**Heterogeneous Rollouts** To lock credit to the right semantic target, the structural boundary must first be made explicit. Given an instruction $x \sim \mathcal{D}$, the policy $\pi_\theta$ generates a trajectory $y = (y_1, \ldots, y_T)$ that decomposes as $y = [y_{\text{tool}} \oplus y_{\text{sum}}]$, where $y_{\text{tool}}$ comprises reasoning traces and structured tool calls, and $y_{\text{sum}}$ is the final user-facing response.

**Gradient Masking** To align optimization with this structure, a binary mask $m_{i,t} \in \{0, 1\}$ identifies *learnable policy tokens* ($m_{i,t} = 1$) versus *frozen environment contexts* ($m_{i,t} = 0$). This mask restricts gradient propagation to agent actions and provides the structural signal for our automatic segment decomposition.

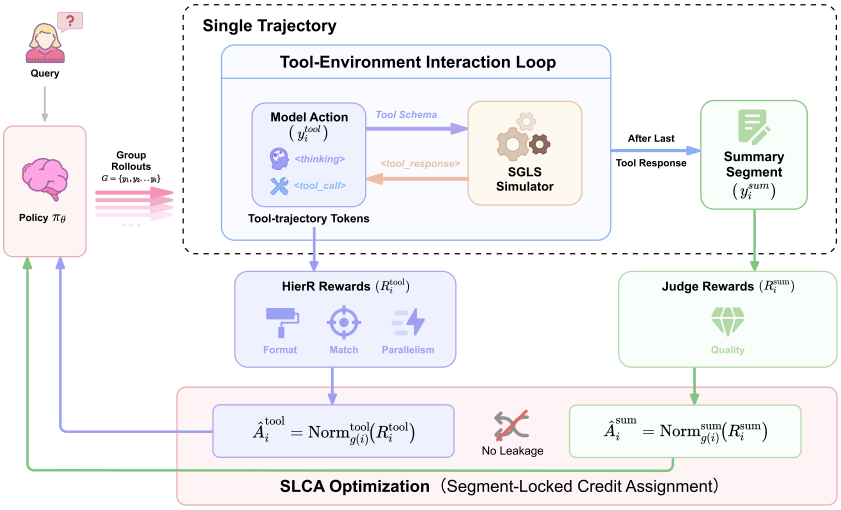



Figure 1: Overview of the SLCA-GRPO Framework. (Top) Infrastructure: SGLS enables scalable exploration via schema-constrained simulation, generating interleaved trajectories. (Middle) Signal: HierR decouples feedback into dense execution rewards ($R^{\text{tool}}$) and terminal outcome rewards ($R^{\text{sum}}$). (Bottom) Optimization: SLCA decouples segment-wise advantages. By normalizing and routing advantages independently ($\hat{A}^{\text{tool}}$ vs. $\hat{A}^{\text{sum}}$), it blocks the defined summary-to-tool support path within each policy update.

**Automatic Segment Decomposition** This mask partitions learnable tokens into disjoint semantic sets without training auxiliary segmenters. For a rollout $i$, the **Summary Segment** $\mathcal{T}_{i,\text{sum}}$ is the *final contiguous run* of learnable tokens. The **Tool Segment** $\mathcal{T}_{i,\text{tool}}$ is the union of *all* preceding learnable runs, capturing the entire reasoning-action chain including any intermediate thinking blocks between tool calls. This deterministic mapping ensures that every gradient-bearing token is uniquely assigned to a semantic role, as long as the trajectory culminates in a single final-answer block (see App. A.1 for edge cases). **Scope.** SLCA targets the *structural* axis (execution vs. articulation) rather than the *temporal* axis within the tool segment (e.g., Action 1 vs. Action 2 in a ReAct chain). Thus, SLCA is not a variant of VinePPO/GiGPO/SPO, which address temporal credit assignment under different grouping assumptions (§2).

### 3.2 HierR and Segment Returns

To make segment-locked routing meaningful, the reward must also separate execution quality from response articulation. We decompose the return into two semantically distinct components: $R(y) = R_{\text{tool}}(y_{\text{tool}}) + R_{\text{sum}}(y_{\text{sum}} \mid y_{\text{tool}}, o)$. **HierR** instantiates these signals as: (i) Process Reward: A dense, structure-aware score for tool correctness and efficiency. (ii) Summary Preference: A terminal score evaluating the final answer quality.

For the $i$-th sampled rollout $y_i$, the segment rewards are computed by applying these functions directly:

$$R_i^{\text{tool}} = S_{\text{process}}(y_{i,\text{tool}}), \quad R_i^{\text{sum}} = S_{\text{summary}}(y_{i,\text{sum}} \mid y_{i,\text{tool}}, o_i). \tag{1}$$

Full reward definitions and matching logic are detailed in App. A.4. Together, SGLS and HierR build on prior tool-use simulation and process-supervision work, but serve a specific role here: providing scalable, segment-specific feedback for testing structural credit routing.

### 3.3 SLCA-GRPO

Given segment boundaries and segment returns, SLCA directly blocks the contamination channel identified in the introduction. Standard GRPO broadcasts a single trajectory-level advantage derived from $R_{\text{total}}$ to all learnable tokens, coupling tool-token gradients with summary rewards.

**Decoupled Advantage Estimation** For each rollout $i$, the two segment rewards $(R_i^{\text{tool}}, R_i^{\text{sum}})$ from Eq. 1 are normalized *separately* within the group:

$$\hat{A}_i^{\text{tool}} = \frac{R_i^{\text{tool}} - \mu_g^{\text{tool}}}{\sigma_g^{\text{tool}} + \epsilon_{\text{norm}}}, \quad \hat{A}_i^{\text{sum}} = \frac{R_i^{\text{sum}} - \mu_g^{\text{sum}}}{\sigma_g^{\text{sum}} + \epsilon_{\text{norm}}}, \tag{2}$$

where $\mu_g$ and $\sigma_g$ denote the mean and standard deviation within the group, and $\epsilon_{\text{norm}}$ is a numerical floor. The token-level advantage $\hat{A}_{i,t}^{\text{SLCA}}$ is then constructed by routing these signals exclusively to their semantic counterparts:

$$\hat{A}_{i,t}^{\text{SLCA}} = \begin{cases} \lambda_{\text{tool}}\, \hat{A}_i^{\text{tool}}, & t \in \mathcal{T}_{i,\text{tool}}, \\ \lambda_{\text{sum}}\, \hat{A}_i^{\text{sum}}, & t \in \mathcal{T}_{i,\text{sum}}, \\ 0, & m_{i,t} = 0, \end{cases} \tag{3}$$

This construction changes only the scalar advantage attached to each token; it does not split the model or require additional rollouts. Tool tokens are updated only by execution quality, while summary tokens are updated only by response quality. Thus, summary rewards can still train final-answer articulation, but no longer supply gradients to tool-decision tokens.

**Robust Optimization Mechanisms** For sparse-tool stability, the implementation uses presence filtering, post-normalization weighting, and omission-penalty routing; details are in App. A.3.

**Theoretical Properties** The key per-update guarantee of SLCA is advantage isolation: the tool-token gradient component is functionally independent of the summary reward,

$$\frac{\partial g_{\text{tool}}^{\text{SLCA}}}{\partial R^{\text{sum}}} = \mathbf{0}. \tag{4}$$

Beyond this isolation property, local score-function analysis shows that SLCA removes a positive summary-noise variance term from tool updates and preserves the tool-execution direction in sign-conflict regimes. These are conditional, per-update guarantees rather than claims of global bias-free optimization; full assumptions and proofs are in App. A.2.

### 3.4 Scalable Exploration via SGLS

To test structural credit routing at scale, SGLS supplies schema-consistent observations without relying on live APIs. It combines deterministic schema validation with frozen cross-family LLM mocking, so invalid calls receive immediate structured feedback and valid calls receive plausible tool responses; deployment details and alignment examples are in App. C.3. The cross-family design mitigates implicit leakage while preserving the semantic topology needed for transfer.

### 3.5 Objective and Training Algorithm

Finally, a single unified policy is trained by replacing GRPO's unified scalar advantage with the routed segment advantage. The objective is PPO-clip (Schulman et al., 2017) with segment-locked advantages. Let $\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|h_{i,t})}{\pi_{\text{old}}(y_{i,t}|h_{i,t})}$; a per-token KL penalty $\beta\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})$ is enforced but omitted below for brevity:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{N}\sum_{i,t} m_{i,t}\cdot\min\left(\rho_{i,t}\hat{A}_{i,t}^{\text{SLCA}}, \text{clip}(\rho_{i,t}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\hat{A}_{i,t}^{\text{SLCA}}\right)\right], \tag{5}$$

where $G$ is the rollout group size and $N = \sum_{i,t} m_{i,t}$ is the total valid token count. The complete training procedure is summarized in Algorithm 1.

## 4 Experiments

Our experiments proceed in three parts: we examine the diagnosed failure under unified or additive credit signals, test transfer across schemas and long-horizon interaction, and ablate SLCA, SGLS, and HierR.

### 4.1 EXPERIMENTAL SETUP

**Datasets and Benchmarks** SLCA-GRPO is evaluated across three dimensions: in-domain mastery, cross-distribution generalization, and collaborative robustness. (i) **Toucan-1.5M (In-Domain):** The Toucan dataset (Xu et al., 2025) is used for both SFT initialization and RL post-training. A multi-stage filtering pipeline (App. C.1) yields 42,423 SFT samples and 31,818 RL training samples (spanning single-turn and decomposed multi-turn trajectories). Evaluation uses a held-out Toucan-Test set of 4,000 samples. (ii) **BFCL V3 (Generalization):** Generalization is evaluated on the Berkeley Function-Calling Leaderboard (Patil et al., 2025). [1] (iii) $\tau^2$**-Bench (Robustness):** Robustness is measured in dynamic dual-control environments (Airline, Retail, Telecom) where agents must collaborate with users to manipulate state, reporting Pass[1] (Barres et al., 2025). Full protocols are in App. C.5.

**Baselines and Training** The methods are implemented on Qwen2.5-7B-Instruct (Qwen Team, 2025a) (default), Qwen2.5-3B-Instruct, and Qwen3-8B-Base (Qwen Team, 2025b) to verify scalability, comparing six paradigms (details in App. C.2): (i) Original Backbones: The original model weights evaluated directly without any exposure to the Toucan training set. (ii) SFT (Toucan): A behavioral cloning baseline fine-tuned on the union of the standard SFT partition (42k) and the raw source trajectories of the RL partition (31k). (iii) SFT+GRPO (Baseline): The standard GRPO algorithm using the same HierR rewards and SGLS environment as SLCA-GRPO, with a unified advantage computed from $R_{\text{total}} = R_{\text{tool}} + R_{\text{sum}}$. (iv) ToolPO (Li et al., 2026b): An additive credit-assignment baseline evaluated with its global outcome and local tool rewards. Tool tokens receive their sum, while summary tokens inherit the global term. The outcome-reward protocol and its sensitivity diagnostic are documented in App. C.4. (v) RLTR (adapted) (Li et al., 2025): A planner-summarizer baseline with a dedicated planner-only SFT initialization and a frozen summarizer, following a two-stage planning pipeline. (vi) SFT+SLCA-GRPO (Ours): Our proposed framework employing Segment-Locked Credit Assignment. The controlled isolation comparison is between SFT+GRPO and SFT+SLCA-GRPO: these two rows share the same data partitions, SFT initialization, SGLS endpoint, mocker configuration, decoding settings, evaluation protocol, rollout group size $G = 16$, one RL epoch, and HierR rewards, differing in unified versus segment-locked advantage computation. The matched ablations use the same per-backbone run set; w/o HierR removes HierR, and w/o SGLS removes schema conditioning and deterministic validation. ToolPO uses the LLM-judge outcome reward in the main tables, while RLTR retains its method-specific protocol; the reference-rule ToolPO sensitivity diagnostic is reported in App. C.4.

### 4.2 MAIN RESULTS

**Toucan-Test** The in-domain evaluation first asks whether structural routing improves execution under matched data, simulator, and reward conditions. Table 1 reports the matched GRPO comparison together with the method-specific baseline rows (full results in App. D). On Qwen2.5-7B-Instruct, SLCA-GRPO has a 79.13% three-run mean for strict Success@0.9, 2.53 pp above matched GRPO. The corresponding mean gaps are +2.35 pp on Qwen2.5-3B-Instruct and +2.05 pp on Qwen3-8B-Base.

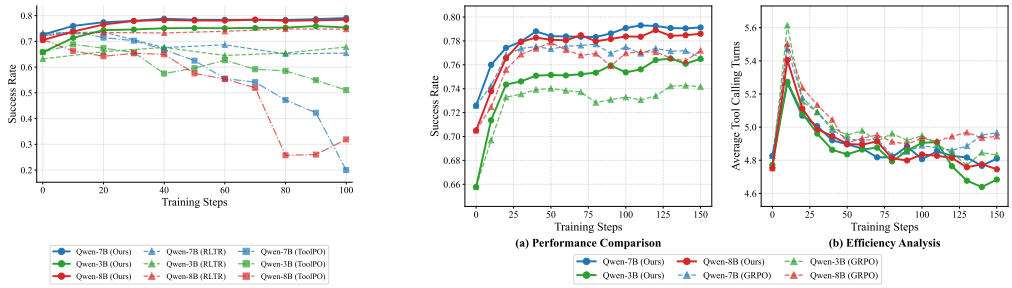
[1] For the main BFCL V3 comparison, relevance detection is excluded and evaluation is restricted to **single-turn** samples. This isolates *atomic* generalization to unseen schemas (e.g., AST); the appendix additionally reports BFCL Multi-Turn accuracy, while $\tau^2$-Bench covers longer-horizon collaboration.

Table 1: Toucan-Test Results. Metrics: Name F1: Multiset F1 score measuring precision and recall of predicted tool-name occurrences against gold names; ArgMatch: Arithmetic mean of argument key and value matching scores; Process: The dense tool-segment reward ($S_{\text{process}}$) defined in Eq. 22 (App. A.4); Success: Strict binary indicator ($\mathbb{I}[S_{\text{process}} \geq 0.9]$) obtained by thresholding the process score. All trained entries report mean±std over three runs; Original rows are point evaluations. The matched rows share the per-backbone SFT initialization, data split, SGLS endpoint, mocker configuration, decoding settings, evaluation protocol, $G = 16$, one RL epoch, and run set. GRPO, SLCA-GRPO, w/o SLCA, and w/o SGLS also share the HierR definition and subweights; w/o HierR removes HierR. ToolPO and RLTR retain their method-specific protocols. The ToolPO rows use the LLM-judge outcome reward; App. C.4 reports how this baseline responds to a rule-based outcome reward under the same estimator.

| Method | Name F1 | ArgMatch | Process | Success |
|---|---|---|---|---|
| ***Qwen2.5-3B-Instruct*** | | | | |
| Original | 0.5207 | 0.5073 | 0.5174 | 0.3816 |
| SFT | 0.7685±.0069 | 0.7132±.0239 | 0.7401±.0050 | 0.6500±.0089 |
| SFT+GRPO | 0.8910±.0087 | 0.8072±.0154 | 0.8577±.0057 | 0.7412±.0115 |
| RLTR | 0.7598±.0149 | 0.6353±.0229 | 0.6924±.0162 | 0.6627±.0151 |
| ToolPO | 0.8050±.0142 | 0.7769±.0160 | 0.8368±.0141 | 0.5128±.0145 |
| **OURS** | **0.9006**±.0087 | **0.8092**±.0112 | **0.8625**±.0049 | **0.7647**±.0093 |
| ***Qwen2.5-7B-Instruct*** | | | | |
| Original | 0.8008 | 0.7388 | 0.7687 | 0.6224 |
| SFT | 0.8390±.0075 | 0.7531±.0224 | 0.8094±.0082 | 0.7214±.0087 |
| SFT+GRPO | 0.8971±.0093 | 0.8350±.0148 | 0.8667±.0061 | 0.7660±.0127 |
| RLTR | 0.7673±.0132 | 0.6346±.0205 | 0.7022±.0143 | 0.6545±.0168 |
| ToolPO | 0.3042±.0168 | 0.3552±.0192 | 0.3408±.0157 | 0.2000±.0136 |
| **OURS** | **0.9164**±.0075 | **0.8353**±.0132 | **0.8766**±.0045 | **0.7913**±.0105 |
| ***Qwen3-8B-Base*** | | | | |
| Original | 0.0887 | 0.1158 | 0.1089 | 0.0471 |
| SFT | 0.8058±.0082 | 0.7437±.0260 | 0.7754±.0060 | 0.6925±.0098 |
| SFT+GRPO | 0.9110±.0092 | 0.8334±.0135 | 0.8767±.0050 | 0.7697±.0141 |
| RLTR | 0.8780±.0125 | 0.7468±.0219 | 0.7941±.0147 | 0.7454±.0167 |
| ToolPO | 0.6854±.0168 | 0.8180±.0177 | 0.7510±.0165 | 0.3199±.0156 |
| **OURS** | **0.9177**±.0075 | 0.8320±.0140 | **0.8786**±.0046 | **0.7902**±.0105 |

**Comparison with Prior Credit-Assignment Baselines** The next comparison evaluates prior paradigms that either augment or separate the tool-planning signal rather than route segment advantages. RLTR (pipeline separation) and ToolPO (additive augmentation) provide complementary reference points for the segment-locked estimator. Their training traces show capacity-dependent format and invocation patterns (App. D.3); the main-table ToolPO rows use the LLM-judge outcome reward described in App. C.4.

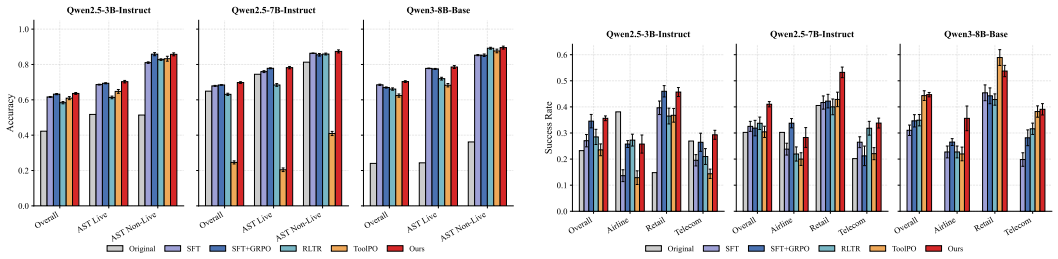
**Training Dynamics and Efficiency** Training traces reveal whether the final gains arise from stable correction of the misattribution channel rather than late-stage overfitting. Figure 2 contrasts training behavior from two views. In the cross-scale training curves (Figure 2a), the plotted traces are representative single runs. SLCA-GRPO rises across the three backbones. The 7B ToolPO curve is a separate diagnostic: it declines after step 40, with the structural divergence appearing after step 80 (App. D.3); RLTR changes more slowly under its coarse completeness reward. Across the representative traces (3B/7B/8B; Figure 2b), SLCA-GRPO ends with *fewer* average tool turns and *higher* success rates than standard GRPO. The gap in trajectory length is narrow, and the ordering is consistent across the plotted traces. Standard GRPO stabilizes at longer trajectories without a corresponding success gain, compatible with "performative execution": padding trajectories to exploit summary rewards rather than improving tool-call precision. These dynamics provide evidence for the failure mode diagnosed in the introduction: additive or unified signals can improve proxy rewards while degrading executable tool behavior.




(a) **Training curves across 3B/7B/8B.** Representative single-run traces.

(b) **Across 3B/7B/8B: success vs. tool turns.** The plotted traces show SLCA-GRPO (solid) ending with shorter trajectories than GRPO (dashed).

Figure 2: **Training Dynamics. (a) Baseline comparison across 3B/7B/8B** for SLCA / ToolPO / RLTR (the ToolPO diagnostic is described in App. D.3), and **(b) cross-scale dynamics** of success rate and average tool turns over training steps.




(a) **BFCL V3 (Single-Turn; mean±std).** Trained bars show three-run means with one-standard-deviation error bars; Original is a point evaluation.

(b) $\tau^2$**-Bench (Overall Pass**[1]**; mean±std).** Trained rows are three-run means with one-standard-deviation error bars; Original is a point evaluation. The 8B Original point is omitted because its four domain scores are zero.

Figure 3: **OOD Generalization across Scales. (a) Atomic generalization on BFCL V3** and **(b) Collaboration robustness on $\tau^2$-Bench** for the three backbones (3B, 7B, 8B), comparing SLCA-GRPO against standard GRPO, RLTR, and ToolPO. Per-domain $\tau^2$-Bench details are in App. D.6.

**Generalization on BFCL and $\tau^2$-Bench** The OOD evaluation then tests whether correcting structural credit assignment transfers beyond the in-domain simulator setting. Figure 3 compares SLCA-GRPO with matched GRPO and the method-specific baselines on two OOD benchmarks across scales (protocols in App. C.5). On BFCL (Figure 3a), SLCA reaches **70.31±0.50%** Overall Accuracy on Qwen3-8B-Base, compared with 66.96±0.32% for matched GRPO and 68.50±0.35% for SFT. The 7B BFCL gap over standard GRPO is +1.36 pp; the corresponding gaps are +0.40 pp on 3B and +3.35 pp on 8B. On $\tau^2$-Bench, the matched gaps are +1.09 pp (3B), +9.15 pp (7B), and +10.03 pp (8B), all based on three-run means. ToolPO outcome-reward details and the LLM-judge protocol used in the scale rows are documented in App. C.4. Multi-Turn BFCL results and per-domain $\tau^2$-Bench breakdowns are in App. D.5 and App. D.6. Robustness under larger candidate tool spaces ($K$ up to 150) with hard-negative distractors is evaluated in App. D.11.

### 4.3 Ablation Studies

Finally, the ablations target the method's three design requirements: routing, scalable simulation, and dense segment feedback. Qwen2.5-7B-Instruct is used as the primary testbed (Table 2), with detailed ablation across scales in App. D.10.

(i) **w/o SLCA (Unified Advantage)**: Reverting to the standard unified advantage ($R_{\text{tool}} + R_{\text{sum}}$) changes both normalization and token support; it lowers Toucan Success by 2.53 pp on 7B under the matched setting. The support-control experiment in App. D.9 separately measures the contribution

of removing the summary-to-tool support path. (ii) **w/o SGLS:** Removing deterministic schema validation and schema-conditioned tool-response prompting lowers the reported OOD transfer metrics (Table 2); its Multi-Turn effect is reported in App. D.5. Schema-constrained simulation supplies the controlled exploration interface. (iii) **w/o HierR:** Relying solely on sparse summary rewards leads to slow convergence and poor argument alignment on complex queries. The matched "w/o SLCA" condition compares the unified and segment-locked estimators under fixed reward definitions, so it measures the combined estimator change rather than either operation in isolation; the unified reward ratio sweep in App. D.8 examines this distinction directly. Together, these ablations test the three design requirements in Section 3: routing addresses the cross-segment path, SGLS stabilizes schema-grounded exploration, and HierR supplies dense execution feedback. The summary advantage ablation and the 7B support control are reported in Apps. D.2 and D.9.

Table 2: Ablation Studies on Qwen2.5-7B-Instruct. Entries report mean±std over three independent runs. Full results are provided in App. D.10. Process is an evaluation metric computed post hoc for all settings, including w/o HierR.

| | Toucan-Test | | BFCL | $\tau^2$-Bench |
|---|---|---|---|---|
| **Setting** | **Process** | **Success** | **Acc** | **Pass**[1] |
| **SLCA-GRPO** | 0.8766±.0045 | **0.7913**±.0105 | **0.6977**±.0047 | **0.4102**±.0101 |
| w/o SLCA | 0.8667±.0061 | 0.7660±.0127 | 0.6841±.0011 | 0.3187±.0297 |
| w/o SGLS | 0.8787±.0051 | 0.7802±.0112 | 0.6875±.0048 | 0.3619±.0195 |
| w/o HierR | 0.8521±.0083 | 0.7344±.0121 | 0.6907±.0046 | 0.3450±.0144 |

**Reward and response-mocker checks** We also test SLCA with a binary $R_{\text{succ}}$ judge without per-example gold-call matching and with a GPT-OSS-120B response mocker; the matched results are reported in Apps. D.12 and C.3.

## 5 Conclusion

We identified Global Signal Conflation as a structural pathology in tool-calling RL and proposed SLCA-GRPO, which decouples advantage estimation at the segment level. Across three backbones, SLCA-GRPO improves success by **+2.53 pp** on 7B in-domain Toucan, **+1.36 pp** on BFCL, and **+9.15 pp** on $\tau^2$-Bench. The corresponding matched gaps are +2.35/+0.40/+1.09 pp on 3B and +2.05/+3.35/+10.03 pp on 8B for Toucan, BFCL, and $\tau^2$-Bench, respectively. The Toucan breakdown shows higher mean Process and Success for SLCA-GRPO across the reported backbones, while higher Summary scores for matched GRPO on 3B and 8B do not translate into higher task Success. Controlled comparisons and ablations support the role of segment-level routing, and the reward-swap check shows that the improvement is not tied to a single reward formulation. SLCA is complementary to temporal credit-assignment methods.

## 6 Limitations

**Segment decomposition.** SLCA assumes a clear boundary between tool calls and free-form text. Settings without one, such as inline code generation, would require learned segmentation. **Reward and bias.** $S_{\text{process}}$ relies on matching to gold trajectories and may under-reward alternative valid tool sequences when multiple API plans solve the same request. The execution-based judge check in App. D.12 provides a complementary reward formulation. Toucan strict Success@0.9 thresholds this score instead of using an independent task outcome label. $\hat{A}^{\text{sum}}$ inherits a grouping bias in summary tokens (App. A.2); explicit subgrouping would shrink group size exponentially. **Intra-tool temporal credit.** SLCA routes one scalar $\hat{A}^{\text{tool}}$ to every tool token, so it does not distinguish a correct Action 1 from a failing Action 2. This temporal issue is separate from cross-segment credit. A combination with VinePPO/GiGPO/SPO remains untested. **Scale and simulation.** Experiments use models up to 8B with SGLS. Larger-scale training and direct real-API comparisons remain future work.

### AI USE STATEMENT

During manuscript preparation, generative AI tools assisted with language editing, numerical consistency checks, table and figure formatting, and documentation. The authors designed and verified the experiments, supplied and checked the data, and take responsibility for the analyses and final scientific content.

### REPRODUCIBILITY STATEMENT

Model configurations, training budgets, reward protocols, and evaluation procedures are documented in the main paper and appendix. The supplementary material upload includes a README with reproduction instructions and the code/data release information. The supplied package contains the processed splits and points reviewers to the corresponding implementation and evaluation artifacts. No trained checkpoints are included.

## REFERENCES

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

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1171–1179. Curran Associates, Inc., 2015.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4299–4307. Curran Associates, Inc., 2017.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 10818–10838, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pp. 51797–51830, 2025. doi: 10.52202/085713-1544.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025a. doi: 10.1038/s41586-025-09422-z.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pp. 126866–126898, 2025b. doi: 10.52202/085713-3815. URL https://openreview.net/forum?id=9osvTOYbT4.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11143–11156. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.664.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model Context Protocol (MCP): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*, 2026. doi: 10.1145/3796519.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 29557–29590, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in LLM-as-a-judge. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=grIvSXVJ65.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116. Association for Computational Linguistics, 2023.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. DeepAgent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, pp. 2219–2230. Association for Computing Machinery, 2026b. doi: 10.1145/3774904.3792460.

Zhiwei Li, Yong Hu, and Wenqing Wang. Encouraging good processes without the need for good answers: Reinforcement learning for LLM agent planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1654–1666. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-industry.116.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022.

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

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48371–48392, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pp. 116896–116926, 2025. doi: 10.52202/085713-3524.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025a.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 627–635. PMLR, 2011.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024.

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

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1889–1897. PMLR, 2015.

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

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Xiaoxuan Wang, Bo Liu, Song Jiang, Jingzhou Liu, Jingyuan Qi, Xia Chen, and Baosheng He. From solving to verifying: A unified objective for robust reasoning in LLMs. *arXiv preprint arXiv:2511.15137*, 2025.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in LLM-as-a-judge. *arXiv preprint arXiv:2410.21819*, 2025.

Quan Wei, Siliang Zeng, Chenliang Li, Zhongruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via fine-grained reward structure and credit assignment. *arXiv preprint arXiv:2505.11821*, 2025a.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7909–7928. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.emnlp-main.401.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. TOUCAN: Synthesizing 1.5m tool-agentic data from real-world MCP environments. *arXiv preprint arXiv:2510.01179*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pp. 125532–125554, 2025. doi: 10.52202/085713-3775. URL https://openreview.net/forum?id=2a36EMSSTp.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836. Curran Associates, Inc., 2020.

# A Additional Formalism and Proofs

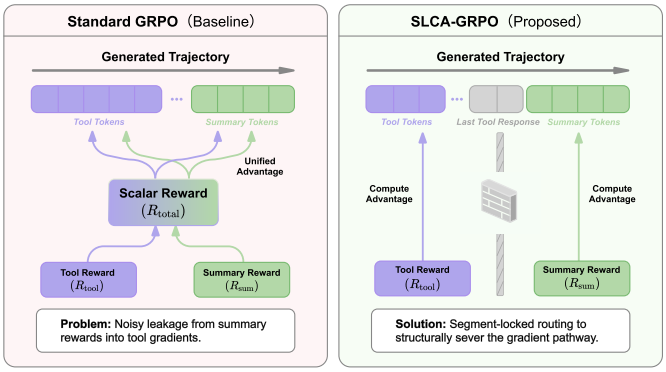



Figure 4: Mechanism of Cross-Segment Credit Misattribution and segment-locked routing. Unified advantage broadcast leaks summary rewards into tool tokens, whereas SLCA blocks this path by routing segment-wise advantages only to their corresponding tokens.

## A.1 Mask-Based Segment Decomposition

We derive segment boundaries using only the standard token-level mask $m_{i,t} \in \{0, 1\}$ from the agent loop. Here $m_{i,t} = 1$ denotes a policy-generated (learnable) token and $m_{i,t} = 0$ denotes an environment-injected token. Let $T_i$ be the unpadded length of rollout $i$.

**Runs of ones** Consider the maximal contiguous runs where $m_{i,t} = 1$. Let $\{(s_{i,k}, e_{i,k})\}_{k=1}^{K_i}$ denote the start and end indices (left-closed, right-open) of these runs so that

$$m_{i,t} = 1 \iff t \in \bigcup_{k=1}^{K_i} [s_{i,k}, e_{i,k}), \tag{6}$$

with disjoint intervals ordered by time.

**Segment definition** We define the **Summary** segment as the last run of learnable tokens:

$$\mathcal{T}_{i,\text{sum}} \triangleq [s_{i,K_i}, e_{i,K_i}), \tag{7}$$

and the **Tool** segment as the union of all earlier learnable runs:

$$\mathcal{T}_{i,\text{tool}} \triangleq \bigcup_{k=1}^{K_i-1} [s_{i,k}, e_{i,k}). \tag{8}$$

**Edge cases** If $K_i = 1$ (no environment injection), then $\mathcal{T}_{i,\text{tool}} = \emptyset$ and the entire generation is treated as Summary. If tool usage is required but the rollout contains no parseable tool call, we trigger the omission-penalty guard (Appendix A.3).

## A.2 Theoretical Analysis: Proofs of Main Properties

This section provides the proofs and modeling assumptions behind Proposition A.1 (Per-Step Advantage Isolation), Theorem A.3 (Nuisance-Variance Removal), and Corollary A.4 (Directional Fidelity), which are summarized in §3.3. The variance and direction results analyze a local score-function estimator that isolates the noise channel induced by unified reward broadcasting; exact GRPO group normalization introduces cross-sample dependencies, which we discuss explicitly below.

**Setup and Definitions** Let a trajectory $y$ be partitioned into a tool segment $y_{\text{tool}}$ and a summary segment $y_{\text{sum}}$. The total reward is $R = R^{\text{tool}} + R^{\text{sum}}$. Let $u_t = \nabla_\theta \log \pi_\theta(y_t \mid h_t)$ be the score function. For the tool-token component, the accumulated score for trajectory $i$ is $U_{i,\text{tool}} = \sum_{t \in \mathcal{T}_{i,\text{tool}}} u_{i,t}$. Let $w_i$ be the importance weight (e.g., length normalization). Let $\mathcal{F}_{\text{tool}}$ be the $\sigma$-algebra generated by $y_{\text{tool}}$.

**Proposition A.1** (Per-Step Advantage Isolation)**.** *Under SLCA routing (Eq. 3), the tool-token component of the gradient estimator has no functional dependence on $R^{sum}$.*

*Proof.* The SLCA-routed tool-token component can be written as

$$g_{\text{tool}}^{\text{SLCA}} = \sum_i w_i \lambda_{\text{tool}} \hat{A}_i^{\text{tool}} \sum_{t \in \mathcal{T}_{i,\text{tool}}} u_{i,t}. \tag{9}$$

The routed advantage $\hat{A}_i^{\text{tool}}$ is computed only from the tool rewards $\{R_j^{\text{tool}}\}$ within the rollout group, while summary tokens are outside the summation. The tool-token gradient component $g_{\text{tool}}^{\text{SLCA}}$ has no functional dependence on $R^{\text{sum}}$, and $\partial g_{\text{tool}}^{\text{SLCA}} / \partial R^{\text{sum}} = \mathbf{0}$. Shared parameters may still transfer information across future optimization steps; the proposition is a per-update statement about the routed gradient estimator. □

**Part I: Conditional Nuisance-Variance Removal** We model the summary reward $R^{\text{sum}}$ as containing a stochastic noise term $\xi$ (e.g., phrasing randomness) that is independent of the tool execution logic.

**Assumption A.2** (Nuisance Noise Decomposition)**.** Conditioned on the tool trajectory $y_{\text{tool}}$, the summary reward decomposes into a deterministic expectation and a zero-mean noise term:

$$R^{\text{sum}} = \bar{R}^{\text{sum}}(y_{\text{tool}}) + \xi, \quad \text{where } \mathbb{E}[\xi \mid y_{\text{tool}}] = 0,\ \mathbb{V}[\xi \mid y_{\text{tool}}] = \sigma_\xi^2 > 0. \tag{10}$$

**Theorem A.3** (Nuisance-Variance Removal: restated from Theorem A.3)**.** *Let $g_{STD}$ and $g_{SLCA}$ be local score-function gradient estimators for tool tokens under unified reward broadcasting and SLCA, respectively. In this local analysis, SLCA removes the irreducible conditional variance term induced by summary-reward noise:*

$$\mathbb{E}[\mathbb{V}(g_{STD} \mid \mathcal{F}_{tool})] - \mathbb{E}[\mathbb{V}(g_{SLCA} \mid \mathcal{F}_{tool})] = \mathbb{E}\left[w_i^2 \|U_{tool}\|^2 \cdot \sigma_\xi^2\right] > 0. \tag{11}$$

*Proof.* We analyze the estimators for a single trajectory $i$ after fixing the local normalization scale of the update:

$$g_{\text{STD}} = w_i (R^{\text{tool}} + \bar{R}^{\text{sum}} + \xi) U_{\text{tool}}, \tag{12}$$

$$g_{\text{SLCA}} = w_i R^{\text{tool}} U_{\text{tool}}. \tag{13}$$

This is the local score-function form of the GRPO update after suppressing group mean and standard-deviation terms. Exact GRPO uses group-normalized rewards, so $\mu_g$ and $\sigma_g$ are functions of all samples in the group and create cross-sample dependencies. This theorem does not give an exact closed form for the fully normalized GRPO covariance; it isolates the summary-noise channel that unified broadcasting necessarily exposes to tool-token gradients and SLCA removes. Conditioned on $\mathcal{F}_{\text{tool}}$, the terms $w_i, R^{\text{tool}}, \bar{R}^{\text{sum}}, U_{\text{tool}}$ are deterministic. The only random variable is $\xi$. For vector-valued gradients, $\mathbb{V}[\cdot]$ denotes the trace covariance, equivalently the expected squared $L_2$ deviation from the conditional mean. We apply the Law of Total Variance: $\mathbb{V}[X] = \mathbb{E}[\mathbb{V}[X|\mathcal{F}]] + \mathbb{V}[\mathbb{E}[X|\mathcal{F}]]$.

**1. Conditional variance of the standard estimator:**

- *Conditional Variance:* $\mathbb{V}[g_{\text{STD}} \mid \mathcal{F}_{\text{tool}}] = \mathbb{V}[w_i \xi U_{\text{tool}} \mid \mathcal{F}_{\text{tool}}] = w_i^2 \|U_{\text{tool}}\|^2 \sigma_\xi^2$.
- *Conditional Expectation:* $\mathbb{E}[g_{\text{STD}} \mid \mathcal{F}_{\text{tool}}] = w_i (R^{\text{tool}} + \bar{R}^{\text{sum}}) U_{\text{tool}}$.

Thus, by the law of total variance,

$$\mathbb{V}[g_{\text{STD}}] = \mathbb{E}[w_i^2 \|U_{\text{tool}}\|^2 \sigma_\xi^2] + \mathbb{V}[w_i (R^{\text{tool}} + \bar{R}^{\text{sum}}) U_{\text{tool}}]. \tag{14}$$

**2. Conditional variance of the SLCA estimator:**

- *Since $g_{SLCA}$ is $\mathcal{F}_{tool}$-measurable (independent of $\xi$):*

$$\mathbb{V}[g_{\text{SLCA}} \mid \mathcal{F}_{\text{tool}}] = 0. \tag{15}$$

**3. Comparison:** Taking expectations over $\mathcal{F}_{\text{tool}}$ gives the stated identity:

$$\mathbb{E}[\mathbb{V}(g_{\text{STD}} \mid \mathcal{F}_{\text{tool}})] - \mathbb{E}[\mathbb{V}(g_{\text{SLCA}} \mid \mathcal{F}_{\text{tool}})] = \mathbb{E}[w_i^2 \|U_{\text{tool}}\|^2 \sigma_\xi^2]. \tag{16}$$

Since generation noise $\sigma_\xi^2 > 0$ and gradient norm $\|U_{\text{tool}}\|^2 > 0$ almost everywhere, the removed nuisance-variance term is strictly positive. This theorem intentionally makes a conditional variance claim: the systematic term $\mathbb{V}[w_i(R^{\text{tool}} + \bar{R}^{\text{sum}})U_{\text{tool}}]$ may differ from $\mathbb{V}[w_i R^{\text{tool}} U_{\text{tool}}]$ depending on task structure and covariance, which is why we frame SLCA as a bias–variance trade-off rather than an unconditional variance dominance statement. □

**Optimization implication** For SGD on non-convex $L$-smooth functions, standard convergence bounds scale with the gradient-variance term (e.g., $O(\sigma^2/\epsilon^4)$ to reach an $\epsilon$-stationary point). Theorem A.3 shows that SLCA removes a positive conditional variance component from tool-token gradients; when this nuisance component dominates systematic covariance changes, the resulting smoother gradients should translate into faster and more stable optimization.

**Part II: Resolution of Advantage Confusion (Directional Fidelity)** We define a Conflict Regime where the tool segment receives a below-group-mean normalized advantage ($\hat{A}^{\text{tool}} < 0$) but the summary segment receives an above-group-mean normalized advantage ($\hat{A}^{\text{sum}} > 0$).

**Theorem A.4** (Directional Fidelity: restated from Corollary A.4)**.** *Let $\mathbf{d}^* = -\nabla \log \pi(y_{tool})$ be the ideal descent direction to penalize tool failure. Let normalized scores be $\hat{A}^{tool} = -\alpha$ ($\alpha > 0$, penalty) and $\hat{A}^{sum} = \beta$ ($\beta > 0$, reward). Let $\sigma_{tool}$ and $\sigma_{sum}$ denote the group standard deviations of the raw segment rewards. The expectations below are taken over the stated rollout randomness, with the trajectory partition, group standard deviations, and $\alpha, \beta$ held fixed.*

- ***SLCA Update:*** *$\mathbb{E}[g_{SLCA}] \propto -\alpha \nabla \log \pi$. Inner product with $\mathbf{d}^*$: $\alpha \|\nabla\|^2 > 0$.* ***(Correct Direction)***
- ***Standard GRPO Update:*** *The trajectory-level raw deviation is proportional to $-\alpha\sigma_{tool} + \beta\sigma_{sum}$, so $\mathbb{E}[g_{STD}] \propto (\beta\sigma_{sum} - \alpha\sigma_{tool})\nabla \log \pi$. Inner product with $\mathbf{d}^*$ is $(\alpha\sigma_{tool} - \beta\sigma_{sum})\|\nabla\|^2$.*

*If $\beta\sigma_{sum} > \alpha\sigma_{tool}$ (the raw summary deviation dominates the raw tool penalty), Standard GRPO updates in the wrong direction ($< 0$), reinforcing the failure. The simpler condition $\beta > \alpha$ is recovered when the two raw reward components have equal group standard deviations. SLCA maintains fidelity with respect to the tool-execution signal.*

A concrete instantiation of the conflict regime in Theorem A.4 is visualized in Appendix D.4: under an imperfect tool trajectory ($R^{\text{tool}}$=0.65, below the group mean) paired with a perfect summary ($R^{\text{sum}}$=1.0, above the group mean), standard GRPO reinforces the inefficient policy, whereas SLCA routes the tool-level penalty into $\hat{A}^{\text{tool}}$ alone and recovers the correct single parallel invocation.

**Bias–Variance Trade-off of the $\hat{A}^{\text{sum}}$ Estimator** Within a rollout group, trajectories may diverge after the tool segment and produce distinct post-tool states. SLCA still normalizes $R^{\text{sum}}$ across the group rather than within subgroups that share an identical post-tool state, so $\hat{A}^{\text{sum}}$ is not a state-conditional value estimate and can conflate summary quality with post-tool state variation.

This cross-state normalization is not unique to SLCA: standard GRPO normalizes $R_{\text{total}}$ across the same group, and methods such as MT-GRPO (Wei et al., 2025a) and GRPO-Verif (Wang et al., 2025) also do not compute exact state-conditional advantages without a learned value function or tree-based rollouts. SLCA *narrows* the scope of this bias:

- Under standard GRPO, both tool and summary tokens receive $A_{\text{total}}$, the z-score of $(R^{\text{tool}} + R^{\text{sum}})$, which includes cross-state variation in $R^{\text{sum}}$.

- Under SLCA, tool tokens receive $\hat{A}^{\text{tool}}$, the z-score of $R^{\text{tool}}$ alone, computed across rollouts from the same initial state (shared prompt $x$), without cross-state summary bias. Only summary tokens receive the biased $\hat{A}^{\text{sum}}$.

We frame this as a bias–variance trade-off. The unified estimator avoids grouping bias but exposes tool-token gradients to conditional nuisance variance from the summary-reward channel (Theorem A.3). SLCA accepts summary-side grouping bias but confines it to summary tokens and removes that direct variance channel. The support-control experiment in Table 21 provides a cleaner decomposition: closing the summary-to-tool $S \to T$ support path has a 5.51 pp main effect on $\tau^2$-Bench, whereas the matched w/o SLCA comparison changes normalization and token support together.

Explicit subgroup partitioning, which would normalize $R^{\text{sum}}$ only within trajectories sharing the same tool outcome, reduces the effective group size to $G/C$ per subgroup, where $C$ is the number of distinct tool outcomes. For multi-tool-call settings with $K$ calls, the effective size decays as $G/C^K$, making explicit partitioning impractical without increasing $G$. This is a design trade-off; soft-grouping or hierarchical alternatives remain open.

### A.3 SLCA Practical Details

This section details the numerical stability mechanisms and edge-case handling strategies implemented in SLCA-GRPO.

**Presence-Based Filtering (Numerical Stability)** Tool usage behavior is naturally sparse and dynamic; some rollouts may lack a tool segment (i.e., $|\mathcal{T}_{i,\text{tool}}| = 0$) due to omission or early termination. Naively normalizing rewards within a group where only one sample contains a tool segment leads to a variance collapse (denominator becomes 0 or unstable), or yields an unnormalized advantage equal to the raw score. To resolve this, we strictly normalize each segment type using only the subset of samples where that segment is present.

Let $L_i^s \triangleq |\mathcal{T}_{i,s}|$ denote the length of segment $s \in \{\text{tool}, \text{sum}\}$ in rollout $i$. We define the valid indices set for group $g$ as:

$$\mathcal{I}_g^s \triangleq \{j \mid g(j) = g \text{ and } L_j^s > 0\}. \tag{17}$$

We compute the segment-wise advantage $\hat{A}_i^s$ using a conditional normalization scheme. With a small stability constant $\epsilon_{\text{norm}}$, the formulation is:

$$\hat{A}_i^s = \begin{cases} 0 & \text{if } L_i^s = 0 \text{ or } |\mathcal{I}_{g(i)}^s| < 2, \\ \dfrac{R_i^s - \mu_{g(i)}^s}{\sigma_{g(i)}^s + \epsilon_{\text{norm}}} & \text{otherwise,} \end{cases} \tag{18}$$

where the mean $\mu_g^s$ and standard deviation $\sigma_g^s$ are computed exclusively over the valid set $\mathcal{I}_g^s$:

$$\mu_g^s = \frac{1}{|\mathcal{I}_g^s|} \sum_{j \in \mathcal{I}_g^s} R_j^s, \quad \sigma_g^s = \text{Std}\left(\{R_j^s\}_{j \in \mathcal{I}_g^s}\right). \tag{19}$$

Here $\epsilon_{\text{norm}}$ denotes a numerical floor for stable division, not a tuned reward or optimization hyperparameter. When the effective group size $|\mathcal{I}_g^s| < 2$, we explicitly zero out the advantage ($\hat{A}_i^s = 0$) to disable updates, rather than falling back to standard statistics (e.g., $\mu = 0, \sigma = 1$), which would amplify raw scores as advantages.

**Post-Normalization Weighting** The segment weights $\lambda_{\text{tool}}$ and $\lambda_{\text{sum}}$ must be applied *after* the normalization step. Applying linear scaling *before* z-score normalization cancels the scale in the idealized case without a numerical floor:

$$\text{Norm}(\lambda \cdot X) = \frac{\lambda X - \text{Mean}(\lambda X)}{\text{Std}(\lambda X)} = \frac{\lambda(X - \mu)}{\lambda \sigma} = \text{Norm}(X). \tag{20}$$

With the fixed $\epsilon_{\text{norm}}$ used in our implementation, this identity holds up to the numerical floor. We construct the final routed advantage $\hat{A}_{i,t}^{\text{SLCA}}$ by multiplying the normalized scalar $\hat{A}_i^s$ by $\lambda_s$ during the token-level routing phase (as shown in Eq. 3).

**Anti-Hacking Omission Penalty Guard** We strictly route the omission penalty $R_{\text{pen}} < 0$ to the Summary Segment:

$$R_i^{\text{tool}} = 0, \qquad R_i^{\text{sum}} = R_{\text{pen}}. \tag{21}$$

In our implementation, $R_{\text{pen}} = -0.5$. When the ground truth requires tool use but the rollout contains no <tool_call> opening tag, this value directly *overrides* the weighted trajectory score rather than being subtracted as an additive penalty; the normalized success score is then clipped to zero. By this design, the negative signal propagates only through the summary generation tokens. This effectively penalizes the decision to "answer immediately without calling," correctly aligning the gradient direction to suppress premature summarization. In a representative SLCA trace, no-call episodes start at roughly 10–15% and approach zero by update 50. In the uniform sensitivity setting, the parallelism weight is $S_{\text{par}} = 0.20$; its BFCL and $\tau^2$-Bench results remain above matched GRPO (App. D.7).

### A.4 HierR: Process Reward Definitions and Edge Cases

HierR provides dense, structure-preserving shaping for the tool segment and a terminal preference score for the summary. We define a process score as a weighted sum of five components:

$$S_{\text{process}} = 0.10\, S_{\text{fmt}} + 0.25\, S_{\text{name}} + 0.15\, S_{\text{key}} + 0.20\, S_{\text{val}} + 0.30\, S_{\text{par}}. \tag{22}$$

Let $\mathcal{Y}$ be the predicted tool-call multiset and $\mathcal{G}$ be the ground-truth multiset.

**(1) Format adherence ($S_{\text{fmt}}$)** We measure syntactic validity and schema-parseability:

$$S_{\text{fmt}} = \frac{N_{\text{valid}}}{\max(N_{\text{open}}, N_{\text{close}})} \in [0, 1], \tag{23}$$

where $N_{\text{valid}}$ is the number of call blocks that pass JSON parsing and schema validation, and $N_{\text{open}}, N_{\text{close}}$ count <tool_call> and </tool_call> tags.

**(2) Tool-name match ($S_{\text{name}}$)** We compute an F1-style score over tool-name multisets $\mathcal{N}_Y, \mathcal{N}_G$:

$$S_{\text{name}} = \frac{2|\mathcal{N}_Y \cap \mathcal{N}_G|}{|\mathcal{N}_Y| + |\mathcal{N}_G|} \in [0, 1], \tag{24}$$

where $\cap$ denotes multiset intersection.

**(3) Argument-key match ($S_{\text{key}}$) and value match ($S_{\text{val}}$)** For each gold call $g \in \mathcal{G}$, match it to the best predicted call $y^* \in \mathcal{Y}$ with the same tool name. Let $\mathcal{K}_y, \mathcal{K}_g$ be argument-key sets. We define key match as Jaccard:

$$S_{\text{key}}(y, g) = \frac{|\mathcal{K}_y \cap \mathcal{K}_g|}{|\mathcal{K}_y \cup \mathcal{K}_g|}. \tag{25}$$

Value match averages a loose indicator over matched keys:

$$S_{\text{val}}(y, g) = \frac{1}{|\mathcal{K}_g|} \sum_{k \in \mathcal{K}_y \cap \mathcal{K}_g} \mathbb{I}_{\text{loose}}\big(y[k], g[k]\big), \tag{26}$$

where $\mathbb{I}_{\text{loose}}$ allows simple type casting and string normalization. We aggregate over all $g$ by averaging the best-match scores. Extra predicted calls are not directly averaged into $S_{\text{key}}$ or $S_{\text{val}}$ once all gold calls have been matched; they are penalized through $S_{\text{name}}$, $S_{\text{par}}$, and, when invalid, $S_{\text{fmt}}$.

**(4) Parallelism constraint ($S_{\text{par}}$)** To prevent redundancy or "parallelism collapse" at the initial tool step, we impose a hard cardinality check:

$$S_{\text{par}} = \mathbb{I}\big(|Y_{\text{init}}| = |G_{\text{init}}|\big) \in \{0, 1\}, \tag{27}$$

where $Y_{\text{init}}$ and $G_{\text{init}}$ denote the predicted and gold call lists in the first call step.

### A.5 SUMMARY SCORE AND OMISSION EDGE CASE

We derive the summary preference score $S_{\text{summary}} \in [0, 1]$ using an LLM-as-a-judge approach (Wataoka et al., 2025; Li et al., 2026a). Specifically, we employ a structured evaluation prompt to assess the utility and faithfulness of the final response, normalizing the raw ratings to the unit interval. We query the judge for every rollout, including cases in which mandatory tool use is omitted. For a rollout that triggers the omission guard, the policy reward is overridden by $R_{\text{sum}} = R_{\text{pen}} = -0.5$; the normalized guard advantage is retained for the policy update.

**Judge Model and Deployment** The judge is GPT-OSS-120B (OpenAI, Apache 2.0), a 120B-parameter open-weight language model served via a vLLM OpenAI-compatible API endpoint. It is a frozen reward producer: its parameters are not optimized by policy training, although its scalar $R_{\text{sum}}$ enters the policy loss. The role can be served by a local model or an API endpoint and is separate from the response mocker used by SGLS. The deployment and decoding parameters are listed in Table 3. The deterministic setting (`temperature`$= 0.0$) makes the judge output deterministic for a fixed input; rollout and summary sampling can still vary across runs.

Table 3: Summary Judge: Deployment and Decoding Parameters.

| Parameter | Value |
|---|---|
| *Model Specification* | |
| Model | GPT-OSS-120B |
| Parameters | 120B |
| Serving Framework | vLLM (OpenAI-compatible API) |
| *Decoding* | |
| Temperature | 0.0 (fully greedy / deterministic) |
| Max Tokens | 8,192 |
| top_p | 1.0 (vLLM default; no nucleus filtering) |
| top_k | 0 (disabled; no top-$k$ filtering) |
| *Reliability* | |
| Timeout | 540s (including queue wait) |
| Max Retries | 3 (exponential backoff) |

**Prompt Structure and Input Variables** The full evaluation prompt is provided in the Prompts section. Three variables are injected into the template: `{question_content}` (the user's original question), `{tool_responses}` (tool return values only, excluding `<tool_call>` content), and `{final_response}` (the model's final summary). The judge evaluates along three dimensions (*Result Interpretation*, *Information Completeness*, and *Expression Quality*) and produces a categorical rating on a 5-point scale.

**Factual vs. Stylistic Dimensions** The three evaluation dimensions differ in nature: *Result Interpretation* and *Information Completeness* are factual; the latter is causally dependent on tool execution quality, since incomplete tool results necessarily limit response completeness. *Expression Quality* is partly stylistic (clarity, organization). SLCA's decoupling guarantee (Proposition A.1) holds regardless of what $R^{\text{sum}}$ measures: it prevents any component of $S_{\text{summary}}$, whether factual or stylistic, from contaminating tool-token advantages. The legitimate causal link (correct tools $\rightarrow$ better summaries) is preserved via the systematic component $\bar{R}^{\text{sum}}(y_{\text{tool}})$ in Assumption A.2; the direct summary-to-tool pathway is blocked. BFCL and $\tau^2$-Bench use judge-free evaluation metrics, providing checks beyond the judge-based metric.

**Score Normalization** The categorical rating is mapped to $[0, 1]$ via linear normalization $(r - 1)/4$, where $r$ is the raw integer score. Table 4 lists the complete mapping.

Table 4: Summary Judge: Rating-to-Score Mapping.

| Rating | Raw Score ($r$) | Normalized $S_{\text{summary}}$ |
|---|---|---|
| very poor | 1 | 0.00 |
| poor | 2 | 0.25 |
| acceptable | 3 | 0.50 |
| good | 4 | 0.75 |
| excellent | 5 | 1.00 |

**Output Parsing and Failure Handling** The judge output is parsed via the regex pattern:

```
<response_quality>.*?<rating>(.*?)</rating>.*?</response_quality>
```

Only the five predefined categorical values listed in Table 4 are accepted. Any other output (e.g., numeric scores, free-text descriptions, or malformed XML) is treated as a parse failure and assigned `JUDGE_FAILURE_SCORE` $= 0.0$. This conservative default prevents noisy or malformed judge outputs from injecting spurious reward signals into training.

**Omission Guard** The guard overrides the policy reward when a required tool call is absent, so a summary cannot receive a positive policy reward for answering without tool support. The judge call is still made for the rollout; its output does not replace the omission reward. This keeps the reward protocol and group normalization statistics fixed across the compared conditions.

**Safeguards Against Judge Artifacts** We employ the following safeguards to mitigate known failure modes of LLM-as-a-judge:

1. **Verbosity/style bias:** The structured rubric evaluates three factual dimensions rather than holistic preference, constraining the judge to content-level assessment. The judge receives only tool responses and the final summary (no intermediate reasoning, chain-of-thought, or `<tool_call>` content), limiting stylistic influence. The discrete 5-point categorical scale further constrains output granularity.
2. **Reward hacking:** Two safeguards are used: (a) the omission guard overrides the policy reward with $R_{\text{pen}}$ and routes its normalized signal to the available Summary tokens when mandatory tool use is absent; (b) SLCA's advantage routing (Eq. 3) isolates $S_{\text{summary}}$ from tool-token gradients.
3. **Preference leakage:** The same judge model, prompt template, and decoding parameters are used identically across all compared methods. The judge weights are frozen throughout training. For cross-domain benchmarks, $S_{\text{summary}}$ is not used: BFCL evaluates tool accuracy directly, and $\tau^2$-Bench uses the official Pass$^1$ metric.

### A.6 SGLS Scope and Evaluation

SGLS is a schema-guided simulator: it preserves tool names, required fields, and response structure while allowing concrete observations to differ from those of a live API. We evaluate this structural alignment empirically through the *w/o SGLS* ablation, the cross-domain benchmarks, and the alignment examples in App. C.3.

**Scope of the semantic argument** Raw simulated and real observations can have disjoint supports: random hashes, timestamps, and instance IDs can make $D_{\text{TV}}(P_{\text{real}}, P_{\text{sim}}) = 1$, so a standard simulation lemma on raw observations is vacuous. The useful alignment is at the schema and semantic-interaction level: tool names, required fields, call structure, and response roles remain aligned even when concrete values differ. A Wasserstein-style metric on a semantic representation would be more meaningful than total variation, but we do not define such a representation or claim a bound here. SGLS treats the schema as a control plane, and its contribution is assessed through the *w/o SGLS* ablation, the cross-domain benchmarks, and the alignment examples reported in the rest of this appendix.

### A.7 Algorithm Details

Algorithm 1 summarizes the complete SLCA-GRPO training procedure with SGLS rollouts and HierR segment rewards.

**Algorithm 1** SLCA-GRPO training with SGLS and HierR

**Input:** dataset $\mathcal{D}$, schemas $\Sigma(x)$, group size $G$, PPO clip $\epsilon_{\text{clip}}$
**for** each iteration **do**
  sample $x \sim \mathcal{D}$
  **for** $i = 1$ to $G$ **do**
    rollout $y_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ by interacting with SGLS (schema validation + tool responses)
    record mask $m_{i,t}$ and segment sets $\mathcal{T}_{i,\text{tool}}, \mathcal{T}_{i,\text{sum}}$
    compute HierR segment returns $(S_i^{\text{tool}}, S_i^{\text{sum}})$ with omission-penalty guard if needed
  **end for**
  compute $\hat{A}_i^{\text{tool}} = \text{Norm}(S_i^{\text{tool}})$ and $\hat{A}_i^{\text{sum}} = \text{Norm}(S_i^{\text{sum}})$ with presence-based filtering
  route advantages to tokens via Eq. 3 to obtain $\hat{A}_{i,t}^{\text{SLCA}}$
  update $\theta$ by maximizing Eq. 5
**end for**

## B Empirical Analysis of Optimization Stability

As an optimization diagnostic, we track the $L_2$ norm of the gradient vector $\|\nabla_\theta\|_2$ throughout training across all model scales. A lower and smoother gradient norm is one descriptive indicator of update variability.

**Visual Analysis.** Figure 5 illustrates the training dynamics.

- **SFT+GRPO (w/o SLCA) (Orange):** Exhibits frequent high-magnitude spikes and broader fluctuations. These patterns are compatible with updates in which summary and tool signals point to different changes.
- **SLCA-GRPO (Blue):** Maintains a generally smoother trajectory in these runs. Its segment-specific support removes the direct summary-to-tool contribution to tool-token updates.

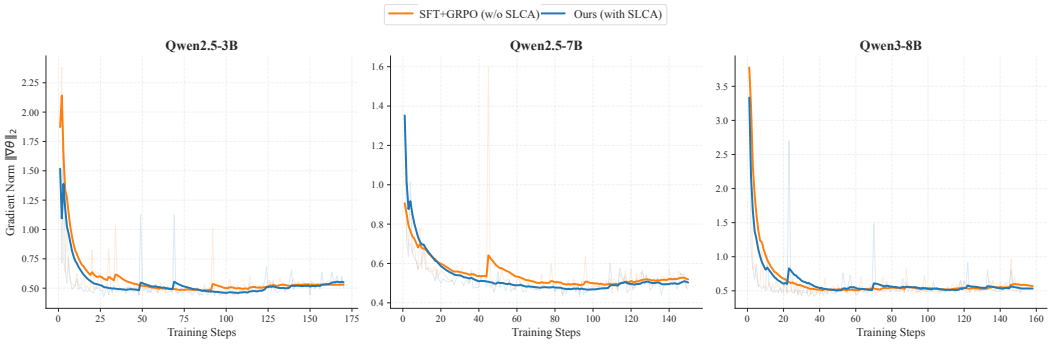



Figure 5: Gradient Norm Dynamics ($\|\nabla_\theta\|_2$) during Training. We compare the gradient norms of SFT+GRPO (w/o SLCA) (Orange) and SLCA-GRPO (Blue) across Qwen2.5-3B, 7B, and Qwen3-8B. These are representative single-run traces and provide a descriptive view of update variability under the two estimators.

**Quantitative Gradient Stability** We quantify this diagnostic by computing the standard deviation (Std) of the gradient norm over the training steps. As reported in Table 5, SLCA-GRPO has a lower observed standard deviation across all model scales in these runs.

On **Qwen2.5-3B-Instruct**, gradient-norm volatility is reduced by **25.0%**. On **Qwen3-8B-Base**, which has the highest baseline Std in this table, the reduction is 2.5%.

**Cross-Scale Analysis: Gradient Stability vs. Task Performance** The relative reduction decreases across the three backbones (25.0% → 13.1% → 2.5%). Gradient-norm standard deviation captures one aggregate variability measure, so these values do not identify the contribution of each noise source.

The matched $\tau^2$-Bench gaps are +1.09 pp (3B), +9.15 pp (7B), and +10.03 pp (8B), all based on three-run means. These aggregate results do not establish a scale trend for gradient stability. On the simpler Toucan single-turn evaluation, the three-run Success gaps range from +2.05 to +2.53 pp. Separately, we measured the gradient cosine similarity $\cos(\nabla_{\text{tool}}, \nabla_{\text{sum}})$ on the 3B backbone and found it near zero ($[-0.03, 0.08]$) throughout training for *both* SLCA and standard GRPO. In this diagnostic, advantage magnitudes, rather than gradient direction, are the more visible source of cross-segment mismatch. Because the measured tool/summary gradient cosine is near zero, we refer to SLCA as an "advantage-level firewall" rather than a "gradient firewall": the diagnostic concerns the magnitude of the assigned advantages, not a measured directional conflict between the two gradient components.

Table 5: **Quantification of Gradient Stability.** We report the Standard Deviation (Std) of the gradient norm. SLCA-GRPO has lower observed volatility in these representative runs.

| **Model Backbone** | **Ours Std** ($\sigma_{\text{SLCA}}$) | **Baseline Std** ($\sigma_{\text{Base}}$) | **Improvement** (↓) |
|---|---|---|---|
| **Qwen2.5-3B-Instruct** | **0.156** | 0.209 | **25.0%** |
| **Qwen2.5-7B-Instruct** | **0.094** | 0.108 | **13.1%** |
| **Qwen3-8B-Base** | **0.312** | 0.320 | **2.5%** |

# C Datasets and Training Details

## C.1 Data Processing Pipeline

As illustrated in Figure 6, this pipeline is designed to strictly prevent test-set leakage and optimize data composition for different learning stages.

The pipeline consists of four distinct phases:

**Phase 1: Noise Filtering and Schema Validation** Starting with 119,279 Toucan raw trajectories, we first remove approximately 40,000 samples identified as irrelevant or containing low-quality formatting. The remaining 79,279 candidates undergo strict filtering, which removes 1,038 samples containing broken JSON, non-standard tool tokens, or hallucinations to ensure simulator stability. This results in a high-quality valid pool of **78,241** trajectories.

**Phase 2: Evaluation Isolation.** To ensure rigorous zero-shot evaluation, we randomly sample a held-out Toucan-Test set of **4,000** samples from the valid pool before any training splits are made. This set comprises:

- **3,000 Native Single-turn samples:** Testing atomic tool-use capabilities.
- **1,000 Decomposed Multi-turn samples:** Extracted from distinct multi-turn dialogues to test context retention and state tracking.

The remaining 74,241 samples constitute the Total Training Pool.

**Phase 3: Trajectory Decomposition Strategy** For Reinforcement Learning, training on long-horizon dialogs from scratch is unstable. We decompose multi-turn dialogs into atomic *Context-Action* pairs. We exclude the first tool-call turn during decomposition and extract samples starting from the second tool invocation (Turn $\geq 2$). Each decomposed sample provided to the RL agent thus contains historical context from previous user-agent interactions. The Toucan trajectories use real

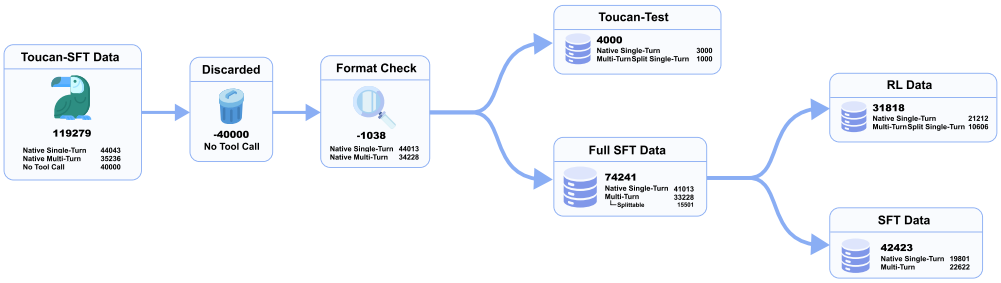



Figure 6: **Data Processing Pipeline.** The workflow applies strict relevance filtering and format check before isolating the evaluation set. The remaining training pool is split into SFT and RL partitions; the RL set combines native single-turn and decomposed multi-turn trajectories formatted for the SGLS.

MCP teacher traces. The segment mask is derived deterministically from the structured trajectory trace and the environment boundary; no annotated segment or token labels and no learned segmenter are used.

**Phase 4: SFT vs. RL Partitioning** We split the training pool (74,241 samples) into two partitions:

- **SFT Partition (42,423 samples):** Representing approximately $57\%$ of the data, this set prioritizes diversity in format (Human/GPT styles) and includes the majority of native multi-turn data to establish a robust instruction-following prior.
- **RL Partition (31,818 samples):** Optimized for exploration, this set is constructed with a specific compositional ratio:

$$\text{RL Composition} \approx \frac{2}{3} \times \text{Native Single-turn} + \frac{1}{3} \times \text{Decomposed Multi-turn}$$

  Specifically, it contains **21,212** native single-turn samples and **10,606** decomposed multi-turn samples. This stratified mix balances the learning of simple function execution with complex, history-dependent reasoning.

**Total Supervision Visibility** As noted in the experimental setup, our SFT baseline is trained on the union of both partitions (Total 74,241 samples), so the comparison uses matched data exposure.

### C.2 Training Implementation Details

**Implementation Environment and Hyperparameters** We fine-tune the model on the Toucan supervision trajectories using the standard cross-entropy objective. The SFT training is conducted on 8 NVIDIA H20 GPUs with *bfloat16* precision. We employ the AdamW optimizer (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) with a cosine learning rate scheduler. For the RL stage, training is scaled across 32 NVIDIA H20 GPUs (4 nodes). The maximum sequence lengths for prompts and generations are extended to 16k and 8k tokens, respectively, to accommodate long-context reasoning. Detailed hyperparameters for both stages are listed in Table 6.

**Baseline Configurations and Controlled Variables** The matched set comprises SFT+GRPO, SFT+SLCA-GRPO, w/o SLCA, w/o SGLS, and w/o HierR. Within each backbone, all five use that backbone's SFT checkpoint, the same data split, SGLS endpoint, mocker configuration, decoding settings, evaluation protocol, rollout group size $G = 16$, one RL epoch, and the same run set. GRPO, SLCA-GRPO, w/o SLCA, and w/o SGLS use the same HierR definition and subweights; w/o HierR removes HierR, and w/o SGLS removes schema conditioning and deterministic validation:

- **SFT (Data-Equivalent Baseline):** By exposing the model to the full dataset in a supervised manner, this baseline serves as a data-equivalent benchmark to determine whether RL optimization yields gains beyond simply scaling up demonstration data.

- **SFT+GRPO (Standard Baseline):** This baseline uses the same SGLS environment and HierR reward definitions as our method. Its estimator aggregates the two rewards into an unweighted sum ($R_{\text{total}} = 1.0 \cdot R_{\text{tool}} + 1.0 \cdot R_{\text{sum}}$) and broadcasts the resulting advantage to all learnable tokens.
- **SFT+SLCA-GRPO (Ours):** Following the same data split as SFT+GRPO, it computes independent advantages for execution and summarization. The matched comparison uses the same 1:1 weighting ($\lambda_{\text{tool}} = \lambda_{\text{sum}} = 1$) and routes signals via Eq. 3.
- **Ablation definitions:** w/o SLCA uses the unified estimator, w/o SGLS removes schema conditioning and deterministic validation while retaining the mocker endpoint, and w/o HierR removes the hierarchical reward.

Table 6: Detailed Hyperparameters and Environment Settings. We list the configurations for both the supervised fine-tuning (SFT) and the subsequent RL training stages.

| **Fine-Tuning Hyperparameters** | | **RL Hyperparameters** | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| Hardware Resources | 8 × H20 GPUs | Hardware Resources | 32 × H20 (4 Nodes) |
| Optimizer | AdamW | Precision | bfloat16 |
| Total Batch Size | 32 | Total Batch Size | 128 |
| Per-Device Batch | 1 | Mini-batch Size | 32 |
| Grad Accumulation | 4 | Rollout Group Size ($G$) | 16 |
| Learning Rate | $5 \times 10^{-6}$ | Learning Rate | $1 \times 10^{-6}$ |
| Number of Epochs | 2 | Number of Epochs | 1 |
| LR Scheduler | Cosine | Max Response Len | 8192 |
| Warmup Ratio | 0.1 | KL Coefficient ($\beta_{\text{KL}}$) | 0.001 |
| Max Context Len | 16384 | Max Prompt Len | 16384 |
| – | | PPO Clip ($\epsilon_{\text{clip}}$) | 0.2 (symmetric) |
| – | | Dual-Clip Constant ($c$) | 3.0 |
| – | | KL Estimator | low-var KL |
| – | | Loss Aggregation | token-mean |
| – | | No-Call Penalty ($R_{\text{pen}}$) | $-0.5$ (override) |

The PPO clip is symmetric: $\epsilon_{\text{clip}} = \epsilon_{\text{low}} = \epsilon_{\text{high}} = 0.2$, without asymmetric DAPO-style clipping. The dual-clip constant $c = 3.0$ applies only to negative-advantage samples when the likelihood ratio is large. The no-call value $R_{\text{pen}} = -0.5$ is used when tool use is required but no tool call is emitted; it overrides the weighted score rather than subtracting from it.

**Training-step convention** All RL curves report policy-update steps, not PPO inner mini-batch passes. The main runs use rollout group size $G = 16$ and one RL epoch over the post-filtered rollout stream; this corresponds to approximately 100 logged policy-update steps in the default 7B setting. We use this convention consistently across all training-dynamics figures.

### C.3 SGLS Implementation Details

**Frozen LLM Specification** The frozen LLM used in SGLS is **Qwen3-235B-A22B-Instruct** (July 2025 checkpoint), a 235B-parameter Mixture-of-Experts (MoE) model with 22B active parameters per forward pass. This model belongs to a *different architectural family* from the policy backbone (Qwen2.5-7B-Instruct, dense architecture, September 2024). The cross-family, cross-generational design limits implicit leakage and same-backbone favoritism: (i) the representation spaces are structurally distinct (MoE routing vs. dense layers), (ii) the models use different tokenizers and pre-training corpora, and (iii) the frozen LLM's role is limited to template-constrained mocking of valid tool calls; it never provides planning guidance or reward signals. The Toucan teacher trajectories used to construct the training data are real MCP trajectories curated for tool coverage and quality. They provide the tool-reference source for $R_{\text{tool}}$ and are not an input to the SLCA estimator.

**Deployment Configuration** The frozen LLM is served via vLLM 0.11.0 with the deployment parameters listed in Table 7.

Table 7: SGLS Frozen LLM: Deployment and Decoding Parameters.

| Parameter | Value |
|---|---|
| *vLLM Deployment* | |
| Tensor-Parallel Size (TP) | 8 |
| Max Model Length | 32,768 |
| GPU Memory Utilization | 0.86 |
| Max Concurrent Sequences | 300 |
| Precision (dtype) | bfloat16 |
| Prefix Caching | Enabled |
| Expert Parallel (MoE) | Enabled |
| *Decoding (API Call Side)* | |
| Temperature | 0.6 |
| Max Tokens | 4,096 |
| top_p | 1.0 (vLLM default; no nucleus filtering) |
| top_k | 0 (disabled; no top-$k$ filtering) |
| Stop Tokens | Model EOS (vLLM built-in) |

**Two-Stage Pipeline** SGLS operates as a two-stage pipeline to minimize unnecessary LLM invocations:

1. **Stage 1: Deterministic Schema Validation (no LLM):** Checks tool existence against the current instance's available tool set; if the tool name is not found, returns a hard-coded error string. Then verifies required parameters against the schema definition; missing parameters trigger a deterministic error message listing the absent fields.

2. **Stage 2: Template-Constrained LLM Mocking (valid calls only):** For calls that pass Stage 1, the tool definition JSON and the parsed `<tool_call>` payload are injected into a fixed prompt template. The frozen LLM generates a mock `<tool_response>` output. Post-processing extracts the response content via tag matching and handles truncation edge cases (e.g., responses with opening but no closing tags).

This hybrid architecture gives schema violations in early-training rollouts deterministic feedback, while the LLM is invoked only for semantically plausible mock responses to correctly formed calls.

***w/o SGLS* fallback environment** The *w/o SGLS* ablation keeps the same frozen mocker and rollout interface, and removes only schema conditioning and deterministic validation. Both variants call the same external vLLM `/v1/chat/completions` endpoint with the same Qwen3-235B-A22B model, temperature (0.6), max-token budget (4,096), and timeout (540s). Thus the ablation is *prompt-unconstrained*, not model-unconstrained: the API-server role instruction, `<tool_response>` output contract, prohibitions against explanations or `<tool_call>` emission, few-shot examples, incoming-request format, and rollout limits are unchanged. The only prompt-side removal is the `Tool Definition` JSON-schema block; the corresponding code path also removes per-instance schema caching and tool-definition parsing. As a result, unknown tools, missing required parameters, and type-invalid arguments no longer trigger deterministic error strings; every call is forwarded to the mocker, which generates a `<tool_response>` shell whose content is determined by the LLM. Post-processing only strips/extracts tool-response tags and does not validate JSON fields.

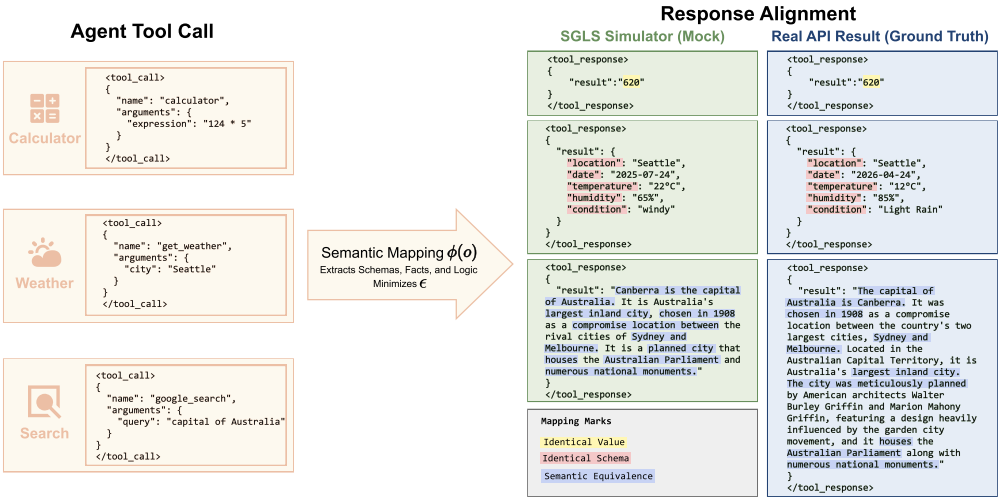



Figure 7: **SGLS mock responses align with real-API outputs across three representative tool domains.** The Semantic Mapping $\phi(o)$ extracts schemas, facts, and logic, realizing three levels of alignment (Mapping Marks): *Identical Value* for deterministic tools (*calculator*), *Identical Schema* for query-style tools (*weather*; individual values need not match a specific real-API snapshot but remain within plausible ranges), and *Semantic Equivalence* for open-ended tools (*search*; factually consistent content differing only in phrasing).

**Response-mocker robustness** An alternate response-mocker check replaces the frozen response mocker with GPT-OSS-120B while keeping the policy initialization, data, validators, rewards, rollout budget, and evaluation protocol fixed. The response mocker generates tool observations; it is separate from the summary judge that produces $R_{\text{sum}}$. In this check the mocker and judge are intentionally from the same GPT-OSS-120B family; BFCL and $\tau^2$-Bench do not use the judge reward, so their transfer measurements are not affected by that shared-family choice. The summary-judge prompt and decoding remain unchanged. Table 8 reports the matched results.

Table 8: Response-mocker check on Qwen2.5-7B-Instruct. GPT-OSS-120B replaces the response mocker; entries are mean $\pm$ standard deviation under the matched policy setup.

| **Method** | **Toucan Succ. (%)** | **BFCL Acc. (%)** | **$\tau^2$-Bench Pass$^1$ (%)** |
|---|---|---|---|
| SFT+GRPO (w/o SLCA) | $75.74 \pm 1.47$ | $68.08 \pm 0.36$ | $32.74 \pm 2.66$ |
| **SLCA-GRPO** | $\mathbf{78.57 \pm 1.15}$ | $\mathbf{69.32 \pm 0.56}$ | $\mathbf{39.90 \pm 1.46}$ |
| $\Delta$ | **+2.83** | **+1.24** | **+7.16** |

The SLCA advantage remains positive on all three benchmarks after the response-mocker replacement.

### C.4 Baseline Comparison: RLTR and ToolPO Implementation

To provide context for the comparisons in §4.2, we re-implement RLTR (Li et al., 2025) and ToolPO (Li et al., 2026b) within the same verl training framework. The matched GRPO and SLCA-GRPO runs share the controls listed in Table 6; RLTR and ToolPO retain their method-specific initialization and reward/protocol structure. The ToolPO rows are our implementations under the stated outcome-reward choices, not reproductions of the original ToolPO environment.

**SFT Initialization** For each backbone, the matched methods start from that backbone's SFT checkpoint (e.g., `qwen2.5-7B-sft`); the RLTR planner checkpoint follows its method-specific construction. RLTR requires a dedicated *Planner SFT*: we remove summary tokens from the SFT

data and append an <answer> marker to delineate the planner's output boundary, following the original paper's design.

**RLTR Implementation** We re-implement the Planner-Summarizer pipeline:

- **Reward:** $R = R_{\text{comp}} + R_{\text{repeat}} + R_{\text{error}}$, where $R_{\text{comp}}$ is an LLM-based Completeness Checker (Qwen3-30B-A3B, $N$=3 samples averaged), $R_{\text{repeat}} = -0.1 \times$ repeat_count, and $R_{\text{error}} = -0.2 \times$ error_count. Format violations yield $R = -1.0$.
- **Advantage:** Only process (tool) tokens receive non-zero advantages; summary token advantages are set to zero.
- **Evaluation:** A 2-stage pipeline: the Planner generates the tool trajectory, then a frozen SFT Summarizer produces the final answer. For BFCL, only the Planner checkpoint is used.

**ToolPO Implementation** We report a reference-rule ToolPO condition and an LLM-judge condition as a protocol-sensitivity diagnostic on 7B. The 3B and 8B ToolPO rows in the scale tables use the LLM-judge Solved/Unsolved outcome reward; no reference-rule runs are available for those scales.

- **Outcome reward protocols:** The reference-rule condition uses the available gold tool calls, while the sensitivity condition uses an LLM judge that classifies each rollout as Solved or Unsolved, denoted $R_{\text{succ}}^{\text{judge}}$. Both are evaluated on the 7B backbone below.
- $C(a_t^{\textbf{call}})$ **adaptation:** Tool-call correctness via strict matching against gold tool calls (exact name match + all gold keys present + lenient value matching). $R_{\text{action}} = \sum C/|\text{gold_flat}|$, normalized to $[0, 1]$.
- **Advantage:** $A_{\text{tool}} = A_{\text{global}} + \lambda \cdot A_{\text{tool_local}}$ for tool tokens, $A_{\text{sum}} = A_{\text{global}}$ for summary tokens, with $\lambda = 1.0$ and group-wise normalization. We adopt $\lambda$=1.0 following the original ToolPO specification (Li et al., 2026b), which gives the local action reward parity with the global outcome reward; we did not further tune $\lambda$ to avoid introducing a confound absent from the original design.

**Reference rule protocol** We implement the reference rule as an explicit gold-call conjunction using the same matching primitives as the HierR process reward. Let $\mathcal{N}_Y$ and $\mathcal{N}_G$ be the predicted and gold tool-name multisets, and let $y^* : \mathcal{G} \to \mathcal{Y}$ be the one-to-one matching used above, pairing each gold call with the best unused predicted call of the same name. Let $Y_{\text{init}}$ and $G_{\text{init}}$ denote first-round calls:

$$R_{\text{succ}}^{\text{rule}} = \mathbb{I}\Big[N_{\text{valid}} = N_{\text{open}} = N_{\text{close}},\ \mathcal{N}_Y = \mathcal{N}_G,\ |Y_{\text{init}}| = |G_{\text{init}}|, \\ \forall g \in \mathcal{G} :\ \big[\mathcal{K}_g^{\text{req}} \subseteq \mathcal{K}_{y^*(g)} \ \wedge\ \forall k \in \mathcal{K}_g^{\text{req}},\ \mathbb{I}_{\text{loose}}(y^*(g)[k], g[k]) = 1\big]\Big]. \tag{28}$$

Here $N_{\text{valid}} = N_{\text{open}} = N_{\text{close}}$ requires every parsed call to be valid with balanced opening and closing tags. The name multiset equality rejects missing or extra calls and incorrect multiplicities; $\mathcal{K}_g^{\text{req}}$ is the set of required keys in gold call $g$, and the matched-call key condition requires all of them. Values use the same case-insensitive, type-coercing matcher as HierR. The initial cardinality check enforces the parallel-call count. Cross-round call order is not checked. The conjunction reuses the gold-call checks already used by $R_{\text{action}}$, so this row is a protocol comparison rather than an independent reward source.

Table 9: ToolPO outcome reward protocol comparison on Qwen2.5-7B-Instruct. Entries are mean ± standard deviation over three matched runs. The reference rule row uses Eq. 28; Unified GRPO and SLCA-GRPO are shown as contextual matched baselines under their standard reward protocol.

| Condition | Toucan Succ. (%) | BFCL Acc. (%) | $\tau^2$-Bench Pass[1] (%) |
|---|---|---|---|
| ToolPO, LLM-judge $R_{\text{succ}}^{\text{judge}}$ | 20.00 ± 1.36 | 24.59 ± 0.87 | 30.44 ± 2.15 |
| ToolPO, reference rule $R_{\text{succ}}^{\text{rule}}$ | 77.35 ± 1.19 | 68.86 ± 0.43 | 34.18 ± 2.42 |
| Unified GRPO | 76.60 ± 1.27 | 68.41 ± 0.11 | 31.87 ± 2.97 |
| **SLCA-GRPO** | **79.13 ± 1.05** | **69.77 ± 0.47** | **41.02 ± 1.01** |

The two ToolPO rows use the same additive estimator, initialization, data, training budget, and evaluation. Replacing the LLM-judge outcome reward with the reference rule removes the observed

7B collapse in this ToolPO comparison; the rule-based ToolPO remains close to Unified GRPO and below SLCA on $\tau^2$-Bench. This comparison isolates sensitivity to the outcome-reward protocol.

**Discussion: ToolPO-7B Format Behavior** Under the LLM-judge protocol, ToolPO has low format validity on Qwen2.5-7B-Instruct (`format_passed` = 38%), whereas the 3B and 8B runs retain higher format validity. The matched GRPO and SLCA-GRPO rows use the same 7B SFT initialization, SGLS environment, and RL data; ToolPO retains its method-specific initialization, additive estimator, and outcome reward protocol. The 7B result is also reflected in the three-run mean (Toucan Success $0.200 \pm 0.014$). The training trace shows an increase in unmatched opening tags after step 80 (App. D.3). This is a diagnostic of the LLM-judge condition. The reference-rule comparison above keeps the ToolPO estimator fixed while changing the outcome reward.

**Transparency Note** We document the following necessary adaptations for reproducibility:

1. The reference-rule condition uses the available gold tool calls; the LLM-judge condition is retained as a protocol-sensitivity diagnostic.
2. RLTR's Planner SFT is trained with summary removal and `<answer>` markers, rather than using the general SFT directly.
3. RLTR's penalty coefficients ($\lambda_{\text{repeat}}$=0.1, $\mu_{\text{error}}$=0.2) are set to reasonable values, as the original paper does not specify them.
4. We do not force HierR into ToolPO's additive update, because doing so would change ToolPO's original reward protocol rather than evaluate the method-faithful baseline.
5. All methods use vanilla policy loss mode, so the policy-loss implementation is held fixed.

### C.5 Evaluation Protocols

For full reproducibility, we document the complete evaluation protocol for each benchmark, including software versions, model configurations, and scoring logic. The matched GRPO and SLCA-GRPO rows share the benchmark protocol; the ToolPO and RLTR choices are listed in App. C.4.

**$\tau^2$-Bench Protocol** We use the official $\tau^2$-Bench framework (Barres et al., 2025) without modifications to prompts, tools, tasks, or scoring logic. Table 10 summarizes the configuration.

Table 10: $\tau^2$-Bench evaluation protocol.

| Parameter | Value |
|---|---|
| `tau2` package | v0.2.1.dev0 (from `@v0.2.0` tag) |
| Evaluation framework | evalscope 1.3.0 |
| Task split | airline, retail, telecom (all 3 official domains) |
| Dataset source | `evalscope/tau2-bench-data` (ModelScope) |
| Task filtering | `LLMGTAgent.check_valid_task()` |
| Aggregation | `mean_and_pass_hat_k` (built-in) |
| Trials per task | 1 (default `repeats=1`; Pass[1]) |
| *User Simulator LLM* | |
| Model | DeepSeek-V3.2 |
| Temperature | 0.7 |
| Max tokens | 1024 |
| Timeout | 200s |
| Max retries | 8 |

**BFCL V3 Protocol** We evaluate on BFCL V3 (Patil et al., 2025) using the function-calling protocol. Table 11 lists the configuration.

Table 11: BFCL V3 evaluation protocol.

| Parameter | Value |
|---|---|
| Evaluation framework | evalscope 1.3.0 |
| Evaluation mode | `is_fc_model=True` |
| Temperature | 0 (greedy) |
| Max tokens | 8192 |
| `parallel_tool_calls` | True |
| `underscore_to_dot` | True |
| Batch size | 256 |

*Chat template alignment.* The vLLM serving endpoint uses a Toucan-specific chat template that injects a fixed system prompt and skips upstream system messages. This is a **chat template alignment** choice ensuring inference consistency with training, not a modification of the BFCL benchmark protocol. The skipped upstream system messages contain only generic formatting instructions; skipping them is a compatibility step and does not change instance-specific content or evaluation semantics. BFCL tool definitions are injected through the standard `tools` parameter in the function-calling interface, which the template processes correctly. The `underscore_to_dot=True` flag reconciles naming-convention differences between training-time schemas and BFCL's schema format. For the main BFCL V3 comparison, we exclude the relevance-detection subset and restrict evaluation to single-turn samples. These choices are held fixed for the matched GRPO and SLCA-GRPO rows; the Multi-Turn extension is reported separately below.

**Toucan In-Domain Protocol** The Toucan evaluation uses a custom asynchronous evaluation system (independent of evalscope). Table 12 lists the agent and environment configuration.

Table 12: Toucan in-domain evaluation protocol.

| Parameter | Value |
|---|---|
| Dataset | `tool_call_rl_40k_v1.parquet` (4,000 held-out) |
| Agent loop max turns | 10 |
| Temperature | 0.0 (greedy) |
| Max tokens | 12,288 |
| Top-$p$ | 1.0 |
| Concurrency | 256 |
| Retries | 3 |
| *Tool Simulator (same as SGLS)* | |
| Model | Qwen3-235B-A22B |
| Temperature | 0.6 |
| Max tokens | 4,096 |
| Timeout | 540s |
| *LLM Judge* | |
| Model | GPT-OSS-120B |
| Temperature | 0.0 (deterministic) |
| Max tokens | 8,192 |
| Scoring | 5-point categorical $\rightarrow [0, 1]$ via $(r-1)/4$ |
| Summary-judge threshold | rating $\in$ {good, excellent} ($\geq 0.75$) |

The primary Success metric in Table 1 is $\mathbb{I}[S_{\text{process}} \geq 0.9]$; it is distinct from the summary-judge threshold above.

*Process Score sub-item weights.* The $S_{\text{process}}$ metric (Eq. 22) decomposes into five sub-items: format ($w$=0.10), name ($w$=0.25), key ($w$=0.15), value ($w$=0.20), and parallel ($w$=0.30). Name matching uses multiset F1 over predicted and gold tool-name occurrences; key matching uses gold-call greedy Jaccard similarity; value matching uses lenient comparison (case-insensitive, automatic numeric type conversion).

### C.6 Details on Tool Space Scalability

We evaluate tool-space scalability via a controlled candidate-size sweep (K-sweep) under two distractor sampling strategies (Hard vs. Random). This appendix documents the dataset statistics, the K-sweep candidate-set construction procedure, the hard-negative mining mechanism, and the context-length profiling.

**Dataset and Global Tool Pool** All K-sweep datasets are derived from the Toucan evaluation set containing 4,000 instances. The global tool pool comprises 1,568 tools in total, exhibiting a long-tailed distribution of schema lengths. In the original data, the allowed-tool list contains 5.44 tools on average, and the gold-tool set size is 1.54 on average. These statistics motivate a controlled augmentation protocol to probe robustness under larger candidate sets.

**Controlled Candidate-Set Construction (K-sweep)** For each instance $i$, let $\mathcal{A}_i$ denote the original allowed tool set and $\mathcal{G}_i \subseteq \mathcal{A}_i$ denote the gold tool set. Given a target candidate size $K$, we construct a controlled candidate set $\tilde{\mathcal{A}}_i$ as follows:

- **Gold preservation:** Always retain all gold tools $\mathcal{G}_i$ in $\tilde{\mathcal{A}}_i$.
- **Original baseline ($K = 0$):** If $K = 0$, keep the instance unchanged, i.e., $\tilde{\mathcal{A}}_i = \mathcal{A}_i$.
- **Expansion ($|\mathcal{A}_i| < K$):** Add $K - |\mathcal{A}_i|$ distractor tools sampled from the global tool pool, excluding tools already present in $\mathcal{A}_i$.
- **Contraction ($|\mathcal{A}_i| > K$):** Keep all gold tools and uniformly subsample $K - |\mathcal{G}_i|$ tools from $\mathcal{A}_i \setminus \mathcal{G}_i$.

This construction enforces a precise candidate-set size while guaranteeing that the target tool(s) remain available, thereby isolating the effect of tool-space size on tool selection.

**Distractor Sampling Strategies.** We implement two strategies for selecting distractors in the expansion step:

- **Random Sampling (Control):** Distractors are drawn uniformly at random (without replacement) from the global tool pool after excluding $\mathcal{A}_i$.
- **Hard-Negative Sampling (Semantic Distractors):** We mine semantically similar distractors based on tool-description similarity. Specifically, we encode each tool's description text (tool name concatenated with its functionality docstring) using the sentence-transformer `sentence-transformers/all-MiniLM-L6-v2`, and precompute a cosine similarity matrix over the global pool. For each instance, we retrieve the most similar non-gold tools to the gold tool(s) (excluding tools already in $\mathcal{A}_i$) and take the top-scoring tools as distractors until the required budget is met. This procedure produces functionally related distractors, maximizing the probability of semantic interference.

**Context Length Profiling and Skip Policy** All token statistics below are computed using the Qwen2.5-7B-Instruct tokenizer with a maximum context length of 32,768 tokens. During K-sweep dataset generation, if an instance exceeds the context limit after tool-schema injection, it is automatically skipped. Tables 13 and 14 report empirical system-prompt length statistics and skip rates for Random and Hard strategies, respectively.

Table 13: System prompt length statistics vs. candidate set size ($K$) for **Random** distractors (Qwen2.5 tokenizer).

| Candidate Size ($K$) | Original | 20 | 50 | 100 | 150 |
|---|---|---|---|---|---|
| **Valid Samples** | 3,991 | 3,980 | 3,882 | 2,607 | 443 |
| **Avg. Tokens** | 943 | 4,614 | 11,064 | 17,177 | 23,635 |
| **Std. Dev.** | 1,114 | 3,679 | 5,256 | 1,550 | 733 |
| **Skip Rate** | 0.2% | 0.5% | 2.9% | 34.8% | 88.9% |

Table 14: System prompt length statistics vs. candidate set size ($K$) for **Hard** distractors (Qwen2.5 tokenizer).

| **Candidate Size ($K$)** | **Original** | **20** | **50** | **100** | **150** |
|---|---|---|---|---|---|
| **Valid Samples** | 3,991 | 3,959 | 3,793 | 3,023 | 2,034 |
| **Avg. Tokens** | 943 | 3,693 | 8,522 | 14,483 | 20,609 |
| **Std. Dev.** | 1,114 | 2,735 | 3,806 | 2,595 | 1,719 |
| **Skip Rate** | 0.2% | 1.0% | 5.2% | 24.4% | 49.2% |

Hard sampling tends to yield shorter prompts and lower skip rates at fixed $K$ than Random sampling, since semantically similar tools are more likely to match the length profile of tools already present in the original candidate set. We report main-paper scalability results up to $K = 100$ and treat $K = 150$ as a stress-test regime with high context-limit attrition.

# D Additional Experimental Results

## D.1 Toucan-Test: full HierR decomposition

Table 15 expands Table 1 by reporting two additional components: (i) *Parallel*, which captures the parallelism/cardinality constraints in HierR, and (ii) *Summary*, an LLM-judge score over the user-facing segment $y_{\text{sum}}$. We include RLTR and ToolPO as prior credit-assignment baselines. Reporting these terms separately is useful here: RLTR does not optimize the summary segment inside RL, while ToolPO can attain a non-trivial summary score even when tool execution degrades.

Table 15: Comprehensive Results on Toucan-Test across Scales. This table expands upon the main results by including all sub-metrics. In addition to Name F1, ArgMatch, Process, and Success (defined in Table 1), we report: Parallel: The parallelism/cardinality constraint score defined in Eq. 27; Summary: The user-facing response quality score evaluated by the LLM-judge. SLCA-GRPO has the highest mean Success and Process values across scales. RLTR and ToolPO are included for context; their method-specific protocols are described in App. C.4. RLTR's Summary column is marked "–" as its frozen summarizer is not optimized during RL. Trained rows report mean±std over three runs; Original rows are point evaluations.

| **Method** | **Name F1** | **ArgMatch** | **Parallel** | **Process** | **Summary** | **Success** |
|---|---|---|---|---|---|---|
| ***Qwen2.5-3B-Instruct*** | | | | | | |
| Original | 0.5207 | 0.5073 | 0.5412 | 0.5174 | 0.3477 | 0.3816 |
| SFT | 0.7685±.0069 | 0.7132±.0239 | 0.7772±.0071 | 0.7401±.0050 | 0.5512±.0047 | 0.6500±.0089 |
| SFT+GRPO | 0.8910±.0087 | 0.8072±.0154 | 0.9176±.0080 | 0.8577±.0057 | 0.7842±.0122 | 0.7412±.0115 |
| RLTR | 0.7598±.0149 | 0.6353±.0229 | 0.6729±.0098 | 0.6924±.0162 | – | 0.6627±.0151 |
| ToolPO | 0.8050±.0142 | 0.7769±.0160 | 0.8946±.0137 | 0.8368±.0141 | 0.6189±.0117 | 0.5128±.0145 |
| **Ours** | **0.9006**±.0087 | **0.8092**±.0112 | **0.9214**±.0071 | **0.8625**±.0049 | 0.7635±.0089 | **0.7647**±.0093 |
| ***Qwen2.5-7B-Instruct*** | | | | | | |
| Original | 0.8008 | 0.7388 | 0.8003 | 0.7687 | 0.5882 | 0.6224 |
| SFT | 0.8390±.0075 | 0.7531±.0224 | 0.8397±.0083 | 0.8094±.0082 | 0.6346±.0054 | 0.7214±.0087 |
| SFT+GRPO | 0.8971±.0093 | 0.8350±.0148 | 0.9091±.0079 | 0.8667±.0061 | 0.8241±.0108 | 0.7660±.0127 |
| RLTR | 0.7673±.0132 | 0.6346±.0205 | 0.6928±.0119 | 0.7022±.0143 | – | 0.6545±.0168 |
| ToolPO | 0.3042±.0168 | 0.3552±.0192 | 0.3438±.0143 | 0.3408±.0157 | 0.6054±.0128 | 0.2000±.0136 |
| **Ours** | **0.9164**±.0075 | **0.8353**±.0132 | **0.9251**±.0068 | **0.8766**±.0045 | **0.8450**±.0087 | **0.7913**±.0105 |
| ***Qwen3-8B-Base*** | | | | | | |
| Original | 0.0887 | 0.1158 | 0.1208 | 0.1089 | 0.0322 | 0.0471 |
| SFT | 0.8058±.0082 | 0.7437±.0260 | 0.8133±.0071 | 0.7754±.0060 | 0.6261±.0058 | 0.6925±.0098 |
| SFT+GRPO | 0.9110±.0092 | 0.8334±.0135 | 0.9287±.0065 | 0.8767±.0050 | 0.8339±.0110 | 0.7697±.0141 |
| RLTR | 0.8780±.0125 | 0.7468±.0219 | 0.7523±.0097 | 0.7941±.0147 | – | 0.7454±.0167 |
| ToolPO | 0.6854±.0168 | 0.8180±.0177 | 0.6642±.0135 | 0.7510±.0165 | 0.6789±.0137 | 0.3199±.0156 |
| **Ours** | **0.9177**±.0075 | 0.8320±.0140 | **0.9303**±.0064 | **0.8786**±.0046 | 0.8239±.0079 | **0.7902**±.0105 |

**Process and summary scores.** Across scales, SLCA-GRPO has higher mean *Process* and strict *Success* scores than matched GRPO, and higher mean *Parallel* scores. These metrics capture structural constraints beyond tool-name matching. On Qwen2.5-3B and Qwen3-8B-Base, standard GRPO attains a higher *Summary* score but a lower *Success* rate than SLCA-GRPO. This pattern is compatible with Cross-Segment Credit Misattribution: a plausible summary can coexist with erroneous tool behavior when a mixed advantage is broadcast to all tokens. Under the LLM-judge diagnostic, ToolPO shows the same separation on Qwen2.5-7B: it reaches *Summary* 0.6054 while its *Process* score is 0.3408. On Qwen2.5-7B, SLCA-GRPO has higher reported means for both execution and *Summary*; the two scores move in the same direction in this comparison.

### D.2 Summary Advantage Ablation

We isolate the learned summary-reward branch while keeping the omission safeguard unchanged. Let $\mathcal{O}$ denote rollouts that trigger the omission guard. After computing the ordinary group-normalized summary advantages, we set $\hat{A}_i^{\mathrm{sum}} = 0$ for $i \notin \mathcal{O}$ and retain the computed guard advantage for $i \in \mathcal{O}$. The summary tokens remain in the loss mask and under the per-token KL constraint. We query the judge for every rollout, including omission cases; its parameters remain frozen. For a guard rollout, the policy reward is overridden by $R_{\mathrm{sum}} = R_{\mathrm{pen}} = -0.5$.

Table 16: Summary advantage ablation on Qwen2.5-7B-Instruct. Process and Summary are unit-interval scores; benchmark columns are percentages. The no-call column is the end-of-training rate. Entries in the score columns are mean ± standard deviation over three matched runs.

| Condition | No-call (%) | Process | Toucan (%) | Summary | BFCL (%) | $\tau^2$ (%) |
|---|---|---|---|---|---|---|
| SFT | – | 0.8094 ± 0.0082 | 72.14 ± 0.87 | 0.6346 ± 0.0054 | 67.89 ± 0.28 | 32.54 ± 1.92 |
| Unified GRPO | 0.6 | 0.8667 ± 0.0061 | 76.60 ± 1.27 | 0.8241 ± 0.0108 | 68.41 ± 0.11 | 31.87 ± 2.97 |
| $\hat{A}^{\mathrm{sum}} = 0$ (except omission penalty episodes) | 0.5 | 0.8748 ± 0.0061 | 78.72 ± 1.22 | 0.6512 ± 0.0208 | 69.44 ± 0.55 | 36.24 ± 1.58 |
| $\hat{A}^{\mathrm{sum}} = 0$ (including omission penalty episodes) | 14.2 | 0.7506 ± 0.0227 | 67.54 ± 1.94 | 0.5587 ± 0.0248 | 61.53 ± 1.74 | 29.44 ± 2.28 |
| **SLCA-GRPO** | 0.4 | **0.8766 ± 0.0045** | **79.13 ± 1.05** | **0.8450 ± 0.0087** | **69.77 ± 0.47** | **41.02 ± 1.01** |
| RLTR | – | 0.7022 ± 0.0143 | 65.45 ± 1.68 | – | 63.11 ± 0.62 | 33.72 ± 2.38 |

The omission-exception condition remains close to SLCA on Process, Toucan, and BFCL, while its $\tau^2$-Bench score is lower by 4.78 pp and its Summary score is close to the SFT value. The full-zero condition is reported as a guard diagnostic: removing both the summary advantage and the guard raises the no-call rate to 14.2% and lowers all listed scores. Its no-call rollouts receive no advantage-weighted policy gradient, while the KL term remains active. The two rows are not interpreted as a comparison of the summary branch alone.

### D.3 Training Dynamics against RLTR and ToolPO

Static benchmark numbers can obscure how proxy objectives and execution quality move during training. We analyze the full training traces across all three model scales and observe different behaviors for ToolPO's additive advantage design.

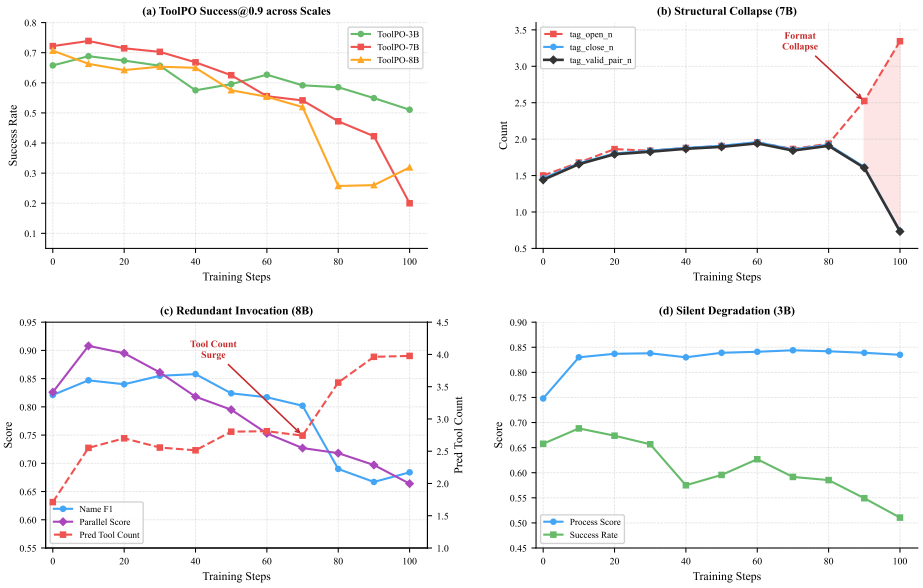



Figure 8: **ToolPO training behavior under the LLM-judge diagnostic (representative single-run traces). (a)** Success@0.9 across three scales: a decline on 3B (0.65→0.51), a sharper decline on 7B (0.72→0.20), and an overall decline with a brief mid-training rebound on 8B, ending near 0.32. **(b)** 7B: `tag_open_n` diverges from `tag_close_n` after step 80, reducing the valid parsed-call count from 1.91 to 0.73. **(c)** 8B: `pred_tool_count` surges 1.71→3.98 (gold ≈2.0), driving Name F1 and Parallel Score down. **(d)** 3B: Process Score plateaus at ∼0.84, below the $S_{\text{process}} \geq 0.9$ threshold, explaining why proxy metrics improve while strict Success declines.

**ToolPO training trace** Here `tag_valid_pair_n` denotes the number of parsed, schema-valid call blocks. It is not a count formed by pairing the opening and closing tag totals.

The traces show different failure patterns across model scales (Figure 8a):

- **3B: Silent degradation (Figure 8d).** The proxy metrics increase ($R_{\text{succ}}$: 0.24→0.50; summary: 0.56→0.62), and `process_score` rises from 0.75 to a plateau of ∼0.84. Strict Success@0.9 declines (0.65→0.51). The observed gap is consistent with the threshold $S_{\text{process}} \geq 0.9$: the Process Score remains near ∼0.84. Format integrity is preserved (`format_passed` ≈ 0.98, `tag_valid_pair_n` ≈ 2.0), so the decline is not accompanied by a structural-format change in this trace.
- **7B: Structural format decline (Figure 8b).** The trace shows more opening than closing tool-call tags (`tag_open_n`: 1.50→3.34; `tag_close_n`: 1.46→0.74), while the summary score remains non-zero. The valid parsed-call count drops from 1.91 to 0.73, and Success@0.9 drops from 0.72 to 0.20. The change is concentrated between steps 80–100, when the gap between `tag_open_n` and `tag_close_n` widens to 2.60.
- **8B: Redundant tool invocation (Figure 8c).** Qwen3-8B-Base keeps `tag_open_n` and `tag_close_n` close in this trace. The trace instead shows more predicted tool calls (`pred_tool_count`: 1.71→3.98, gold ≈2.0). Success@0.9 declines overall, with a brief mid-training rebound, and ends near ∼0.32 as redundant calls accumulate. At steps 70–80, `pred_tool_count` exceeds 3.5 and Name F1 falls from 0.80 to 0.69.

These observations are specific to the LLM-judge diagnostic and do not imply that ToolPO always fails. They motivate the separate matched GRPO comparison, where the summary-to-tool support path is blocked by SLCA's routing.

**RLTR: Coarse-Grained Reward and Training Stagnation** As visualized in Figure 2a, RLTR avoids direct summary-to-tool leakage via pipeline separation, but its planning reward remains sequence-level and coarse. In our runs, `rltr_comp` oscillates between 0.49 and 0.64, while *Process* and *Parallel* improve much more slowly than under SLCA-GRPO. On Qwen3-8B-Base, $R_{\text{comp}}$ doubles (0.22→0.43) while `error_count` increases from 0.39 to 0.59, suggesting that the

completeness reward does not penalize structurally invalid calls (cf. the frozen-summarizer limitation noted in Table 15).

### D.4 Visualizing Credit Misattribution

Figure 9 presents a diagnostic case on a parallel information-retrieval task (comparing the current stock prices of Apple and Microsoft), illustrating how Cross-Segment Credit Misattribution manifests as *tool-use inefficiency masked by a perfect summary*. The reference trajectory consists of a single parallel `get_stock_price` invocation covering both tickers, and the tool-level reward is decomposed as $R_{\text{tool}} = 0.10\,R_{\text{format}} + 0.25\,R_{\text{name}} + 0.15\,R_{\text{key}} + 0.20\,R_{\text{value}} + 0.30\,R_{\text{parallel}}$.

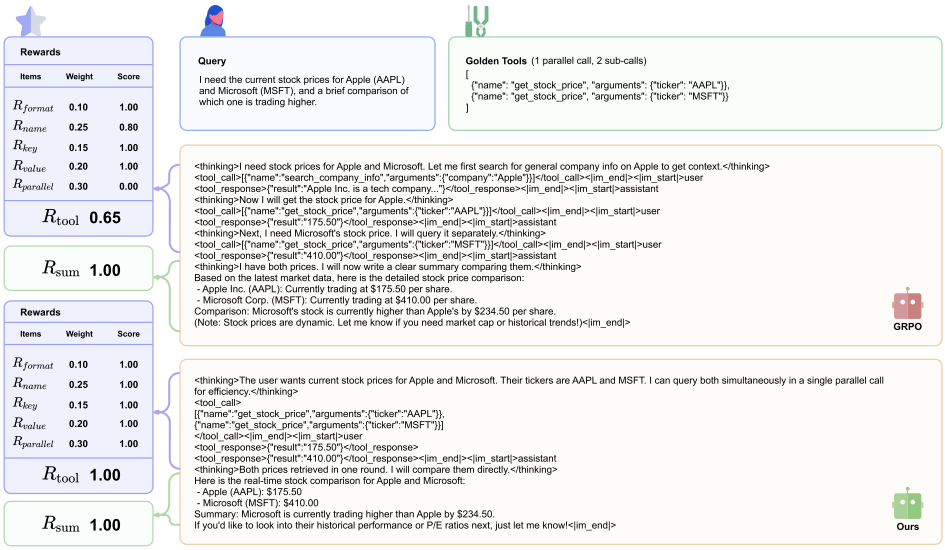



Figure 9: **Diagnostic Case Study: Resolving Cross-Segment Credit Misattribution on a parallel tool-use task. (Top) Reference trajectory and rewards. (Middle) Standard GRPO.** The agent adopts a redundant, trial-and-error trajectory (first issuing an unnecessary `search_company_info` call, then retrieving the two stock prices through *serial* `get_stock_price` invocations in separate turns), yet still produces a fluent, factually correct comparison ($R_{\text{sum}}$=1.00). The tool-level decomposition exposes where the trajectory fails: the `format` reward is unaffected (1.00), the `key` and `value` rewards remain high on the individual calls that do match the schema (1.00 each), the `name` reward is diluted by the redundant `search_company_info` call (0.80), and the `parallel` reward is **0.00** because the two required stock-price queries are issued as separate sequential turns (further preceded by a redundant exploration call), rather than batched into a single parallel invocation. Aggregated under the fixed weights above, this yields $R_{\text{tool}}$=0.65. Because standard GRPO merges $R_{\text{tool}}$ and $R_{\text{sum}}$ into a single trajectory-level advantage, the high summary reward can offset the structural tool penalty in the unified advantage, allowing the inefficient pattern to be reinforced. **(Bottom) SLCA-GRPO (Ours).** The same model, trained under our advantage-isolation objective, produces a trajectory in which both tickers are known a priori and issues a *single parallel tool call* covering AAPL and MSFT simultaneously. All five tool sub-rewards saturate at 1.00, giving $R_{\text{tool}}$=1.00 while preserving $R_{\text{sum}}$=1.00. By routing $A_{\text{tool}}$ and $A_{\text{sum}}$ through separate optimization pathways, segment routing keeps the summary signal from entering the tool-token update in this example. The tool-execution score increases by $+0.35$ while the summary score remains 1.00. The trajectory is qualitatively similar to the representative single-run `pred_tool_count` increase observed under ToolPO on 8B (Figure 8c); the two diagnostics use different protocols.

### D.5 BFCL: Robustness in Multi-Turn Scenarios

While the main paper (Figure 3a) focuses on atomic schema generalization, Table 17 additionally reports Multi-Turn accuracy, now including RLTR and ToolPO. On Qwen3-8B-Base, standard GRPO

has a lower Multi-Turn mean than SFT (0.0822±0.0035 vs. 0.1544±0.0102), while SLCA-GRPO reaches 0.1489±0.0025. The SLCA mean is close to the SFT value in this three-run comparison; the table does not isolate the source of the Multi-Turn difference. On Qwen2.5-3B-Instruct, SLCA-GRPO also remains below SFT on Multi-Turn (0.0622±0.0025 vs. 0.0867±0.0100). RLTR remains competitive on AST Non-Live, suggesting that planner-only RL can learn basic tool selection under unseen schemas, but it lags on Multi-Turn where end-to-end coordination matters. Under the reported ToolPO evaluation, the Multi-Turn means are 0.0961±0.0067 on 3B and 0.0083±0.0067 on 8B. This spread is reported as context rather than as a claim about the original method across model scales.

Table 17: Comprehensive Generalization Results on BFCL V3 across Scales. This table details BFCL generalization performance. In addition to Overall Acc, AST Live, and AST Non-Live (reported in Figure 3a), we include Multi-Turn accuracy to assess state tracking under schema generalization. Trained rows report mean±std over three runs; Original rows are point evaluations. ToolPO and RLTR retain their method-specific protocols.

| Method | Overall Acc | AST Live | AST Non-Live | Multi-Turn |
|---|---|---|---|---|
| ***Qwen2.5-3B-Instruct*** | | | | |
| Original | 0.4225 | 0.5174 | 0.5139 | 0.0271 |
| SFT | 0.6163±.0026 | 0.6862±.0022 | 0.8104±.0043 | 0.0867±.0100 |
| SFT+GRPO | 0.6321±.0032 | 0.6936±.0030 | 0.8583±.0087 | 0.0600±.0033 |
| RLTR | 0.5838±.0065 | 0.6136±.0074 | 0.8275±.0044 | 0.0494±.0100 |
| ToolPO | 0.6095±.0086 | 0.6477±.0108 | 0.8325±.0139 | 0.0961±.0067 |
| **OURS** | **0.6361**±.0052 | **0.7027**±.0070 | 0.8574±.0083 | 0.0622±.0025 |
| ***Qwen2.5-7B-Instruct*** | | | | |
| Original | 0.6490 | 0.7446 | 0.8130 | 0.1172 |
| SFT | 0.6789±.0028 | 0.7595±.0053 | 0.8635±.0012 | 0.1435±.0120 |
| SFT+GRPO | 0.6841±.0011 | 0.7783±.0026 | 0.8548±.0074 | 0.1443±.0034 |
| RLTR | 0.6311±.0062 | 0.6840±.0078 | 0.8591±.0053 | 0.0750±.0094 |
| ToolPO | 0.2459±.0087 | 0.2043±.0103 | 0.4096±.0118 | 0.0251±.0056 |
| **OURS** | **0.6977**±.0047 | **0.7820**±.0068 | **0.8739**±.0086 | **0.1550**±.0024 |
| ***Qwen3-8B-Base*** | | | | |
| Original | 0.2403 | 0.2435 | 0.3617 | 0.0000 |
| SFT | 0.6850±.0035 | 0.7779±.0015 | 0.8528±.0031 | 0.1544±.0102 |
| SFT+GRPO | 0.6696±.0032 | 0.7750±.0027 | 0.8522±.0071 | 0.0822±.0035 |
| RLTR | 0.6610±.0065 | 0.7197±.0078 | 0.8919±.0053 | 0.0861±.0102 |
| ToolPO | 0.6241±.0090 | 0.6827±.0096 | 0.8765±.0104 | 0.0083±.0067 |
| **OURS** | **0.7031**±.0050 | **0.7856**±.0082 | **0.8954**±.0083 | 0.1489±.0025 |

### D.6 ADDITIONAL ANALYSIS ON $\tau^2$-BENCH RESULTS

**Baseline Configurations and Omissions** The Qwen3-8B-Base *Original* baseline has a 0.00 Pass[1] rate in every domain; it is shown explicitly in Table 18. The SFT baseline serves as the starting point for the 8B scale comparisons.

Table 18: Comprehensive Collaboration Results on $\tau^2$-Bench across Scales. We report Pass[1] rates for dual-control scenarios across three distinct domains: Airline (User Retention), Retail (Order Modification), and Telecom (Plan Management). Pass[1] is the proportion of successful tasks in the finite official task set; because each run evaluates a finite set, the reported values are discrete proportions and their standard deviations reflect variation of those proportions across runs. The Qwen3-8B-Base Original row is included and is zero in all four reported metrics. Trained rows are mean±std over three runs. ToolPO and RLTR retain their method-specific protocols.

| Method | Overall Pass[1] | Airline | Retail | Telecom |
|---|---|---|---|---|
| ***Qwen2.5-3B-Instruct*** | | | | |
| Original | 0.2324 | 0.3810 | 0.1481 | 0.2692 |
| SFT | 0.2705±.0236 | 0.1364±.0227 | 0.3966±.0259 | 0.1954±.0217 |
| SFT+GRPO | 0.3454±.0257 | 0.2576±.0131 | 0.4598±.0217 | 0.2644±.0348 |
| RLTR | 0.2850±.0291 | 0.2727±.0227 | 0.3649±.0303 | 0.2098±.0303 |
| ToolPO | 0.2355±.0226 | 0.1288±.0262 | 0.3678±.0263 | 0.1437±.0179 |
| **OURS** | **0.3563**±.0091 | **0.2576**±.0347 | 0.4569±.0172 | **0.2931**±.0172 |
| ***Qwen2.5-7B-Instruct*** | | | | |
| Original | 0.3023 | 0.3023 | 0.4054 | 0.2018 |
| SFT | 0.3254±.0192 | 0.2381±.0227 | 0.4167±.0253 | 0.2647±.0208 |
| SFT+GRPO | 0.3187±.0297 | 0.3375±.0177 | 0.4221±.0260 | 0.2124±.0375 |
| RLTR | 0.3372±.0238 | 0.2195±.0271 | 0.4001±.0302 | 0.3182±.0264 |
| ToolPO | 0.3044±.0215 | 0.2000±.0243 | 0.4286±.0278 | 0.2208±.0231 |
| **OURS** | **0.4102**±.0101 | 0.2827±.0380 | **0.5323**±.0203 | **0.3380**±.0190 |
| ***Qwen3-8B-Base*** | | | | |
| Original | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| SFT | 0.3104±.0200 | 0.2273±.0227 | 0.4540±.0303 | 0.1983±.0259 |
| SFT+GRPO | 0.3466±.0236 | 0.2652±.0131 | 0.4425±.0303 | 0.2816±.0303 |
| RLTR | 0.3490±.0218 | 0.2273±.0227 | 0.4282±.0217 | 0.3161±.0217 |
| ToolPO | 0.4432±.0186 | 0.2197±.0262 | 0.5891±.0303 | 0.3822±.0217 |
| **OURS** | **0.4469**±.0084 | **0.3561**±.0473 | 0.5374±.0217 | **0.3908**±.0217 |

**Domain-Specific Performance Analysis** The SLCA-GRPO mean is higher than the matched GRPO mean on Overall Pass[1] for all three backbones. Domain-level orderings vary by scale: on Qwen2.5-3B-Instruct, SLCA-GRPO ties GRPO on Airline, trails it on Retail, and leads on Telecom; RLTR has the highest Airline mean. On Qwen2.5-7B-Instruct, the Airline mean is below GRPO while Retail and Telecom are higher. On Qwen3-8B-Base, SLCA-GRPO leads on Airline and Telecom, while ToolPO has the highest Retail mean.

The inclusion of RLTR and ToolPO reveals additional patterns. RLTR trails SLCA on 7B (0.3372 vs. 0.4102) and 3B (0.2850 vs. 0.3563), constrained by its frozen summarizer in tasks requiring full conversational agent loops. On Qwen3-8B-Base, the reported ToolPO and SLCA-GRPO Overall means are 44.32±1.86% and 44.69±0.84%, a difference of 0.37 pp. ToolPO has the higher Retail mean (58.91±3.03% vs. 53.74±2.17%), while SLCA-GRPO is higher on Airline and Telecom. These results show that the relative ordering depends on the domain and protocol.

### D.7 HierR Weight Sensitivity Analysis

To examine sensitivity to HierR weight choices, we evaluate three $S_{\text{process}}$ weight configurations on Qwen2.5-7B-Instruct (three-run mean±std), summarized in Table 19.

Table 19: HierR weight sensitivity on Qwen2.5-7B-Instruct. Entries are mean±std over three runs.

| Config | $S_{\text{process}}$ Weights | | | | | BFCL Acc (%) | $\tau^2$ Pass[1] (%) |
|---|---|---|---|---|---|---|---|
| | $S_{\text{fmt}}$ | $S_{\text{name}}$ | $S_{\text{key}}$ | $S_{\text{val}}$ | $S_{\text{par}}$ | | |
| **Default (SLCA)** | 0.10 | 0.25 | 0.15 | 0.20 | 0.30 | **69.77±0.47** | **41.02±1.01** |
| Uniform (SLCA) | 0.20 | 0.20 | 0.20 | 0.20 | 0.20 | 69.18±0.55 | 38.62±1.86 |
| Value-heavy (SLCA) | 0.05 | 0.20 | 0.20 | 0.35 | 0.20 | 69.44±0.48 | 39.35±1.73 |
| w/o HierR | — | | | | | 69.07±0.46 | 34.50±1.44 |
| GRPO (unified adv.) | same as Default | | | | | 68.41±0.11 | 31.87±2.97 |

The Uniform configuration sets $S_{\text{par}} = 0.20$ and remains above the matched GRPO baseline on BFCL and $\tau^2$-Bench (+0.77 pp and +6.75 pp, respectively). The Default configuration has the highest values among the tested settings.

### D.8 Unified Reward Ratio Sensitivity

We vary the relative weight of the two raw rewards in the unified GRPO estimator,

$$\hat{A}_i^{\text{uni}}(\alpha) = \text{Norm}\big(\alpha R_i^{\text{tool}} + R_i^{\text{sum}}\big), \qquad \alpha = \frac{\lambda_{\text{tool}}}{\lambda_{\text{sum}}}, \tag{29}$$

while keeping the SFT initialization, data split, SGLS, HierR components, rollout budget, and evaluation protocol fixed. The displayed ratios include the existing 1:1 condition and the three additional tested ratios.

Table 20: Unified reward ratio sensitivity on Qwen2.5-7B-Instruct. Entries are mean ± standard deviation over three matched runs. The ratio is applied before the unified group normalization; the scale-invariance identity holds up to the numerical floor described in App. A.3.

| Reward ratio | Toucan Success@0.9 (%) | BFCL Acc. (%) | $\tau^2$-Bench Pass[1] (%) |
|---|---|---|---|
| Unified 1:1 | 76.60 ± 1.27 | 68.41 ± 0.11 | 31.87 ± 2.97 |
| Unified 2:1 | 77.54 ± 1.24 | 68.83 ± 0.31 | 34.62 ± 2.71 |
| **Unified** 3:1 | **77.88 ± 1.30** | **68.95 ± 0.36** | **35.24 ± 2.58** |
| Unified 5:1 | 77.02 ± 1.41 | 68.60 ± 0.44 | 32.90 ± 2.88 |
| **SLCA-GRPO** | **79.13 ± 1.05** | **69.77 ± 0.47** | **41.02 ± 1.01** |

The unified reward ratio sweep recovers part of the SLCA gap within the tested ratios. The best tested unified ratio remains below SLCA on BFCL and $\tau^2$-Bench, while increasing the tool weight to 5:1 reduces the $\tau^2$-Bench score relative to the best tested ratio. The pattern indicates a trade-off between tool and summary rewards in the unified estimator rather than a single uniformly favorable weight.

### D.9 Support Control on Qwen2.5-7B-Instruct

We evaluate four support conditions at the primary 7B scale. Let $T = \lambda_{\text{tool}} \text{Norm}(R_{\text{tool}})$ and $S = \lambda_{\text{sum}} \text{Norm}(R_{\text{sum}})$, with both weights set to one. The four cells differ only in which normalized components reach each token segment; additive sums are left unrescaled, as in the tested estimator. All cells use the same initialization, data, rollout budget, and evaluation protocol. Unified is shown as a joint-normalization reference and is not part of the factorial effects.

Table 21: Support control on Qwen2.5-7B-Instruct. The four support cells use separately normalized components. Unified is a joint-normalization reference. Entries are mean ± standard deviation over three matched runs.

| Condition | Tool tokens | Summary tokens | Toucan (%) | BFCL (%) | $\tau^2$ (%) |
|---|---|---|---|---|---|
| **Both closed (SLCA)** | $T$ | $S$ | 79.13 ± 1.05 | 69.77 ± 0.47 | 41.02 ± 1.01 |
| $S \to T$ open | $T + S$ | $S$ | 77.21 ± 1.24 | 68.72 ± 0.29 | 33.95 ± 2.71 |
| $T \to S$ open | $T$ | $T + S$ | 79.02 ± 1.12 | 69.84 ± 0.44 | 41.36 ± 1.24 |
| Both open | $T + S$ | $T + S$ | 78.30 ± 1.09 | 69.30 ± 0.41 | 37.42 ± 2.10 |
| Unified (joint normalization) | joint normalized reward | joint normalized reward | 76.60 ± 1.27 | 68.41 ± 0.11 | 31.87 ± 2.97 |

The factorial effects are computed from the four support cells rather than from a diagonal contrast:

Table 22: Factor effects from the 7B support control. Values are computed from the cell means in Table 21 and are reported in percentage points. Main effects average the two simple effects over the other factor; interaction entries use the unhalved difference-in-differences contrast.

| Effect | Toucan | BFCL | $\tau^2$-Bench |
|---|---|---|---|
| Close $S \to T$ | 1.32 | 0.80 | 5.51 |
| Open $T \to S$ | 0.49 | 0.33 | 1.91 |
| Interaction | 1.20 | 0.51 | 3.13 |

Closing the $S \to T$ path gives the larger estimated effect on all three benchmarks in this control. Opening the $T \to S$ path has a smaller effect, and the difference between that condition and SLCA is $-0.11/+0.07/+0.34$ pp on Toucan, BFCL, and $\tau^2$-Bench. These results are support control effects under the tested additive estimator; they are not presented as a routing-only causal estimate.

### D.10 Full Ablation Studies Across Scales

**Component contributions across scales.** Table 23 reports ablation results across the three backbones. The w/o SLCA rows have lower reported Toucan Success means by 2.05 to 2.53 pp across the three backbones. The matched BFCL and $\tau^2$-Bench means are also lower in the reported blocks; the 3B BFCL gap is 0.40 pp. The w/o HierR rows show the largest Toucan Success drops in each block and lower means on most other Toucan metrics. Removing deterministic schema validation and schema-conditioned tool-response prompting has a smaller effect on in-domain Success; it also changes BFCL and $\tau^2$-Bench, with the largest BFCL drop at 8B.

### D.11 Scaling to Large Tool Spaces Across Families

Table 24 extends the scalability analysis across all three backbones, including an extreme stress test at $K = 150$. The table compares the two sampling strategies at each candidate-set size:

- **Semantic interference:** Across all models, performance drops under Hard-Negative sampling compared to Random sampling (e.g., Qwen3-8B drops from ∼60% to ∼36% at $K = 100$). The stress test makes semantically similar tools harder to distinguish than random candidates at the same set size.
- **Effect of context limits:** At $K = 150$ for Qwen2.5-7B, the Random-sampling rows are near-tied, while the Hard-sampling gap is +2.50 pp. These results show how candidate-set construction and context length interact with the estimator.
- **Across tested families:** SLCA-GRPO is higher than GRPO in the reported Hard-sampling rows (e.g., +2.25 pp for 3B at $K = 150$).

Table 23: Comprehensive Ablation Studies Across Model Families. Comparison of Full SLCA-GRPO against variants removing key components. Metrics include Toucan-Test (Name F1, ArgMatch, Process, Success), BFCL (Accuracy), and $\tau^2$-Bench (Pass[1]). Process is computed as a post-hoc evaluation metric for all settings, including w/o HierR. Trained rows in all three blocks report mean±std over three runs; Original evaluations are point values where shown. The w/o SLCA condition is the SFT+GRPO estimator.

| Method | Toucan-Test | | | | BFCL | $\tau^2$-Bench |
|---|---|---|---|---|---|---|
| | Name F1 | ArgMatch | Process | Success | Acc | Pass[1] |
| ***Qwen2.5-3B-Instruct*** | | | | | | |
| **SLCA-GRPO** | **0.9006**±.0087 | **0.8092**±.0112 | **0.8625**±.0049 | **0.7647**±.0093 | **0.6361**±.0052 | **0.3563**±.0091 |
| w/o SLCA | 0.8910±.0087 | 0.8072±.0154 | 0.8577±.0057 | 0.7412±.0115 | 0.6321±.0032 | 0.3454±.0257 |
| w/o SGLS | 0.8927±.0075 | 0.7982±.0109 | 0.8559±.0050 | 0.7584±.0101 | 0.6304±.0048 | 0.2995±.0182 |
| w/o HierR | 0.8399±.0084 | 0.7850±.0161 | 0.8264±.0075 | 0.6552±.0107 | 0.5993±.0046 | 0.3152±.0145 |
| ***Qwen2.5-7B-Instruct*** (three-run mean±std) | | | | | | |
| **SLCA-GRPO** | **0.9164**±.0075 | **0.8353**±.0132 | 0.8766±.0045 | **0.7913**±.0105 | **0.6977**±.0047 | **0.4102**±.0101 |
| w/o SLCA | 0.8971±.0093 | 0.8350±.0148 | 0.8667±.0061 | 0.7660±.0127 | 0.6841±.0011 | 0.3187±.0297 |
| w/o SGLS | 0.9163±.0081 | 0.8298±.0126 | 0.8787±.0051 | 0.7802±.0112 | 0.6875±.0048 | 0.3619±.0195 |
| w/o HierR | 0.9167±.0089 | 0.8284±.0138 | 0.8521±.0083 | 0.7344±.0121 | 0.6907±.0046 | 0.3450±.0144 |
| ***Qwen3-8B-Base*** | | | | | | |
| **SLCA-GRPO** | **0.9177**±.0075 | 0.8320±.0140 | **0.8786**±.0046 | **0.7902**±.0105 | **0.7031**±.0050 | **0.4469**±.0084 |
| w/o SLCA | 0.9110±.0092 | 0.8334±.0135 | 0.8767±.0050 | 0.7697±.0141 | 0.6696±.0032 | 0.3466±.0236 |
| w/o SGLS | 0.9123±.0084 | 0.8276±.0110 | 0.8751±.0058 | 0.7832±.0127 | 0.6704±.0047 | 0.3853±.0146 |
| w/o HierR | 0.8985±.0105 | 0.8215±.0121 | 0.8648±.0075 | 0.7496±.0139 | 0.6603±.0045 | 0.3647±.0146 |

Table 24: Comprehensive Scalability Results Across Model Families. We report **Success@0.9** as the candidate set size $K$ increases. Comparison is made between Hard-Negative (Semantic) and Random sampling to isolate the impact of semantic interference. The *Original* column reports the multi-run mean at $K = 0$, matching the main results; the $K > 0$ columns retain the stress-test evaluations. For $K \geq 100$, especially $K = 150$, these stress-test values are computed on context-valid survivors after skip filtering.

| | Candidate Tool Set Size ($K$) | | | | |
|---|---|---|---|---|---|
| **Method** | **Original** | **20** | **50** | **100** | **150** |
| **Hard-Negative Sampling** | | ***Qwen2.5-3B-Instruct*** | | | |
| SFT+GRPO | 0.7412 | 0.3400 | 0.2921 | 0.2662 | 0.2285 |
| **OURS** | **0.7647** | **0.3584** | **0.3209** | **0.2865** | **0.2510** |
| **Random Sampling** | | ***Qwen2.5-3B-Instruct*** | | | |
| SFT+GRPO | 0.7412 | 0.6399 | 0.5701 | 0.4950 | 0.4360 |
| **OURS** | **0.7647** | **0.6637** | **0.5854** | **0.5050** | **0.4413** |
| **Hard-Negative Sampling** | | ***Qwen2.5-7B-Instruct*** | | | |
| SFT+GRPO | 0.7660 | 0.3650 | 0.3325 | 0.3111 | 0.2570 |
| **OURS** | **0.7913** | **0.3862** | **0.3452** | **0.3304** | **0.2820** |
| **Random Sampling** | | ***Qwen2.5-7B-Instruct*** | | | |
| SFT+GRPO | 0.7660 | 0.6749 | 0.6060 | 0.5477 | 0.5155 |
| **OURS** | **0.7913** | **0.7030** | **0.6377** | **0.5776** | 0.5129 |
| **Hard-Negative Sampling** | | ***Qwen3-8B-Base*** | | | |
| SFT+GRPO | 0.7697 | 0.3976 | 0.3615 | 0.3508 | 0.3080 |
| **OURS** | **0.7902** | **0.4143** | **0.3662** | **0.3601** | **0.3135** |
| **Random Sampling** | | ***Qwen3-8B-Base*** | | | |
| SFT+GRPO | 0.7697 | 0.7044 | 0.6468 | 0.5775 | 0.5729 |
| **OURS** | **0.7902** | **0.7312** | **0.6753** | **0.6029** | **0.5844** |

### D.12 Execution-Based Reward Experiment

To test whether SLCA's gains extend beyond gold-trajectory matching rewards, we replace $R_{\text{tool}} = S_{\text{process}}$ with execution-based $R_{\text{succ}}$. $R_{\text{sum}} = S_{\text{summary}}$ remains unchanged. Both methods use Qwen2.5-7B-Instruct with identical data, SGLS, and compute.

For this test, the binary $R_{\text{succ}}$ judge labels each rollout as Solved or Unsolved without per-example gold-call matching. Table 25 reports the matched comparison.

Table 25: Execution-based $R_{\text{succ}}$ comparison on Qwen2.5-7B-Instruct. Entries are mean $\pm$ standard deviation over three runs; data, initialization, SGLS, and compute are matched.

| **Method** | **Toucan Succ. (%)** | **BFCL Acc. (%)** | **$\tau^2$-Bench Pass[1] (%)** |
|---|---|---|---|
| SFT+GRPO (w/o SLCA) | $73.51 \pm 1.45$ | $66.28 \pm 0.32$ | $25.54 \pm 2.13$ |
| **SLCA-GRPO** | $\mathbf{76.43 \pm 1.21}$ | $\mathbf{67.61 \pm 0.51}$ | $\mathbf{30.18 \pm 1.76}$ |
| $\Delta$ | **+2.92** | **+1.33** | **+4.64** |

The SLCA advantage persists under the execution-based reward. Absolute scores are lower than under dense HierR rewards because the execution signal is sparser.

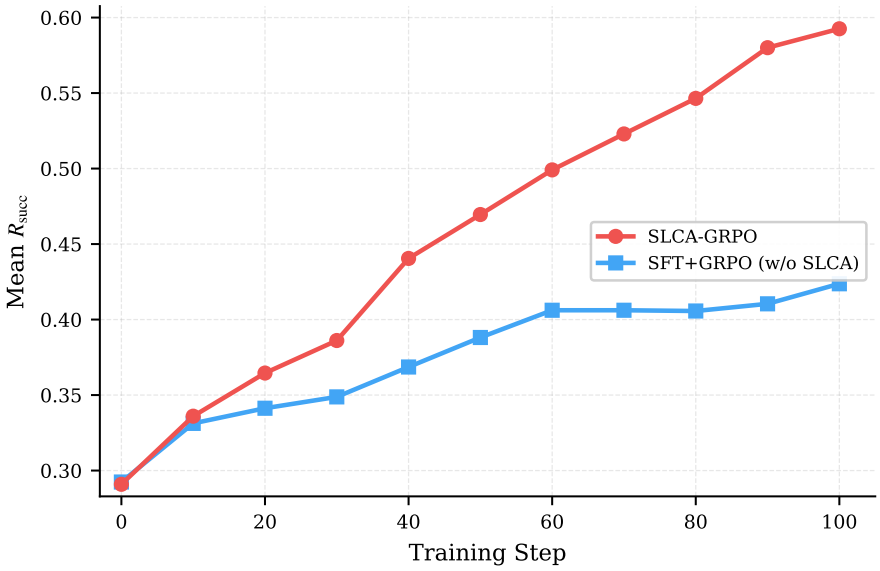



Figure 10: **Training dynamics under execution-based $R_{\text{succ}}$ (7B).** The curves are single-run checkpoint evaluations on the same fixed set of 3,991 valid Toucan validation examples. The plotted $R_{\text{succ}}$ is distinct from Toucan Success@0.9. At step 100, SLCA-GRPO reaches $R_{\text{succ}}$=0.593, compared with 0.424 for SFT+GRPO (w/o SLCA).

**Training Dynamics under $R_{\text{succ}}$** The $R_{\text{succ}}$ learning curve (Figure 10) reveals a clear divergence: at step 0, both methods start at $R_{\text{succ}} \approx 0.29$; by step 100, SLCA-GRPO reaches 0.592583 (2365/3991), whereas the SFT+GRPO (w/o SLCA) baseline reaches 0.423703 (1691/3991), a +16.9 pp gap. The logged validation series is `val-aux/toucan_eval_v4/reward/tool_call/r_succ/mean@1`; each checkpoint reports the exact empirical proportion on the same fixed set of 3,991 examples from one run per method, rather than an across-run mean. This execution-based metric is separate from Toucan's strict Success@0.9 measure.

# E PROMPTS

## E.1 SUMMARY JUDGE PROMPTS

```
## Task
Evaluate the quality of an AI assistant's final response based on the
    tool results it received.

## Assessment Criteria
### Response Quality (1-5 scale)
- 1 (very poor): Response is completely wrong, irrelevant, or ignores
    tool results
- 2 (poor): Seriously misinterprets tool results or misses critical
    information
- 3 (acceptable): Basically correct but has minor errors or incomplete
    coverage
- 4 (good): Correctly uses tool results, response is clear and complete
- 5 (excellent): Perfect interpretation of results, professional and
    helpful response

## User Question
```
{question_content}
```

## Tool Results
```
{tool_responses}
```

## Model's Final Response
```
{final_response}
```

## Evaluation Points
1. **Result Interpretation**: Does the model correctly understand and
    explain the data returned by tools?
2. **Information Completeness**: Does the response address all parts of
    the user's question based on the tool results?
3. **Expression Quality**: Is the response clear, well-organized, and
    helpful to the user?

## Output Format
<response>
  <response_quality>
    <reasoning>Brief analysis of the response quality based on the three
    evaluation points above.</reasoning>
    <rating><!-- one of: very poor, poor, acceptable, good, excellent
    --></rating>
  </response_quality>
</response>
```

## E.2 SGLS SIMULATOR PROMPTS

```
# CRITICAL ROLE: You are an API SERVER, NOT an AI assistant or agent.

You are role-playing as a **backend API server** that executes function
    calls and returns results.
You are NOT making tool calls - you are RESPONDING to them as if you
    were the actual API endpoint.

## YOUR ONLY OUTPUT FORMAT:
```

```
<tool_response>{"key": "value", ...}</tool_response>

## ABSOLUTE PROHIBITIONS:
- NEVER output <tool_call> tags (you are the SERVER, not the client)
- NEVER output markdown code blocks
- NEVER output explanations or conversational text
- ONLY output the <tool_response> tag with JSON inside

## Examples of CORRECT server responses:

Input: calculator(expression="124 * 5")
<tool_response>{"result": 620}</tool_response>

Input: get_weather(city="Seattle")
<tool_response>{"temp": "12$^\circ$C", "condition":
    "Rainy"}</tool_response>

Input: google_search(query="capital of Australia")
<tool_response>{"snippets": ["Canberra is the capital city of
    Australia."]}</tool_response>

---
## NOW EXECUTE THIS API CALL:

Tool Definition:
%s

Incoming Request:
%s

Return the execution result in <tool_response>...</tool_response> format:
```