State-Aware Policy Optimization for a Reliable Multi-Turn, Multi-Tool Scientific Agent in Kinetic Biological Models

Ansh Kumar [1] **Sanjana Balaji Kuttae** [1] **Charlie George Barker** [1]
Thomas Rückle [1] **Gurdeep Singh**[†] [1] **Ahmad Wisnu Mulyadi**[†] [1]

Anastasios Siokis [2] **Sven Jager** [3] **Thomas Klabunde** [2] **Tommaso Andreani**[†] [2]

Abstract

Large language model (LLM)-based agents show promise for scientific tasks requiring multi-turn reasoning with multi-tool use, but often fail due to error propagation, where early mistakes cascade through downstream steps. In kinetic biological modeling, including quantitative systems pharmacology (QSP), multi-step workflows involve simulating and analyzing complex dynamical systems. Standard reinforcement learning (RL) assumes stateless input, yet these scientific workflows require stateful simulation backends that preserve environment state across tool calls. We address this limitation through a state-aware Group Relative Policy Optimization (GRPO) framework that decomposes multi-turn interactions into per-turn episodes while reconstructing environment state by replaying prior tool calls on a live simulation backend. Combined with a hybrid reward function consisting of deterministic verification and a frozen LLM judge, our approach enables verifiable, sample-efficient optimization. We apply this to Talk2BioModels (T2B), an open-source agent for kinetic biological models. Using only a compact Qwen2.5-3B-Instruct backbone, our optimized agent achieves 98% tool correctness, 88% argument correctness, and 77% task completion, outperforming 24 frontier and open-source baselines on a 324-turn benchmark spanning 10 scenarios derived from 20 biological models.

[1]Virtual Patient Engine, BioMed X Germany GmbH, Heidelberg, Germany [2]Translational Medicine Unit (TMU) - Disease Modeling, Sanofi R&D, Frankfurt am Main, Germany [3]Digital R&D, Data & Computational Science, Sanofi Digital, Frankfurt am Main, Germany. [†]Correspondence to: Ahmad Wisnu Mulyadi <mulyadi@bmedx.com>, Gurdeep Singh <gsingh@bmedx.com>, Tommaso Andreani <tommaso.andreani@sanofi.com>.

Published in the AI for Science workshop (ICML 2026). Copyright 2026 by the author(s).

1. Introduction

The rapid development of large language models (LLMs) has enabled agentic AI frameworks capable of structured reasoning and iterative tool use, allowing them to solve complex, multi-step tasks (Singh et al., 2025b; Huang et al., 2025; Gao et al., 2025). In scientific domains, such systems offer the potential to act as AI-driven scientific agents, providing conversational access to computational analyses. This is particularly relevant in kinetic biological modeling (*e.g.*, quantitative systems pharmacology, QSP), where researchers rely on mathematical models to describe complex biological processes. These models are often formulated as ordinary differential equation (ODE) systems that capture biochemical reaction networks, cellular signaling pathways, and pharmacokinetic/pharmacodynamic relationships. Working with such models requires a high degree of rigor and reliability when interacting with simulation tools (Saini & Farnoud, 2025; Shahin et al., 2025), including precise parameter specification, careful state management, and multi-step analysis, where errors can propagate across steps.

Current agentic systems struggle in such settings, especially over long, multi-turn interactions. While small language models (SLMs) may suffice for repetitive tasks (Belcak et al., 2025), they often exhibit incomplete execution in long-horizon, multi-turn interactions (Wan et al., 2025). Common failure modes include incorrect tool selection, argument parsing errors, insufficient tool documentation, and limited context management, all of which can propagate errors and undermine reasoning capabilities (Xiong et al., 2025; Shah et al., 2026). Addressing these limitations requires improving both reasoning and decision-making during tool use. Supervised fine-tuning (SFT) enables task-specific behavior and improves instruction following (Wei et al., 2022; Chung et al., 2024; Pareja et al., 2024; Gao et al., 2025). Reinforcement learning from human feedback (RLHF), in contrast, optimizes behavior using reward signals, often via actor-critic methods such as proximal policy optimization (PPO) (Christiano et al., 2017; Schulman et al., 2017). Alternative approaches, such as direct preference

optimization (DPO), aim to avoid the instability of RL by removing explicit policy optimization (Rafailov et al., 2023; Meng et al., 2024). Despite these differences, RL-based methods remain attractive for complex scientific tasks. They have been shown to encourage emergent behaviors such as reflection, verification, and adaptation, and often improve generalization beyond training data (Guo et al., 2025; Chu et al., 2025). Recently proposed Group Relative Policy Optimization (GRPO) further improves efficiency by eliminating the critic model, thereby reducing memory requirements relative to PPO, while maintaining strong instruction-following performance (Shao et al., 2024).

More recent studies have explored applying GRPO to optimize tool-calling agent policies. Approaches such as Nemotron-Tool-N1 (Zhang et al., 2025) and ToolRL (Qian et al., 2025) focus on reward design for tool calls. Multi-turn extensions such as RC-GRPO (Zhong et al., 2026) and MT-GRPO (Wei et al., 2025) improve rollout diversity and per-turn credit assignment. Frameworks such as RLFactory (Chai et al., 2025b) and AgentFly (Wang et al., 2025) incorporate environment interaction by reconstructing state during rollouts. However, existing approaches do not fully address the challenges of persistent, stateful environments. Most methods either assume stateless interactions (Zhang et al., 2025; Qian et al., 2025), reconstruct state without isolating per-turn credit (Chai et al., 2025b; Wang et al., 2025), or decompose interactions into independent steps evaluated against static ground truth (Wei et al., 2025; Feng et al., 2025). These assumptions break down in scientific settings, where each step depends on an evolving simulator state that must be rebuilt through a sequence of tool calls.

To address this gap, we propose a framework for learning robust scientific agent policies for multi-turn, multi-tool interactions using state-aware GRPO. Our approach decomposes interactions into turn-level episodes while reconstructing the environment state before each rollout by replaying prior tool calls against a live simulation backend. This design preserves the benefits of per-turn credit assignment (Wei et al., 2025) while maintaining faithful execution of stateful processes (Chai et al., 2025b). It also reduces error accumulation compared to full-trajectory rollouts. Following the paradigm of RL with verifiable rewards (RLVR) (Lambert et al., 2024; Guo et al., 2025), we adopt a reward-model-free approach that combines deterministic tool-use verification with a frozen LLM-based judge (Zheng et al., 2023). We evaluate our framework on Talk2BioModels (T2B) (Singh et al., 2025b; Wehling et al., 2025), an open-source agent for kinetic biological modeling, using a synthetic multi-turn, multi-tool benchmark derived from its data generation pipeline. Employing a compact Qwen2.5-3B-Instruct backbone (Qwen et al., 2024), our framework achieves 98% tool correctness, 88% argument correctness, and 77% task completion, outperforming all 24 evaluated frontier and open-source baselines, including Claude Opus 4.6 (Anthropic, 2025), GPT-5.4 (Singh et al., 2025a), DeepSeek-V3 (DeepSeek-AI et al., 2024), and O3-mini (OpenAI, 2025). While broadly applicable, this approach is particularly relevant for domains such as biological modeling, where reliable multi-step computational modeling is essential.

In summary, our contributions in this study are as follows:

- **State-aware Per-turn Optimization Framework**: We propose a two-stage framework combining SFT and state-aware GRPO with verifiable rewards, which formulates multi-turn interactions as per-turn episodes by reconstructing environment state and replaying prior tool calls on a live simulation backend. This enables finer-grained credit assignment in stateful scientific tool-use settings while reducing cascading errors in long-horizon rollouts.
- **Optimized Scientific Agent for Biological Models**: We apply our proposed framework to T2B, improving its capabilities as a scientific agent by enhancing tool-calling consistency and accuracy for reliable interrogation of kinetic biological models.
- **Scientific Agent using a Compact Model**: Using our proposed framework, a 3B-parameter scientific agent achieves strong performance gains over several frontier models in tool correctness, argument correctness, and task completion, revealing the effectiveness of the approach for improving smaller models on scientific tasks.

2. Proposed Framework

We denote a benchmark dataset comprising tool-based question-answer (QA) pairs as $\mathbb{X} = \{(\boldsymbol{q}_n, (\boldsymbol{f}_n, \boldsymbol{a}_n))\}_{n=1}^{N}$, where N is the total number of samples and $\boldsymbol{f} \in \{f_1, f_2, \ldots, f_K\}$ indicates which of the K available tools is invoked to produce the answer. In a multi-turn setting, each n-th element unfolds as a sequence of alternating questions, tool invocations, and answers, defined as $(\boldsymbol{q}_n, (\boldsymbol{f}_n, \boldsymbol{a}_n)) = [q_n^1, (f_n^1, a_n^1), \ldots, q_n^T, (f_n^T, a_n^T)]$, where T denotes the total number of turns. Hereafter, indices are omitted when unambiguous for de-cluttering. Our main objective is to optimize the agent policy π_θ for reliable tool calling, enabling correct answer generation within an RL-based framework, as illustrated in Fig. 1.

2.1. Synthetic Benchmark Construction

Following the self-instruct paradigm (Wang et al., 2023), a prior study (Wehling et al., 2025) introduced a synthetic benchmark construction pipeline to generate QA pairs for evaluating Talk2BioModels (T2B) in terms of task completion. We extend this pipeline to support multi-turn, multi-

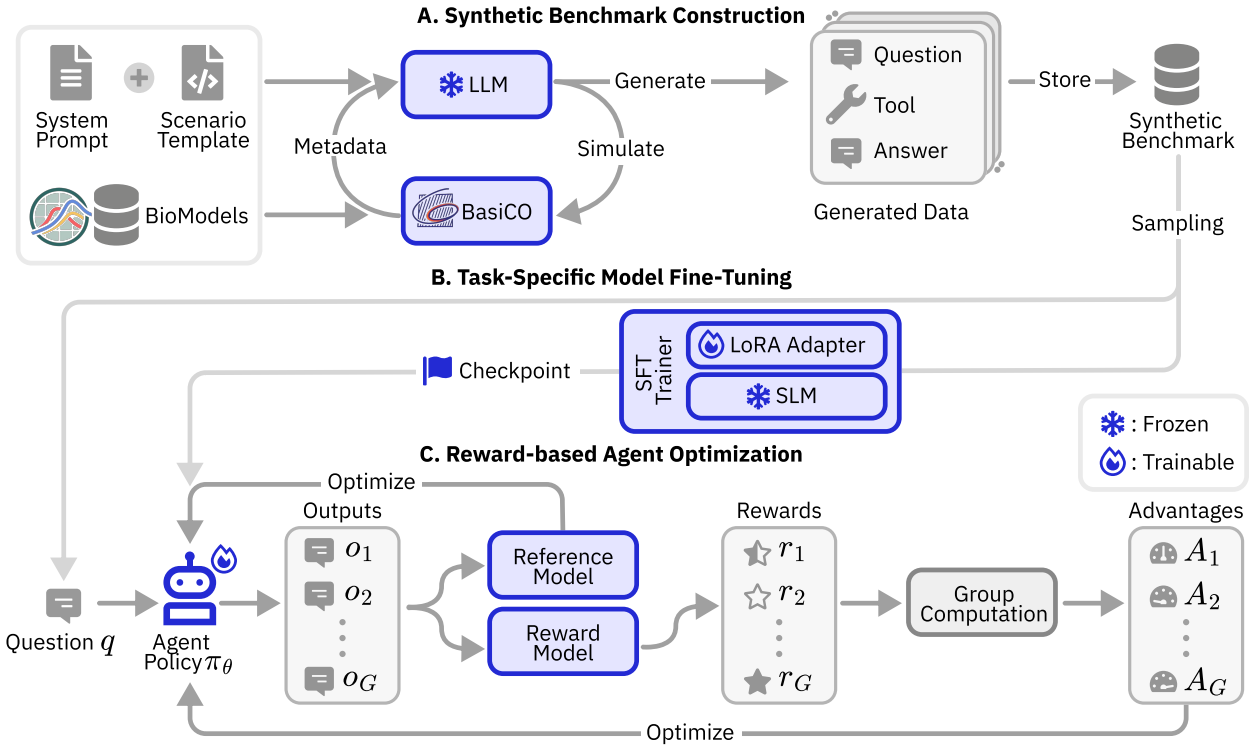

Figure 1. Overview of the proposed framework, including (a) synthetic benchmark construction and (b,c) staged SFT and state-aware GRPO optimization with verifiable rewards for multi-turn, multi-tool calling agent training.

tool invocation using five T2B tools: (i) `get_modelinfo`, which retrieves descriptive metadata of a biological model; (ii) `simulate_model`, which performs time-course simulation of a biological model; (iii) `steady_state`, which computes steady-state concentrations of model species; (iv) `ask_question`, which queries simulation and steady-state results; and (v) `parameter_scan`, which explores how parameter variations in the model affect concentrations of selected species. We select these tools due to their complex argument structures and design multi-turn scenarios involving their combined usage (see Appendix A). For each multi-turn scenario $\mathbb{S}$, we set a system prompt ϕ (see Appendix A.2) for a frozen LLM π_{QA} to produce a question q together with its answer a, guided by an expected tool f that is iteratively specified after each turn in a scenario $s \in \mathbb{S}$ until a predefined number N_s of samples is reached, as formalized as

$$\{(\boldsymbol{q}_n, (\boldsymbol{f}_n, \boldsymbol{a}_n))\}_{n=1}^{N_s} \leftarrow \pi_{QA}(\phi, s), \quad \forall\, s \in \mathbb{S}, N_s \leq N.$$

2.2. Task-Specific Model Fine-tuning

Prior to reward-based optimization, we first ensure that the base model acquires the capabilities required for the target tasks, such as reading simulated time-series data in dataframe format. We apply supervised fine-tuning (SFT) with low-rank adaptation (LoRA) (Hu et al., 2022) on the base model, optimizing the objective $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(q,a)\sim\mathbb{X}_{\text{SFT}}} \sum_{t=1}^{|a|} \log \pi_\theta\left(a_t \mid q, a_{<t}\right)$, where $\mathbb{X}_{\text{SFT}}$ denotes the SFT dataset, q is the prompt, and a is the sequence of answer tokens, including tool calls and final responses.

2.3. Reward-based Agent Optimization

We adopt GRPO (Shao et al., 2024) to optimize the agent's policy π_θ using $\mathcal{L}_{\text{GRPO}} = \mathbb{E}_{q\sim\mathbb{X},\mathbb{O}\sim\pi_{\theta_{old}}}\left(\frac{1}{G}\sum_{g=1}^{G} \min(\rho_g(\theta)A_g, [\rho_g(\theta)]_{1-\epsilon}^{1+\epsilon}A_g) - \beta\mathbb{D}_{KL}(\pi_\theta\|\pi_{ref})\right)$, where $\mathbb{O} = \{\boldsymbol{o}_g\}_{g=1}^{G}$ denotes a group of G sampled outputs produced by the agent policy for prompt q, $\rho_g(\theta)$ is the standard importance ratio between the current and behavior policies, ϵ is the clipping parameter, and β is the KL penalty coefficient. On top of GRPO, we further consider two key factors to improve optimization effectiveness: verifiable reward design and state-aware per-turn policy optimization.

Verifiable Reward Design Following evidence that learned reward models are prone to overoptimization and reward hacking in structured settings (Gao et al., 2023; Guo et al., 2025), we adopt a *reward-model-free* design, consistent with the RLVR paradigm (Lambert et al., 2024). This design is motivated by three factors. First, tool invocation correctness is directly verifiable via rule-based checks, re-

moving the need for a learned proxy. Second, prior work shows that rule-based rewards can effectively train strong tool-calling agents (Zhang et al., 2025; Qian et al., 2025; Zhong et al., 2026). Third, cascaded gating ensures that argument and task completion rewards are only computed when tool execution is correct, preventing contamination from invalid tool calls.

In particular, each sampled output is scored using a composite reward r_g combining deterministic heuristics and a frozen LLM judge as

$$r_g = \lambda_{\text{tool}}\, r_g^{\text{tool}} + \lambda_{\text{arg}}\, r_g^{\text{arg}} + \lambda_{\text{task}}\, r_g^{\text{task}}, \qquad (1)$$

where r_g^{tool} is a deterministic tool-sequence correctness score, r_g^{arg} is a deterministic argument-schema correctness score with soft numerical tolerance, and r_g^{task} is an LLM-as-a-judge (Zheng et al., 2023) score for task completion using a frozen, pre-specified LLM, with λ_* denoting the corresponding weighting coefficients. The group-relative advantages are then computed as $A_g = (r_g - \mu(\boldsymbol{r}))/\sigma(\boldsymbol{r})$, where $\mu(\boldsymbol{r})$ and $\sigma(\boldsymbol{r})$ denote the mean and standard deviation of rewards within the group, respectively.

State-Aware Per-Turn Policy Optimization In most multi-turn scientific workflows, later turns strongly depend on the state produced in earlier ones. In the T2B scientific agent setting, users are able to query simulated data from biological models, where the correctness of each response depends on whether the underlying simulation and its outputs are valid. Standard GRPO treats each prompt as stateless, which is incompatible with such dependencies. Existing multi-turn GRPO methods either (i) use full-trajectory rollouts with live execution (Zhong et al., 2026), which accumulate errors and provide only trajectory-level credit, (ii) decompose into per-turn episodes but evaluate against static ground truth without live execution (Zhang et al., 2025; Qian et al., 2025; Wei et al., 2025; Feng et al., 2025; Chai et al., 2025a), which fails in stateful environments where rewards depend on simulator outputs, or (iii) reconstruct state during rollout (Chai et al., 2025b; Wang et al., 2025), which entangles state reconstruction with generation and prevents clean per-turn credit assignment.

Compared to full-trajectory rollouts, per-turn decomposition avoids cascading errors and enables fine-grained credit assignment without stepwise estimators. Unlike static scoring, live execution on reconstructed state grounds rewards in simulator outputs, while replaying ground-truth tool calls ensures consistent state independent of policy errors, separating state construction from evaluation. An ideal approach should also preserve GRPO's efficiency by requiring only G rollouts per turn rather than full multi-turn trajectories.

Hence, we propose a per-turn episode formulation with stateful context reconstruction, as shown in Algorithm 1.

Algorithm 1 State-Aware Per-Turn GRPO with verifiable rewards

Require: Conversation $\mathbb{X}_i = [(q_1, f_1, a_1), \ldots, (q_T, f_T, a_T)]$
for each turn $t = 1, \ldots, T$ **do**
  $\mathcal{E}_t \leftarrow$ NewEnvironment(π_θ)
  $\mathcal{P}_t \leftarrow \{(f_j, \mathbf{a}_j) \mid j < t,\ f_j \in \mathcal{F}_{\text{state}}\}$ {Prior state-building calls}
  for each $(f_j, \mathbf{a}_j) \in \mathcal{P}_t$ **do**
    $\mathcal{E}_t$.Execute($f_j, \mathbf{a}_j | \pi_\theta$) {Reconstruct state}
  end for
  $\mathcal{E}_t$.ClearScoringArrays() {Only policy calls count}
  $\mathbf{p}_t \leftarrow$ BuildPrompt($C_{<t}, q_t$) {Full history + current query}
  $\mathcal{O}_t = \{o_g\}_{g=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathbf{p}_t, \mathcal{E}_t)$ {G rollouts with live tools}
  $r_g \leftarrow \lambda_{\text{tool}} r_g^{\text{tool}} + \lambda_{\text{arg}} r_g^{\text{arg}} + \lambda_{\text{task}} r_g^{\text{task}}$
  $A_g \leftarrow (r_g - \mu(\mathbf{r}))/\sigma(\mathbf{r})$
  Update π_θ using $\mathcal{L}_{\text{GRPO}}$ with $\{(o_g, A_g)\}_{g=1}^{G}$
end for

Specifically, we decompose each multi-turn conversation into turn-level GRPO episodes and initialize a fresh environment for each turn. Let $\mathcal{F}_{\text{state}}$ denote the subset of tools that modify the simulator state, and $C_{<t}$ the conversation history preceding turn t. The environment reconstructs the current state by replaying state-building tool calls from the ground-truth conversation history. These calls are then re-executed against the live simulation backend, ensuring that the state is faithfully restored for the specific turn. We treat this process as a pre-warming phase where intermediate scoring traces are discarded after state reconstruction, so that only newly generated policy rollouts contribute to the reward signal. Finally, the policy generates G rollouts for the current turn, which are evaluated in the reconstructed live environment and optimized accordingly.

3. Results

3.1. Experimental Setup

Dataset Using our synthetic benchmark pipeline, we construct 10 scenarios (Table 5) derived from 20 ODE-based human biological models from the BioModels repository (Malik-Sheriff et al., 2020) (Table 3) encoded in the Systems Biology Markup Language (SBML) covering various biological settings. Each model defines species (quantities of entities such as molecules or populations), reactions (rules describing how species are transformed or interact), and parameters (numerical values controlling reaction rates and system behavior). For each scenario, we first use BasiCO

Table 1. Quantitative performance on the held-out test split (324 turns, 120 conversations). All models ranked by composite score. Parameter counts shown for open-weight models.

Model	Tool Correctness	Argument Correctness	Task Completion	Latency (s/turn)
Qwen-SFT+GRPO (ours, 3B)	**0.98**	**0.88**	**0.77**	6.5
Qwen-SFT (ours, 3B)	0.97	0.86	0.75	8.0
Claude Opus 4.6	0.82	0.63	0.73	14.0
Claude Sonnet 4.5	0.82	0.59	0.74	11.6
DeepSeek-V3-0324 (685B)	0.81	0.59	0.69	6.3
GPT-5.4	0.82	0.55	0.71	6.2
Claude Haiku 4.5	0.80	0.59	0.70	8.4
GPT-4o	0.84	0.58	0.66	5.4
O4-mini	0.81	0.60	0.67	10.3
O3-mini	0.81	0.56	0.65	9.4
GPT-5.4-mini	0.78	0.55	0.66	5.3
Qwen2.5-72B-Instruct (72B)	0.74	0.58	0.62	12.3
O3	0.73	0.57	0.62	10.1
GLM-4.7 (358B)	0.71	0.56	0.63	9.8
Llama-3.3-70B-Instruct (70B)	0.76	0.58	0.55	12.3
GLM-4.7-Flash (31B)	0.70	0.51	0.62	8.0
GPT-4o-mini	0.70	0.48	0.55	7.5
Qwen3-32B (32B)	0.79	0.54	0.40	11.3
Mistral-Small-2409 (22B)	0.71	0.43	0.52	5.2
GPT-5.4-nano	0.63	0.44	0.52	5.8
Llama-4-Scout-17B-16E (109B)	0.66	0.45	0.48	3.9
GRPO-only (ours, 3B)	0.66	0.46	0.46	5.1
DeepSeek-V3.1 (685B)	0.58	0.47	0.51	7.1
Claude Sonnet 4.6	0.56	0.47	0.48	17.2
Qwen2.5-3B-Instruct (3B)	0.65	0.26	0.49	4.5
GPT-5-nano	0.51	0.41	0.43	31.1
GPT-5-mini	0.43	0.30	0.36	20.8

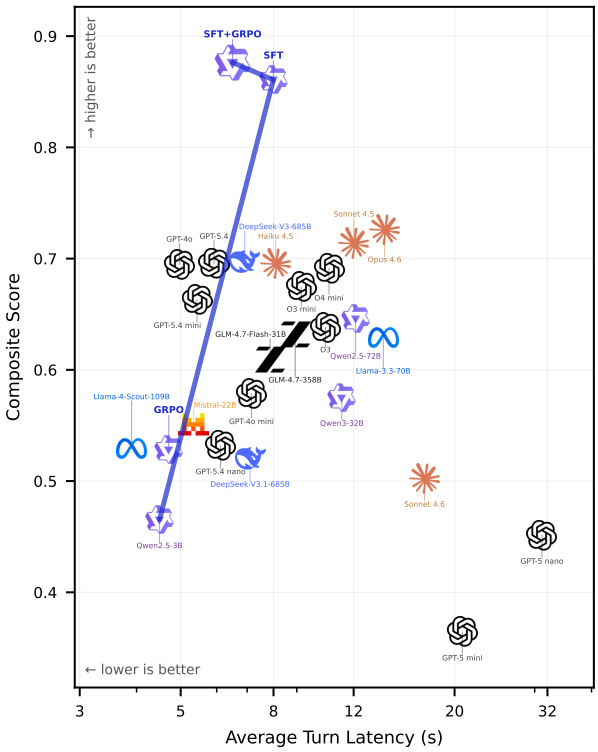

Figure 2. Pareto frontier of composite score vs. average turn latency. Our Qwen-SFT+GRPO lies on the upper-left frontier. The arrow shows the Base→SFT→SFT+GRPO trajectory.

(Bergmann, 2023) to run simulations[1] and generate scientifically grounded model behaviors. These simulations are then used to guide a frozen LLM (GPT-5-nano (Singh et al., 2025a)) in generating multi-turn question-answer trajectories. This process yields 1,200 conversations, which are split in a scenario-stratified manner into 960/120/120 train, validation, and test samples. The test set contains 324 evaluation turns. Additional dataset statistics are provided in Appendix A.3.

Implementation Details We extend T2B, originally built on LangGraph (LangChain, Inc., 2024), to support policy optimization using the Transformer Reinforcement Learning (TRL) library (von Werra et al., 2020). We use Qwen2.5-3B-Instruct (Qwen et al., 2024) as the base model, trained with LoRA adapters in two stages: (i) SFT on multi-turn traces, followed by (ii) state-aware GRPO with verifiable rewards. During RL training, we incorporate state-aware online scoring by executing tool calls and reconstructing turn-level context at each step. We compare against 24 baselines spanning OpenAI (Singh et al., 2025a; OpenAI, 2025), Anthropic (Anthropic, 2025), DeepSeek (DeepSeek-AI et al., 2024), Meta (Grattafiori et al., 2024; Meta AI, 2025), Qwen (Yang et al., 2025), GLM (GLM Team et al., 2025), and Mistral (Mistral AI, 2024) model families. All models are evaluated using the same stateful execution harness and LLM judge. Evaluation uses the model's own generated tool calls against the live simulator, so errors propagate across turns; the 73.3% perfect conversation rate validates performance under self-generated state. Frontier models are evaluated zero-shot, framing our comparison as a demonstration of targeted optimization rather than a claim about general capabilities. Experiments are conducted on two NVIDIA B200 180 GB GPUs: one for training and local model serving, and one hosting the frozen LLM judge. Additional implementation details are provided in Appendix B.

[1]All floating-point arguments derived from the simulated data are rounded to four significant figures.

3.2. Performance Comparison

Table 1 summarizes performance across all evaluated models. Our Qwen-SFT+GRPO achieves the best results on all three metrics, outperforming all 24 evaluated baselines. Compared to the strongest baseline (Claude Opus 4.6), it improves tool correctness by +16 pp, argument correctness by +25 pp, and task completion by +4 pp. The largest gains are in argument correctness: all baseline models do not exceed 63%, whereas our 3B model reaches 88%. Fig-

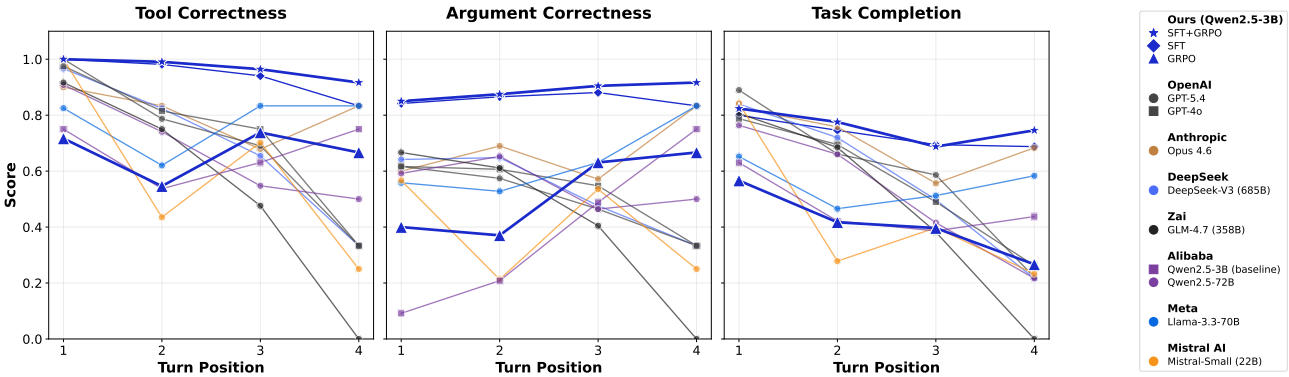

Figure 3. Performance by turn position across representative models (top performers from each family). Qwen-SFT+GRPO (dark blue, thick line) maintains near-perfect tool and argument correctness across all turn positions, while baseline models degrade at later turns where state-dependent interactions dominate. All models show declining task completion at Turn 4, but GRPO degrades least. This robustness at later positions directly reflects the benefit of state-aware training with prior-turn replay.

ure 2 shows the composite-latency Pareto frontier, where Qwen-SFT+GRPO achieves the highest composite score at competitive latency.

Beyond the reported quantitative results, we analyzed turn-level multi-turn conversations (Figure 3), where baseline models degraded at later turns due to difficulty handling state-dependent interactions requiring prior context, while the GRPO-optimized model remained stable. We define a perfect conversation as one in which the model correctly performed tool parsing and invocation at every turn and produced the correct final answer. Under this criterion, Qwen-SFT+GRPO achieved 73.3% perfect conversations, compared to 28.3% for the strongest baseline (Claude Opus 4.6), a 2.6× improvement. The primary bottleneck arose in cases where the correct tool was selected but the arguments were incorrect: Claude Opus 4.6 produced incorrect arguments on 16.4% of turns, compared to 10.5% for our model. This disparity suggests that tool selection and argument construction are governed by different failure modes, and that argument construction benefits most from domain-specific optimization.

Argument errors are compounded in multi-turn scenarios, as illustrated by three representative cases (Table 2) where baseline models selected the correct tool but produced incorrect arguments, directly illustrating the 88% vs. <63% argument correctness gap. In **Case A** (Scenario 4), both Claude Opus 4.6 and DeepSeek-V3-0324 issued excess tool calls at Turn 2 and produced incorrect arguments at Turn 3: Claude Opus omitted the required `time_data` field and fabricated the experiment name, while DeepSeek-V3 appended compartment-annotation suffixes to species identifiers. DeepSeek-V3's errors cascaded to Turn 4, resulting in zero scores across all metrics. In **Case B** (Scenario 6), Claude Opus fabricated the experiment name while DeepSeek-V3 failed to expand parameter values, with both errors cascading to Turn 3. In **Case C** (Scenario 8, Appendix D), both baselines omitted the required `time_data` field entirely at Turn 1. These results indicate argument construction errors rather than tool-selection failures, and our optimized model produced correct arguments across all turns and cases. We provide a detailed per-tool breakdown in Appendix D.

The impact of state dependencies on error propagation is further supported by our out-of-distribution evaluation (Appendix F). In the novel composition category (Scenario 18), where familiar tools are arranged in an unseen ordering, tool correctness drops to 71% and argument correctness to 64%, compared to 89%/78% overall OOD performance. This confirms that state-dependent interactions are a primary source of cascading failures.

4. Discussion

Complementary Roles of SFT and GRPO The ablation results (Table 6) show that SFT and GRPO play complementary roles. SFT yields the largest improvement, increasing argument correctness from 26% to 86% (+60 pp) by learning structured tool-call formats and domain-specific argument schemas. GRPO then further refines the policy, improving tool correctness from 97% to 98% and argument correctness from 86% to 88%. The per-tool analysis (Table 7) shows that GRPO's gains are concentrated on tools with complex argument structures. For example, task completion of the `simulate_model` tool improves from 87.2% to 92.2% (+5.0 pp), while task completion of `parameter_scan` improves from 80.7% to 84.8% (+4.1 pp). We also conducted a turn-position analysis (Figure 3), showing that GRPO maintains near-perfect performance at later turns where state-dependent interactions dominate, while baseline models degrade significantly, consistent with the benefit of prior-turn state reconstruction.

Table 2. Three representative multi-turn argument failure cases. Each block shows a scenario with the user query, ground-truth tool and arguments, and per-turn model behavior. Correct outputs are marked in blue, and incorrect or missing outputs in red. Free-form responses without tool invocation are labeled "skipped." We include Claude Opus 4.6 and DeepSeek-V3-0324 as representative baselines that exhibit common argument failure patterns shared across all evaluated frontier models.

Case A (Scenario 4): IL-6 signaling (Dwivedi et al., 2014)				
	Turn 1	**Turn 2**	**Turn 3**	**Turn 4**
Query	*"What parameters are defined for BIOMD537?"*	*"Simulate 2016 h, 14 steps, IL6{serum}=2.63e-4. What is CRP{serum}"?*	*"Please run the steady state... IL6=2.99e-4, sgp130=2.748, sR_IL6_sgp130=0.0605."*	*"Concentrations and transition times for the species."*
Ground Truth	`get_modelinfo`; `parameters`:true, all others false	`simulate+ask_question`; duration=2016, interval=144; species=["IL6"]; experiment="simulate_BIOMD…537"	`steady_state`; species=["IL6", "sgp130", "sR_IL6_sgp130"]; time={1000,100}; experiment="ss_BIOMD…537"	`ask_question`; experiment="ss_BIOMD…537"; question_context="steady_state"
Qwen-SFT+GRPO	all arguments passed correctly	match ground truth	match ground truth	match ground truth
Claude Opus 4.6	match ground truth	4 calls (+2 extra; inserted get_modelinfo)	time_data omitted; exp fabricated	match ground truth
DeepSeek-V3-0324	match ground truth	4 calls (+2 extra; simulate×3)	3 calls; species with {serum} suffix	called simulate_model instead
Case B (Scenario 6): NFAT activation (Fisher et al., 2006)				
	Turn 1	**Turn 2**	**Turn 3**	
Query	*"Simulate BIOMD123, 3000 s, 5 steps, Calcineurin nuc.=2.59e-5 μmol."*	*"Please run a parameter scan on the model... Ca Cyt=1.174, NFAT_Nuc=5.37e-4. k7: 0.00821→0.01321, step 5e-4."*	*"Concentration of Phospho. NFAT in nucleus?"*	
Ground Truth	`simulate`; duration=3000, int=600; species=["Active Calcineurin in nucleus"]	`param_scan`; duration=3000, interval=10; parameter=k7, 11 values; experiment="param_scan_BIOMD…123"	`ask_question`; experiment="simulate_BIOMD…123"; question_context="simulation"	
Qwen-SFT+GRPO	match ground truth	interval=10; k7 match	match ground truth	
Claude Opus 4.6	match ground truth	experiment fabricated ("param_scan_k7_...")	experiment cascaded from Turn 2	
DeepSeek-V3-0324	match ground truth	values=[start,end,step] not expanded; exp fabricated	experiment cascaded from Turn 2	
Case C (Scenario 8): Calcium-dependent NFAT dynamics (Fisher et al., 2006)				
	Turn 1	**Turn 2**		
Query	*"Please run the steady state simulation on BIOMD122... Calcineurin cyt=1.35e-5, NFAT-Calc=3.71e-6, Phospho. NFAT=3.03e-4 μmol."*	*"Please run a parameter scan on the model by updating the initial... k9 4.334→5.334, step 0.1; NFAT_nuc=4.99e-4, NFAT_Cyt=1.14e-4."*		
Ground Truth	`steady_state`; time={1000,100} in `arg_data`; species=["Active Calc…","NFAT Calc…","Phospho. NFAT…"]	`param_scan`; duration=400, int=16; parameter=k9; experiment="param_scan_BIOMD…122"		
Qwen-SFT+GRPO	match ground truth	match ground truth		
Claude Opus 4.6	time_data omitted entirely	match ground truth		
DeepSeek-V3-0324	time_data omitted; exp fabricated	match ground truth		

Understanding Failure Modes in Tool-Calling Agents The largest performance difference is observed in argument correctness, where all evaluated baseline models do not exceed 63%, whereas our 3B-parameter SFT+GRPO model reaches 88%. Case studies in Table 2 indicate that this gap is associated with consistent failure patterns rather than random errors. Such failure patterns include namespace confusion (*e.g.*, appending compartment suffixes to species identifiers in biological models), schema nesting errors (placing fields at incorrect levels of the argument hierarchy), and numerical reasoning failures (such as miscalculating intervals from duration and step count). These patterns suggest that general-purpose models lack structured knowledge of tool schemas, which can be mitigated through domain-specific policy optimization. In contrast, the results of the tool selection suggest that this aspect is comparatively easier. For example, GPT-4o achieves 84% tool correctness without domain-specific training, but only 58% argument correctness. This disparity indicates that tool selection and argument construction are governed by different failure modes, and that argument construction is the primary bottleneck where targeted optimization yields the largest gains.

Task Completion under Mixed Verifiable and LLM-Based Rewards Despite large gains in tool and argument correctness, our optimized agent shows only marginal improvement in task completion, with a $+2$ pp increase from SFT to SFT+GRPO, reflecting a fundamental distinction in what each metric captures. In particular, tool and argument correctness are optimized using deterministic, verifiable rewards, which together account for two-thirds of the total reward signal (each weighted equally at $\lambda = 0.33$). In contrast, task completion is evaluated by a frozen LLM-based judge on open-ended responses and contributes the remaining third. While all three components receive equal weight, in scientific tool-use settings, correct tool execution is a prerequisite for meaningful downstream results, and our

framework successfully optimizes this objective. However, the LLM-judged component provides a noisier signal for improving natural-language summarization, resulting in only marginal gains. Nevertheless, our model still achieves the highest task completion among all evaluated models (77% vs. 74% for Claude Sonnet 4.5), suggesting that accurate tool use provides a strong foundation for answer quality.

We also conducted a turn-level performance analysis by breaking it down by scenario (Figure 8), revealing two distinct sources of task completion degradation. First, scenarios involving individual `ask_question` turns (*e.g.*, Turn 3 in Scenarios 1, 3 and 9; Turn 2 in Scenario 2) exhibit reduced task completion in all models evaluated, including our optimized agent. This tool requires a downstream LLM to parse large tabular data and extract precise numerical responses, a process that becomes unreliable when the data exceed the effective context capacity of the model. Second, prompts invoking multi-tool combination turns (*e.g.*, `simulate_model` followed by `ask_question` in Scenario 4 Turn 2, and `steady_state` followed by `ask_question` in Scenarios 7 and 10 Turn 2) exhibit more severe degradation, as the agent must both execute a state-building tool and immediately query its results within a single turn. Since our per-turn decomposition treats each turn as an independent episode, it does not explicitly train the agent on intra-turn tool composition, which may compound the tabular data parsing challenge. These observations suggest two directions for future work: (i) replacing the current `ask_question` tool with a dedicated data-analysis agent or a more capable LLM backend for handling large tabular outputs, and (ii) developing finer-grained decomposition strategies that explicitly optimize intra-turn multi-tool orchestration and execution.

Limitations Despite encouraging results, several limitations remain. First, performance varies across scenarios; in Scenario 5 (Appendix E), the model omits a required field not penalized by the reward, revealing that unverified fields can affect downstream behavior. Second, a training-inference gap exists: training uses ground-truth state replay while evaluation uses self-generated tool calls. Our stateless ablation (Appendix G), which removes state replay to approximate trajectory-level error propagation, quantifies this gap ($-3/-5/-6$ pp), and the 73.3% perfect conversation rate under self-generated state confirms the approach remains effective despite this asymmetry. Third, the synthetic benchmark is grounded in real COPASI (Hoops et al., 2006)/BasiCO (Bergmann, 2023) computations (only question phrasing is LLM-generated), but may not fully capture expert query diversity (Appendix F). Fourth, performance may improve further with models pretrained on biomedical corpora (Labrak et al., 2024; Bolton et al., 2024), which offer stronger domain-specific priors. Fifth, our current framework focuses on a single-agent setting. Extending it to multi-agent systems, such as those combining simulation and knowledge-graph agents (Singh et al., 2025b), is a promising direction for future work, particularly in the context of hierarchical reward optimization (Geng et al., 2024; 2025).

Finally, a key limitation is that our framework assumes biological models are already fully specified, with predefined species, parameters, and initial conditions. In practice, models must often be initialized from unstructured literature, requiring the extraction and grounding of entities and interventions from free-text sources. This upstream step remains a major bottleneck (Tiwari et al., 2021; Karr et al., 2022) and is not addressed by our framework. Future work could integrate agentic extraction and mapping pipelines to convert literature-derived information into structured, simulation-ready model configurations, enabling a more complete end-to-end workflow.

5. Conclusion

We propose a state-aware GRPO framework for training scientific agents that decomposes multi-turn interactions into per-turn episodes with environment state reconstruction via prior-turn replay. Applied to Talk2BioModels with a compact Qwen2.5-3B-Instruct backbone, our approach achieves 98% tool correctness, 88% argument correctness, and 77% task completion, outperforming all 24 evaluated baselines while maintaining competitive latency. A preliminary out-of-distribution evaluation further shows that the optimized policy generalizes to unseen tool compositions and novel tool interfaces to some extent. While our current experiments focus on synthetic scenarios, future work will extend this framework to real-user queries, improve task completion by replacing the current `ask_question` tool with a dedicated data-analysis agent, develop finer-grained decomposition strategies for intra-turn multi-tool coordination, and investigate hierarchical optimization in multi-agent settings.

Funding

This study was funded by Sanofi.

Conflict Interests

AS, SJ, TK and TA are Sanofi employees and may hold shares and/or stock options in the company.

References

Anthropic. The Claude Model Family: Claude Opus 4, Sonnet 4, and Haiku. Technical report, Anthropic, 2025. URL https://www.anthropic.com/research.

Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y. C., and Molchanov, P. Small Language Models are the Future of Agentic AI. *arXiv preprint arXiv:2506.02153*, 2025.

Bergmann, F. T. BASICO: A simplified Python interface to COPASI. *Journal of Open Source Software*, 8(90):5553, 2023.

Bolton, E., Venigalla, A., Yasunaga, M., Hall, D., Xiong, B., Lee, T., Daneshjou, R., Frankle, J., Liang, P., Carbin, M., et al. BioMedLM: A 2.7B Parameter Language Model Trained On Biomedical Text . *arXiv preprint arXiv:2403.18421*, 2024.

Chai, H., Cao, Z., Ran, M., Yang, Y., Lin, J., Peng, X., Wang, H., Ding, R., Wan, Z., Wen, M., Liu, W., Zhang, W., Huang, F., and Wen, Y. PARL-MT: Learning to Call Functions in Multi-Turn Conversation with Progress Awareness. *arXiv preprint arXiv:2509.23206*, 2025a.

Chai, J., Yin, G., Xu, Z., Yue, C., Jia, Y., Xia, S., Wang, X., Jiang, J., Li, X., Dong, C., He, H., and Lin, W. RLFactory: A Plug-and-Play Reinforcement Learning Post-Training Framework for LLM Multi-Turn Tool-Use. *arXiv preprint arXiv:2509.06980*, 2025b.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep Reinforcement Learning from Human Preferences. *Advances in neural information processing systems*, 30, 2017.

Chu, T., Zhai, Y., Yang, J., and Ma, Y. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. In *International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 10818–10838, 2025.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Dwivedi, G., Fitz, L., Hegen, M., Martin, S., Harrold, J., Heatherington, A., and Li, C. A Multiscale Model of Interleukin-6-Mediated Immune Regulation in Crohn's Disease and Its Application in Drug Discovery and Development. *CPT: Pharmacometrics & Systems Pharmacology*, 3(1):1–9, 2014.

Feng, L., Xue, Z., Liu, T., and An, B. Group-in-Group Policy Optimization for LLM Agent Training. In *Advances in Neural Information Processing Systems*, 2025. arXiv preprint arXiv:2505.10978.

Fisher, W. G., Yang, P.-C., Medikonduri, R. K., and Jafri, M. S. NFAT and NFκB activation in T lymphocytes: a model of differential activation of gene expression. *Annals of biomedical engineering*, 34(11):1712–1728, 2006.

Gao, L., Schulman, J., and Hilton, J. Scaling Laws for Reward Model Overoptimization. In *International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 10835–10866, 2023.

Gao, S., Zhu, R., Kong, Z., Noori, A., Su, X., Ginder, C., Tsiligkaridis, T., and Zitnik, M. TxAgent: An AI agent for therapeutic reasoning across a universe of tools. *arXiv preprint arXiv:2503.10970*, 2025.

Geng, M., Pateria, S., Subagdja, B., and Tan, A.-H. HiSOMA: A hierarchical multi-agent model integrating self-organizing neural networks with multi-agent deep reinforcement learning. *Expert Systems with Applications*, 252:124117, 2024.

Geng, M., Pateria, S., Subagdja, B., Li, L., Zhao, X., and Tan, A.-H. L2M2: A Hierarchical Framework Integrating Large Language Model and Multi-agent Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*, 2025.

GLM Team, Zeng, A., Lv, X., Zheng, Q., et al. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *arXiv preprint arXiv:2508.06471*, 2025.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. COPASI: a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, 2006. doi: 10.1093/bioinformatics/btl485.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

Huang, K., Zhang, S., Wang, H., Qu, Y., Lu, Y., Roohani, Y., Li, R., Qiu, L., Zhang, J., Di, Y., et al. Biomni: A General-Purpose Biomedical AI Agent. *bioRxiv*, pp. 2025–05, 2025.

Karr, J. R., Malik-Sheriff, R. S., Osborne, J., Gonzalez-Parra, G., Forgoston, E., Bowness, R., Liu, Y., Thompson, R., Anwar, A., Yennamalli, R. M., et al. Model integration in computational biology: the role of reproducibility, credibility and utility. *Frontiers in Systems Biology*, 2: 822606, 2022.

Labrak, Y., Bazoge, A., Morin, E., Gourraud, P.-A., Rouvier, M., and Dufour, R. BioMistral: A Collection of Open-Source Pretrained Large Language Models for Medical Domains. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5848–5864, 2024.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. TULU 3: Pushing Frontiers in Open Language Model Post-Training. *arXiv preprint arXiv:2411.15124*, 2024.

LangChain, Inc. LangGraph: Build Resilient Language Agents as Graphs, 2024. URL https://github.com/langchain-ai/langgraph.

Malik-Sheriff, R. S., Glont, M., Nguyen, T. V., Tiwari, K., Roberts, M. G., Xavier, A., Vu, M. T., Men, J., Maire, M., Kananathan, S., et al. Biomodels: 15 years of sharing computational models in life science. *Nucleic acids research*, 48(D1):D407–D415, 2020.

Meng, Y., Xia, M., and Chen, D. SimPO: Simple Preference Optimization with a Reference-Free Reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Meta AI. Llama 4: Multimodal Intelligence. Technical report, Meta, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Mistral AI. Mistral Small Instruct 2409. https://mistral.ai/news/september-24-release, 2024.

NVIDIA. Llama-3.3-Nemotron-Super-49B-v1.5: NVIDIA Open Model for Reasoning, Coding, and Agentic Tasks. Technical report, NVIDIA, 2025. URL https://huggingface.co/nvidia/Llama-3_3-Nemotron-Super-49B-v1_5.

OpenAI. OpenAI o3 and OpenAI o4-mini System Card. Technical report, OpenAI, 4 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf. System card.

Pareja, A., Nayak, N. S., Wang, H., Killamsetty, K., Sudalairaj, S., Zhao, W., Han, S., Bhandwaldar, A., Xu, G., Xu, K., et al. Unveiling the secret recipe: A guide for supervised fine-tuning small LLMs. *arXiv preprint arXiv:2412.13337*, 2024.

Qian, C., Acikgoz, E. C., He, Q., Wang, H., Chen, X., Hakkani-Tür, D., Tur, G., and Ji, H. ToolRL: Reward is All Tool Learning Needs. *arXiv preprint arXiv:2504.13958*, 2025.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *Advances in neural information processing systems*, 36: 53728–53741, 2023.

Saini, A. and Farnoud, A. QSP-Copilot: An AI-Augmented Platform for Accelerating Quantitative Systems Pharmacology Model Development. *CPT: Pharmacometrics & Systems Pharmacology*, 2025. doi: 10.1002/psp4.70127.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shah, M. B., Morovati, M. M., Rahman, M. M., and Khomh, F. Characterizing Faults in Agentic AI: A Taxonomy of Types, Symptoms, and Root Causes. *arXiv preprint arXiv:2603.06847*, 2026.

Shahin, M. H., Goswami, S., Lobentanzer, S., and Corrigan, B. W. Agents for change: Artificial intelligent workflows for quantitative clinical pharmacology and translational sciences. *Clinical and Translational Science*, 2025. doi: 10.1111/cts.70188.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. DeepSeekMath: Pushing The Limits of Mathematical Reasoning in Open Language Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al. OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*, 2025a.

Singh, G., Wehling, L., Mulyadi, A. W., Sreenath, R. H., Klabunde, T., Andreani, T., and McCloskey, D. Talk2Biomodels and Talk2KnowledgeGraph: AI agent-based application for prediction of patient biomarkers and reasoning over biomedical knowledge graphs. In *ICLR 2025 Workshop on Machine Learning for Genomics Explorations*, 2025b.

Tiwari, K., Kananathan, S., Roberts, M. G., Meyer, J. P., Sharber, M. U., Zeke, T., Liénard, P., Vinod, R. S. M., Glont, M., Hermjakob, H., and Malik-Sheriff, R. S. Reproducibility in systems biology modelling. *Molecular Systems Biology*, 17(2):e9982, 2021.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Wan, G., Ling, M., Ren, X., Han, R., Li, S., and Zhang, Z. Compass: Enhancing agent long-horizon reasoning with evolving context. *arXiv preprint arXiv:2510.08790*, 2025.

Wang, R., Genadi, R. A., El Bouardi, B., Wang, Y., Koto, F., Liu, Z., Baldwin, T., and Li, H. AgentFly: Extensible and Scalable Reinforcement Learning for LM Agents. *arXiv preprint arXiv:2507.14897*, 2025.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Wehling, L., Singh, G., Mulyadi, A. W., Sreenath, R. H., Hermjakob, H., Nguyen, T. V., Rückle, T., Mosa, M. H., Cordes, H., Andreani, T., et al. Talk2Biomodels: AI Agent-Based Open-Source LLM Initiative for Kinetic Biological Models. *BMC bioinformatics*, 26(1):276, 2025.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

Wei, Q., Zeng, S., Li, C., Brown, W., Frunza, O., Deng, W., Schneider, A., Nevmyvaka, Y., Zhao, Y. K., Garcia, A., and Hong, M. Reinforcing Multi-Turn Reasoning in LLM Agents via Turn-Level Reward Design. *arXiv preprint arXiv:2505.11821*, 2025.

Xiong, Q., Huang, Y., Jiang, Z., Chang, Z., Zheng, Y., Li, T., and Li, M. Butterfly effects in toolchains: A comprehensive analysis of failed parameter filling in LLM tool-agent systems. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 16712–16729. Association for Computational Linguistics, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhang, S., Dong, Y., Zhang, J., Kautz, J., Catanzaro, B., Tao, A., Wu, Q., Yu, Z., and Liu, G. Nemotron-Research-Tool-N1: Exploring Tool-Using Language Models with Reinforced Reasoning. *arXiv preprint arXiv:2505.00024*, 2025.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Zhong, H., Zhai, J., Song, L., Bian, J., Liu, Q., and Tan, T. RC-GRPO: Reward-Conditioned Group Relative Policy Optimization for Multi-Turn Tool Calling Agents. *arXiv preprint arXiv:2602.03025*, 2026.

A. Synthetic Benchmark Construction

A.1. Synthetic Data Generation Pipeline

We constructed a synthetic dataset using 20 models selected from the BioModels repository (Malik-Sheriff et al., 2020) (Table 3), spanning diverse areas of biology. For each model, we manually extracted the required simulation times from the corresponding publications. Scientifically grounded question-answer pairs were generated using the BasiCO (Bergmann, 2023) library, a Python interface to COPASI (Hoops et al., 2006), by extracting model species, parameters, and metadata, such as names, descriptions, units, and compartment information, using the `get_species()`, `get_parameters()`, and `get_model_info()` functions.

The `get_modelinfo` tool retrieves descriptive metadata about a model, including its name, compartments, species, parameters, and other relevant information. For this tool, we provided the LLM with metadata obtained using the `get_model_info()` function from BasiCO. The LLM was instructed to generate natural-language questions that a user might plausibly ask when exploring a new biological model (e.g., requesting a description or name of the model), and the corresponding answers. The system prompt used for this tool was provided as Prompt S4 in Section A.2.

The `simulate_model` tool performs a time-course simulation of a model under specified initial conditions. The LLM was provided with the list of species obtained using `get_species()` function from BasiCO. The LLM was instructed to generate a question that selected up to three species whose initial concentrations could be modified prior to simulation and up to three species whose concentrations should be measured after the simulation. We then extracted the original initial concentrations of the selected species using the BasiCO library `get_species()` function and perturbed them by randomly scaling their values to lie between half and twice the original concentration. The question explicitly described these requested modifications and measurements in natural language. The answer associated with this tool invocation confirmed whether the simulation completed successfully. The corresponding system prompt was provided as Prompt S2 in the Appendix. We also executed a time-course simulation using the `run_time_course()` function from BasiCO, and saved the results for later use by the `ask_question` tool.

The `steady_state` tool computes the steady-state concentrations of model species under specified initial conditions. For this tool, the LLM was provided with the list of model species obtained from `get_species()`, and instructed to generate a question specifying up to three species for concentration modification prior to the steady-state computation and up to three species for post-simulation measurement. The initial concentrations of the selected species were perturbed using the same random scaling strategy described above. The question explicitly described these requested modifications and measurements in natural language. The answer to this tool invocation indicated whether the model successfully reached a steady state. The system prompt for this tool was provided as Prompt S3 in the Appendix. We also computed the steady-state solution using the `run_steadystate()` function from BasiCO, and saved the results for later use by the `ask_question` tool.

The `ask_question` tool queries results produced by the `simulate_model` and `steady_state` tools. For this tool, we generated questions that asked for the final concentrations of specific species identified earlier in the interaction. Since the species to be measured were already selected during the generation of the simulation or steady-state questions, the follow-up questions were constructed to explicitly query these species. To generate answers, we extracted the final concentrations of the requested species from the stored simulation results. For time-course simulations, the answer reported the final concentrations of the queried species. For steady-state simulations, if the model failed to reach steady state, the answer did not report concentrations and instead indicated that steady state was not achieved. The system prompt used for this tool is provided in Section A.2.

The `parameter_scan` tool explores how varying a model parameter affected the concentrations of selected species under specified initial conditions. For this tool, the LLM was provided with the list of species obtained from `get_species()` and was instructed to generate a question that selected up to three species for concentration modification and up to three species for post-scan measurement. In addition, the LLM was provided with a list of parameters and instructed to select one parameter for scanning. Using the `get_parameters()` function from BasiCO, we retrieved the initial value of the selected parameter and varied it over a range from half to twice its original value. The scan was performed using a randomly chosen number of steps between 5 and 10. Species concentrations were perturbed prior to the scan using the same random scaling strategy described earlier. The question explicitly described these requested modifications and measurements in natural language. The answer to this tool invocation confirmed that the parameter scan completed successfully. The corresponding system prompt was provided as Prompt S6 in the Section A.2.

A.2. System Prompts

PROMPT S1: SELECT MODIFIABLE AND MEASURABLE SPECIES FOR `SIMULATE_MODEL` AND `STEADY_STATE`

Given: The list of species in a systems biology model.
Tasks: Identify a list of 1-2 species whose initial concentrations can be modified. Identify a list of 1-2 different species whose concentrations can be measured.
Constraints: The two lists must not contain the same species. If the model appears to be a QSP model, prioritize pharmacologically relevant species (e.g., drug, biomarker, disease marker). Prefer species that are clinically measurable for the measurement list and species that can be pharmacologically modulated for the modification list. Do not label or name the lists in the response.

PROMPT S2: PREPARE Q&A FOR `SIMULATE_MODEL`

Given: A list of species with updated initial concentrations. A list of species with their expected concentrations after simulation. A model ID (unless this is a follow-up question). Model units.
Tasks: Write a natural-language question requesting a simulation of the model for a specified duration and time step, explicitly stating the new initial concentrations. Ask for the concentrations of the specified species at the end of the simulation. Write a corresponding natural-language answer containing the expected concentrations.
Guidelines: Species names and numerical values must exactly match those provided. Units must appear inline with numerical values and nowhere else. Write as an expert in systems biology and pharmacology.

PROMPT S3: PREPARE Q&A FOR `STEADY_STATE`

Given: A list of species with updated initial concentrations. A list of species with their expected steady-state concentrations and/or transition times. A model ID (unless this is a follow-up question). Model units.
Tasks: Write a natural-language question requesting that the model be run to steady state, explicitly stating the new initial concentrations. Ask for the steady-state concentrations and/or transition times of the specified species. Write a corresponding natural-language answer containing the expected results.
Guidelines: Species names and numerical values must exactly match those provided. Units must appear only inline with numerical values. Write as an expert in systems biology and pharmacology.

PROMPT S4: RETRIEVE MODEL INFORMATION

Given: Information from one or more of the following model components: species, parameters, compartments, units, description, model name.
Tasks: Write a natural-language question asking to retrieve one or more of these components from a systems biology model with a given ID. Write a natural-language answer containing exactly the provided information.
Guidelines: Species names and numerical values must exactly match those provided. Units must appear inline with numerical values and nowhere else. Write as an expert in systems biology and pharmacology.

PROMPT S5: SELECT SPECIES/PARAMETERS FOR `PARAMETER_SCAN`

Given: The species and parameters of a systems biology model.
Tasks: Randomly select 1-3 species whose initial concentrations will be modified before running a parameter scan. Select 1 species or parameter whose value will be scanned. Select 1-3 species whose concentrations will be measured after the scan. Return only the three lists based on the provided species and parameters.

PROMPT S6: PREPARE Q&A FOR `PARAMETER_SCAN`

Given: A list of species with new initial concentrations. A species or parameter to be scanned. A species to be observed after the scan. A model ID (unless this is a follow-up question). Model units.
Tasks: Write a natural-language question requesting a parameter scan. Specify the new initial concentrations. Define the scan range with a start value, end value, and step size for the selected species or parameter. State the simulation duration and time step. Specify the species to be observed. Write a natural-language answer confirming that the parameter scan has been completed.
Guidelines: Species names and numerical values must exactly match those provided. Units must appear inline with numerical values and nowhere else. Write as an expert in systems biology and pharmacology.

A.3. Additional Information

Table 3. Manually curated BioModels with ground-truth simulation time points.

Model ID	Model Name	Simulation Time	PubMed ID
BIOMD0000000112	Clarke2006_Smad_signalling	480 min	17186703
BIOMD0000000163	Zi2007_TGFbeta_signaling	8 hrs	17895977
BIOMD0000000186	Ibrahim2008 - Mitotic Spindle Assembly Checkpoint - Dissociation variant	4000 sec	18253502
BIOMD0000000187	Ibrahim2008 - Mitotic Spindle Assembly Checkpoint - Convey variant	4000 sec	18253502
BIOMD0000000424	Faratian2009 - Role of PTEN in Trastuzumab resistance	60 min	19638581
BIOMD0000000357	Lee2010_ThrombinActivation_OneForm_reduced	900 sec	20435402
BIOMD0000000364	Lee2010_ThrombinActivation_OneForm	900 sec	20435402
BIOMD0000000399	Jenkinson2011_EGF_MAPK	60 min	21548948
BIOMD0000000328	Bucher2011_Atorvastatin_Metabolism	600 min	21548957
BIOMD0000000463	Heldt2012 - Influenza Virus Replication	12 hrs	22593159
BIOMD0000000362	Butenas2004_BloodCoagulation	1200 sec	15039440
BIOMD0000000120	Chan2004_TCell_receptor_activation	70 sec	15255048
BIOMD0000000335	Hockin2002_BloodCoagulation	700 sec	11893748
BIOMD0000000528	Fribourg2014 - Dynamics of viral antagonism and innate immune response (H1N1 influenza A virus - Cal/09)	600 days	17532343
BIOMD0000000529	Fribourg2014 - Dynamics of viral antagonism and innate immune response (H1N1 influenza A virus - NC/99)	8 hrs	24594370
BIOMD0000000889	Fribourg2014 - Model of influenza A virus infection dynamics of viral antagonism and innate immune response.	8 hrs	24594370
BIOMD0000000122	Fisher2006_Ca_Oscillation_dpdnt_NFAT_dynamics	400 sec	17031595
BIOMD0000000123	Fisher2006_NFAT_Activation	3000 sec	17031595
BIOMD0000000537	Dwivedi2014 - Crohns IL6 Disease model - Anti-IL6R Antibody	2016 hrs	24402116
BIOMD0000000788	Schropp2019 - Target-Mediated Drug Disposition Model for Bispecific Antibodies	150 days	30480383

Table 4. List of tools used to optimize the Talk2BioModels (T2B) agent.

Tool Name	Description	Expected Returns
`get_modelinfo`	Retrieve model metadata (species, parameters, compartments, units, description, name).	Model metadata summary.
`simulate_model`	Run time-course simulation with specified initial conditions and time settings.	Simulation results (time-series).
`steady_state`	Compute steady state given initial conditions.	Steady-state results (concentrations, transition time).
`parameter_scan`	Scan a parameter over values and observe species responses.	Parameter scan results.
`ask_question`	Answer a question over simulation or steady-state results using the dataframe.	Numeric/text answer derived from results.

Table 5. Scenario templates and synthetic benchmark generated by our pipeline.

Scenario	Tool chain (multi-turn)	Turns/Conv	#Conv	#Turns	Example
1	`get_modelinfo` → `simulate_model` → `ask_question`	3	120	360	Units query for BIOMD0000000112.
2	`steady_state` → `ask_question`	2	120	240	Steady state with new initial conditions.
3	`simulate_model` → `ask_question` → `ask_question`	3	120	360	Simulation with two follow-up queries.
4	`get_modelinfo` → `simulate_model`, `ask_question` → `steady_state` → `ask_question`	4	120	480	Compartments and species query.
5	`parameter_scan`	1	120	120	Scan a parameter and confirm completion.
6	`simulate_model` → `parameter_scan` → `ask_question`	3	120	360	Simulate then scan a parameter.
7	`get_modelinfo` → `steady_state`, `ask_question` → `parameter_scan`	3	120	360	Info then steady state and scan.
8	`steady_state` → `parameter_scan`	2	120	240	Steady state followed by scan.
9	`simulate_model` → `simulate_model` → `ask_question`	3	120	360	Two simulations then query.
10	`get_modelinfo` → `steady_state`, `ask_question` → `ask_question`	3	120	360	Info then steady state with follow-ups.
Total			**1200**	**3240**	

B. Implementation Details

We trained the policy model using a two-stage pipeline with LoRA (Hu et al., 2022) adapters: SFT followed by GRPO with verifiable rewards. For SFT, we trained Qwen2.5-3B-Instruct with TRL's (von Werra et al., 2020) `SFTTrainer` on scenario-stratified multi-turn splits, using assistant-only teacher-forcing over tool calls and final answers. We used a tool-calling chat template with structured response formatting to preserve schema-valid invocation patterns. LoRA settings were $r = 16$, $\alpha = 32$, and dropout $= 0.05$, with adaptation on attention and feed-forward projections.

For GRPO, we initialized from the SFT adapter and trained at turn level. In each update, we sampled $G = 6$ completions per prompt at temperature 0.9 and used a composite verifiable reward combining three components: (i) deterministic tool correctness via exact-match sequence comparison ($\lambda_{\text{tool}} = 0.33$), (ii) deterministic argument correctness with soft numeric tolerance bands ($\lambda_{\text{arg}} = 0.33$), and (iii) task-completion scoring from a frozen LLM judge ($\lambda_{\text{task}} = 0.33$). No learned reward model was used. Cascaded gating ensures argument and task scores are zeroed when tool correctness is zero, preventing gradient signal from incorrect tool invocations.

The argument scoring function assigns graduated credit based on relative numeric error: exact matches receive 1.0, errors within 1% receive 0.9, within 10% receive 0.7, within 100% receive 0.3, and larger errors receive 0.0. Non-numeric arguments use exact string matching. Unverifiable fields (experiment names, free-text questions, ambiguous interval semantics) are stripped before comparison.

Note that training uses these soft tolerance bands while evaluation uses near-exact matching (relative tolerance 10^{-9}). This is a deliberate design choice, not a misalignment: binary argument scoring on numeric floats would provide near-zero gradient signal to a 3B model (most outputs would score 0.0), whereas graduated credit creates a smooth optimization landscape that rewards progressively more precise argument generation. Crucially, soft argument scoring is safe because it is gated behind binary tool correctness the model only receives argument reward when it has already called the correct tool, preventing partial-credit exploitation of the kind that motivated our switch from soft to binary tool scoring (see Section 2.3).

The task-completion judge is Llama-3.3-Nemotron-Super-49B-v1.5-FP8 (NVIDIA, 2025), a frozen open-weight model used to score open-ended task completion. When the policy ends generation on a tool call without producing a final text answer, actual tool execution outputs are used as a proxy answer for judging.

State-aware online scoring replays prior-turn expected tool calls from the dataset to reconstruct turn-local context (e.g., populating simulation dataframes before scoring an `ask_question` turn). The KL penalty coefficient was $\beta = 0.1$, with a generation batch size of 6.

As for evaluation metrics, we used a stateful execution harness and computed argument correctness, tool correctness, and task completion on held-out conversations. For *argument correctness*, we evaluated whether each tool call included the correct input parameters given the user query and expected tool usage, with

$$\text{Argument Correctness} = \frac{1}{|\mathcal{I}|} \sum_{i=1}^{|\mathcal{I}|} m(\widetilde{\boldsymbol{a}}_i, \boldsymbol{a}_i), \tag{2}$$

where $\mathcal{I}$ indexes the expected tool calls within a turn, $\widetilde{\boldsymbol{a}}_i$ and $\boldsymbol{a}_i$ denote the predicted and ground-truth arguments, and $m(\cdot,\cdot) \in \{0, 0.5, 1\}$ returns 1 if all fields match (using near-exact numeric comparison with relative tolerance 10^{-9}), 0.5 if a strict subset matches, and 0 otherwise. We estimated *tool correctness* by comparing the executed tool calls against the expected ones for each turn using exact matching with ordering, defined as

$$\text{Tool Correctness} = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \mathbb{I}\big[\hat{t}_i = t_i\big] \,, \tag{3}$$

where $\hat{t}_i$ and t_i denote the predicted and expected tool names at position i in the ordered sequence for a given turn. Task completion was computed using an LLM judge over stateful traces, and aggregated over turns as

$$\text{Task Completion} = \frac{1}{|\mathcal{U}|} \sum_{u=1}^{|\mathcal{U}|} s_u, \tag{4}$$

where $s_u \in [0, 1]$ is the judge score for turn u and $\mathcal{U}$ is the set of evaluated turns.

C. Ablation Studies

Table 6 reports ablations isolating the contribution of each design choice. We evaluate four factors:

- **Training stages** (top section of Table 6): SFT provides the largest absolute improvement, lifting argument correctness from 26% to 86% (+60 pp) by teaching structured tool-call formatting and domain-specific argument schemas. GRPO further refines the policy (+1/+2/+2 pp). The modest GRPO gain on task completion is consistent with the reward design ($\lambda = 0.33$ each); deterministic tool and argument rewards provide clearer gradient signal than the LLM-judged task completion (see Section D).

- **SFT initialization** (middle section): Without SFT, GRPO-only achieves only 66%/46%/46%, confirming that SFT provides the structured tool-calling foundation that GRPO refines. The −32/−42/−31 pp gap versus SFT+GRPO demonstrates that RL alone cannot discover reliable tool-calling behavior from scratch in a 3B model.

- **State-aware vs. stateless (trajectory-level proxy)** (bottom section): The stateless variants disable ground-truth state replay, causing errors from prior turns to propagate into subsequent turns without correction, thereby approximating full-trajectory rollouts where the model must operate on its own (potentially erroneous) state. This degrades both settings, but the effect is substantially larger for SFT+GRPO (−3/−5/−6 pp) than GRPO-only (−5/−4/−3 pp). The SFT+GRPO model has learned to *exploit* prior-turn context (experiment names, species lists, simulation outputs) for precise argument construction; when that context is corrupted by error propagation, its reliance on it causes larger regression on argument and task metrics. The GRPO-only model never learned reliable tool formatting, so it has less context-dependent behavior to lose. This interaction confirms that SFT initialization and state-aware training are complementary, and validates the benefit of per-turn state reconstruction over trajectory-level approaches.

- **Multi-seed variance**: Evaluation (n=2, seeds 42 and 0) of SFT+GRPO confirms negligible training variance: tool ±0.001, argument ±0.000, task ±0.005.

Table 6. Stage-wise ablation. Top: progressive training stages. Middle and bottom: state-aware vs. stateless comparison (paired by initialization). Stateless variants disable ground-truth state replay, causing errors to propagate across turns as in full-trajectory rollouts. Δ shows the degradation from removing state reconstruction.

Model Variant	Tool Correctness	Argument Correctness	Task Completion	Δ Tool	Δ Arg	Δ Task
Qwen2.5-3B baseline	0.65	0.26	0.49			
+ SFT	0.97	0.86	0.75			
+ SFT+GRPO (state-aware)	**0.98**	**0.88**	**0.77**			
SFT+GRPO (state-aware)	0.98	0.88	0.77			
SFT+GRPO (stateless)	0.95	0.83	0.71	−3	−5	−6
GRPO-only (state-aware)	0.66	0.46	0.46			
GRPO-only (stateless)	0.61	0.42	0.43	−5	−4	−3

D. Additional Analyses

Table 7 presents a per-tool breakdown across the three training stages. Key observations: (i) The base model can often identify the correct single tool (e.g., 96.7% for `simulate_model`) but catastrophically fails at argument generation (5.0%), confirming that tool selection and argument construction are distinct capabilities. (ii) SFT provides the largest absolute gains, particularly on argument correctness for complex tools like `steady_state` (0.083 → 0.889) and `parameter_scan` (0.0 → 0.521). (iii) GRPO further refines accuracy, with notable improvements on `simulate_model` task completion (0.872 → 0.922, +5.0 pp) and `parameter_scan` task completion (0.807 → 0.848, +4.1 pp). (iv) On multi-tool turns (`simulate_model+ask_question`), GRPO achieves perfect tool and argument correctness (1.000), while task completion (67.9%) is lower due to the open-ended nature of the LLM judge metric on these complex turns; this pattern is consistent across all evaluated models, indicating a metric-level property rather than a framework limitation.

Table 7. Per-tool breakdown on the held-out test split. Multi-tool entries denote turns requiring multiple tool invocations. N is the number of evaluated turns per tool type.

Tool(s)	Model	N	Tool	Arg	Task
ask_question	Base	96	0.688	0.667	0.464
	SFT	96	0.969	0.969	0.716
	SFT+GRPO	96	0.990	0.990	0.723
get_modelinfo	Base	48	0.812	0.229	0.704
	SFT	48	1.000	0.854	0.777
	SFT+GRPO	48	1.000	0.875	0.787
simulate_model	Base	60	0.967	0.050	0.845
	SFT	60	1.000	0.917	0.872
	SFT+GRPO	60	1.000	0.917	0.922
steady_state	Base	36	0.722	0.083	0.451
	SFT	36	0.944	0.889	0.703
	SFT+GRPO	36	0.972	0.917	0.717
parameter_scan	Base	48	0.375	0.000	0.230
	SFT	48	0.958	0.521	0.807
	SFT+GRPO	48	0.958	0.521	0.848
simulate_model + ask_question	Base	12	0.000	0.000	0.000
	SFT	12	0.917	0.917	0.704
	SFT+GRPO	12	1.000	1.000	0.679
steady_state + ask_question	Base	24	0.125	0.104	0.106
	SFT	24	0.958	0.896	0.510
	SFT+GRPO	24	0.958	0.896	0.506

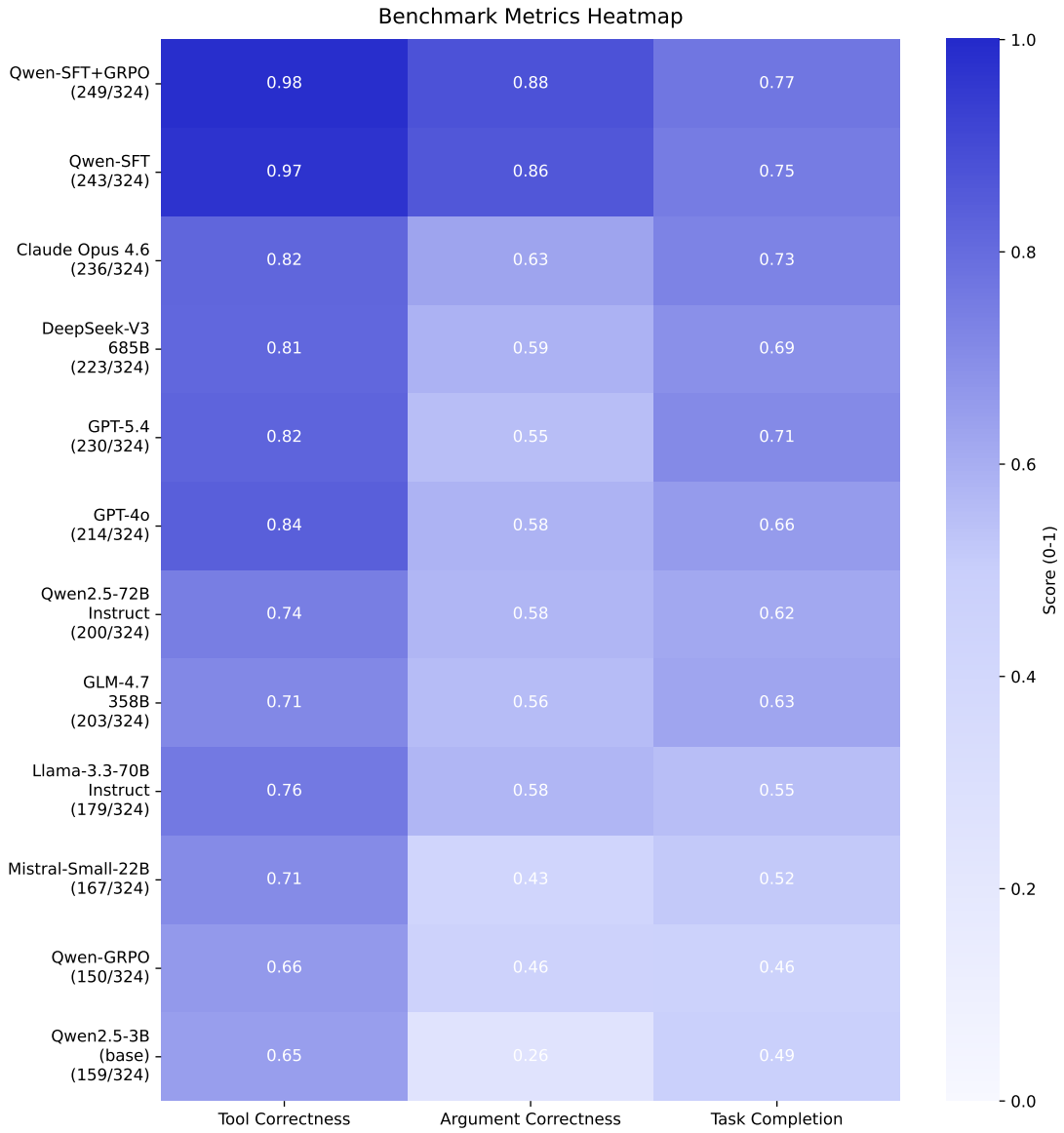

Figure 4. Benchmark metrics heatmap across representative models (top performers from each family). Models are sorted by composite performance (top to bottom). Numbers in parentheses indicate successfully scored turns out of 324 total. Qwen-SFT+GRPO (top) dominates on tool and argument correctness; all baseline models score at most 0.63 on argument correctness.

E. Per-Scenario Analysis

Figures 5-7 present per-scenario heatmaps for each evaluation metric across all 10 scenarios and 12 representative models (top performers from each model family, plus our ablation variants).

Tool Correctness (Fig. 5) Qwen-SFT+GRPO achieves perfect tool correctness (1.000) on 8 out of 10 scenarios, with the only imperfect scores on Scenario 5 (`parameter_scan` only, 0.933) and Scenario 7 (0.944). In contrast, baseline models show high variance across scenarios: GPT-4o ranges from 0.58 (Scenario 10) to 1.00 (Scenario 1), while Mistral-Small-2409 scores below 0.65 on 4 of 10 scenarios. The base Qwen2.5-3B model struggles most on multi-tool scenarios (Scenarios 4, 7, 10) where it must chain multiple tool invocations.

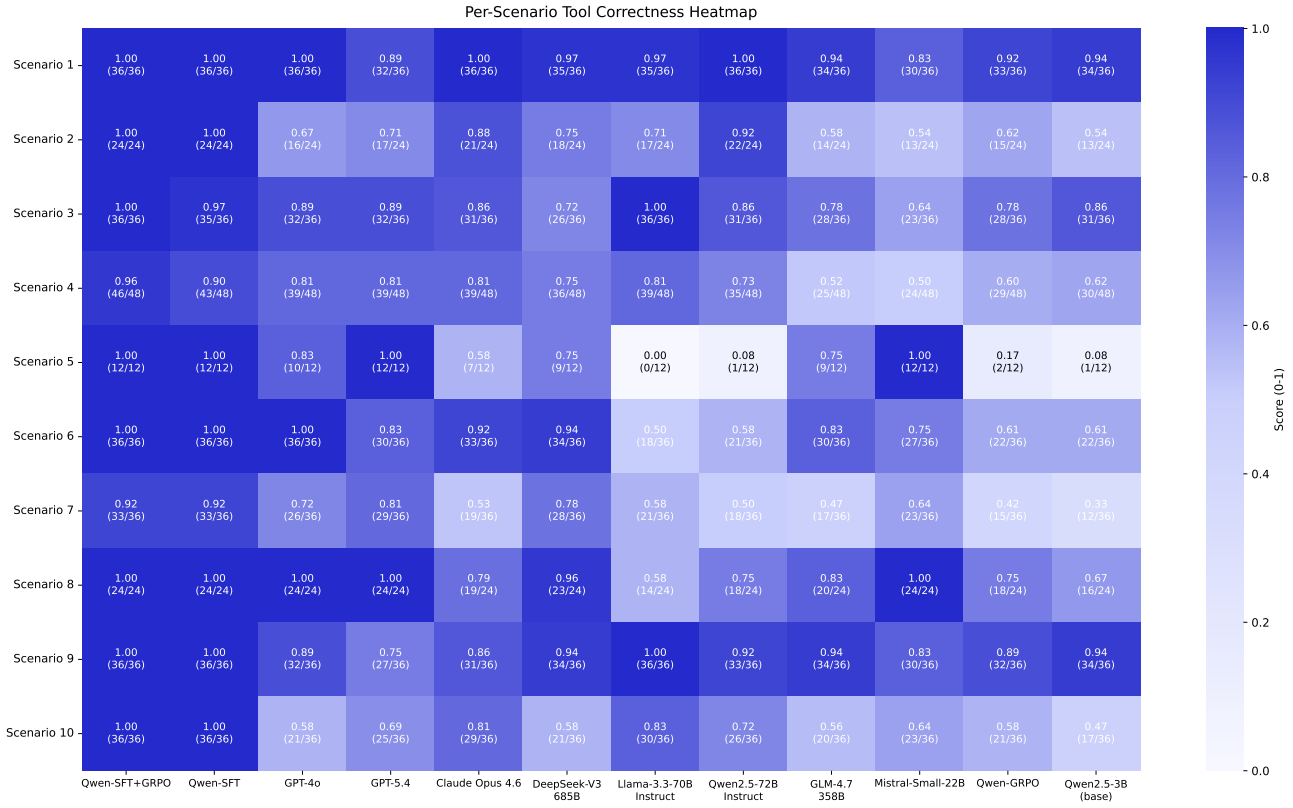

Figure 5. Per-scenario tool correctness heatmap. Each cell shows the mean score with count in parentheses. Qwen-SFT+GRPO achieves perfect or near-perfect tool selection across nearly all scenarios.

Argument Correctness (Fig. 6) The argument correctness gap between our optimized models and baselines is most pronounced. Qwen-SFT+GRPO exceeds 0.80 on 8 of 10 scenarios, whereas the best baseline model (Claude Opus 4.6) never exceeds 0.63 on any scenario. Scenarios involving `parameter_scan` (5, 6, 7, 8) are consistently harder for all models due to the complex argument schema (scan ranges, step sizes, multiple species). The SFT→GRPO gain is most visible on Scenarios 2, 8, and 10 where argument structures involve steady-state and multi-tool chains.

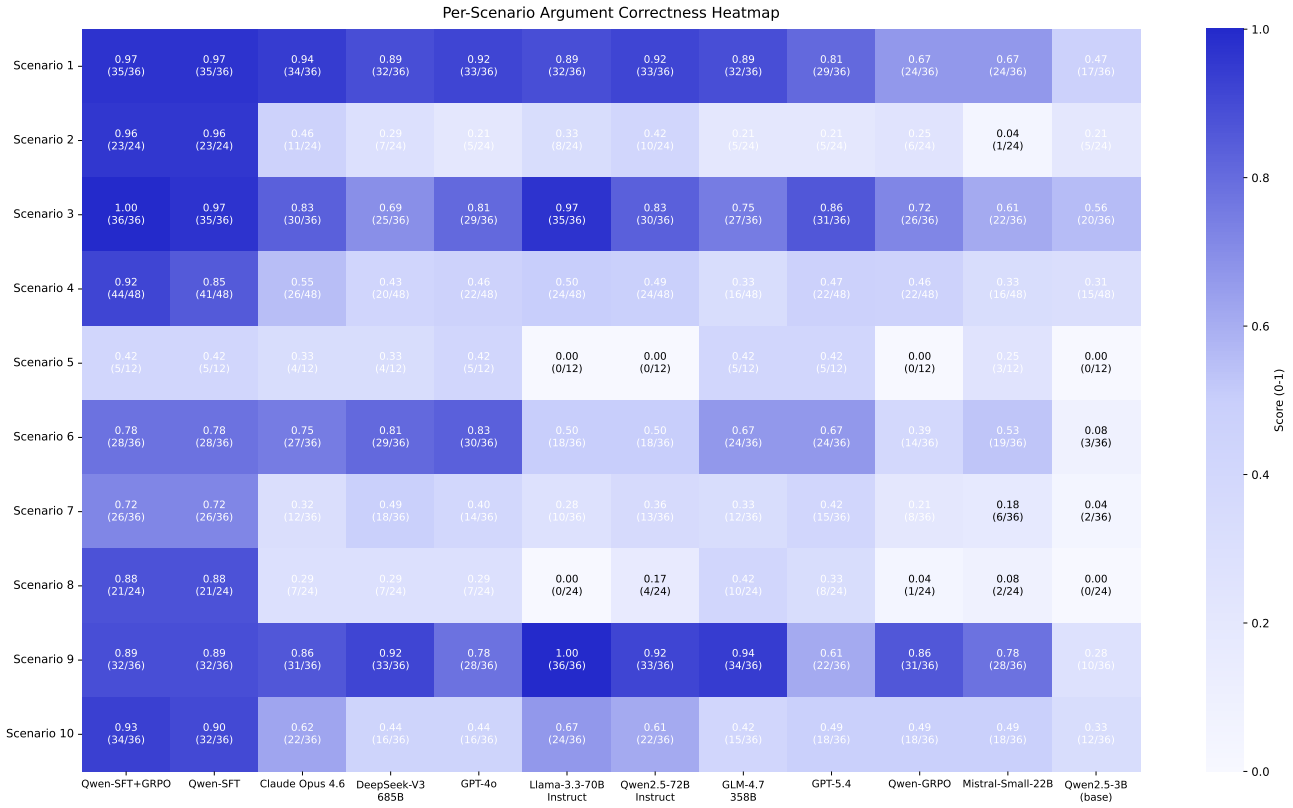

Figure 6. Per-scenario argument correctness heatmap. The gap between Qwen-SFT+GRPO/SFT and baseline models is largest here, demonstrating the value of domain-specific fine-tuning for structured argument generation.

Task Completion (Fig. 7) Task completion scores are more uniform across models, reflecting the open-ended nature of the LLM judge metric. Qwen-SFT+GRPO leads on most scenarios but the margins are smaller, particularly on Scenarios 5-8 involving `parameter_scan` and multi-tool chains. The narrower gaps on this metric are consistent with the reward design, which weights all three components equally ($\lambda = 0.33$ each); while tool and argument rewards are deterministic, task completion reflects natural-language answer quality, where all models produce lower scores on complex tool outputs.

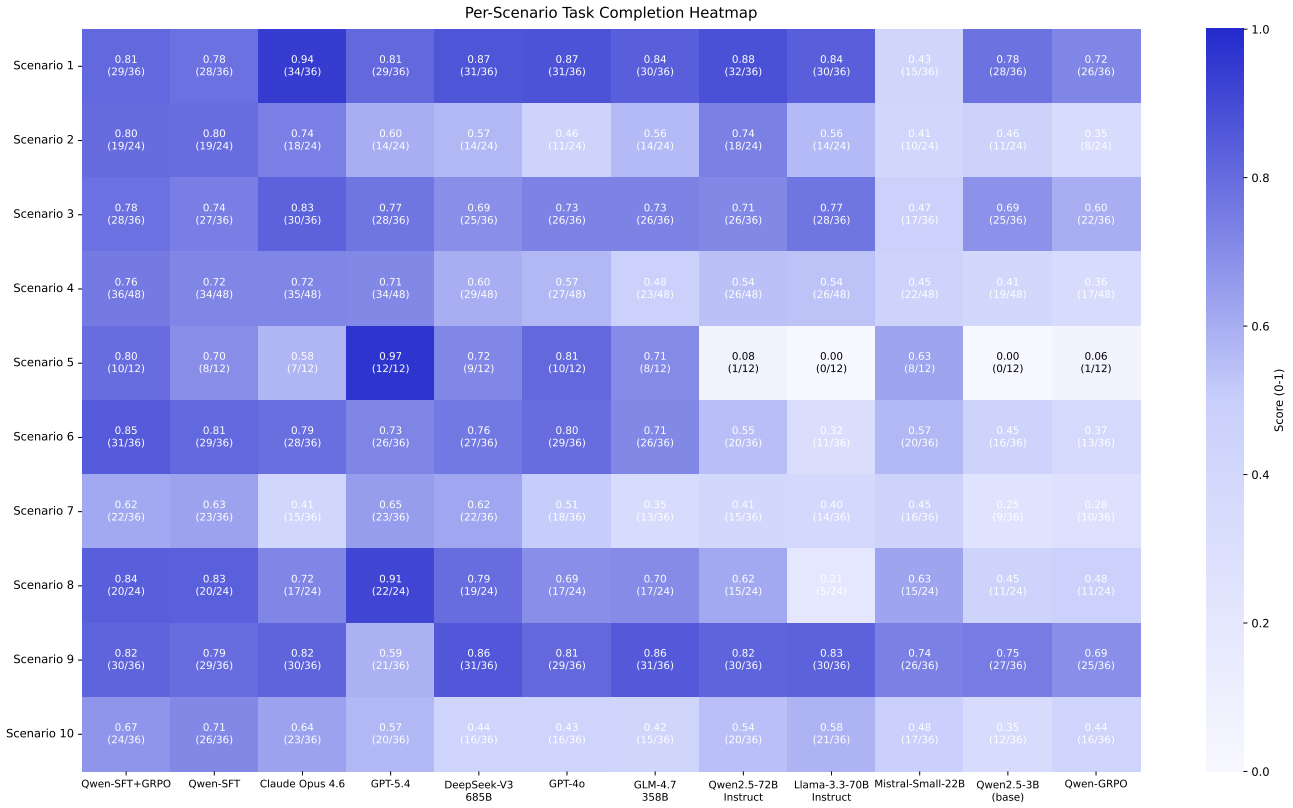

Figure 7. Per-scenario task completion heatmap. Scores are more uniform across models on this metric, reflecting the open-ended nature of LLM judge evaluation. Qwen-SFT+GRPO leads on most scenarios, with narrower margins on parameter scan scenarios (5-8) where all models produce lower scores.

Turn-Level Breakdown by Scenario (Fig. 8) Figure 8 decomposes Figure 3 into individual scenarios, revealing where task completion degrades and why. Two patterns emerge. First, standalone `ask_question` turns consistently show reduced task completion across all models. In Scenario 1 (Turn 3), Scenario 2 (Turn 2), Scenario 3 (Turns 2 and 3), and Scenario 9 (Turn 3), task completion drops even when tool and argument correctness remain high. These turns require parsing large simulation or steady-state dataframes to extract precise numeric answers, a task that degrades when the tabular output exceeds the answering model's effective capacity. Second, multi-tool combination turns exhibit more severe degradation. In Scenario 4 (Turn 2: `simulate_model+ask_question`), Scenario 7 (Turn 2: `steady_state+ask_question`), and Scenario 10 (Turn 2: `steady_state+ask_question`), all models including our optimized agent show sharper drops in task completion. These turns require executing a state-building tool and immediately querying its results within a single turn, compounding the dataframe parsing challenge with intra-turn tool coordination. Notably, tool and argument correctness remain near-perfect for Qwen-SFT+GRPO on these same turns, confirming that the task completion bottleneck arises from answer generation rather than tool execution. This breakdown motivates two future directions: replacing the current `ask_question` tool with a dedicated data-analysis agent capable of handling large tabular outputs, and developing decomposition strategies that explicitly optimize intra-turn multi-tool interactions.

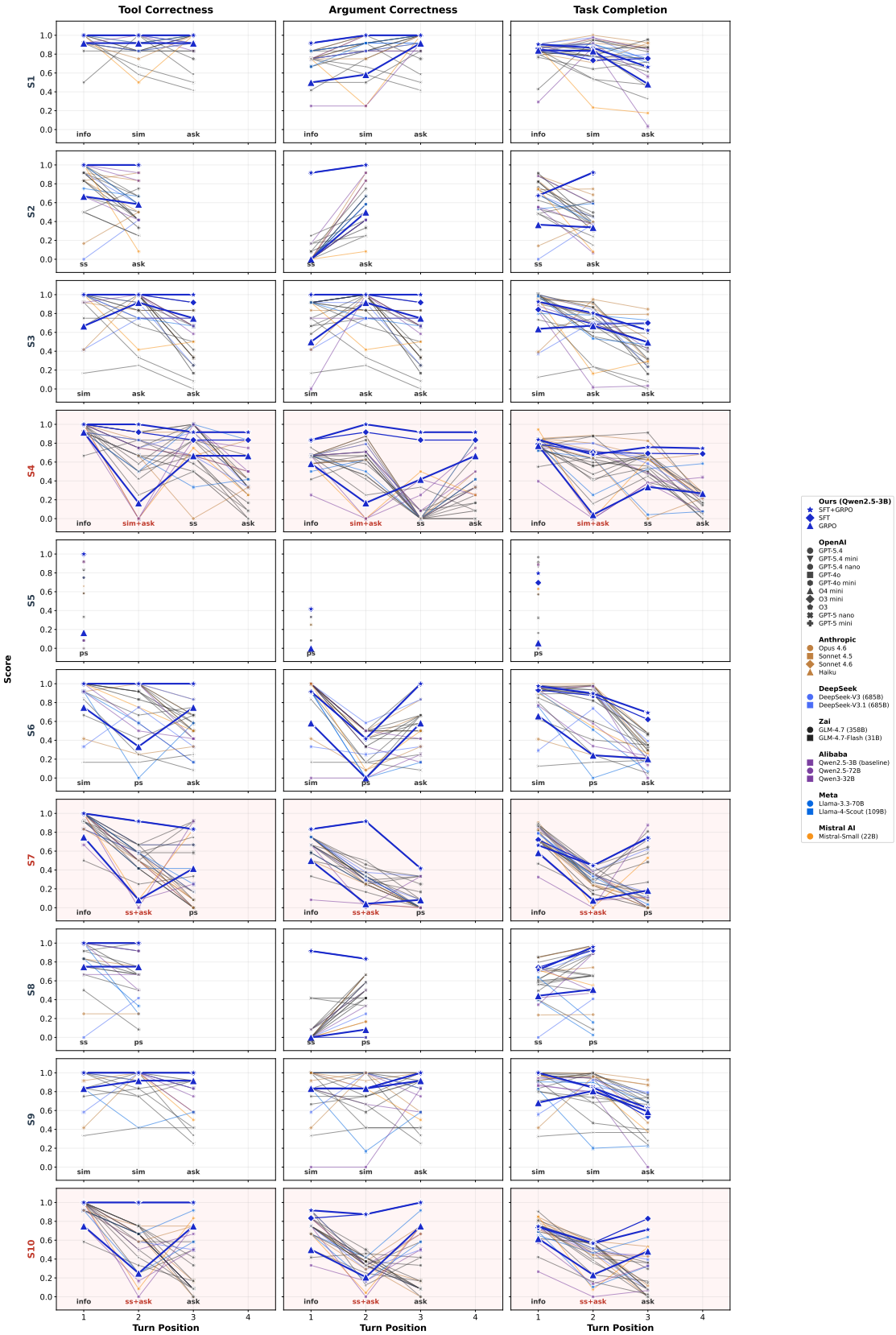

Figure 8. Per-scenario turn-level performance breakdown across all evaluated models and three metrics. Each row corresponds to one scenario (S1-S10, see Table 5 for full definitions); columns show tool correctness, argument correctness, and task completion. Tool labels below each turn indicate which tool(s) are invoked at that position; multi-tool turns are highlighted in red. Standalone `ask_question` turns and multi-tool combination turns (highlighted rows: Scenarios 4, 7, 10) exhibit the sharpest task completion drops, even when tool and argument correctness remain high.

F. Out-of-Distribution Generalization

To further assess whether the optimized policy generalizes beyond the training distribution, we evaluate Qwen-SFT+GRPO on a separate out-of-distribution (OOD) benchmark comprising 9 scenarios (340 conversations, 554 turns) that differ from the 10 in-distribution training scenarios in tool composition, tool novelty, or both. In particular, the OOD scenarios are organized into three categories (Table 8):

1. **Novel composition of training tools** (Scenario 18): Tool chains that use only tools seen during training but in combinations never encountered (*e.g.*, `parameter_scan` → `steady_state` → `ask_question`).

2. **Novel tools only** (Scenarios 11, 16): Single-turn invocations of tools that have zero training examples (`search_models`, `get_annotation`). The model must rely entirely on schema-following capabilities acquired through SFT and GRPO.

3. **Mixed training and novel tools** (Scenarios 12-15, 17, 19): Multi-turn chains that interleave training tools with novel tools (*e.g.*, `simulate_model` → `custom_plotter` → `ask_question`), testing compositional generalization across both familiar and unseen tool interfaces.

Furthermore, Table 9 presents the per-category and overall results. Overall, the optimized agent achieves 89% tool correctness, 78% argument correctness, and 71% task completion on the OOD benchmark. While these scores represent a decrease from the in-distribution results (98%/88%/77%), the degradation is moderate and non-uniform across categories. The mixed category, where novel tools are interleaved with training tools, shows the strongest transfer (tool=0.96, arg=0.83, task=0.81), indicating that familiarity with a subset of the tool chain provides sufficient scaffolding for the model to generalize to unseen tools via schema following. The novel tools category (S11, S16), despite having zero training examples, achieves 95% tool correctness and 91% argument correctness, demonstrating that the SFT+GRPO pipeline equips the model with robust schema-following capabilities that transfer to entirely new tool interfaces. The weakest performance occurs in the novel composition category (S18: `parameter_scan` → `steady_state` → `ask_question`), where all individual tools were seen during training but their specific combination was not. This scenario achieves only 71% tool correctness and 64% argument correctness. Inspection reveals that the primary failure mode is on `parameter_scan` turns, consistent with the in-distribution finding that this tool has the most complex argument schema (Table 7). When combined with the reversed ordering relative to training scenarios (where `steady_state` typically precedes or follows `parameter_scan` in different configurations), the model struggles with argument construction, particularly for scan range specification and interval computation.

These results suggest that our optimization framework provides meaningful generalization to out-of-distribution tool compositions and novel tool interfaces, with the strongest transfer occurring when at least part of the tool chain overlaps with training experience.

Table 8. Out of distribution evaluation scenarios. Tools marked with * are novel (zero training examples). Category labels: **NC** = novel composition of training tools, **NT** = novel tools only, **MX** = mixed training and novel tools.

Scenario	Category	#Turns	Tool Chain
18	NC	3	`parameter_scan` → `steady_state` → `ask_question`
11	NT	1	`search_models`*
16	NT	1	`get_annotation`*
12	MX	2	`search_models`* → `get_modelinfo`
13	MX	2	`simulate_model` → `custom_plotter`*
14	MX	2	`get_modelinfo` → `get_annotation`*
15	MX	3	`simulate_model` → `custom_plotter`* → `ask_question`
17	MX	3	`simulate_model` → `ask_question` → `custom_plotter`*
19	MX	3	`search_models`* → `simulate_model` → `parameter_scan`

Table 9. Out of distribution generalization results for Qwen-SFT+GRPO, grouped by category. In-distribution (ID) results are shown for reference. N denotes the number of evaluated turns.

Category	N	**Tool**	**Arg**	**Task**
Novel composition (S18)	163	0.71	0.64	0.48
Novel tools (S11, S16)	23	0.95	0.91	0.72
Mixed (S12-15, 17, 19)	368	0.96	0.83	0.81
OOD overall	**554**	**0.89**	**0.78**	**0.71**
ID reference (Table 1)	324	0.98	0.88	0.77

G. Sensitivity Analysis

To assess the robustness of our framework to hyperparameter choices, we conduct a sensitivity analysis over the KL penalty coefficient β and learning rate (Table 10), which are the two hyperparameters most critical to GRPO training stability and convergence.

Table 10. Sensitivity analysis over KL penalty coefficient β and learning rate. The default configuration (β=0.1, LR $= 5 \times 10^{-7}$) achieves the best results. Reducing the KL penalty while increasing the learning rate causes KL divergence and catastrophic forgetting of structured tool-calling behavior.

Hyperparameter	**Value**	**Tool**	**Arg**	**Task**
Default (β=0.1, LR $= 5\times10^{-7}$)		0.98	0.88	0.77
β=0.004, LR $= 1\times10^{-6}$		0.94	0.83	0.71

The default configuration uses a higher KL penalty (β=0.1) paired with a conservative learning rate (5×10^{-7}). This combination constrains the policy to remain close to the SFT reference distribution while making incremental improvements through reward maximization. The alternative configuration (β=0.004, LR $= 1\times10^{-6}$) reduces the KL constraint by $25\times$ while doubling the learning rate, effectively giving the optimizer greater freedom to deviate from the reference policy. This aggressive setting led to KL divergence during training, manifesting as rapid entropy collapse and catastrophic forgetting of structured tool-calling behavior. The resulting model suffers degradation across all three metrics: tool correctness drops from 98% to 94% (-4 pp), argument correctness from 88% to 83% (-5 pp), and task completion from 77% to 71% (-6 pp).

These results are consistent with prior findings that GRPO training is sensitive to the balance between exploration and reference anchoring (Shao et al., 2024). In structured tool-calling tasks, where the output must conform to precise schemas, insufficient KL regularization allows the policy to drift into degenerate modes (e.g., omitting required argument fields or generating malformed JSON). The stronger KL penalty preserves the schema-following behavior acquired during SFT while still permitting reward-driven refinement of argument values and tool sequencing. This finding underscores the importance of conservative hyperparameter selection for RL post-training of tool-calling agents, where the structured output format is fragile relative to free-text generation tasks.