The Interplay of Harness Design and Post-Training in LLM Agents

Kyungmin Kim[1,*], **Youngbin Choi**[1,*], **Seoyeon Lee**[1], **Suhyeon Jun**[2],
Dongwoo Kim[1,2,†], **Sangdon Park**[1,2,†]
[1]Graduate School of Artificial Intelligence, POSTECH,
[2]Department of Computer Science and Engineering, POSTECH,
{kkm959595, choi.youngbin, seoyeon26, suhyeonjun, dongwoo.kim, sangdon}@postech.ac.kr

Abstract

Tool-integrated LLM agents are often wrapped within a *harness*: the scaffolding that determines which tools are exposed, how they are described, and what auxiliary information accompanies each per-step observation. While agents are routinely post-trained, this scaffolding is typically treated as a *fixed* engineering detail, with design effort limited to the training-free regime. Moreover, existing post-training algorithms assume a static environment, even though tool environments and tasks often shift upon deployment. To address this gap, we extend `ALFWorld` (i) to treat the harness as a controllable design dimension and (ii) to support evaluation under task and tool environment shifts. Building on this, we systematically analyze how the harness design influences post-training in both in-distribution and out-of-distribution (OOD) settings. We empirically show that *harness-aware post-training* not only improves in-distribution performance but also enables agents to robustly adapt to OOD settings. Under a harness with minimal design effort, post-training suffers a drastic performance drop under stronger tool environment shifts, further highlighting the importance of harness-aware post-training under such shifts.

1 Introduction

LLM agents are designed to solve complex problems that often require multi-step reasoning [1, 2, 3, 4, 5]. Among them, *tool-integrated agentic systems* extend LLM capabilities by proactively interacting with environments through predefined tools [6, 7, 8, 9]. Accordingly, performance depends not only on producing correct final outputs but also on appropriately invoking tools, which has motivated increasing efforts to post-train pretrained LLMs for effective tool use [10].

A factor that is often overlooked in this line of work is the *harness*: the code that wraps the LLM, determining what is stored, retrieved, and presented to it, and what is done with its outputs [11, 12, 13]. While the harness is, in principle, an engineering detail, in practice it is a major determinant of overall performance: even with the same task and model, two harnesses that differ only in how they present auxiliary information can produce drastically different success rates [4, 14]. For this reason, harness design has attracted increasing attention, ranging from manually curated prompts [15, 16, 17] to recent attempts at automating the design itself [18, 19]. All of these efforts, however, are limited to a training-free regime, treating the harness as a way to elicit better performance, typically from a closed-source model. How harness design interacts with post-training, in contrast, remains largely unexamined.

[*]Equal contribution.
[†]Co-advising.

RLxF: Reinforcement Learning from World Feedback Workshop in ICML 2026

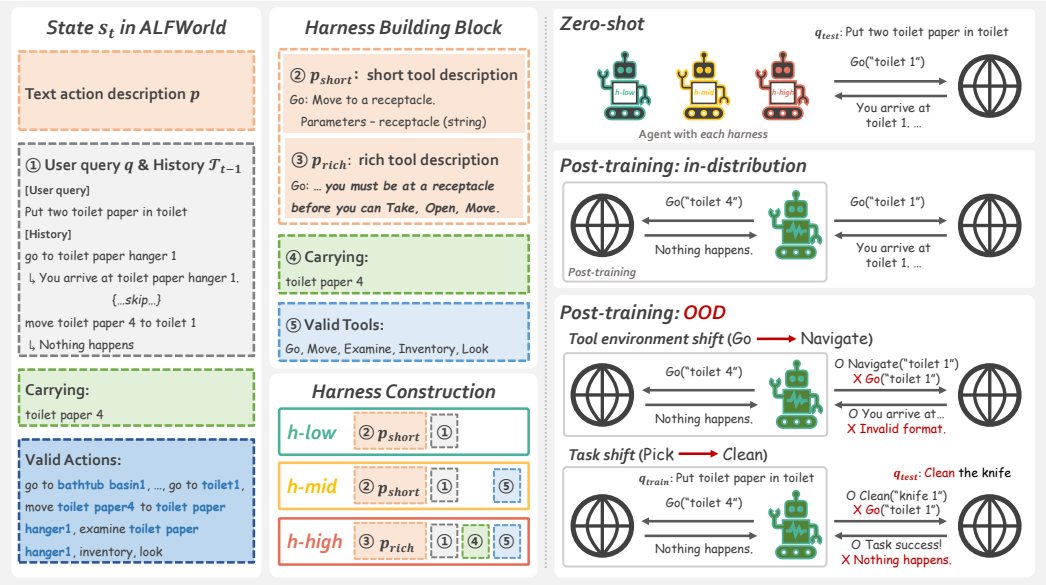

Figure 1: Overview of experimental design, built on our extension of `ALFWorld` [20] tailored to tool-integrated agentic tasks. (Left) State of the original text-based `ALFWorld` environment. (Middle) Harness building blocks for tool-integrated agentic tasks; different combinations of these blocks yield harnesses with varying levels of informativeness. (Right) Agents under each harness are evaluated in zero-shot and post-training regimes, where the latter spans in-distribution and OOD settings.

In parallel, most existing post-training approaches for tool-integrated agents implicitly assume a static deployment scenario, in which both the task distribution and the tool environment, defined as the set of tools and their invocation protocols, remain fixed [21, 22, 23, 24, 25, 26, 27]. In practice, this assumption is often violated: real-world deployments expose agents to shifts in both the task distribution and the tool environment. A first source of shift arises from changes in the task distribution, a standard out-of-distribution (OOD) problem at the input level [28, 29, 30]. For example, an agent originally designed for workload scheduling may later be repurposed for itinerary planning, shifting objectives, constraints, and tool usage patterns, which we refer to as *task shift*. A second, more subtle source stems from changes in the tool environment itself: tools may be updated, modifying invocation protocols while preserving functionality, which we refer to as *tool environment shift*. This introduces a distinct OOD setting in which the task distribution remains fixed but the action interface itself changes, and it remains comparatively underexplored [31, 32].

In this paper, we ask the following question: *how does harness design influence post-training LLM agents in both in-distribution and OOD settings?* To this end, we extend `ALFWorld` [20], a text-based planning environment, into a benchmark tailored to tool-integrated agentic tasks. Specifically, we (i) treat the harness as a controllable design dimension and (ii) modularize the benchmark to support evaluation both in-distribution and under tool environment shift and task shift settings (Fig. 1). While existing post-training algorithms attain high success rates on `ALFWorld` [25], we observe that this performance largely stems from *harness-aware post-training*, where a specific harness is fixed a priori (Fig. 3). In particular, the original benchmark provides the agent with the complete set of feasible tool calls at every step, which constitutes a highly informative harness. Such a harness, however, comes at a cost: it can be constructed only when the transition dynamics of the `ALFWorld` environment are fully specified. Yet this cost is rarely made explicit in prior work. To address this gap, we provide three harnesses with varying levels of informativeness and empirically examine *whether the additional cost of harness design is indispensable for post-training*.

We empirically show that harness-aware post-training not only improves in-distribution performance but also enables agents to robustly adapt to OOD settings. Notably, models post-trained on harnesses with low design effort suffer from a drastic performance drop under more severe tool environment shift. Furthermore, harness informativeness tends to improve model performance under the task shift scenario, highlighting the importance of harness-aware post-training for the OOD robustness of agents. In addition, we observe that applying a harness only after post-training under low-effort

harness yields performance inferior to harness-aware post-training, providing actionable insight into when to apply the harness during post-training.

2 Preliminaries

We outline the formulation of multi-step tool-integrated agentic tasks and representative post-training methods, along with the notation used throughout the paper.

2.1 Multi-step Tool-integrated Agentic Tasks

Let $\mathcal{V}$ be a token vocabulary. Given a system prompt $p \in \mathcal{V}^*$ and a user query $q \in \mathcal{V}^*$, an LLM agent $\pi : \mathcal{S} \to \mathcal{A}$ generates a sequence of action tokens $a_t \coloneqq \pi(s_t)$ at each time step $t \in \{1, \ldots, T\}$. Here, $s_t \in \mathcal{S}$ is the state at step t provided to the agent, and T is the total number of tool calling steps. Specifically, $s_t \coloneqq (p, q, \mathcal{T}_{t-1})$, where $\mathcal{T}_{t-1}$ is a history of tool calling steps up to the $(t-1)$-th step, and $\mathcal{T}_0$ is an empty sequence.

The tool environment $\texttt{TE} : \mathcal{V}^* \to \mathcal{V}^*$ then plays two roles: in the *tool calling phase*, it returns the tool calling result $e_t \coloneqq \texttt{TE}((s_t, a_t))$; in the *state reconstruction phase*, it aggregates e_t with a_t and $\mathcal{T}_{t-1}$ into a history $\mathcal{T}_t \coloneqq \texttt{TE}((s_t, a_t, e_t))$, provided to the agent as the next state $s_{t+1} = (p, q, \mathcal{T}_t)$. As such, the t-th tool calling step can be represented as:

$$\underbrace{s_t = (p, q, \mathcal{T}_{t-1}) \xrightarrow[+a_t = \pi(s_t)]{} (p, q, \mathcal{T}_{t-1}, a_t) \xrightarrow[+e_t = \texttt{TE}((s_t, a_t))]{}}_{\text{Tool Calling Phase}} \underbrace{(p, q, \mathcal{T}_{t-1}, a_t, e_t) \xrightarrow[+\mathcal{T}_t = \texttt{TE}((s_t, a_t, e_t))]{-(\mathcal{T}_{t-1}, a_t, e_t)} s_{t+1} = (p, q, \mathcal{T}_t).}_{\text{State Reconstruction Phase}}$$

Remark. While "harness" generally refers to any wrapper around the agent [17], we restrict it to wrappers acting through p and TE. These are the core components of the agentic workflow that shape what the agent observes in our environment. As detailed in §3, harness, tool schema, and task type jointly affect these transition dynamics. In particular, tool environment shift and task shift pose challenges for training an agent that remains robust to deployment-time workflows differing from those encountered during training.

2.2 Training Tool-integrated Agentic Systems

We model a multi-step tool calling process as a Markov decision process (MDP) at a sequence level. The goal is to find a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, parameterized by $\theta \in \Theta$, that maximizes $\mathcal{J}(\theta) = \mathbb{E}[R(\tau)]$, where $\tau \coloneqq \{(s_1, a_1, s_2), \ldots, (s_T, a_T, s_{T+1})\}$ is an episode-level trajectory and $R : (\mathcal{S} \times \mathcal{A} \times \mathcal{S})^T \to \mathbb{R}$ is an *episode-level* reward function. A common choice is $R(\tau) \coloneqq \sum_{t=1}^{T} \gamma^{t-1} r_t$, where $\gamma \in [0, 1]$ is a discount factor and $r_t \coloneqq r(s_t, a_t, s_{t+1})$ is a *step-level* reward.

Group-relative policy optimization (GRPO) [33] is a widely adopted algorithm for post-training LLMs due to its computational efficiency in estimating $\nabla_\theta \mathcal{J}(\theta)$. However, as GRPO uniformly assigns the episode-level advantage across all steps, it suffers from a credit assignment problem in long-horizon tasks [34, 35]. Unlike GRPO, group-in-group policy optimization (GiGPO) [25] differentiates the advantage assigned to each step within the same episode by introducing a step-level advantage on top of the episode-level advantage. A formal description of both algorithms is provided in Appendix C.

Yet both algorithms assume a static agentic workflow. Furthermore, existing works that adopt such algorithms in `ALFWorld` fix a highly informative harness in advance, rather than treating it as a design variable. As such, they lack a controlled analysis of how harness-aware post-training affects performance and OOD robustness of LLM agents.

3 Experiment Setup

3.1 `ALFWorld` for Tool-based Agentic Tasks

`ALFWorld` [20] is a text-based planning environment with $3,827$ task instances across six categories of common household activities: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick 2). Every task

requires multi-step decision making, since it consists of a sequence of sub-goals. This structure makes ALFWorld a natural testbed for evaluating the planning capability of LLM agents.

We extend ALFWorld by reformulating its original tasks as tool-integrated reasoning problems. Specifically, text actions in ALFWorld can naturally be interpreted as tool calls: verbs correspond to tools and entities serve as arguments. For instance, a text action "Go to drawer 1" in the original ALFWorld environment is mapped to a tool call "Go(receptacle='drawer 1')".

In subsequent sections, we describe how our extended benchmark enables a controlled study of harness design in post-training LLM agents by modularizing three components: harness (§3.2), tool schema (§3.3), and task type (§3.4).

3.2 Harness

In tool-integrated agentic tasks, the harness determines how much guidance the agent receives about using the available tools. To this end, we instantiate three harnesses with varying degrees of informativeness, each building on the previous one: h-low, h-mid, and h-high.

Harness Ver.	Tool Description in p	Valid tools: (in $\mathcal{T}_t$)	Carrying: (in $\mathcal{T}_t$)
h-low	Short	–	–
h-mid	Short	✓	–
h-high	Rich	✓	✓

Table 1: Each harness version, characterized by how it shapes the system prompt p and the per-step history $\mathcal{T}_t$ produced by TE. Valid tools: and Carrying: are auxiliary lines appended to $\mathcal{T}_t$, listing the tools admissible at the current state and the object the agent is currently carrying, respectively.

The base version, h-low, is designed with low design effort: every tool carries only a one-line description in p (e.g., "Go to a receptacle" for Go), and each per-step history $\mathcal{T}_t$ contains nothing beyond the raw observation. h-mid additionally augments every per-step history with the set of tools admissible at the current state, leaving the one-line descriptions in p unchanged. Finally, h-high further expands each tool description in p into a richer form, covering its preconditions, interactions with other tools, and role in completing tasks. It also appends the object the agent is currently carrying to each per-step history, which it would otherwise have to retrieve through a separate Inventory call. Table 1 summarizes the three versions, and the corresponding tool descriptions and per-step histories are deferred to Appendix B.1.

Informative harnesses like h-mid and h-high, however, do not come for free. They must either be hand-designed by experts with sufficient prior knowledge of the environment, or be discovered through costly exploratory interaction with it. This cost is rarely made explicit. Most prior studies on ALFWorld in fact operate under a specific harness that goes beyond even h-high, providing the agent with the full set of feasible tool calls at every step, yet this is rarely flagged as an assumption. We instead treat the harness as a controllable variable, with h-low, h-mid, and h-high corresponding to three distinct levels of informativeness. We acknowledge that h-mid and h-high expose information that would otherwise require costly exploration to obtain. *We view* h-mid *and* h-high *as representing harnesses obtained through different amounts of such exploration, treating informativeness itself as a design choice that shapes agent performance* (**RQ1** in §4).

3.3 Tool Schema

The tool schema determines the surface form of admissible tool calls. Taking v1.0 as the base schema, we provide two environment shift scenarios of varying degrees, where the stronger shift (v2.0) extends the milder one (v1.1) through additional updates to the tool invocation protocols. Specifically, v1.0 is the base schema itself, consisting of 13 tools defined in ALFWorld with intuitive verb-style names such as Go, Take, Heat, Slice, and Look. The second, v1.1, applies paraphrasing: a semantics-preserving renaming of every tool name and parameter key from v1.0 into more descriptive forms (e.g., Go $\rightarrow$ NavigateTo, with its parameter receptacle $\rightarrow$ destination), leaving the cardinality of the tool set unchanged. The third, v2.0, additionally applies grouping by structural and functional similarity: building on v1.1, tools that share structural or functional roles are consolidated into a smaller, higher-level set, reducing the cardinality from 13 to 5. Each consolidated tool exposes its sub-operations through a discrete action parameter. For example, NavigateTo, OpenContainer, and CloseContainer are merged into a single ReceptacleControl tool, whose

action parameter takes one of three values: navigate_to, open_container, or close_container. Table 2 provides an illustrative example of the three versions, with full details in Appendix B.2.

Across the three versions, both p and TE are updated accordingly, with details deferred to Appendix B.2. Tool names, parameter keys, and descriptions in p are revised to match the active schema, while the overall structure of p is preserved. TE's validation logic is adjusted accordingly. As an illustration, the same operation "move to drawer 1" issued at step t takes a distinct valid form under each schema (Table 2). A tool call has a valid format only under the schema in which it is defined. Otherwise, even though the call expresses the same operation, TE returns $e_t =$ "Invalid tool format". We aim to analyze *whether harness-aware post-training enables agents to robustly adapt to such shifts in tool invocation protocols* (**RQ3** in §4).

Tool Schema	Example
v1.0	Go(receptacle="drawer 1")
v1.1	NavigateTo(destination="drawer 1")
v2.0	ReceptacleControl(action="navigate_to", target="drawer 1")

Table 2: Tool schema versions and the valid form of tool calls for the same operation "move to drawer 1" under each schema. v1.1 applies paraphrasing to v1.0, while v2.0 additionally groups tools by structural and functional similarity.

3.4 Task Type

The task type determines the distribution from which the user query q is drawn. We group the six task categories of ALFWorld by the minimum number of sub-goals required for completion into three difficulty levels: t-easy, t-med, and t-hard (Table 3). For example, task instances in the Pick category (e.g., "Put a plate on the coffee table") are categorized as t-easy, requiring a sequence of four sub-goals: find an object ("plate"), pick it up, find the correct location ("coffee table") to place it, and put it down there. This grouping serves as the basis for the task shift regime: training on one difficulty level and evaluating on the others constitutes a controlled shift in the input distribution, with the tool schema held fixed.

While task instances are categorized into three task groups, some sub-goals are shared across instances from different groups. For example, every task instance shares the same initial sub-goal: to find an object of the desired type. Because the corresponding tool calls are reused across instances (possibly with different arguments), an ideal post-training method should learn to generate such tool calls in a way that generalizes to task instances unseen during training, rather than overfitting to the training distribution. While these household activities are simple from a human perspective, results in §4 show that they present non-trivial challenges to LLM agents in both in-distribution and OOD scenarios. We further ask *whether the prior knowledge encoded in an informative harness helps an agent generalize across task groups* (**RQ3** in §4).

Group	Task	# Sub-goals	Example
t-easy	Pick	4	"Put a plate on the coffee table"
	Look	3	"Examine a book under the lamp"
t-med	Clean	5	"Clean the knife and put in the drawer"
	Heat	5	"Heat a mug and put on the coffee table"
	Cool	5	"Put a cool bottle on the countertop"
t-hard	Pick 2	8	"Put two pencils in the drawer"

Table 3: Six task categories of ALFWorld grouped into three difficulty levels (t-easy, t-med, t-hard) by the minimum number of sub-goals required for completion.

4 Analysis

In this section, we present a controlled analysis of how harness design influences the post-training of LLM agents, both in-distribution and under OOD settings. Specifically, we investigate the following research questions:

- **RQ1:** Does the influence of harness informativeness, observed at zero-shot (Obs. 1), extend to harness-aware post-training (Obs. 2)?
- **RQ2:** Can a harness be applied only after post-training, or should it be in place during training (Obs. 3)?
- **RQ3:** Does harness-aware post-training enable robustness to tool environment shift (Obs. 4) and task shift (Obs. 5)?

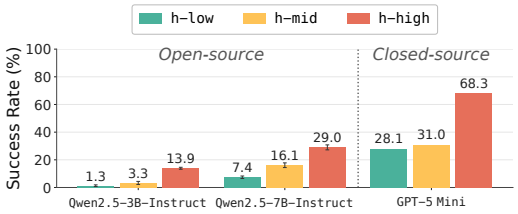

Figure 2: Zero-shot success rates without post-training: Pretrained agents are evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under each harness with tool schema fixed to `v1.0`. Zero-shot performance improves with harness informativeness, more pronounced for high-capacity models.

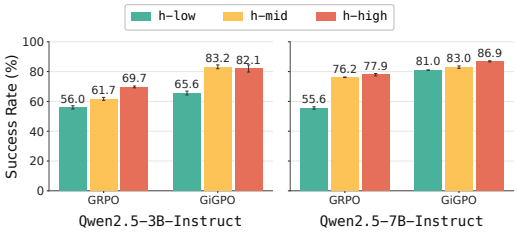

Figure 3: In-distribution success rates after post-training: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{all}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness and schema. Post-training performance tends to improve with harness informativeness.

4.1 Evaluation Protocol

Among $3,827$ task instances in `ALFWorld`, $3,553$ form the training set $\mathcal{D}^{\text{tr}}_{\text{all}}$ and the remaining 274 form the test set $\mathcal{D}^{\text{te}}_{\text{all}}$. The test set corresponds to the standard seen/unseen split of `ALFWorld` (140 seen and 134 unseen instances). Both sets are partitioned by task difficulty: $\mathcal{D}^{\text{tr}}_{\text{all}} := \mathcal{D}^{\text{tr}}_{\text{easy}} \cup \mathcal{D}^{\text{tr}}_{\text{med}} \cup \mathcal{D}^{\text{tr}}_{\text{hard}}$ and $\mathcal{D}^{\text{te}}_{\text{all}} := \mathcal{D}^{\text{te}}_{\text{easy}} \cup \mathcal{D}^{\text{te}}_{\text{med}} \cup \mathcal{D}^{\text{te}}_{\text{hard}}$. Here, $\mathcal{D}^{\text{tr}}_{\text{easy}}$, $\mathcal{D}^{\text{tr}}_{\text{med}}$, $\mathcal{D}^{\text{tr}}_{\text{hard}}$ refer to the training instances within each difficulty-based task group (`t-easy`, `t-med`, `t-hard`) defined in §3.4. The test splits are defined analogously. We report the *success rate*, defined as the fraction of test task instances in which the agent completes all required sub-goals within the maximum number of tool calling steps.

For *zero-shot analysis*, we evaluate pretrained LLM agents directly under each of the three harnesses $\{$`h-low`, `h-mid`, `h-high`$\}$ with tool schema fixed to `v1.0` (Obs. 1). Specifically, we evaluate `GPT-5 Mini` as a closed-source model, and instruction-tuned versions of `Qwen2.5-3B` and `Qwen2.5-7B` as open-source models. For `GPT-5 Mini`, we set its reasoning effort to "high" to use the most capable inference setting.

For *post-training analysis*, we post-train the same open-source models with GRPO and GiGPO as RL algorithms. For in-distribution and tool environment shift scenarios, agents are post-trained on the full training split $\mathcal{D}^{\text{tr}}_{\text{all}}$ under each of the three harnesses, with tool schema fixed to `v1.0`. Post-trained agents are then evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness, either with schema `v1.0` (in-distribution, Obs. 2) or with `v1.1` and `v2.0` corresponding to tool environment shifts of varying degrees (Obs. 4). For the task shift scenario, agents are instead post-trained on a single difficulty-specific split ($\mathcal{D}^{\text{tr}}_{\text{easy}}$, $\mathcal{D}^{\text{tr}}_{\text{med}}$, or $\mathcal{D}^{\text{tr}}_{\text{hard}}$) under each of the three harnesses, with tool schema fixed to `v1.0`, and evaluated on the test splits of the remaining groups under the same harness and schema (Obs. 5). All open-source results are averaged over three random seeds, with standard deviations reported in the corresponding figures and tables. Additional training and evaluation details are provided in Appendix C and Appendix D.

Remark. Prior post-training studies on `ALFWorld` typically operate under a single (harness, task type) configuration, training only on the full training split $\mathcal{D}^{\text{tr}}_{\text{all}}$. We instead treat harness and task type as controllable variables, running each RL algorithm across all combinations of three harness levels and four task type settings (the full training split and three difficulty-specific splits), for both open-source models. This amounts to a total compute budget of approximately $1,800$ H200 GPU-hours.

4.2 Zero-shot and In-distribution Performance (RQ1)

Observation 1 (Zero-shot)**.** *Harness informativeness monotonically improves zero-shot performance of LLM agents, and the magnitude of the gain scales with model capacity.*

Across all three models, performance improves monotonically with harness informativeness (Fig. 2). In particular, `GPT-5 Mini` shows the largest gain, confirming that the magnitude of the gain scales with model capacity. Furthermore, while open-source models fail at Pick 2 (`t-hard`), reporting 0.0 across most harness configurations, `GPT-5 Mini` achieves 61.0 under `h-high` (Table 10). Even under

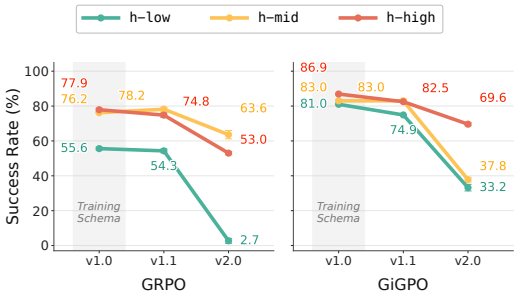

Figure 4: Post-training success rates of `Qwen2.5-7B-Instruct` under tool environment shift: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{all}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness but with schema `v1.1` (mild shift) or `v2.0` (stronger shift). Across both algorithms, more informative harnesses yield greater robustness to such shifts.

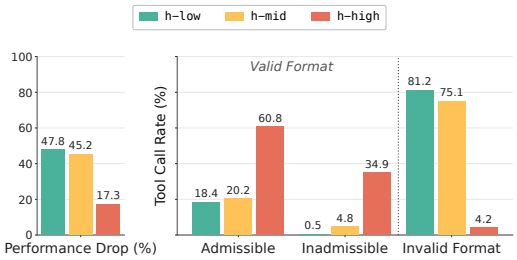

Figure 5: In-depth analysis of post training under the tool shift from `v1.0` (training) to `v2.0` (evaluation). (Left) Performance drop under `v2.0` relative to that of in-distribution. (Right) Breakdown of generated tool calls into three categories: *admissible* (correct format and executable), *inadmissible* (correct format but rejected by the environment, *i.e.,* "`Nothing happens`"), and *invalid* (malformed, *e.g.,* non-existent tool name or argument errors).

`h-low`, `GPT-5 Mini` achieves 17.1 at `Pick 2`, further illustrating that model capacity is essential to drive harness-induced gains.

Observation 2 (In-distribution)**.** *The monotonic harness gain observed at zero-shot (Fig. 2) largely carries over after post-training under both algorithms.*

The monotonic harness gain observed at zero-shot extends to post-training under both algorithms (Fig. 3). For example, `Qwen2.5-3B-Instruct` post-trained with GRPO under `h-high` outperforms `Qwen2.5-7B-Instruct` post-trained with GRPO under `h-low` by 14.1 points, indicating that the choice of harness can outweigh the effect of model capacity even after post-training. We also observe that GiGPO consistently outperforms GRPO across all configurations, consistent with its finer credit assignment in long-horizon tasks.

4.3 Post-hoc Harness Application (RQ2)

Observation 3 (Post-hoc Gap)**.** *Applying a harness only after training recovers little of the benefit of training with it in place.*

Once the harness is treated as a controllable variable, a natural question arises: should a harness be applied at training time, or in a post-hoc manner after post-training under a low-effort harness? We find that harness-aware post-training is preferable across all model and harness configurations (Fig. 6). The gap is particularly large for `Qwen2.5-7B-Instruct` post-trained with GRPO, where training-time harness application outperforms post-hoc application by 20.7 points under `h-mid` and 22.5 points under `h-high`. This suggests that the harness is better specified before post-training, allowing the agent to adapt to the harness it will ultimately use.

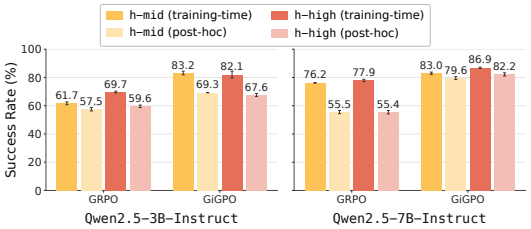

Figure 6: Training-time vs. post-hoc harness application: in-distribution success rates on $\mathcal{D}^{\text{te}}_{\text{all}}$. For each harness {`h-mid`, `h-high`}, we compare an agent post-trained with that harness in place (*training-time*) against an agent post-trained under `h-low` with that harness applied only at evaluation (*post-hoc*). Training-time application consistently outperforms post-hoc application.

4.4 OOD Robustness (RQ3)

Observation 4 (Tool Environment Shift)**.** *Harness-aware post-training is robust to tool environment shift, while post-training under* `h-low` *with low design effort suffers a drastic performance drop under stronger shift.*

Harness	Training-time Form (v1.0)	Test-time Form (v2.0)	Agent Output
h-low	Go(receptacle="countertop 1")	ReceptacleControl(action="navigate_to", target="countertop 1")	GoTo("countertop 1")
	Go(receptacle="shelf 1")	ReceptacleControl(action="navigate_to", target="shelf 1")	GoToLocation("shelf 1")
	Go(receptacle="diningtable 1")	ReceptacleControl(action="navigate_to", target="diningtable 1")	GoToDiningTable("diningtable 1")
h-mid	Go(receptacle="countertop 1")	ReceptacleControl(action="navigate_to", target="countertop 1")	navigate_to("countertop 1")
	Go(receptacle="dresser 1")	ReceptacleControl(action="navigate_to", target="dresser 1")	go_to("dresser 1")
	Go(receptacle="dresser 1")	ReceptacleControl(action="navigate_to", target="dresser 1")	goto_dresser()

Table 4: Example failure modes of `Qwen2.5-7B-Instruct` post-trained with GiGPO under `h-low` and `h-mid`, with tool schema `v1.0` at training time and `v2.0` at evaluation. The corresponding valid forms under `v1.0` and `v2.0` are shown for reference. The agents fail to adapt to the test-time schema, generating plausible-looking but non-existent tool names.

Harness-aware post-training not only improves in-distribution performance (Obs. 2), but also remains robust under tool environment shift (Fig. 4). While the agent post-trained under a harness with low design effort (`h-low`) adapts to the mild shift (`v1.1`), it suffers a drastic performance drop under `v2.0`. In particular, `Qwen2.5-7B-Instruct` post-trained with GRPO under `h-low` achieves 2.7 under `v2.0`, which is 10.8 points below the base model without post-training (13.5, Table 13).

While `Qwen2.5-7B-Instruct` post-trained with GiGPO under `h-low` performs well in-distribution (Fig. 3), it exhibits a substantial drop under `v2.0` (Fig. 4), indicating that the agent fails to adapt to the test-time schema. As such, we conduct an in-depth analysis on the tool calling patterns of `Qwen2.5-7B-Instruct` post-trained with GiGPO, where tool schema is fixed to `v1.0` at training time and `v2.0` at evaluation.

The model post-trained under `h-high` generates 95.7% valid tool calls under `v2.0`, indicating that it well adapts to the tool environment shift at the surface level (Fig. 5). However, 34.9% of these calls are not admissible at the current state, resulting in a performance drop of 17.3 points (from 86.9 to 69.6) relative to its in-distribution result. By contrast, the models post-trained under `h-mid` and `h-low` fail to adapt at the schema level, returning e_t = "`Invalid tool format`" in 75.1% and 81.2% of attempts, respectively. Performance degrades by 45.2 and 47.8 points under `v2.0`, significantly larger than that of `h-high`. This underscores the importance of training with an informative harness for OOD robustness, particularly under stronger tool environment shift. Representative failure modes are provided in Table 4.

Observation 5 (Task Shift)**.** *Prior knowledge encoded in harness boosts inter-task transfer of agent performance.*

Performance improves monotonically with harness informativeness in the task shift scenario as well, where this trend holds across both algorithms (Fig. 7). GRPO improves from 33.1 under `h-low` to 50.6 under `h-high` ($+17.5$), and GiGPO from 28.8 to 73.4 ($+44.6$, Table 16). This improvement arises not only from the in-distribution task category (`Pick 2`), but also from the OOD categories, where agents post-trained under `h-low` fail to transfer (Fig. 7).

Specifically, `Qwen2.5-7B-Instruct` post-trained with GRPO under `h-low` achieves 0.9 on `Heat`, which rises to 29.1 under `h-high` (Table 16). A similar pattern holds with GiGPO under `h-low`, which achieves 0.0 on `Cool` and rises to 51.4 under `h-high`. This indicates that the prior knowledge encoded in an informative harness helps the agent generalize across task groups. As in the tool environment shift scenario, harness-aware post-training improves OOD robustness.

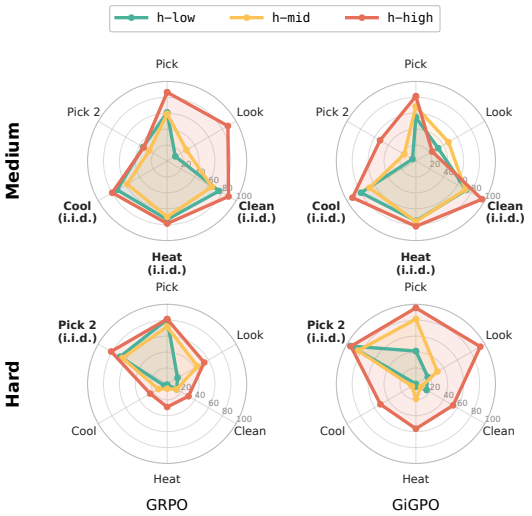

Figure 7: Post-training success rates of `Qwen2.5-7B-Instruct` under task shift: agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{med}}$ (top row) and $\mathcal{D}^{\text{tr}}_{\text{hard}}$ (bottom row) under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness and schema. In-distribution task categories are highlighted in **bold**: **`Clean`**, **`Heat`**, **`Cool`** for $\mathcal{D}^{\text{tr}}_{\text{med}}$, and **`Pick 2`** for $\mathcal{D}^{\text{tr}}_{\text{hard}}$. Post-training under informative harnesses tends to improve performance on OOD task categories.

5 Conclusion

In this paper, we studied harness design in the context of post-training tool-integrated LLM agents, a factor that prior work has either treated as a fixed engineering detail or examined only in the training-free regime. Building on `ALFWorld`, we constructed a benchmark that treats the harness as a controllable design dimension and supports four evaluation regimes: zero-shot, and post-training under the in-distribution and two OOD scenarios (tool environment shift and task shift). Across these regimes, we instantiated three harnesses with varying levels of informativeness and analyzed how harness design influences post-training of LLM agents in both in-distribution and OOD settings.

Our experiments establish that harness design and post-training cannot be treated as separable design choices. Performance improves monotonically with harness informativeness at zero-shot, and this trend extends to post-training under the in-distribution scenario. This improvement, however, requires the harness to be in place during post-training: applying it only afterward recovers little of the benefit of training with one in place. Under OOD settings, harness-aware post-training remains robust to tool environment shift and helps the agent generalize across task groups, whereas post-training under a harness with low design effort suffers a drastic performance drop under stronger tool environment shift and fails to transfer to OOD task categories. These results demonstrate that harness-aware post-training is a prerequisite for, rather than a supplement to, robust performance of tool-integrated LLM agents.

Limitations

Our analysis is conducted on a single environment, `ALFWorld`. Although our benchmark built on `ALFWorld` enables a controlled study of harness design under in-distribution and OOD settings, validating our findings on benchmarks with different task distributions and tool environments is an important direction.

We examine harness-aware post-training with two open-source models, `Qwen2.5-3B-Instruct` and `Qwen2.5-7B-Instruct`, and two RL algorithms, GRPO and GiGPO, spanning 24 training configurations per model and amounting to approximately $1{,}800$ H200 GPU-hours in total. Extending this to a broader range of models and RL algorithms, beyond what our compute budget allowed, is left to future work.

We instantiate three harnesses (`h-low`, `h-mid`, `h-high`) as representative levels of informativeness. In practice, informative harnesses such as `h-mid` and `h-high` must be obtained either through expert design or costly exploration of the environment (§3.2). While GiGPO under `h-low` performs comparably to harness-aware counterparts under the in-distribution scenario, it still breaks down under stronger tool environment shift, indicating that finer credit assignment alone cannot replace an informative harness. A natural direction is therefore to develop harness-aware post-training methods that jointly optimize the harness and the LLM agent. Our benchmark, which supports arbitrary harness designs, can serve as a testbed for such algorithms.

References

[1] 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.

[2] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[3] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

[4] John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pages 50528–50652, 2024.

[5] Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28489–28503, 2025.

[6] 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 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[7] 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.

[8] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[9] Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. APIGen: Automated PIpeline for generating verifiable and diverse function-calling datasets. In *Advances in Neural Information Processing Systems*, 2024.

[10] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

[11] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

[12] Anthropic. Effective harnesses for long-running agents. `https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents`, 2026. Anthropic Engineering Blog.

[13] OpenAI. Harness engineering: Leveraging Codex in an agent-first world. `https://openai.com/index/harness-engineering/`, 2026. OpenAI Blog.

[14] Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[15] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. In *International Conference on Learning Representations*, volume 2025, pages 64485–64515, 2025.

[17] Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

[18] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

[19] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. In *International Conference on Learning Representations*, volume 2025, pages 3841–3865, 2025.

[20] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[21] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

[22] Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. Agentic RL scaling law: Spontaneous code execution for mathematical problem solving. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[23] Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[24] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems*, pages 105523–105553, 2025.

[25] 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*, 2026.

[26] Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. s3: You don't need that much data to train a search agent via RL. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[27] Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. Agent-R1: Training powerful LLM agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

[28] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[29] Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, FangYuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. Revisiting out-of-distribution robustness in NLP: Benchmarks, analysis, and LLMs evaluations. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[30] Linyi Yang, Yaoxian Song, Xuan Ren, Chenyang Lyu, Yidong Wang, Jingming Zhuo, Lingqiao Liu, Jindong Wang, Jennifer Foster, and Yue Zhang. Out-of-distribution generalization in natural language processing: Past, present, and future. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 4533–4559, 2023.

[31] Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. Learning evolving tools for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[32] Guangrui Li, Yaochen Xie, Yi Liu, Ziwei Dong, Xingyuan Pan, Tianqi Zheng, Jason Choi, Michael J. Morais, Binit Jha, Shaunak Mishra, Bingrou Zhou, Chen Luo, Monica Xiao Cheng, and Dawn Song. The world won't stay still: Programmable evolution for agent benchmarks. *arXiv preprint arXiv:2603.05910*, 2026.

[33] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[34] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic Reinforced Policy Optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[35] Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chang Liu, and Peilin Zhao. Proximity-based multi-turn optimization: Practical credit assignment for llm agent training. *arXiv preprint arXiv:2602.19225*, 2026.

[36] Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2026.

[37] Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*, 2025.

[38] 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 *Forty-second International Conference on Machine Learning*, 2025.

A Related Work

A.1 Training Tool-integrated Agentic Systems

A substantial body of prior work aims to improve the performance of tool-integrated agentic systems under static tool environments. Most approaches train an LLM agent to generate valid tool calling sequences using supervised fine-tuning (SFT) with expert demonstrations or reinforcement learning (RL).

Zhang et al. [36] propose two SFT strategies that incorporate additional trajectories generated by the agent itself. The first performs "implicit world modeling" to learn transition dynamics from agent–environment interactions before fine-tuning on expert trajectories. The second constructs "self-reflection data" by contrasting agent-generated trajectories with expert demonstrations, encouraging the model to internalize why expert solutions are preferable. Similarly, Qian et al. [24] design reward functions that align generated trajectories with expert demonstrations, supervising both intermediate tool calls and final outcomes.

In parallel, a complementary line of work develops reinforcement learning methods tailored to tool-integrated agents [21, 22, 23, 26, 37, 27]. A key distinction from conventional RL is the masking of feedback tokens from the tool environment, ensuring that parameter updates depend solely on agent-generated tokens. Additional refinements include filtering trajectories containing invalid or incomplete tool calls [23] and introducing multi-turn optimization strategies, such as turn-level ratios in group relative policy optimization (GRPO). Beyond individual algorithms, prior studies examine the interaction between SFT and RL for improving generalization [38]: SFT followed by RL can enhance training stability, while RL-only approaches have been reported to achieve stronger generalization in certain tool-integrated reasoning tasks [23].

Despite their methodological differences, these approaches share two assumptions: that the tool environment is static throughout training and evaluation, and that the harness is fixed and treated as part of the environment rather than as a design variable. As a consequence, neither the harness's role in shaping what the post-trained agent learns, nor the agent's robustness to changes in the harness or the tool environment, is systematically examined.

A.2 Harness Design for Tool-integrated Agentic Systems

The harness, as introduced in Section 1, is widely recognized as a substantial determinant of agent performance, with recent work reporting accuracy gaps of up to roughly six-fold across harnesses [18]. For tool-integrated agentic systems, the harness specifies, among other things, which tools are exposed to the agent and how they are described, what auxiliary information accompanies each per-step observation, and how the resulting context is assembled into a prompt.

In practice, harnesses have been hand-designed for closed-source models. Benchmarks for interactive LLM agents, including those built on `ALFWorld`, typically report results obtained under carefully crafted harnesses, often containing detailed descriptions of available actions, structured per-step observations, and various forms of auxiliary information [36]. A recent line of work seeks to remove the human from this loop: Lee et al. [18] introduce Meta-Harness, which casts harness design as a search problem and uses a high-capacity LLM agent to read candidate harnesses and propose new ones automatically. Even with such automated search and strong base models, however, the gains from harness design alone are not always satisfactory. Furthermore, the underlying LLM is held fixed throughout, with the implications for harness design in the post-training regime remaining unaddressed.

In this work, we study how harness design and post-training jointly shape the resulting agent: we treat the harness as an explicit, controlled design dimension and post-train independently under harnesses providing three different levels of auxiliary information, characterizing their interplay both in-distribution and under task and tool environment shift.

A.3 Tool-integrated Agentic Systems under Dynamic Environments

Chen et al. [31] introduce ToolQA-D, a benchmark in which API names, signatures, and behaviors drift over time with the agent unaware of these changes, and propose ToolEVO, a framework that uses Monte Carlo Tree Search to enable an agent to detect and adapt to such variability at inference

time. More recently, Li et al. [32] propose ProEvolve, a graph-based framework in which the data, tools, and schema of an agent environment are represented as a typed relational graph and evolved through programmable graph transformations, allowing controlled evaluation of agents' robustness as environments change. While these efforts share our motivation, they neither treat the harness as a design dimension nor consider the post-training regime, leaving the interplay between the two under such shifts uncharacterized.

B Benchmark Details

B.1 Harness Configurations

Tool	h-low / h-mid	h-high
Go	Move to a receptacle.	Move to a receptacle in the room. This sets your current location to that receptacle; you must be at a receptacle before you can Take from it, Open/Close it, Examine it, or Move an object to it.
Open	Open a receptacle.	Open a closed container at your current location (drawer, cabinet, fridge, microwave). You must be at the container (use Go) and it must currently be closed. After opening, its contents become visible and you can Take items from it.
Close	Close a receptacle.	Close an open container at your current location. Normally not required for task completion; useful only when a goal explicitly requires closed state.
Take	Take an object from a receptacle.	Pick up an object from a receptacle at your current location. Preconditions: you must be at the receptacle (use Go), the object must be visible, and if the receptacle is a closed container (drawer/cabinet/fridge/microwave) you must Open it first. You can only carry one object at a time.
Move	Place an object in or on a receptacle.	Place the object you are carrying in/on a receptacle at your current location. Preconditions: you must be carrying the object and be at the target receptacle (for closed containers, Open it first). This action COMPLETES the task when the carried object and the target receptacle both match the goal.
Clean	Clean an object using a receptacle.	Clean an object you are carrying using a sinkbasin at your current location. Preconditions: you must be carrying the object and be at the sinkbasin.
Heat	Heat an object using a receptacle.	Heat an object you are carrying using a microwave at your current location. Preconditions: you must be carrying the object and be at the microwave.
Cool	Cool an object using a receptacle.	Cool an object you are carrying using a fridge at your current location. Preconditions: you must be carrying the object and be at the fridge.
Slice	Slice an object using a sharp object.	Slice an object at your current location using a knife (or butterknife) you are carrying. Preconditions: you must be carrying the sharp object and be at the target object's location.
Look	Look around your current location.	Look around your current surroundings. If not at any receptacle, reports all visible receptacles in the room. If at a receptacle, reports what is on/in it.
Inventory	Check your current inventory.	Report what object you are currently carrying. You can carry at most one object at a time.
Examine	Examine a receptacle or an object.	Examine a receptacle or object at your current location. For containers this reveals their contents; for objects this reports their state. Useful for re-checking an already-opened receptacle without Go-ing again.
Use	Use an object.	Activate an object at your current location, most commonly turning on a lamp (desklamp / floorlamp). In 'look at object in light' tasks, turning on the lamp while holding the target object completes the task.

Table 5: One-line versus rich tool descriptions in p, for all 13 tools (under tool schema `v1.0`). The one-line form is shared by `h-low` and `h-mid`; the rich form is used only by `h-high`. Rows are grouped according to the `v2.0` consolidation (Table 9).

Harness Ver.	Per-step history $\mathcal{T}_t$
h-low	Your task is to: <GOAL>. Current location: <LOC> You are now at step <N> and your current observation is: <STEP_OBS>
h-mid	Your task is to: <GOAL>. Current location: <LOC> You are now at step <N> and your current observation is: <STEP_OBS> Valid tools: <TOOL_A>, <TOOL_B>, <TOOL_C>
h-high	Your task is to: <GOAL>. Current location: <LOC> Carrying: <ITEM_OR_"nothing"> You are now at step <N> and your current observation is: <STEP_OBS> Valid tools: <TOOL_A>, <TOOL_B>, <TOOL_C>

Table 6: Per-step history $\mathcal{T}_t$ under each harness version. The three harnesses share a common core, with each richer version appending one additional auxiliary line (`Valid tools:` under `h-mid`, and additionally `Carrying:` under `h-high`).

B.2 Tool Schema Versions

Meaning	v1.0	v1.1
Move to a receptacle	Go	NavigateTo
Open a container	Open	OpenContainer
Close a container	Close	CloseContainer
Pick up an object	Take	Pickup
Place an object in/on a receptacle	Move	Place
Wash an object (in sinkbasin)	Clean	Wash
Heat an object (in microwave)	Heat	Warm
Cool an object (in fridge)	Cool	Chill
Slice an object (with a knife)	Slice	Cut
Survey the surroundings	Look	SurveyRoom
Check items currently carried	Inventory	CheckCarried
Examine a receptacle or object	Examine	Inspect
Activate a device (e.g., lamp)	Use	Activate

Table 7: Tool name mapping from v1.0 to v1.1, covering all 13 tools defined for ALFWorld. Rows are grouped according to the v2.0 consolidation (Table 9).

Meaning	v1.0	v1.1
Destination of a navigation/placement action	receptacle	destination
Object being acted upon	object	item
Source receptacle (e.g., for Take)	receptacle	source
Washing receptacle (sinkbasin)	receptacle	washer
Heating receptacle (microwave)	receptacle	heater
Cooling receptacle (fridge)	receptacle	cooler
Slicing instrument	knife	blade
Target of an examination	target	subject

Table 8: Parameter key mapping from v1.0 to v1.1, covering all parameter roles shared across tools.

v2.0 Tool	Merged v1.1 Tools	action Values	Unified Parameters
ReceptacleControl	NavigateTo, OpenContainer, CloseContainer	navigate_to, open_container, close_container	action, target
ObjectTransport	Pickup, Place	pickup, place	action, item, location
ObjectTransform	Wash, Warm, Chill, Cut	wash, warm, chill, cut	action, item, instrument
Observe	SurveyRoom, CheckCarried, Inspect	survey_room, check_carried, inspect	action, subject
Activate	Activate	– (single operation)	device

Table 9: Tool consolidation from v1.1 (13 tools) to v2.0 (5 tools). Each v2.0 group merges one or more v1.1 tools sharing structural or functional similarity, distinguished within the group by a discrete action parameter. Parameter keys are unified across merged operations.

C Experimental Details

Multi-step Tool Calling Process as Markov Decision Process. We illustrate in detail how multi-step tool calling process is modeled as a Markov decision process (MDP) at a sequence level. Specifically, the MDP is defined by a tuple $\mathcal{M} \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mathcal{P}_0)$ as the following.

- $\mathcal{S} \coloneqq \mathcal{V}^*$: State space
- $\mathcal{A} \coloneqq \mathcal{V}^*$: Action space
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$: State transition probability such that $\sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s') = 1$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. In addition, we let $\mathcal{P}(s'|s, a) \coloneqq \mathcal{P}(s, a, s')$.
- $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$: Step-level reward function evaluated at each step, where we let
$$r_t \coloneqq r(s_t, a_t, s_{t+1}).$$
- $\mathcal{P}_0 : \mathcal{S} \to [0, 1]$: Initial state probability distribution

Then, the multi-step tool calling process can be formalized as follows. Before the process starts, the initial state $s_1 \coloneqq (p, q)$ is sampled from $\mathcal{P}_0$, where $p \in \mathcal{V}^*$ is a system prompt and $q \in \mathcal{V}^*$ is a user query. In each tool calling step t, the LLM agent $\pi_\theta : \mathcal{S} \to \mathcal{A}$ first observes the state s_t. Specifically, $s_t \coloneqq (p, q, \mathcal{T}_{t-1}) \in \mathcal{S}$, where $\mathcal{T}_{t-1}$ is a history of tool calling steps up to the $(t-1)$-th step (possibly in a summarized version), and $\mathcal{T}_0$ is an empty sequence. The agent then generates an action sequence $a_t \in \mathcal{A}$. The environment then samples the next state s_{t+1} from the probability distribution $\mathcal{P}(\cdot|s_t, a_t)$, where

$$\mathcal{P}(s_{t+1}|s_t, a_t) \coloneqq \begin{cases} 1 & s_{t+1} = (p, q, \mathcal{T}_t), \\ 0 & \text{o.w.} \end{cases}$$

Here, $e_t = \texttt{TE}((s_t, a_t))$ and $\mathcal{T}_t = \texttt{TE}((s_t, a_t, e_t))$. Note that the transition dynamics is deterministic and $\texttt{TE}$-dependent. In addition, the agent receives the step level reward r_t. The tool calling process proceeds for $T \in \mathbb{N}$ steps, where the process terminates either when it succeeds in a given task or when the agent reaches the maximum number of tool calling steps. Finally, by letting $\tau \coloneqq \{(s_1, a_1, s_2), \ldots, (s_T, a_T, s_{T+1})\} \subset (\mathcal{S} \times \mathcal{A} \times \mathcal{S})^*$ be an episode-level trajectory and $\mathcal{R} : (\mathcal{S} \times \mathcal{A} \times \mathcal{S})^* \to \mathbb{R}$ be an episode-level reward, the objective can be formulated as

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{P}_0, \mathcal{P}, \pi_\theta} \left[\mathcal{R}(\tau) \right],$$

where $\mathcal{R}(\tau) \coloneqq \sum_{t=1}^{T} \gamma^{t-1} r_t$ is a typical choice ($\gamma \in [0, 1]$).

Group-relative policy optimization (GRPO) [33]. It is a widely adopted algorithm for post-training LLMs due to its computational efficiency in calculating the gradient $\nabla_\theta \mathcal{J}(\theta)$. Given a system prompt and a question pair (p, q), a behavior policy $\pi_{\theta_{\text{old}}}$ samples a group of G episodes $\{\tau^i\}_{i=1}^G$, where $\tau^i \coloneqq \{(s_1^i, a_1^i, s_2^i), \ldots, (s_{T^i}^i, a_{T^i}^i, s_{T^i+1}^i)\}$ is the i-th episode. Here, we let $a_t^i \coloneqq (a_{t1}^i, \ldots, a_{tn_t^i}^i)$, where $n_t^i \coloneqq |a_t^i|$. GRPO algorithm then optimizes the following token-level objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{(p,q)\sim P_0, \\ \{a_t^i\}\sim\pi_{\theta_{\text{old}}}}} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{\sum_{t=1}^{T^i} n_t^i} \sum_{t=1}^{T^i} \sum_{k=1}^{n_t^i} \Bigg\{ \min \left(\rho_{tk}^i \hat{A}^i, \text{clip} \left(\rho_{tk}^i, 1-\varepsilon, 1+\varepsilon \right) \hat{A}^i \right) - \beta D_{\text{KL}} \left(\pi_\theta(\cdot \mid s_{<tk}^i) \,\|\, \pi_{\text{ref}}(\cdot \mid s_{<tk}^i) \right) \Bigg\} \Bigg].$$

where $s_{<tk}^i \coloneqq (p, q, \mathcal{T}_{t-1}^i, a_{t1}^i, \ldots, a_{t(k-1)}^i)$ denotes the context preceding token a_{tk}^i, and the token-level ratio ρ_{tk}^i and the episode-level advantage $\hat{A}^i$ are defined as follows:

$$\rho_{tk}^i = \frac{\pi_\theta(a_{tk}^i \mid s_{<tk}^i)}{\pi_{\theta_{\text{old}}}(a_{tk}^i \mid s_{<tk}^i)}, \quad \hat{A}^i = \frac{R^i - \mu}{\sigma}.$$

Here, $\{R^i\}_{i=1}^G$ are episode-level rewards evaluated on $\{\tau^i\}_{i=1}^G$, and $\mu = \texttt{mean}(\{R^i\}_{i=1}^G)$, $\sigma = \texttt{std}(\{R^i\}_{i=1}^G)$ are their sample mean and standard deviation, respectively. Here, π_{ref} is a fixed reference policy, typically initialized from the initial policy and kept frozen throughout training. In practice, the per-token KL term is computed using the low-variance estimator of $D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$, as adopted in GRPO [33].

Group-in-group policy optimization (GiGPO) [25]. Unlike GRPO, GiGPO differentiates the advantage assigned to each step within the same episode by additionally considering a step-level advantage $\hat{A}^i_t$ in addition to an episode-level advantage $\hat{A}^i$. To define a step-level advantage, GiGPO algorithm requires a state identifier $h : \mathcal{V}^* \times \mathcal{V}^* \to \{0, 1\}$ that verifies whether two different per-step initial states, $s^i_{t_1}$ and $s^j_{t_2}$, *i.e.,* the states from which steps t_1 and t_2 begin, are semantically equivalent (possibly from different episodes, where $i \neq j$). Specifically, given a set of step-level transitions across all G episodes $\mathcal{G} := \left\{(s^i_t, a^i_t, s^i_{t+1})\right\}_{i\in[G],t\in[T^i]}$, GiGPO partitions $\mathcal{G} = \cup_{\tilde{s}\in\mathcal{AN}} \mathcal{G}(\tilde{s})$ into disjoint groups, where steps within each group $\mathcal{G}(\tilde{s})$ shares the same per-step initial state $\tilde{s} \in \mathcal{V}^*$ w.r.t. the state identifier h as follows:

$$\mathcal{G}(\tilde{s}) := \left\{(s^i_t, a^i_t, s^i_{t+1}) \in \mathcal{G} : h(s^i_t, \tilde{s}) = 1\right\}.$$

Here, $\mathcal{AN}$ is an anchor set that partitions $\mathcal{G}$ based on the semantic equivalence of the per-step initial state that jointly satisfies

$$\mathcal{G} = \cup_{\tilde{s}\in\mathcal{AN}}\mathcal{G}(\tilde{s}),$$
$$\mathcal{G}(\tilde{s}) \cap \mathcal{G}(\hat{s}) = \emptyset \; \forall \tilde{s}, \hat{s} \in \mathcal{AN} \text{ s.t. } h(\tilde{s}, \hat{s}) = 0.$$

Then, every step level transition $(s^i_t, a^i_t, s^i_{t+1})$ is assigned an index g^i_t that specifies which group that it belongs to, defined as $g^i_t := (\tilde{s} \in \mathcal{AN} \text{ s.t. } h(s^i_t, \tilde{s}) = 1)$. GiGPO algorithm then optimizes the following token-level objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{(p,q)\sim P_0,\\ \{a^i_t\}\sim\pi_{\theta_{\text{old}}}}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{\sum_{t=1}^{T^i} n^i_t}\sum_{t=1}^{T^i}\sum_{k=1}^{n^i_t} \Bigg\{\min\Big(\rho^i_{tk}(\hat{A}^i + \omega\hat{A}^i_t), \text{clip}\left(\rho^i_{tk}, 1-\varepsilon, 1+\varepsilon\right)(\hat{A}^i + \omega\hat{A}^i_t)\Big) - \beta D_{\text{KL}}\left(\pi_\theta(\cdot \mid s^i_{<tk}) \,\|\, \pi_{\text{ref}}(\cdot \mid s^i_{<tk})\right)\Bigg\}\Bigg].$$

where the token level ratio ρ^i_{tk} and the episode-level advantage $\hat{A}^i$ are defined in a same manner as GRPO, and the step-level advantage $\hat{A}^i_t$ is defined as :

$$\hat{A}^i_t = \frac{R^i_t - \mu_{g^i_t}}{\sigma_{g^i_t}}.$$

Here, $R^i_t := \sum_{\tilde{t}=t}^{T^i} \gamma^{\tilde{t}-t} r^i_{\tilde{t}}$ is a step-level return of the t-th step within the i-th episode. For each anchor $\tilde{s} \in \mathcal{AN}$, $\mu_{\tilde{s}}$ and $\sigma_{\tilde{s}}$ denote the sample mean and standard deviation of the step-level returns $\{R^i_t\}$ within $\mathcal{G}(\tilde{s})$. The advantage $\hat{A}^i_t$ then uses the statistics of the group g^i_t to which the t-th step of the i-th episode belongs. Note that the discount factor $\gamma \in [0, 1]$ may be different from that used to define the episode-level reward R^i. $\omega \in \mathbb{R}^+$ is a hyperparameter that balances episode-level and step-level advantages.

Training. We fine-tune Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct with GRPO, using group-normalized advantages over G=8 trajectories per prompt. We train for 500 steps with a constant learning rate of 1×10^{-6}, a batch of 16 task prompts per step, a PPO minibatch size of 256, and a low-variance KL penalty (coefficient 0.01) applied to the loss. A randomly sampled 10% of the training data is held out for validation and used to select the final checkpoint. Rollouts use vLLM with a sampling temperature of 1.0 and a maximum of 50 turns per episode over the 13 ALFWorld tools, with a maximum prompt length of 3072 tokens and a maximum response length of 1024 tokens. The reward is sparse: a successful trajectory receives 10, while each invalid action incurs a penalty of -0.1. We use $4\times$ H200 GPUs for training.

Evaluation. We evaluate on the ALFWorld test split of 274 games (140 seen, 134 unseen), with the evaluation harness matched to the training harness and a sampling temperature of 0.4. Each configuration is run with three seeds, and we report the mean success rate; the corresponding per-configuration standard deviations are reported alongside the means in all result tables. For GPT-5-mini, we use a reasoning effort of `high`; since reasoning models require substantially more

tokens, we set the maximum response length to 4096 tokens. Due to budget constraints, GPT-5-mini is run with a single seed. For the tool environment shift scenario, we refer to the paraphrased schema `v1.1` as the *mild* shift and the consolidated schema `v2.0` as the *strong* shift, reflecting the increasing degree of surface-level deviation from the `v1.0` training schema.

D Additional Experimental Results

Model Type	Model	Test-time Harness	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2	
Closed-source	GPT-5 Mini (high)	h-low	71.2	16.1	19.0	12.8	15.2	17.1	28.1
		h-mid	67.8	12.9	20.7	20.5	34.8	12.2	31.0 (+2.9)
		h-high	93.2	77.4	36.2	66.7	78.3	61.0	68.3 (+40.2)
Open-source	Qwen2.5-3B-Instruct	h-low	$6.2_{\pm 2.6}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$1.3_{\pm 0.6}$
		h-mid	$11.9_{\pm 4.5}$	$2.2_{\pm 1.9}$	$0.0_{\pm 0.0}$	$3.4_{\pm 3.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$3.3_{\pm 1.1}$ (+2.0)
		h-high	$35.0_{\pm 2.6}$	$15.1_{\pm 1.9}$	$11.5_{\pm 1.0}$	$12.8_{\pm 0.0}$	$2.2_{\pm 2.2}$	$0.0_{\pm 0.0}$	$13.9_{\pm 0.7}$ (+12.6)
	Qwen2.5-7B-Instruct	h-low	$19.8_{\pm 3.5}$	$18.3_{\pm 4.9}$	$0.6_{\pm 1.0}$	$3.4_{\pm 3.0}$	$2.9_{\pm 3.3}$	$0.0_{\pm 0.0}$	$7.4_{\pm 0.9}$
		h-mid	$44.6_{\pm 2.0}$	$20.4_{\pm 3.7}$	$9.8_{\pm 3.6}$	$8.5_{\pm 3.9}$	$5.1_{\pm 1.3}$	$0.0_{\pm 0.0}$	$16.1_{\pm 1.7}$ (+8.7)
		h-high	$54.8_{\pm 3.9}$	$18.3_{\pm 6.7}$	$26.4_{\pm 2.6}$	$35.9_{\pm 0.0}$	$24.6_{\pm 3.3}$	$1.6_{\pm 1.4}$	$29.0_{\pm 1.8}$ (+21.6)

Table 10: Zero-shot success rates without post-training:Pretrained agents are evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under each harness with tool schema fixed to `v1.0`. GPT-5 Mini is run only once due to the limited budget.

Model	Algorithm	Harness	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2	
Qwen2.5-3B-Instruct	GRPO	h-low	$80.2_{\pm 2.6}$	$38.7_{\pm 3.2}$	$66.1_{\pm 3.6}$	$64.1_{\pm 0.0}$	$66.7_{\pm 1.3}$	$0.0_{\pm 0.0}$	$\underline{56.0}_{\pm 1.1}$
		h-mid	$80.2_{\pm 2.6}$	$60.2_{\pm 4.9}$	$67.2_{\pm 1.7}$	$70.9_{\pm 1.5}$	$68.8_{\pm 4.5}$	$11.4_{\pm 3.7}$	$\underline{61.7}_{\pm 1.0}$
		h-high	$81.4_{\pm 1.7}$	$77.4_{\pm 5.6}$	$75.9_{\pm 3.0}$	$71.8_{\pm 2.6}$	$83.3_{\pm 1.3}$	$21.1_{\pm 3.7}$	$\underline{69.7}_{\pm 0.6}$
	GiGPO	h-low	$91.5_{\pm 0.0}$	$89.2_{\pm 6.7}$	$72.4_{\pm 7.5}$	$67.5_{\pm 3.0}$	$64.5_{\pm 1.3}$	$0.0_{\pm 0.0}$	$\underline{65.6}_{\pm 1.3}$
		h-mid	$94.9_{\pm 1.7}$	$63.4_{\pm 4.9}$	$93.7_{\pm 2.6}$	$85.5_{\pm 3.0}$	$82.6_{\pm 2.2}$	$65.0_{\pm 6.1}$	$\underline{83.2}_{\pm 1.3}$
		h-high	$85.9_{\pm 5.4}$	$88.2_{\pm 1.9}$	$89.7_{\pm 1.7}$	$84.6_{\pm 5.1}$	$79.7_{\pm 5.5}$	$61.8_{\pm 1.4}$	$\underline{82.1}_{\pm 2.4}$
Qwen2.5-7B-Instruct	GRPO	h-low	$80.8_{\pm 2.0}$	$0.0_{\pm 0.0}$	$56.9_{\pm 1.7}$	$75.2_{\pm 3.0}$	$52.9_{\pm 1.3}$	$43.9_{\pm 2.4}$	$\underline{55.6}_{\pm 0.8}$
		h-mid	$79.7_{\pm 1.7}$	$86.0_{\pm 1.9}$	$78.2_{\pm 1.0}$	$71.8_{\pm 2.6}$	$66.7_{\pm 2.5}$	$75.6_{\pm 2.4}$	$\underline{76.2}_{\pm 0.2}$
		h-high	$82.5_{\pm 2.0}$	$86.0_{\pm 1.9}$	$81.6_{\pm 1.0}$	$75.2_{\pm 1.5}$	$79.7_{\pm 4.5}$	$61.0_{\pm 4.2}$	$\underline{77.9}_{\pm 0.8}$
	GiGPO	h-low	$86.4_{\pm 1.7}$	$75.3_{\pm 3.7}$	$75.9_{\pm 0.0}$	$76.9_{\pm 0.0}$	$89.9_{\pm 1.3}$	$78.9_{\pm 2.8}$	$\underline{81.0}_{\pm 0.0}$
		h-mid	$90.4_{\pm 2.0}$	$80.6_{\pm 6.5}$	$87.4_{\pm 1.0}$	$84.6_{\pm 0.0}$	$72.5_{\pm 1.3}$	$78.0_{\pm 2.4}$	$\underline{83.0}_{\pm 0.8}$
		h-high	$89.3_{\pm 2.0}$	$88.2_{\pm 1.9}$	$92.5_{\pm 1.0}$	$76.1_{\pm 1.5}$	$86.2_{\pm 2.5}$	$85.4_{\pm 2.4}$	$\underline{86.9}_{\pm 0.4}$

Table 11: In-distribution success rates after post-training: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{all}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness and schema.

	Algorithm	Post-hoc Harness	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2	
Qwen2.5-3B-Instruct	GRPO	h-mid	$79.7_{\pm 3.4}$	$49.5_{\pm 9.9}$	$67.2_{\pm 3.0}$	$63.2_{\pm 1.5}$	$68.8_{\pm 1.3}$	$0.0_{\pm 0.0}$	$57.5_{\pm 1.2}$ ($\mathbf{-4.2}$)
		h-high	$82.5_{\pm 1.0}$	$50.5_{\pm 3.7}$	$72.4_{\pm 1.7}$	$68.4_{\pm 1.5}$	$65.9_{\pm 1.3}$	$0.0_{\pm 0.0}$	$59.6_{\pm 0.9}$ ($\mathbf{-10.1}$)
	GiGPO	h-mid	$89.8_{\pm 1.7}$	$95.7_{\pm 1.9}$	$77.0_{\pm 2.0}$	$73.5_{\pm 1.5}$	$73.9_{\pm 2.2}$	$0.0_{\pm 0.0}$	$69.3_{\pm 0.0}$ ($\mathbf{-13.9}$)
		h-high	$89.3_{\pm 2.0}$	$93.5_{\pm 3.2}$	$73.6_{\pm 5.5}$	$70.9_{\pm 1.5}$	$72.5_{\pm 1.3}$	$0.0_{\pm 0.0}$	$67.6_{\pm 1.1}$ ($\mathbf{-14.5}$)
Qwen2.5-7B-Instruct	GRPO	h-mid	$77.4_{\pm 2.6}$	$0.0_{\pm 0.0}$	$56.9_{\pm 3.0}$	$76.9_{\pm 4.4}$	$56.5_{\pm 3.8}$	$42.3_{\pm 1.4}$	$55.5_{\pm 1.1}$ ($\mathbf{-20.7}$)
		h-high	$80.2_{\pm 3.5}$	$0.0_{\pm 0.0}$	$58.0_{\pm 3.6}$	$72.6_{\pm 1.5}$	$55.8_{\pm 2.5}$	$40.7_{\pm 1.4}$	$55.4_{\pm 1.2}$ ($\mathbf{-22.5}$)
	GiGPO	h-mid	$85.9_{\pm 1.0}$	$55.9_{\pm 7.4}$	$75.3_{\pm 1.0}$	$82.9_{\pm 1.5}$	$91.3_{\pm 0.0}$	$78.0_{\pm 2.4}$	$79.6_{\pm 1.0}$ ($\mathbf{-3.4}$)
		h-high	$86.4_{\pm 0.0}$	$80.6_{\pm 5.6}$	$75.9_{\pm 0.0}$	$79.5_{\pm 2.6}$	$91.3_{\pm 0.0}$	$78.9_{\pm 1.4}$	$82.2_{\pm 1.1}$ ($\mathbf{-4.7}$)

Table 12: Training-time vs. post-hoc harness application: in-distribution success rates on $\mathcal{D}^{\text{te}}_{\text{all}}$. For each harness {`h-mid`, `h-high`}, we compare an agent post-trained with that harness in place (*training-time*) against an agent post-trained under `h-low` with that harness applied only at evaluation (*post-hoc*). The parenthesized value is the change in overall accuracy relative to post-training under the same harness applied throughout. Training-time application consistently outperforms post-hoc application.

		Harness	v1.0							v1.1							v2.0						
			$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2		Pick	Look	Clean	Heat	Cool	Pick 2		Pick	Look	Clean	Heat	Cool	Pick 2	
Qwen2.5-3B-Instruct	Zero-shot	h-low	6.2	0.0	0.0	0.0	0.0	0.0	1.3	26.0	14.0	0.0	0.0	0.0	0.0	7.2	18.6	10.8	0.0	0.0	0.7	0.0	5.4
		h-mid	11.9	2.2	0.0	3.4	0.0	0.0	3.3	46.3	25.8	8.0	4.3	8.0	0.0	16.5	14.1	18.3	0.0	0.0	0.7	0.0	5.2
		h-high	35.0	15.1	11.5	12.8	2.2	0.0	13.9	40.7	20.4	10.9	14.5	8.0	0.0	16.8	35.6	28.0	5.7	7.7	8.7	0.0	14.6
	GRPO	h-low	80.2	38.7	66.1	64.1	66.7	0.0	56.0	52.0	28.0	10.9	14.5	5.8	0.0	19.7	17.5	7.5	0.6	0.9	2.2	0.0	5.2
		h-mid	80.2	60.2	67.2	70.9	68.8	11.4	61.7	72.3	47.3	56.3	53.8	62.3	14.6	53.2	48.0	26.9	12.1	20.5	26.1	7.3	24.3
		h-high	81.4	77.4	75.9	71.8	83.3	21.1	69.7	71.2	63.4	70.1	70.9	84.8	26.0	65.6	73.4	58.1	66.1	51.3	79.0	26.8	60.9
	GiGPO	h-low	91.5	89.2	72.4	67.5	64.5	0.0	65.6	87.0	93.5	69.5	68.4	57.2	0.0	63.4	20.9	90.3	27.0	27.4	38.4	0.0	30.8
		h-mid	94.9	63.4	93.7	85.5	82.6	65.0	83.2	92.7	45.2	87.9	74.4	83.3	54.5	76.4	89.8	37.6	47.7	35.0	63.8	47.2	56.4
		h-high	85.9	88.2	89.7	84.6	79.7	61.8	82.1	81.9	83.9	82.2	87.2	81.2	63.4	80.0	85.9	71.0	61.5	64.1	79.7	60.2	71.0
Qwen2.5-7B-Instruct	Zero-shot	h-low	19.8	18.3	0.6	3.4	2.9	0.0	7.4	15.8	25.8	5.7	0.9	1.4	0.0	7.9	35.6	24.7	5.7	3.4	6.5	1.6	13.5
		h-mid	44.6	20.4	9.8	8.5	5.1	0.0	16.1	58.8	22.6	35.1	20.5	22.5	6.5	30.3	58.8	32.3	16.1	18.8	17.4	8.9	26.6
		h-high	54.8	18.3	26.4	35.9	24.6	1.6	29.0	39.5	16.1	27.0	35.9	22.5	8.9	26.3	54.8	46.2	31.6	35.9	26.8	16.3	35.8
	GRPO	h-low	80.8	0.0	56.9	75.2	52.9	43.9	55.6	77.4	0.0	59.2	63.2	55.1	45.5	54.3	2.3	0.0	8.0	0.9	0.7	1.6	2.7
		h-mid	79.7	86.0	78.2	71.8	66.7	75.6	76.2	83.6	79.6	87.4	76.9	63.8	74.0	78.2	78.0	77.4	73.0	29.1	52.9	64.2	63.6
		h-high	82.5	86.0	81.6	75.2	79.7	61.0	77.9	78.0	83.9	73.6	76.9	76.1	61.8	74.8	66.7	76.3	28.7	48.7	51.4	56.1	53.0
	GiGPO	h-low	86.4	75.3	75.9	76.9	89.9	78.9	81.0	87.6	21.5	75.9	73.5	90.6	79.7	74.9	36.2	0.0	61.5	30.8	40.6	8.1	33.2
		h-mid	90.4	80.6	87.4	84.6	72.5	78.0	83.0	92.7	82.8	87.4	82.9	69.6	78.0	83.0	32.2	24.7	52.3	52.1	41.3	17.9	37.8
		h-high	89.3	88.2	92.5	76.1	86.2	85.4	86.9	85.9	79.6	92.0	75.2	75.4	81.3	82.5	73.4	84.9	94.8	22.2	66.7	65.0	69.6

Table 13: Post-training success rates under tool environment shift: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{all}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness but with schema `v1.1` (mild shift) or `v2.0` (stronger shift). Zero-shot results are included for reference, where the **Harness** column refers to the test-time harness.

Model	Algorithm	Harness	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2	
Qwen2.5-3B-Instruct	GRPO	h-low	$74.6_{\pm 1.7}$	$54.8_{\pm 3.2}$	$0.6_{\pm 1.0}$	$0.0_{\pm 0.0}$	$2.9_{\pm 1.3}$	$1.6_{\pm 2.8}$	$\underline{23.1}_{\pm 0.6}$
		h-mid	$84.7_{\pm 1.7}$	$76.3_{\pm 3.7}$	$6.9_{\pm 0.0}$	$12.8_{\pm 6.8}$	$9.4_{\pm 5.0}$	$13.0_{\pm 3.7}$	$\underline{33.7}_{\pm 1.4}$
		h-high	$88.7_{\pm 2.6}$	$71.0_{\pm 8.5}$	$74.7_{\pm 5.0}$	$62.4_{\pm 5.9}$	$66.7_{\pm 3.3}$	$0.8_{\pm 1.4}$	$\underline{63.1}_{\pm 1.7}$
	GiGPO	h-low	$89.8_{\pm 1.4}$	$88.2_{\pm 3.0}$	$2.9_{\pm 0.8}$	$0.0_{\pm 0.0}$	$9.4_{\pm 4.1}$	$38.2_{\pm 6.1}$	$\underline{37.2}_{\pm 1.0}$
		h-mid	$92.1_{\pm 1.6}$	$90.3_{\pm 5.3}$	$19.5_{\pm 2.9}$	$39.3_{\pm 6.7}$	$36.2_{\pm 3.7}$	$4.9_{\pm 2.0}$	$\underline{46.6}_{\pm 2.4}$
		h-high	$93.8_{\pm 2.1}$	$51.6_{\pm 2.6}$	$30.5_{\pm 2.9}$	$43.6_{\pm 2.1}$	$50.7_{\pm 1.0}$	$41.5_{\pm 2.0}$	$\underline{53.4}_{\pm 1.0}$
Qwen2.5-7B-Instruct	GRPO	h-low	$64.4_{\pm 1.7}$	$38.7_{\pm 3.2}$	$36.2_{\pm 3.4}$	$37.6_{\pm 5.3}$	$36.2_{\pm 1.3}$	$12.2_{\pm 2.4}$	$\underline{39.2}_{\pm 1.7}$
		h-mid	$77.4_{\pm 3.5}$	$81.7_{\pm 6.7}$	$0.6_{\pm 1.0}$	$44.4_{\pm 6.5}$	$36.2_{\pm 4.5}$	$28.5_{\pm 5.1}$	$\underline{42.7}_{\pm 1.0}$
		h-high	$88.7_{\pm 1.0}$	$90.3_{\pm 0.0}$	$71.3_{\pm 2.6}$	$63.2_{\pm 7.4}$	$58.0_{\pm 3.3}$	$56.1_{\pm 2.4}$	$\underline{71.5}_{\pm 1.6}$
	GiGPO	h-low	$90.4_{\pm 1.0}$	$86.0_{\pm 1.9}$	$51.7_{\pm 0.0}$	$34.2_{\pm 1.5}$	$29.0_{\pm 5.5}$	$30.1_{\pm 9.9}$	$\underline{54.4}_{\pm 2.4}$
		h-mid	$89.8_{\pm 1.7}$	$90.3_{\pm 3.2}$	$33.9_{\pm 1.0}$	$35.0_{\pm 1.5}$	$26.1_{\pm 4.3}$	$61.0_{\pm 2.4}$	$\underline{55.2}_{\pm 0.8}$
		h-high	$88.1_{\pm 2.9}$	$91.4_{\pm 1.9}$	$63.2_{\pm 2.0}$	$53.0_{\pm 5.3}$	$65.2_{\pm 4.3}$	$38.2_{\pm 3.7}$	$\underline{66.9}_{\pm 0.8}$

Table 14: Post-training success rates of under task shift: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{easy}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness and schema.

Model	Algorithm	Harness	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2	
Qwen2.5-3B-Instruct	GRPO	h-low	$6.2_{\pm 2.6}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$\underline{1.3}_{\pm 0.6}$
		h-mid	$59.9_{\pm 1.0}$	$2.2_{\pm 1.9}$	$64.4_{\pm 5.0}$	$57.3_{\pm 1.5}$	$36.2_{\pm 6.6}$	$3.3_{\pm 3.7}$	$\underline{41.5}_{\pm 2.4}$
		h-high	$81.9_{\pm 1.0}$	$32.3_{\pm 3.2}$	$81.0_{\pm 1.7}$	$89.7_{\pm 0.0}$	$85.5_{\pm 1.3}$	$27.6_{\pm 7.8}$	$\underline{69.7}_{\pm 1.1}$
	GiGPO	h-low	$6.2_{\pm 2.6}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$\underline{1.3}_{\pm 0.6}$
		h-mid	$53.1_{\pm 1.0}$	$18.3_{\pm 4.9}$	$81.0_{\pm 1.7}$	$76.9_{\pm 2.6}$	$81.2_{\pm 1.3}$	$17.9_{\pm 3.7}$	$\underline{57.9}_{\pm 1.1}$
		h-high	$51.4_{\pm 3.9}$	$12.9_{\pm 3.2}$	$91.4_{\pm 1.7}$	$86.3_{\pm 3.0}$	$86.2_{\pm 2.5}$	$10.6_{\pm 1.4}$	$\underline{60.2}_{\pm 0.6}$
Qwen2.5-7B-Instruct	GRPO	h-low	$61.0_{\pm 1.7}$	$11.8_{\pm 1.9}$	$75.3_{\pm 3.6}$	$73.5_{\pm 3.9}$	$72.5_{\pm 2.5}$	$35.0_{\pm 5.1}$	$\underline{58.3}_{\pm 2.0}$
		h-mid	$59.3_{\pm 4.5}$	$28.0_{\pm 6.7}$	$64.9_{\pm 2.0}$	$70.1_{\pm 1.5}$	$58.0_{\pm 2.5}$	$26.0_{\pm 2.8}$	$\underline{53.3}_{\pm 1.9}$
		h-high	$86.4_{\pm 0.0}$	$88.2_{\pm 3.7}$	$89.1_{\pm 1.0}$	$78.6_{\pm 3.9}$	$79.7_{\pm 1.3}$	$34.1_{\pm 4.2}$	$\underline{77.1}_{\pm 0.4}$
	GiGPO	h-low	$54.8_{\pm 2.6}$	$32.3_{\pm 3.2}$	$73.0_{\pm 1.0}$	$75.2_{\pm 3.0}$	$79.7_{\pm 1.3}$	$4.9_{\pm 2.4}$	$\underline{55.7}_{\pm 0.2}$
		h-mid	$68.4_{\pm 3.5}$	$47.3_{\pm 8.1}$	$71.8_{\pm 1.0}$	$76.1_{\pm 3.9}$	$67.4_{\pm 0.0}$	$17.1_{\pm 2.4}$	$\underline{60.0}_{\pm 0.6}$
		h-high	$81.4_{\pm 3.4}$	$23.7_{\pm 1.9}$	$96.0_{\pm 1.0}$	$82.1_{\pm 4.4}$	$92.0_{\pm 3.3}$	$52.0_{\pm 6.1}$	$\underline{75.4}_{\pm 1.3}$

Table 15: Post-training success rates of under task shift: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{med}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness and schema. For `Qwen2.5-3B-Instruct` under `h-low`, post-training fails to take effect, so zero-shot results are reported instead.

Model	Algorithm	Harness	$\mathcal{D}^{\text{te}}_{\text{easy}}$		$\mathcal{D}^{\text{te}}_{\text{med}}$			$\mathcal{D}^{\text{te}}_{\text{hard}}$	All
			Pick	Look	Clean	Heat	Cool	Pick 2	
Qwen2.5-3B-Instruct	GRPO	h-low	$6.2_{\pm 2.6}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$\underline{1.3}_{\pm 0.6}$
		h-mid	$11.9_{\pm 4.5}$	$2.2_{\pm 1.9}$	$0.0_{\pm 0.0}$	$3.4_{\pm 3.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$\underline{3.3}_{\pm 1.1}$
		h-high	$35.0_{\pm 2.6}$	$15.1_{\pm 1.9}$	$11.5_{\pm 1.0}$	$12.8_{\pm 0.0}$	$2.2_{\pm 2.2}$	$0.0_{\pm 0.0}$	$\underline{13.9}_{\pm 0.7}$
	GiGPO	h-low	$6.2_{\pm 2.6}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$\underline{1.3}_{\pm 0.6}$
		h-mid	$11.9_{\pm 4.5}$	$2.2_{\pm 1.9}$	$0.0_{\pm 0.0}$	$3.4_{\pm 3.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$\underline{3.3}_{\pm 1.1}$
		h-high	$35.0_{\pm 2.6}$	$15.1_{\pm 1.9}$	$11.5_{\pm 1.0}$	$12.8_{\pm 0.0}$	$2.2_{\pm 2.2}$	$0.0_{\pm 0.0}$	$\underline{13.9}_{\pm 0.7}$
Qwen2.5-7B-Instruct	GRPO	h-low	$80.8_{\pm 1.0}$	$15.1_{\pm 9.9}$	$13.8_{\pm 4.6}$	$0.9_{\pm 1.5}$	$4.3_{\pm 2.2}$	$68.3_{\pm 4.2}$	$\underline{33.1}_{\pm 0.2}$
		h-mid	$71.8_{\pm 1.0}$	$44.1_{\pm 1.9}$	$13.8_{\pm 1.7}$	$6.0_{\pm 3.9}$	$13.0_{\pm 4.3}$	$65.0_{\pm 2.8}$	$\underline{36.1}_{\pm 0.0}$
		h-high	$81.4_{\pm 1.7}$	$53.8_{\pm 9.3}$	$31.0_{\pm 3.0}$	$29.1_{\pm 1.5}$	$24.6_{\pm 7.0}$	$81.3_{\pm 1.4}$	$\underline{50.6}_{\pm 1.2}$
	GiGPO	h-low	$41.2_{\pm 1.0}$	$17.2_{\pm 4.9}$	$15.5_{\pm 0.0}$	$3.4_{\pm 1.5}$	$0.0_{\pm 0.0}$	$95.1_{\pm 0.0}$	$\underline{28.8}_{\pm 0.4}$
		h-mid	$81.4_{\pm 1.7}$	$31.2_{\pm 1.9}$	$7.5_{\pm 1.0}$	$18.8_{\pm 3.9}$	$5.8_{\pm 1.3}$	$83.7_{\pm 3.7}$	$\underline{38.8}_{\pm 0.6}$
		h-high	$95.5_{\pm 1.0}$	$93.5_{\pm 3.2}$	$54.0_{\pm 7.0}$	$56.4_{\pm 4.4}$	$51.4_{\pm 5.5}$	$94.3_{\pm 3.7}$	$\underline{73.4}_{\pm 2.9}$

Table 16: Post-training success rates of under task shift: Agents are post-trained on $\mathcal{D}^{\text{tr}}_{\text{hard}}$ under each harness with tool schema fixed to `v1.0`, and evaluated on $\mathcal{D}^{\text{te}}_{\text{all}}$ under the same harness and schema. For `Qwen2.5-3B-Instruct`, post-training fails to take effect, so zero-shot results are reported instead.