LEARNING *What* VS. LEARNING *How*: DECOMPOSING TASK AND INTERFACE COMPETENCE IN AGENTIC TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents operate through *harnesses* that define system prompts, action formats, feedback loops, and resource constraints. Although a model may possess the same underlying task knowledge, successful execution does not necessarily follow under every harness. This raises basic questions: when a policy fails under a given harness, is it missing task knowledge, or does it simply lack the competence to operate the interface? Can harness-conditioned training improve task competence while degrading interface competence, or vice versa? To address these, we formulate a two-component view of agentic competence: *task competence*, the knowledge and reasoning required to solve a task, and *interface competence*, the ability to act on that knowledge through a given harness. We demonstrate that harness-conditioned training can yield vastly different outcomes depending on how these competencies interact. While multi-harness distillation aggregates task knowledge learned from different harness trajectories into a shared policy, this knowledge is only realised if the training distribution preserves or improves interface competence for the target harness. Otherwise, harness diversity can induce *interference* in interface competence, sometimes catastrophically collapsing performance on a held-out harness despite improvements in underlying task competence. To further distinguish the effects of task and interface supervision, we introduce *harness rehearsal*: distilling the model on synthetic task trajectories designed to carry minimal task knowledge while exercising the target harness's interface. We find that rehearsal alone raises performance under the target harness, and when recombined with task knowledge acquired under different harnesses, produces gains far larger than either alone. We show that the effect holds across four reasoning and agentic benchmarks and persists through the subsequent reinforcement learning stage.

1 INTRODUCTION

Language model agents are defined not only by what they know, but by what they can do: using tools, executing code, and completing tasks over long interaction horizons. In these settings, an agent acts through a *harness*, an interface that determines what it can observe, which actions it can take, and how those actions are interpreted (Yao et al., 2026; Raj et al., 2026; Gu, 2026). This creates a consequential source of failure: a policy may know how to solve a task yet fail because it cannot express that solution through a given harness. A coding agent may identify the correct fix but mishandle the shell, a clinical reasoning agent may reason correctly about a patient record but fail to retrieve or submit the required information through the available interface. Observed performance under a harness therefore reflects both the quality of reasoning about the task and the ability to carry out the intended actions through the harness.

We study this distinction in terms of *task competence*, the knowledge and reasoning needed to solve the task, and *interface competence*, the ability to carry that solution out through a particular harness. Task competence concerns *what* a policy can solve, whereas interface competence concerns *how* it carries out those solutions through a given harness. The distinction becomes especially important when harnesses are part of the training environment rather than only a deployment interface. We hypothesise that performance under a harness depends on how these two competences interact: task competence can transfer across harnesses, while interface competence can facilitate or hinder that

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

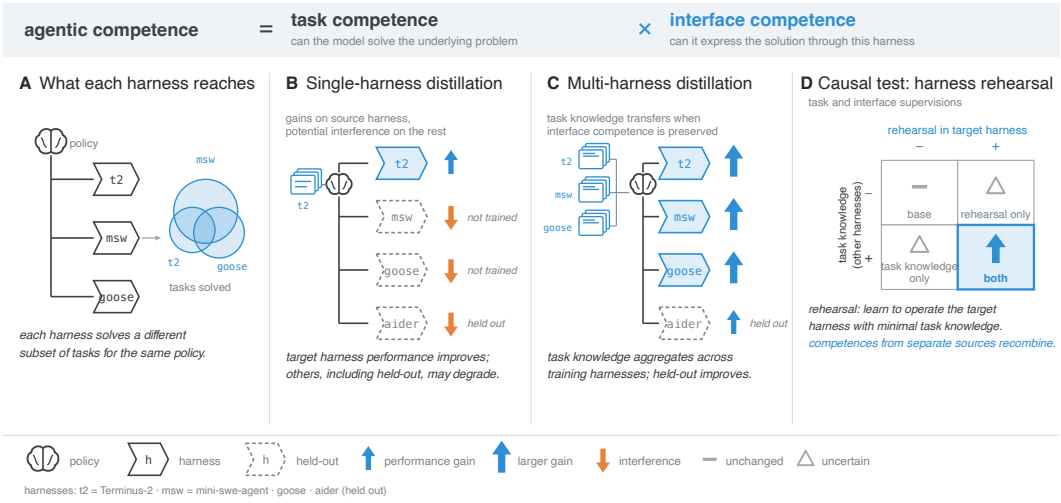

Figure 1: **Overview of our study of task and interface competence across harnesses.** (1) A fixed policy exhibits different capabilities through different harnesses. (2) Single-harness distillation improves performance on the training harness but can interfere with others. (3) Multi-harness distillation preserves gains across training harnesses and can transfer to a held-out harness. (4) Rehearsal separates task supervision from interface practice, allowing us to study how the two contribute to cross-harness transfer.

transfer. We ask four related questions: **Q1. Does the harness change which tasks a policy can solve, or only how well it performs?** For a frozen base model, performance varies substantially across harnesses and the tasks solved through them only partly overlap (§5.1). **Q2. How does supervision from one harness, or from several, transfer to other harnesses?** Distilling on one harness, that is, training on successful trajectories a larger model collected through it, improves that harness and can lower performance through others, in some cases catastrophically. Distilling multiple training harnesses generally raises performance on them and, on a held-out harness that no training stage uses, avoids collapse (§5.2). **Q3. Are task and interface supervision separable, and do they interact?** Harness rehearsal teaches the model to act through a target harness without having to work out the task. Added to traces from other harnesses, it lifts target-harness success far above what either gives alone, which tests a prediction of the hypothesis by intervening on the supervision (§5.3). **Q4. After multi-harness distillation, does reinforcement learning (RL) preserve what was learnt, and does RL on a single harness help or interfere with the others?** The held-out harness did not degrade in any setting tested, and RL on a single harness can improve performance on other harnesses (§5.4).

2 Related Work

Harnesses are becoming an increasingly important part of agentic systems, shaping not only how agents are evaluated but also how they are trained. Harness-Bench quantifies this dependence by systematically varying harness configurations (Yao et al., 2026). One line of work optimises the harness around a policy. Meta-Harness and HarnessOpt-Bench study harness optimisation, while HarnessDev and Self-Harness explore agents that create or improve their own harnesses (Lee et al., 2026; Ursekar et al., 2026; Wu et al., 2026; Zhang et al., 2026). This work treats the harness as an adaptable part of the agent system rather than a mere implementation detail.

A complementary line of work asks what happens when the policy must learn across harnesses. HarnessBandit studies scheduling across harnesses for agentic RL (Wei et al., 2026), *What Does Multi-Harness RL Learn?* examines cross-harness credit assignment and portability (Chen et al., 2026; Le et al., 2026), and *Co-Evolving Harnesses and Models* jointly evolves harnesses and policies (Yu et al., 2026). These works make transfer across harnesses an explicit training problem.

Our work asks a more specific question: *what drives that transfer, and what causes interference?* We separate task supervision from experience with the target interface, allowing us to test when supervision collected through one harness becomes useful through another, whether interface-specific learning transfers or interferes, and whether gains from multi-harness distillation persist through RL.

3 METHOD

Our goal is to understand how training across multiple agent harnesses reshapes what a shared policy can solve and how well it can carry that out through different interfaces. Each question in Section 1 needs a different comparison. Q1 needs no training: we evaluate one frozen policy through several harnesses. Q2 compares policies distilled from a single harness with a policy distilled from several (§3.1). Q3 needs supervision that carries only one of the two kinds of information in a successful trace: how to solve the task or how to act through the harness, which harness rehearsal provides (§3.2). Q4 asks what reinforcement learning (RL) adds after supervised fine-tuning (SFT) (§3.3).

3.1 MULTI-HARNESS DISTILLATION

For Q2, we study how supervision collected through different harnesses combines under distillation. For each $h \in \mathcal{H}_{\text{train}}$, we sample teacher trajectories on the training tasks and retain only verifier-passing trajectories. Collection settings are given in Appendix B.1. Let $\mathcal{T}_h$ denote the resulting pool. We train students either on a single harness-specific pool or on the pooled demonstrations from all training harnesses,

$$\mathcal{T}_{\text{MH}} = \bigcup_{h \in \mathcal{H}_{\text{train}}} \mathcal{T}_h, \qquad \mathcal{L}(\pi) = -\mathbb{E}_{\tau \sim \mathcal{T}_{\text{MH}}} \sum_{t=1}^{|s(\tau)|} \log \pi\big(x_t \mid x_{<t}\big), \tag{1}$$

where $s(\tau)$ is the serialised trajectory and x_t its tokens. The loss covers the full trajectory, including harness-specific interaction and feedback, and we evaluate each trained policy across training and held-out harnesses. Each successful trace provides both task supervision, demonstrating the reasoning needed to find a solution, and interface supervision, showing how to carry it out through the harness. We use single- and multi-harness distillation to study how training through one or more harnesses affects performance through others. Throughout, a *specialist* is a policy distilled from a single harness's pool, and *3H* denotes the policy distilled from all three training pools. Relative to the base policy, training *transfers* to a harness when performance through that harness rises, and *interferes* with it when performance falls. These comparisons reveal patterns of transfer and interference, but cannot isolate which source of supervision drives them. Harness rehearsal (§3.2) separates the two.

3.2 DECOMPOSING TASK AND INTERFACE SUPERVISION WITH HARNESS REHEARSAL

For Q3, harness rehearsal tests this distinction more directly by varying the two sources of supervision separately. Distillation traces collected through other harnesses provide task supervision without exposure to the target interface. Rehearsal provides the complement: interface supervision that does not require the model to work out the solution. Crossing the two lets us test a prediction of the hypothesis, that task supervision from other harnesses has a larger positive effect on target-harness performance when training also includes target-harness interface supervision.

We construct each rehearsal instance from a training task: the environment, resources, and verifier remain unchanged, but the original instruction is replaced with one that states exactly what to do, including which values to extract or submit. The model therefore does not need to work out what solves the task. It only needs to carry out what the instruction says correctly through the target harness h. Formally, for each training task $x \in \mathcal{D}_{\text{train}}$ with a known solution, we construct a rehearsal instruction $\rho(x)$. We run these synthetic rehearsal tasks with a demonstrator policy (the 3H student) through the harness and keep only trajectories that pass the unmodified solution verifier:

$$\mathcal{R}_h \;=\; \big\{\tau : \tau \sim \pi_{\text{demo}} \text{ on } (\rho(x), h),\; x \in \mathcal{D}_{\text{train}},\; \text{verify}(\tau) = 1\big\}. \tag{2}$$

$\mathcal{R}_h$ consists of verified interaction traces serialised in the same format as full distillation trajectories collected through harness h. We obtain the synthetic rehearsal instructions programmatically from

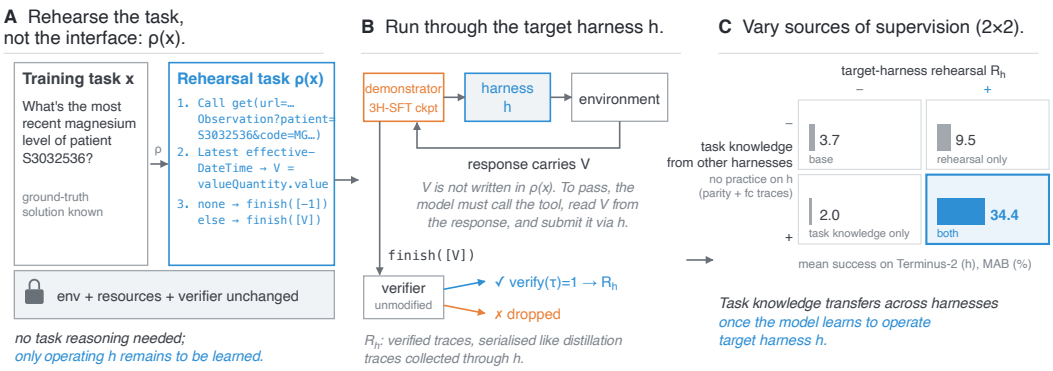

Figure 2: **Harness rehearsal.** (A) We keep a training task's environment, resources and verifier but replace its instruction with $\rho(x)$, which includes instructions on the correct actions, so the model only learns how to carry it out through the target harness. (B) A demonstrator runs $\rho(x)$ through the target harness h, and traces that pass the original verifier form the rehearsal set $\mathcal{R}_h$. (C) Crossing rehearsal with task supervision from other harnesses gives a $2{\times}2$ that separates task supervision from interface supervision.

each instance's ground-truth solution and metadata, so their content is explicitly controlled. For instance, a MedAgentBench rehearsal instruction names the `get` call to make and its URL, which field of the response to read, and what to pass to the `finish` tool (Appendix C.1). A trajectory passes only if the policy issued the call, parsed the response, and submitted the fetched value through the harness's termination idiom.

Varying task and interface supervision separately. We evaluate harness rehearsal in a 2×2 design, varying whether training includes cross-harness distillation and target-harness rehearsal. The four conditions measure each source of supervision in isolation and together, with success on the target harness as the outcome. The hypothesis predicts that the two sources are complementary. The design does not assume that task and interface competence can be directly measured as latent quantities: it varies the training experience associated with each and asks whether their effects on target-harness performance interact. We additionally test alternative explanations with failed-rehearsal, distillation pool size, and leakage controls described in Appendix C.

3.3 Multi-Harness Reinforcement Learning

For Q4, we apply GRPO through the same task containers, harness implementations, and verifiers used for distillation. Each step samples a batch of training tasks and, for every task, rolls out G trajectories *per harness* $h \in \mathcal{H}_{\text{train}}$ against the live policy. The reward is the unmodified verifier's binary task success $r(\tau) \in \{0, 1\}$. Advantages are group-relative within a (task, harness) group,

$$\hat{A}(\tau_i) \;=\; r(\tau_i) - \frac{1}{G}\sum_{j=1}^{G} r(\tau_j), \tag{3}$$

and the policy-gradient loss is taken over the policy's own generated tokens only. In contrast to the full-sequence distillation objective of Eq. 1, harness observations enter as context but not as prediction targets. Because the advantage in Eq. 3 is zero when all rollouts for a task receive the same reward, we run RL on training tasks with non-zero success rate under the initial checkpoint. Evaluation uses the full held-out split. We compare rollouts on all three training harnesses with the same number of rollouts on a single harness (Appendix B.2).

4 Experimental Settings

We present in this section the benchmarks, harnesses, models and evaluation settings in this study.

Benchmarks and harnesses. We evaluate on MedAgentBench (MAB) (Jiang et al., 2025), Reasoning Gym (RG) (Stojanovski et al., 2026), AppWorld (Trivedi et al., 2024), and the Berkeley Function-Calling Leaderboard (BFCL V3) (Patil et al., 2025), using disjoint training and test tasks. For RG, we train on generation seeds 0–1 and test on seed 2 at both easy and hard difficulties. For each benchmark, we train on three harnesses and hold out a fourth (Table 1). Terminus-2, mini-swe-agent, Goose, and Aider are existing agent scaffolds. The remaining harnesses mainly differ in how the model interacts with the environment. FC uses native tool calls, JSON and XML use structured action formats, PYCALL uses Python calls, CODE uses AppWorld's persistent Python environment, and PARITY is MedAgentBench's benchmark-native reference harness. Appendix A gives the full splits, harness configurations, and evaluation details.

Table 1: Training and held-out harnesses for each benchmark.

Benchmark	Training harnesses	Held out
MedAgentBench	PARITY / FC / Terminus-2	XML
Reasoning Gym	Terminus-2 / mini-swe / Goose	Aider
AppWorld	CODE / FC / mini-swe	XML
BFCL single-turn	FC / JSON / Terminus-2	XML
BFCL multi-turn	FC / JSON / PYCALL	XML

Models and training. We distill verifier-passing Qwen3-32B demonstrations into Qwen3-4B in reasoning mode, using single-harness pools or their union, which we call 3H. The 3H pool contains more demonstrations than the specialists, and separate volume-matched controls isolate data-volume effects. On MAB/RG, GRPO starts from SFT checkpoints and samples training tasks with nonzero initial success, and single-/multi-harness controls match total rollout count. Appendix B specifies collection, demonstrators, pool construction, and optimisation. **Evaluation.** We report mean verifier success and pass@4 across evaluation tasks. Appendix A.1 provides benchmark-specific evaluation details and decoding settings.

5 Results

We address the four questions from Section 1 in turn. We begin by characterising how harness choice and harness-conditioned training affect performance, then use harness rehearsal to test the task/interface distinction more directly, and ask whether these patterns persist through reinforcement learning.

5.1 Harness Choice Changes Which Tasks a Policy Solves

We begin with the frozen base policy, asking how harness choice affects both overall performance and which tasks are solved (Q1). For Qwen3-4B, performance varies widely across harnesses (Figure 3a). On MAB, mean success ranges from about 4% to 36%. The sets of solved tasks also differ (Figure 3b): on AppWorld, each of the three harnesses solves between 14 and 18 tasks that neither of the others solves. The share of solved tasks reachable only through a harness other than the strongest one is 17% on MAB and 5% on ReasoningGym, where one harness solves nearly everything the others do, and for the 32B teacher it reaches 45% on AppWorld (Appendix E.1). This suggests that harnesses differ not only in overall strength, but also in which parts of a model's underlying task competence they expose.

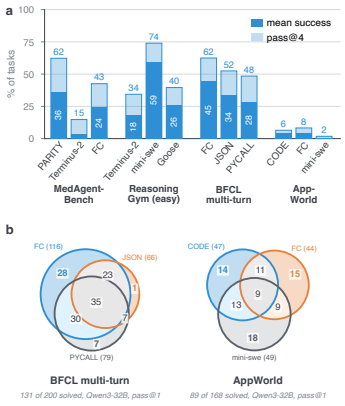

Figure 3: **Harness choice changes task reach.** (a) Qwen3-4B base model performance across harnesses. (b) Tasks each harness solves on datasets, counts per region, Qwen3-32B, pass@1.

5.2 Transfer, Interference and Combination Across Harnesses

Q2 asks how training changes the policy of §5.1: what distilling on one harness does to performance through others, and what happens when supervision from several harnesses is combined. Table 2

Table 2: **Single-harness specialists improve their own interface but often degrade others, while 3H achieves gains beyond single-harness training across harnesses.** Rows denote training pools and columns denote evaluation harnesses. We report mean success (%) with 95% bootstrap confidence intervals ($n = 4$). Arrows indicate change relative to the base model, and bold marks the best training pool in each column. BFCL single-turn and Reasoning Gym Easy appear in Appendix Table A13, and held-out harness results in Figure 4.

MedAgentBench				**AppWorld**			
Pool	PARITY	Terminus-2	FC	Pool	CODE	FC	mini-swe
none (base)	36.1±9.2	3.7±2.3	24.3±8.4	none (base)	4.2±3.9	3.9±3.2	0.5±0.5
PARITY-only	47.3±11.7↑	4.1±2.7↑	21.8±8.6↓	CODE-only	10.7±4.7↑	4.6±2.5↑	7.3±3.4↑
Terminus-2-only	26.4±7.6↓	**21.7±6.4**↑	16.1±5.2↓	FC-only	1.9±1.6↓	1.5±1.2↓	1.8±1.5↑
FC-only	28.7±11.3↓	0.0↓	45.1±9.4↑	mini-swe-only	9.4±4.5↑	4.3±2.3↑	14.0±5.0↑
3H	**67.2±10.2**↑	16.9±5.9↑	**55.3±8.2**↑	3H	**18.4±6.5**↑	**15.2±6.0**↑	**18.7±6.6**↑
BFCL multi-turn				**Reasoning Gym Hard**			
Pool	FC	JSON	PYCALL	Pool	Terminus-2	mini-swe	Goose
none (base)	44.6±5.9	33.5±5.4	28.2±5.1	none (base)	18.1±6.1	44.1±7.9	20.7±6.8
FC-only	55.2±5.3↑	0.4±0.6↓	19.5±3.9↓	Terminus-2-only	42.7±8.5↑	46.1±7.5↑	27.7±6.6↑
JSON-only	29.6±4.8↓	49.0±5.2↑	27.9±4.4↓	mini-swe-only	11.6±3.3↓	33.8±8.1↓	26.4±6.3↑
PYCALL-only	22.0±4.5↓	34.5±5.0↑	44.2±5.2↑	Goose-only	28.2±6.6↑	38.6±7.8↓	44.3±8.5↑
3H	**59.0±5.5**↑	**49.5±5.4**↑	**49.1±5.6**↑	3H	**43.3±7.8**↑	**51.0±8.0**↑	**46.0±8.2**↑

compares the single-harness specialists with 3H on every harness (BFCL single-turn and Reasoning Gym Easy are in Appendix Table A13), and Figure 4 does the same for the held-out harness.

Table 3: **3H scores higher than the best specialist on four of six benchmarks on its own harness.** Mean success (n=4, %).

Benchmark	Harness	Base	Spec.	3H
MedAgentBench	PARITY	36.1	47.3	**67.2**
AppWorld	mini-swe	0.5	14.0	**18.7**
BFCL single-turn	JSON	86.2	**87.9**	86.2
BFCL multi-turn	FC	44.6	55.2	**59.0**
Reasoning Gym Easy	Goose	25.7	**64.6**	60.8
Reasoning Gym Hard	Goose	20.7	44.3	**46.0**

(i) Single-harness distillation produces harness-specific gains. Across benchmarks, distilling on a single harness consistently improves performance on that harness, for example PARITY on MAB (36.1 → 47.3) and mini-swe on AppWorld (0.5 → 14.0). **(ii) But it can degrade performance on other harnesses.** The effect is not simply a lack of transfer: performance on other harnesses often falls below the base policy. On BFCL multi-turn, FC distillation drives JSON from 33.5 to 0.4, and on BFCL single-turn Terminus-2 distillation reduces FC from 87.0 to 21.4 (Appendix Table A13). The effect is clearest when the base policy already performs well on the affected harness. Distillation on one harness can thus impair capabilities the model could previously express through others. **(iii) Multi-harness distillation improves performance on training harnesses.** On BFCL multi-turn, distillation across all three harnesses raises JSON, which FC distillation had driven to 0.4, to 49.5, above the base policy (33.5) and on par with JSON distillation alone (49.0). Whether this restores task or interface competence is not visible here. §5.3 separates the two. **(iv) Multi-harness distillation can combine gains across harnesses.** On AppWorld, 3H outperforms each specialist on that specialist's own harness (CODE 18.4 vs. 10.7, FC 15.2 vs. 1.5, mini-swe 18.7 vs. 14.0, with only the FC intervals disjoint), and BFCL multi-turn and RG Hard point the same way, although most of these margins lie within the uncertainty. This is consistent with task competence aggregating across the traces from different harnesses, while performance on any one harness depends on how well the model can carry that competence out through its interface.

Figure 4 shows the held-out harness, which no training stage used, and here the pattern is asymmetric. Single-harness distillation can collapse a harness it never saw: in four of the six settings the worst specialist falls below half of the base score, for example XML on BFCL multi-turn, which FC-only distillation takes from 20.6 to 2.5. 3H does not. It ends 10 to 66 points above the worst specialist in every setting, and its largest fall below base (Reasoning Gym Hard, 17.6 → 9.8) has overlapping intervals. But it improves on base only on MedAgentBench and AppWorld, is level on BFCL and lower on Reasoning Gym, so multi-harness distillation avoids the collapse without reliably carrying performance over (full scores in Appendix Table A14).

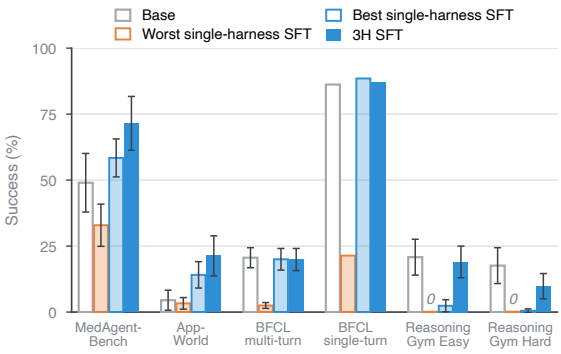

Figure 4: **Transfer to the held-out harness.** Best and worst are the highest and lowest of the three single-harness specialists. Error bars are 95% CIs (full evaluation is in Appendix Table A14).

Taken together, distilling on one harness improves it and can lower performance through others, including a harness it never saw, while distilling on all three training harnesses keeps the gains on the trained harnesses (Table 3) and avoids the held-out collapse. Both patterns are consistent with our hypothesis but do not identify which competence changed. The rehearsal experiments in §5.3 test a prediction that separates the two.

5.3 Task and Interface Supervision Interact

Q3 asks whether task supervision and interface supervision are separable, and whether they interact. §5.2 didn't directly answer this, because a rejection-sampled trace carries both: it demonstrates how to solve the problem and how to carry that out through a particular harness. We therefore construct sources of supervision that emphasise one each. Target-harness rehearsal provides interface supervision without requiring the model to work out the solution, and distillation traces from other harnesses provide task supervision without exposure to the target interface. The task-supervision pool contains PARITY+FC traces for MAB, mini-swe+Goose for RG and JSON+PYCALL for BFCL multi-turn, with targets Terminus-2, Terminus-2 and FC. Table 4 crosses the two sources in a 2×2 design and measures success on the target harness.

The two sources interact strongly on all three benchmarks (Figure 5). On MAB, cross-harness distillation alone gives 2.0 on the target Terminus-2 harness and rehearsal alone 9.5, against 3.7 for the base model. With both, performance rises to 34.4. On RG the two alone give 19.8 and 20.0 against a base of 18.0, and both together 37.0. On BFCL multi-turn's target FC harness they give

Table 4: **Task supervision and harness rehearsal are complementary.** Separating sources of task and interface supervision and recombining the two substantially improves target-harness performance over either component in isolation, and exceeds the native target-harness specialist on MedAgentBench and BFCL multi-turn. Results show success (%) with 95% CIs. Bold indicates the best row in each column excluding baselines.

	MedAgentBench (target: Terminus-2)			**Reasoning Gym** (target: Terminus-2)			**BFCL multi-turn** (target: FC)		
Pool	PARITY	Terminus-2	FC	Terminus-2	mini-swe	Goose	FC	JSON	PYCALL
base	36.1±9.2	3.7±2.3	24.3±8.4	18.0±5.5	59.4±7.9	25.7±6.7	44.6±5.9	33.5±5.4	28.2±5.1
task supervision only	**63.5±10.0**	2.0±1.6	47.1±10.0	19.8±5.9	58.8±7.9	58.4±8.8	8.5±2.8	47.4±5.8	**52.5±5.3**
harness rehearsal only	40.4±10.2	9.5±5.7	38.5±10.9	20.0±5.7	45.4±7.3	20.0±5.8	32.5±5.1	0.0	0.1±0.2
task supervision + harness rehearsal	52.5±9.0	**34.4±8.2**†	**57.0±9.0**	**37.0±7.1**	**62.4±7.8**	**62.9±8.5**	**58.8±5.7**	**50.8±5.9**	50.9±5.0
native target-only	26.4±7.6	21.7±6.4	16.1±5.2	54.7±7.8	55.7±8.4	40.2±7.3	55.2±5.3	0.4±0.6	19.5±3.9
3H	67.2±10.2	16.9±5.9	55.3±8.2	55.3±7.9	63.0±7.6	60.8±8.2	59.0±5.5	49.5±5.4	49.1±5.6

Task-supervision traces exclude the target harness. Bold excludes reference rows below the rule. †: eight-attempt replication, 34.2%.

Table 5: **Single-harness GRPO can transfer beyond the rollout harness.** We start from the 3H-SFT checkpoint and compare GRPO using rollouts from all three training harnesses (3H-GRPO) with GRPO using only the strongest training harness (Single-GRPO: PARITY for MAB and mini-swe for RG), using matched rollout counts (Appendix B.2). We report success (%) with 95% CIs. Bold marks the best result in each block, and † denotes the held-out harness.

MedAgentBench				
	PARITY	Terminus-2	FC	XML†
3H-SFT	67.2±10.2	16.9±5.9	55.3±8.2	71.5±10.0
3H-GRPO	65.8±9.4	**39.7**±9.6	58.4±8.1	72.9±9.0
Single-GRPO	**72.1**±9.8	35.8±9.4	**66.0**±8.6	**76.8**±7.9
Reasoning Gym Easy				
	Terminus-2	mini-swe	Goose	Aider†
3H-SFT	55.3±7.9	63.0±7.6	60.8±8.2	19.0±6.0
3H-GRPO	62.8±7.8	65.0±7.7	**63.9**±8.7	**21.4**±6.7
Single-GRPO	**62.9**±7.9	**67.0**±7.7	62.1±8.4	19.5±6.4
Reasoning Gym Hard				
	Terminus-2	mini-swe	Goose	Aider†
3H-SFT	43.3±7.8	51.0±8.0	**46.0**±8.2	9.8±4.8
3H-GRPO	43.8±8.3	**52.8**±8.1	45.7±8.2	**12.3**±5.7
Single-GRPO	**44.3**±7.9	52.7±8.3	43.8±8.4	11.3±5.0

8.5 and 32.5 against a base of 44.6, and both together 58.8, using 268 rehearsal rows against the 593 trajectories behind the native FC specialist.

This suggests that the two sources of supervision are complementary: task supervision from other harnesses transfers much more effectively to the target harness when the model has also practised with that harness. The combined pool beats native target-harness distillation on MAB (34.4 vs. 21.7) and BFCL multi-turn (58.8 vs. 55.2) but remains below it on RG (37.0 vs. 54.7). Rehearsal is also interface-specific: on BFCL multi-turn, rehearsal alone collapses JSON and PYCALL, and adding it to the task-supervision pool leaves both unchanged, so the target-harness gain does not cost the harnesses that pool supplies. Construction and controls are in Appendix C, replications in Appendix E, and a test on a harness absent from every training pool in Appendix D. This is a controlled intervention on the supervision, and the interaction is what the hypothesis predicts. It does not measure task or interface competence directly, and each condition is a single training run.

5.4 Reinforcement Learning After Multi-Harness Distillation

Multi-harness SFT competence carries through RL. GRPO from the 3H checkpoint generally improves on SFT, and the held-out interface, which no stage trains on, does not degrade in any of the four settings and improves in all four (per-harness results in Appendix Table A19). What multi-harness SFT established therefore survives RL. Single-harness SFT interfered across harnesses (§5.2), so we then ask whether RL must stay multi-harness or can concentrate on one strong harness and let the gains transfer.

Single-harness RL can improve on other harnesses. On MAB, GRPO with only PARITY rollouts improves every interface, including the held-out XML (71.5 → 76.8), and beats cross-harness GRPO on three of the four (Table 5). Reasoning Gym shows the same pattern less cleanly: mini-swe-only GRPO improves all four interfaces on the easy tier and three of four on the hard tier. We do not claim that single-harness RL is gener-

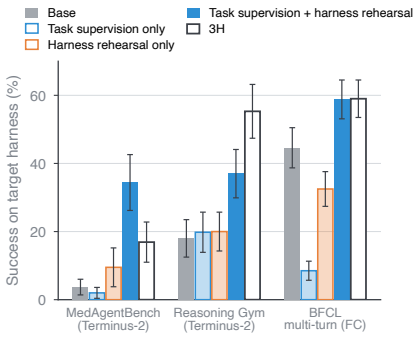

Figure 5: **Neither ingredient works alone.** Target-harness success after training on task supervision from other harnesses, harness rehearsal, or both, with 3H as a reference. Error bars are 95% CIs.

ally better, only that after multi-harness SFT it need not
bring back the interference seen under SFT. This is consistent with the hypothesis: multi-harness SFT establishes interface competence, and RL through one strong harness then improves task competence that the other interfaces can express. It is early evidence, not a prescription: the clearest effect is on MAB, several changes are small relative to their intervals, and each condition is a single run.

6 Conclusion

Harnesses increasingly shape not only how agents are evaluated, but also how they are trained, optimised, and adapted (Yao et al., 2026; Lee et al., 2026; Ursekar et al., 2026; Wu et al., 2026; Zhang et al., 2026; Yu et al., 2026). We study the complementary question of what a policy learns through a harness. Our results support a decomposition into task competence and interface competence. Task competence can transfer across harnesses, while interface-specific learning can either support that transfer or interfere with performance through other interfaces. This perspective helps explain why the same training signal can produce gains under one harness and losses under another. Harness rehearsal provides the clearest evidence for this decomposition. Task supervision collected through other harnesses becomes substantially more useful on a target harness once the model has also practised operating that interface. Multi-harness training can therefore aggregate task competence exposed through different harnesses without requiring every source of supervision to use the target interface.

Several questions remain. Transfer likely depends on how similar the training and target harnesses are, but we do not yet have a formal measure of this distance. Such a measure could help predict when cross-harness supervision will transfer and how much target-interface practice is required. Our RL results also show that, after multi-harness SFT, optimisation through a single harness can improve performance through others, echoing recent work on learning across harnesses (Wei et al., 2026; Le et al., 2026). Running RL directly from the base model would help determine how much of this transfer depends on interface competence established during SFT.

Overall, our results support a two-component view of agentic training: what a policy learns about a task and how it learns to act through a harness are related but distinct. Task competence can transfer across harnesses, while interface competence determines whether that knowledge can be realised through a particular one and can itself transfer or interfere. This decomposition provides a basis for studying when cross-harness transfer should occur, how interface similarity shapes it, and how future training methods might allocate supervision across tasks and interfaces.

AI use statement

In this work, we used generative AI tools to edit and condense text, to design figure layouts and generate plotting code, and to check numerical consistency between the text and the tables. We did not use generative AI to design the experiments, run training or evaluation, or produce the reported results. All AI-assisted text, figures, and code were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artefacts produced with the aid of generative AI.

Reproducibility statement

Section B details training parameters, setup, and collections. We also included the rehearsal details and controls in Appendix in Section C.

References

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. Harnessx: A composable, adaptive, and evolvable agent harness foundry, 2026. URL https://arxiv.org/abs/2606.14249.

Shangding Gu. From model scaling to system scaling: Scaling the harness in agentic ai, 2026. URL https://arxiv.org/abs/2605.26112.

Yixing Jiang, Kameron C Black, Gloria Geng, Danny Park, James Zou, Andrew Y Ng, and Jonathan H Chen. Medagentbench: a virtual ehr environment to benchmark medical llm agents. *Nejm Ai*, 2(9):AIdbp2500144, 2025.

Chenqian Le, Jiayi Cheng, Qijia He, Runhao Li, Yinghao Li, and Xupeng Chen. What does multi-harness rl learn? credit assignment and portability in coding agents, 2026. URL https://arxiv.org/abs/2609.04518.

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.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Harsh Raj, Vipul Gupta, Anas Mahmoud, Razvan-Gabriel Dumitru, Darvin Yi, Aakash Sabharwal, and Yunzhong He. Model or harness? an interaction-centric taxonomy for localizing agent failures, 2026. URL https://arxiv.org/abs/2607.28802.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. *Advances in Neural Information Processing Systems*, 38, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Varun Ursekar, Apaar Shanker, Yash Maurya, Shehab Yasser, Vijay S. Kalmath, Veronica Chatrath, and Yuan Xue. Harnessopt-bench: Evaluating llms at harness optimization, 2026. URL https://arxiv.org/abs/2608.06301.

Hongliang Wei, Xiaobing Tu, Yinggui Wang, Zhengxi Liu, Rongkun Xue, Jinkui Ren, Xiantao Zhang, Debin Zhao, and Xiaopeng Fan. Harnessbandit: Joint learnability-transferability scheduling for multi-harness agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2609.13739.

Yuhao Wu, Jingyuan Zhang, Jiajun Shi, Xinping Lei, Qingshui Gu, Yuxuan Zhang, Zexuan Wang, Chen He, Chen Huang, Maojia Song, Zhiyuan Zeng, Shaowen Wang, Jinkai Liu, Yunfeng Shi, Jiaheng Liu, Shen Yan, Wenhao Huang, Ge Zhang, and Wenxuan Zhang. Harnessdev: Can llms create and evolve their own agent harness?, 2026. URL https://arxiv.org/abs/2609.01437.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-bench: Measuring harness effects across models in realistic agent workflows, 2026. URL https://arxiv.org/abs/2605.27922.

Zhou Yu, Bin Bi, Shiva Kumar Pentyala, Shubham Mehrotra, Sougata Chaudhuri, Shilpa Bhagavath, Zeyuan Chen, Ran Xu, Phil Mui, James Zhu, and Sitaram Asur. Co-evolving harnesses and models: On-policy correction helps weaker models catch up where imitation fails, 2026. URL https://arxiv.org/abs/2609.09134.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves, 2026. URL https://arxiv.org/abs/2606.09498.

The appendix is organised as follows. Appendix A details the benchmarks, harnesses, and evaluation protocol; Appendix B the training configurations and pools; Appendix C the rehearsal construction and its controls; Appendix D the experiment on a harness absent from every distillation pool; and Appendix E the full result tables.

A Benchmarks, Harnesses, and Evaluation

A.1 Benchmark Splits and Evaluation

Table A1 summarises the splits and the evaluation protocol of each benchmark.

Task splits. MAB uses 209 training and 61 test instances; the training split contains 215 instances before filtering. RG uses 96 generator families, with seeds 0–1 yielding 192 training instances and seed 2 yielding 96 test instances per difficulty configuration. Easy and hard tiers therefore test fresh instances of the same families. BFCL splits use seed 20260907. For single-turn, capping each of 13 categories at 150 examples reduces 3,641 examples to 1,406, split into 1,051 training and 355 test examples. Multi-turn uses 800 examples from four categories, split into 600 training and 200 test examples. AppWorld uses the unmodified 94-task training and 168-task `test_normal` splits (56 scenarios, three variants each); `test_challenge` is unused.

Table A1: Benchmark splits and evaluation protocols (n: attempts per task, T: temperature). Sampled evaluation uses four attempts per task; BFCL single-turn uses one greedy attempt. Reasoning Gym tests fresh seeds from the training generator families. BFCL multi-turn pools the four v3 multi-turn categories (BASE, MISS_FUNC, MISS_PARAM, LONG_CONTEXT), 200 tasks each, split 600/200, so the test split holds 50 tasks per category.

Benchmark	Train	Test	Main evaluation
MedAgentBench	209	61	$n = 4, T = 0.7$
Reasoning Gym (per tier)	192	96	$n = 4, T = 0.7$
AppWorld	94	168	$n = 4, T = 0.7$
BFCL single-turn	1,051	355	$n = 1, T = 0$
BFCL multi-turn	600	200	$n = 4, T = 0.7$

Scores. We report mean success (%). On BFCL-MT and AppWorld we average trials within each task first, so every task counts equally. On MAB and RG we average over all trials pooled. Sampled evaluation uses four attempts at T=0.7, greedy uses one attempt at T=0. Analyses of which tasks a harness solves use at-least-once success. BFCL/AppWorld also report the standard unbiased pass@4 estimator, over tasks with at least four attempts.

Evaluation details for supplementary comparisons. The full MAB tables use 61 test tasks, four attempts per task, and $T = 0.7$. Completed trials per cell range from 161–244 for Gemma and 105–244 for Qwen. Gemma serving explicitly selects each harness's template: checkpoint default for PARITY/XML, tool-call template for FC, and an alternation-tolerant template for Terminus-2. The Gemma RG tables use 96 easy-tier tasks, four attempts at $T = 0.7$, and 368–384 completed trials per cell; it includes neither hard-tier nor GRPO runs. The MAB volume controls use the same 61-task, four-attempt protocol.

Uncertainty. We use a nonparametric percentile bootstrap with 1,000 replicates and seed 20260923, reporting the 2.5th and 97.5th percentiles; printed $\pm$ values denote half the interval width. Resampling units are tasks, except in AppWorld, where we resample the 56 scenarios. BFCL single-turn is reported without bootstrap intervals. Exact-zero cells omit intervals. For BFCL-MT and AppWorld, reported means and bootstrap estimates use the same task-level averaging. Task-level and pooled means coincide when attempt counts are equal and can differ when they are unequal. For paired MAB/RG comparisons, we resample per-task differences in success rates using 10,000 replicates and the same seed; two-sided p-values use the doubled, add-one-smoothed sign-crossing fraction.

A.2 Harness Details

Table A2 covers the MedAgentBench and Reasoning Gym harnesses, and Table A3 the BFCL and AppWorld harnesses.

Table A2: **MedAgentBench and Reasoning Gym harnesses.** Custom adapters change the envelope over one action set (`GET`, `POST`, `FINISH`). The Reasoning Gym harnesses are released agents; each writes its answer to `/workspace/answer.txt`. Held-out harnesses are marked †.

Harness	Kind	Acts through	State across turns	History	Ends when
MedAgentBench					
PARITY	custom adapter	text: `GET`, `POST`, `FINISH(...)`	server state only	full, resent each turn	`FINISH`, 8 rounds, or 2 unparseable replies
FC	custom adapter	native tool calls	server state only	full	`finish` call or 8 rounds
Terminus-2	released (2.0.0)	terminal: JSON commands run in a shell	shell and filesystem	agent-managed, summarised	agent marks complete, or budget ends
XML†	custom adapter	XML tags	server state only	full	`<finish>`, 8 rounds, or 2 unparseable replies
Reasoning Gym					
Terminus-2	released (2.0.0)	terminal: JSON commands run in a shell	shell and filesystem	agent-managed, summarised	agent stops or budget ends
mini-swe	released (2.4.6)	shell commands in fenced blocks	shell and filesystem	agent-managed	agent stops or budget ends
Goose	released (1.51.0)	native tool calls, including shell and file editing	session and filesystem	agent-managed	agent stops or budget ends
Aider†	released (0.86.2)	the agent's edit and command interface	filesystem, repository map	agent-managed	agent stops or budget ends

What differs between harnesses. The harnesses fall into three kinds. (i) *Released agent frameworks*, used as provided: Terminus-2 (version 2.0.0), mini-swe-agent (2.4.6 on Reasoning Gym; the AppWorld version is not recorded), Goose (1.51.0) and Aider (0.86.2). Each brings its own system prompt, action format, observation format, history handling and budgets, and the model acts through a shell or file interface. (ii) *Custom adapters that change the envelope only*: PARITY, FC, JSON, PYCALL and XML. For a given benchmark they share the task text, the set of functions, the environment, the verifier and the budgets, and differ in system prompt, action syntax and how observations are phrased. They are selected by an argument to a single code path, so task construction, execution and scoring are identical across them. (iii) The AppWorld CODE, FC and XML adapters, which wrap the same persistent Python session. CODE takes a fenced code block, FC a single `execute_python` tool call, and XML tagged code; the action space is the same in all three.

All four Reasoning Gym harnesses are released agents. On BFCL multi-turn all four harnesses are custom envelopes. MedAgentBench, BFCL single-turn and AppWorld each combine custom envelopes with one released agent (Terminus-2, Terminus-2 and mini-swe-agent). PYCALL is a text format: the adapter parses python-style call lines and invokes the named functions, and does not run an interpreter. This is unlike AppWorld CODE, which executes Python.

B Training Details

This appendix gives the supervised and reinforcement-learning configurations, the training pools, and the experiment frameworks summarised in §4.

Supervised fine-tuning. All arms use one script and one configuration; the pool is the only variable. We fine-tune the base model for 2 epochs with learning rate 10^{-5}, per-device batch size 1 with gradient accumulation 8, sequence length 32,768, Liger kernels, and 8-bit paged AdamW, on a single 80 GB GPU per run. Qwen students train under the model's own chat template; Gemma students additionally require a tool-call-capable template for FC-bearing pools (§4). No arm receives extra epochs, data, or tuning: an arm that trains on 365 rows sees exactly the same optimisation budget as every other 365-row arm. SFT uses cosine decay with 10 warmup steps and bf16 precision.

Framework. Every task is a containerised Harbor task directory: a declarative config, a natural-language instruction, an environment image, and a verifier that writes a binary reward inside the container. Harvesting, RL rollouts, and evaluation run the same task directories, harness implementations, and verifiers through one execution path, so a cross-harness difference cannot be an artefact of two divergent implementations of the same interface; rehearsal instances traverse byte-identical machinery with only the instruction replaced (Appendix C). For RL, the SkyRL trainer is paired

Table A3: **BFCL and AppWorld harnesses.** On BFCL multi-turn the API instances are stateful and persist within a trial. AppWorld CODE, FC and XML run the same persistent Python session. Held-out harnesses are marked †.

Harness	Kind	Acts through	State across turns	History	Ends when
BFCL single-turn					
FC	custom adapter	native tool calls	none	one exchange	after one response
JSON	custom adapter	JSON object of call strings	none	one exchange	after one response
Terminus-2	released (2.0.0)	terminal: shell commands	shell and filesystem	agent-managed	completion, or 8 episodes
XML†	custom adapter	XML-tagged calls	none	one exchange	after one response
BFCL multi-turn					
FC	custom adapter	native tool calls	stateful API instances	full	reply with no calls, or all user turns done
JSON	custom adapter	`{"calls": [...]}`	stateful API instances	full	same
PYCALL	custom adapter	python-style call lines in a fenced block (parsed, not executed)	stateful API instances	full	same
XML†	custom adapter	`<calls>`/`<call>` tags	stateful API instances	full	same
AppWorld					
CODE	custom adapter	Python in a fenced block, run in an IPython session	session and world state	full	`complete_task`, 50 steps, or 40 errors
FC	custom adapter	one `execute_python` tool call, same session	session and world state	full	same
mini-swe	released (version not recorded)	the framework's shell interface	session and world state	framework's own	framework's own
XML†	custom adapter	`<reasoning>`/`<code>` tags, same session	session and world state	full	same

with the Harbor rollout engine: the agent inside each container queries the current policy through a colocated inference engine, and updated weights are broadcast between steps.

B.1 Collection and Training Pools

Collection. We use Qwen3-32B as the teacher and Qwen3-4B with thinking enabled and its default chat template as the student. Demonstrations use the evaluation harness implementations and verifiers; ordinary distillation retains successful trajectories only. MAB retains at most two trajectories per task and harness. MAB collection uses six attempts per training task; RG uses four at $T = 0.6$. BFCL collection uses $T = 1.0$ (one attempt for single-turn and two per collection pass for multi-turn); AppWorld uses four attempts at $T = 0.6$. Rehearsal uses the 3H SFT student as demonstrator, except for the explicit Qwen3-32B rehearsal comparison. Generation sampling is unseeded; MAB/RG SFT and RL also use no fixed seed. Pool artefacts and per-task outcomes specify the realised experiments; split, bootstrap, and subsampling seeds control the corresponding deterministic analyses.

Training pools. Table A4 reports retained trajectories and task coverage. Full mixtures concatenate specialist pools; volume-matched controls are separate experiments. MAB controls sample equal per-harness quotas at 108 or 150 rows (36×3 or 50×3), using random draws or coverage-maximised selection (Table A5). The AppWorld volume-matched mixture contains 74 trajectories covering 43 tasks.

Volume-matched mixtures. Table A5 subsamples the 365-row MAB mixture with equal per-harness quotas: 36×3 or 50×3 traces. Random subsampling at 108 rows was repeated over three seeds, of which the table reports the first; coverage-maximised sampling spreads the budget over as many of the 209 training tasks as possible. The matched specialist is PARITY at 108 rows and FC at 150 rows.

Rehearsal and controls. Rehearsal instructions are constructed programmatically from training-instance solutions and metadata, preserving the environment and verifier (Appendix C). MAB replaces 107 Terminus-2 trajectories or 150 FC trajectories one-for-one; RG replaces 120 Terminus-2 trajectories with transcription rehearsal. The resulting full pools remain at 365 rows (MAB) or 389

Table A4: Size of each distillation pool as retained trajectories / distinct training tasks. Columns (a) to (c) are the single-harness pools in the order given, and the last column is all three combined.

Benchmark	Harness order (a/b/c)	(a)	(b)	(c)	Mixture
MedAgentBench	PARITY / Terminus-2 / FC	108/54	107/77	150/112	365/167
Reasoning Gym	Terminus-2 / mini-swe / Goose	120	114	155	389
AppWorld	CODE / FC / mini-swe	74/46	45/32	89/52	208/60
BFCL single-turn	FC / JSON / Terminus-2	929/915	901/892	849/843	2,679/952
BFCL multi-turn	FC / JSON / PYCALL	593/374	386/279	416/282	1,395/431

Reasoning Gym lists trajectory counts only; distinct-task counts are unavailable.

Table A5: **Part of the 3H gain is data volume, and part of the volume effect is task coverage.** MedAgentBench mixtures subsampled to the row count of the PARITY or FC specialist. The matched mixtures stay below the full 3H pool on PARITY and FC, and the same row budget covers different numbers of tasks. Coverage-maximised mixtures improve on the matched specialist everywhere except the specialist's own FC harness at 150 rows, and every mixture keeps the other harnesses well above the specialists. Values are success (%) with 95% task-bootstrap CIs.

Pool	rows	tasks	PARITY	Terminus-2	FC
none (base)	—	—	36.1±9.2	3.7±2.3	24.3±8.4
3H (full)	365	167	67.2±10.2	16.9±5.9	55.3±8.2
PARITY-only	108	54	47.3±11.7	4.1±2.7	21.8±8.6
3H@108, random draw	108	81	43.8±8.7	17.8±5.5	36.9±7.2
3H@108, coverage-max	108	108	54.5±8.8	15.9±5.3	34.7±8.5
FC-only	150	112	28.7±11.3	0.0	45.1±9.4
3H@150, random draw	150	100	63.2±9.6	12.4±5.0	35.0±8.5
3H@150, coverage-max	150	139	52.3±9.6	17.6±6.3	35.1±8.1

rows (RG). Table A6 gives the component and combined pools. Appendix C.3 compares MAB demonstrators.

BFCL-MT uses a different construction: the task-supervision pool contains JSON and PYCALL demonstrations, excluding the target FC interface. Rehearsal supplies mechanical instructions whose target values must be obtained by executing calls and reading their results, rather than copied from the instruction. The primary combination appends 268 verified harness-rehearsal traces to the task-supervision pool; it is not a size-matched replacement. The success/failure control appends 33 successful or 33 reward-zero rehearsal traces, giving two 835-row pools. The failed-trace supply limits this comparison to 33 rows. A separate XML-rehearsal arm appends 186 traces to the full mixture; XML is therefore seen during training in this arm (Appendix D). BFCL rehearsal pools cap retained traces at two per drill.

B.2 Reinforcement-Learning Configuration

We apply GRPO to MAB and RG, comparing 3H rollouts with single-harness controls at matched total rollout counts. Within each condition the optimisation recipe is fixed. In all cases, the reward is the unmodified verifier's binary task success, and, deliberately, there is no shaping for trajectory length, turn count, or termination. Hyperparameters are given in Table A7, and Table A8 gives the matched rollout budgets; RL is reported for MAB and RG only. The MAB and RG runs both start from the 3H checkpoint. Checkpoints are exported every 5 steps, and both an intermediate (step-5) and the final policy are evaluated under the standard protocol, so RL effects are read as changes in the same task–harness cells as SFT effects.

Learning bands. We restrict RL to a *learning band*: a pre-computed subset of training tasks with non-zero observed success under the initial checkpoint. This focuses rollout compute on tasks for which the initial policy has demonstrated some success. Tasks with observed success rate one remain eligible; a sampled group contributes zero GRPO advantage whenever all its rewards are identical.

Table A6: Rehearsal pools and their controls, as trajectories and distinct training tasks. Failed-rehearsal pools contain traces that did not pass the verifier.

Benchmark	Pool	Trajectories	Tasks
MedAgentBench	PARITY+FC / PARITY+Terminus-2	258 / 215	138 / 98
	rehearsal / failed rehearsal	107 / 107	79 / 70
	PARITY+FC + rehearsal / 32B rehearsal	365 / 365	168 / 169
	FC rehearsal / PARITY+Terminus-2 + FC rehearsal	150 / 365	84 / 160
Reasoning Gym	mini-swe+Goose / rehearsal	269 / 120	—
	mini-swe+Goose + rehearsal	389	—
BFCL multi-turn	task supervision / harness rehearsal	802 / 268	368 / 135
	task supervision + harness rehearsal	1,070	503
	task supervision + 33 successful / failed traces	835 / 835	385 / 390
	3H + XML rehearsal	1,581	537

Slash-separated counts follow pool order. Reasoning Gym distinct-task counts are unavailable.

Table A7: Training settings shared within each comparison. SFT and GRPO optimisation settings, sequence budgets, and hardware. RL epochs vary between single- and multi-harness conditions to match rollout counts.

Supervised fine-tuning		GRPO	
Setting	Value	Setting	Value
epochs	2	advantage estimator	GRPO
learning rate	1×10^{-5}	learning rate	1×10^{-6}
batch / grad. accum.	1 / 8	epochs (multi / single)	1 / 3
sequence length	32,768	train batch (prompts)	24
optimiser	paged AdamW (8-bit)	samples per prompt	6
kernels	Liger	policy mini-batch	24
precision/parallel	single GPU	micro-batch / GPU	1
		KL penalty	none
		loss reduction	token-mean
		std. normalisation	off
		off-policy correction	token-level TIS, clip 2.0
		overlong filtering	on
		context / generation	24,576 / 8,192
		trainer strategy	FSDP, CPU offload
		placement	colocated, 2 GPUs
		inference engine	vLLM, TP= 2, util. 0.75
		weight sync	NCCL
		rollout concurrency	24 trials
		checkpoint interval	5 steps

Sequence budgets are in tokens. RL epochs vary to match rollout counts (Table A8).

The MAB band contains 96 tasks and the RG band 114; both are drawn from the training split only and reused across runs.

C Rehearsal Details and Controls

All rehearsal instances below are reproduced *verbatim* from the training artefacts. Construction invariants, common to every rehearsal: the task's environment, verifier, and solution are copied byte-identical from a training-split instance; only the instruction is replaced by a mechanical one that hands over the solution. Rehearsal *trajectories* are never hand-written: the demonstrator executes these instructions through the unmodified harness, and only trajectories that pass the unmodified verifier enter the pool.

Table A8: **RL budgets.** Single- and multi-harness runs get the same number of rollouts, so single-harness runs make three passes over the tasks and three-harness runs make one.

Condition	Tasks × harnesses	Epochs	Steps	Rollouts
MedAgentBench, 3H	96×3	1	12	1,728
MedAgentBench, parity-only	96×1	3	12	1,728
Reasoning Gym, 3H	114×3	1	14	2,052
Reasoning Gym, mini-swe-only	114×1	3	12–15	2,052

Each batch has 24 task–harness prompts, with six samples each. Reasoning Gym step counts reflect partial batches.

C.1 MEDAGENTBENCH: THE SAME REHEARSAL INSTANCE UNDER TWO ENVELOPES

MAB rehearsal instantiates four templates (task2/3/4/8) over 96 training instances. We show one *read* rehearsal (task4: fetch, filter, transcribe a lab value) in both its Terminus-2 (terminal) and FC (function-calling) phrasings (the same underlying instance, so the pair exhibits the envelope manipulation exactly), and one *write* rehearsal (task3: POST a fixed payload). A shared benchmark boilerplate tail (`## Environment` / `## Required Output`, present in every MAB instruction including native evaluation) is elided here for space.

Read rehearsal, Terminus-2 envelope (`task4_1`, terminal harness):

```
# Terminal Procedure Rehearsal `task4_1`

## Procedure
Execute the following steps exactly, in order. Everything you need is
specified below; read command output carefully and transcribe values exactly.

1. Run: `medagentbench_cli.py get "http://localhost:8080/fhir/Observation
   ?patient=S3032536&code=MG&_count=5000&_format=json"`
2. In the returned JSON, consider only entries whose
   `resource.effectiveDateTime` is at or after `2023-11-12T10:15:00+00:00`.
   Among those, find the entry with the LATEST `resource.effectiveDateTime`
   and read V = its `resource.valueQuantity.value` (copy the number exactly
   as it appears).
3. If there is no such entry, run: `medagentbench_cli.py finish '[-1]'`
   Otherwise, run: `medagentbench_cli.py finish '[V]'`
```

Read rehearsal, FC envelope (same instance, tool-call harness):

```
# Tool-Call Procedure Rehearsal `task4_1`

## Procedure
Execute the following steps exactly, in order, using the provided tools.
Everything you need is specified below; read tool output carefully and
transcribe values exactly.

1. Call the `get` tool with url `http://localhost:8080/fhir/Observation
   ?patient=S3032536&code=MG&_count=5000&_format=json`.
2. In the returned JSON, consider only entries whose
   `resource.effectiveDateTime` is at or after `2023-11-12T10:15:00+00:00`.
   Among those, find the entry with the LATEST `resource.effectiveDateTime`
   and read V = its `resource.valueQuantity.value` (copy the number exactly
   as it appears).
3. If there is no such entry, call the `finish` tool with answer `[-1]`.
   Otherwise, call the `finish` tool with answer `[V]`.
```

Write rehearsal, FC envelope (`task3_1`; the payload is this training instance's own answer, useless on any evaluation instance):

```
1. Call the `post` tool with url `http://localhost:8080/fhir/Observation`
   and payload exactly:
   `{"resourceType": "Observation", "category": [{"coding": [{"system":
   "http://hl7.org/fhir/observation-category", "code": "vital-signs",
   "display": "Vital Signs"}]}], "code": {"text": "BP"},
   "effectiveDateTime": "2023-11-13T10:15:00+00:00", "status": "final",
   "valueString": "118/77 mmHg",
```

```
   "subject": {"reference": "Patient/S2380121"}}`
2. Call the `finish` tool with answer `["DONE"]`.
```

What the model must still do, and what is absent: success requires issuing the call through the harness's native action format, reading the *environment's* response (the lab value in the read rehearsal exists only on the server, and the write rehearsal is verified against server state; an agent that skips to `finish` scores 0 and is filtered), transcribing exactly, and terminating in the harness's required idiom. No clinical rule, threshold, or decision logic appears anywhere; the only content shared with evaluation is template-level payload schema, whose contribution is bounded by the detector measurements of §5.3 (rehearsed templates win $\approx$0 evaluations; 55% of wins land on never-rehearsed templates).

C.2 REASONING-GYM: TRANSCRIPTION REHEARSAL

RG rehearsal is deliberately more extreme: pure transcription, with no query, no schema, and no reusable surface of any kind. The answer string itself is handed over, so *everything* being scored is harness operation (shell quoting, exact file writing, verification, termination):

```
Transcription rehearsal: your only task is to write an exact string to a file.

Write EXACTLY the following content (between the BEGIN/END markers, not
including the marker lines themselves) to the file "/workspace/answer.txt":

BEGIN
-3.0 - 3.0i
END

Preserve every character, including signs, decimals, spaces and line breaks.
Use careful shell quoting (e.g. cat with a quoted heredoc) so nothing is
interpreted by the shell. Verify the file content matches before finishing.
```

The handed-over answer (here `-3.0 - 3.0i`) is the solution to one *training-seed* instance of a procedurally generated family; evaluation uses a disjoint seed of the same family, for which this string is worthless. Because RG rehearsal carries no schema channel at all, the same recombination effect appearing there (Table 4) rules out schema leakage as rehearsal's active ingredient.

C.3 DEMONSTRATOR PROCESS QUALITY

The two MAB rehearsal pools of Table 4 contain the *identical* 107 rehearsal instructions, harvested through the identical unmodified harness and verifier; they differ only in which model executed them: the 4B-family demonstrator (a 3-harness SFT checkpoint) or the 32B teacher. Because rejection sampling filters *outcomes* while students imitate *process*, the two pools nevertheless differ sharply as training data: the 4B demonstrator wanders for a median of 17 assistant turns per verified rehearsal, on instructions that specify 2–4 actions and hand it the answer, while the 32B teacher executes at the length the instruction specifies.

Table A9: **Verified rehearsal traces differ in execution length.** Two demonstrators execute the same 107 rehearsal instructions through the same harness and verifier. Statistics describe assistant turns and characters per verified trajectory.

	4B-demonstrator rehearsal	32B-teacher rehearsal
mean assistant turns	34.7	3.6
median / p90 / max turns	17 / 69 / 287	4 / 5 / 6
mean assistant characters	30,870	13,622

Process quality is a cross-channel property. Holding the task-supervision rows fixed (PARITY+FC, all 32B-authored), swapping only the rehearsal demonstrator flips the effect on the untouched PARITY channel from -11.0 ($63.5 \to 52.5$) to $+6.7$ ($63.5 \to 70.2$, the best PARITY of any checkpoint, above 3H's 67.2). Since rehearsal content is identical and the leak detectors bound task-content transfer at noise level (§5.3), the 17.7-point swing is attributable to trace *process* alone.

Table A10: **Rehearsal raises an unseen harness without moving the trained ones.** BFCL multi-turn, success (%) with 95% task-bootstrap CIs on 200 test tasks, four attempts each (800 trials per cell; 799 for 3H on FC). XML is in no distillation pool; the last arm adds 186 XML rehearsal traces to the 3H pool. Δ is treatment minus control.

Training pool	XML (target)	FC	JSON	PYCALL
none (base)	20.6±3.8	44.6±5.9	33.5±5.4	28.2±5.1
3H (control)	19.9±4.2	59.0±5.5	49.5±5.4	49.1±5.6
3H + XML rehearsal	**36.6±4.8**	60.2±5.8	49.9±5.2	48.9±5.9
Δ vs control	+16.7	+1.2	+0.4	−0.2

Our reading: rehearsal trajectories carry channel-general process skills (termination discipline, read-then-act turn structure) alongside the channel-specific format, and PARITY, with its 8-round budget, prices termination most heavily of the three harnesses. A pool of 35-turn flailing recoveries teaches persistence over termination and taxes PARITY; a pool of $\leq$6-turn crisp executions concentrates exactly the micro-skill PARITY rewards, more concentrated per token than native Terminus-2 traces, whose episodes run to a 50-action budget.

D Rehearsal on a Harness Absent from Every Distillation Pool

This experiment tests rehearsal on a harness that none of the distillation pools contains. The target is XML on BFCL multi-turn, which is held out in every main comparison (Table A3). We train two models that differ in one thing only: the 3H pool alone (1,395 rows), which is the control, and the 3H pool plus 186 verified XML rehearsal traces (1,581 rows). The pools are otherwise identical, including 19 over-length rows that we left in both so that they cancel and do not become a second difference. XML therefore appears in no task-solving pool of either arm, and in the treatment arm only through the rehearsal rows.

Data. BFCL multi-turn here merges the four categories of the benchmark (base, missing function, missing parameter, long context), 200 tasks each, split 600 for training and 200 for testing with seed 20260907. The rehearsal traces come from 138 drill tasks that passed quality control, the same set used for the FC rehearsal (Table A18). The demonstrator is the 3H student (T=1.0, four attempts per drill). We keep verifier-passing traces only and cap them at two per drill: 280 of 551 attempts pass (50.8%), covering 106 of the 138 drills, which leaves 186 rows.

Training and evaluation. Both arms use the recipe of the main experiments (Qwen3-4B with thinking enabled and its default chat template, two epochs, 32,768-token sequences, 8-bit paged AdamW, Liger kernels), with hyperparameters copied unchanged from the control. Evaluation uses the full 200-task test split with four attempts per task at T=0.7, giving 800 trials per cell, and task-level bootstrap intervals (Appendix A.1). Each arm is a single training run with no seed replication. The four attempts per task differ only in generation sampling, and the intervals reflect sampling of tasks for a fixed checkpoint, not variation between training runs.

Result. Distilling on three harnesses leaves XML where the base model was (19.9 against 20.6). Adding 186 rehearsal rows raises it to 36.6, and the 95% intervals do not overlap ($[31.9, 41.4]$ against $[15.6, 24.1]$). The three trained harnesses move by at most 1.2 points, well within their intervals, so the gain is specific to the target interface and not a general capability change.

E Extended Results

E.1 Tasks Solved Across Harnesses and Models

Table A11 extends Section 5.1 to every model and benchmark we evaluated. A task counts as solved through a harness if at least one attempt passes (pass@4); Qwen3-32B on BFCL multi-turn and AppWorld has a single greedy attempt (Table A12). *Others add* is the number of tasks solved

Table A11: **Tasks reachable only through a harness other than the strongest one.** Solved by any: tasks solved through at least one of the three trained harnesses. Strongest harness: the one solving most tasks, with its count. †: four harnesses, including the held-out one.

Benchmark	Model	Attempts	Tasks	Solved by any	Strongest harness	Others add
MedAgentBench	Qwen3-4B	pass@4	61	46	PARITY (38)	+8 (17%)
	Qwen3-32B	pass@4	61	53	PARITY (50)	+3 (6%)
	Gemma-3-4B†	pass@4	61	23	PARITY (16)	+7 (30%)
Reasoning Gym (easy)	Qwen3-4B	pass@4	96	75	mini-swe (71)	+4 (5%)
	Qwen3-32B	pass@4	96	82	mini-swe (77)	+5 (6%)
	Gemma-3-4B†	pass@4	96	31	Terminus-2 (23)	+8 (26%)
Reasoning Gym (hard)	Qwen3-4B	pass@4	96	61	mini-swe (58)	+3 (5%)
	Qwen3-32B	pass@4	96	79	mini-swe (74)	+5 (6%)
BFCL multi-turn	Qwen3-4B	pass@4	200	141	FC (125)	+16 (11%)
	Qwen3-32B	greedy	200	131	FC (116)	+15 (11%)
AppWorld	Qwen3-4B	pass@4	168	16	FC (14)	+2 (12%)
	Qwen3-32B	greedy	168	89	mini-swe (49)	+40 (45%)

Table A12: **The distillation teacher remains sensitive to harness choice.** Qwen3-32B evaluated on each benchmark's full test split with one greedy attempt per task. Values are success (%); harness identities and held-out status are given below.

Benchmark	harness			
	(a)	(b)	(c)	held-out*
AppWorld (test-168)	28.0	26.2	29.2	25.0
BFCL multi-turn (test-200)	58.0	33.0	39.5	—
BFCL single-turn (test-355)	88.5	87.0	—	87.9

: excluded from student distillation. Harness order (a)/(b)/(c)/held-out. AppWorld: CODE/FC/mini-swe/XML; BFCL multi-turn: FC/JSON/PYCALL/XML*; BFCL single-turn: FC/JSON/Terminus-2/XML*. All cells are greedy (T=0), one attempt per task, on the full test split.

through some harness but not through the strongest one, with its share of all tasks solved through any harness. Qwen3-4B solves only 16 AppWorld tasks, so its row is not informative.

E.2 Distillation and Held-Out Transfer

Table A13 completes Table 2, and Table A14 gives the full held-out scores behind Figure 4.

E.3 Rehearsal: Full Tables and Replications

Tables A15 and A16 report every MedAgentBench condition for Qwen3-4B and Gemma-3-4B-it, Table A17 the Gemma replication on Reasoning Gym, and Table A18 the full BFCL multi-turn rehearsal comparison.

E.4 Reinforcement Learning

Table A19 reports every SFT-to-GRPO change behind §5.4.

Table A13: **Single-harness specialists and 3H on BFCL single-turn and Reasoning Gym Easy.** Same format as Table 2: rows are training pools and columns are evaluation harnesses. Values are mean success (%) with 95% bootstrap CIs, arrows show change from the base model and bold marks the best pool per column. Four attempts per task; BFCL single-turn is one greedy run.

BFCL single-turn				**Reasoning Gym Easy**			
Pool	FC	JSON	Terminus-2	Pool	Terminus-2	mini-swe	Goose
none (base)	**87.0**	86.2	22.5	none (base)	18.0±5.5	59.4±7.9	25.7±6.7
FC-only	85.1 ↓	21.4 ↓	0.0 ↓	Terminus-2-only	54.7±7.8 ↑	55.7±8.4 ↓	40.2±7.3 ↑
JSON-only	85.1 ↓	**87.9** ↑	1.5 ↓	mini-swe-only	18.4±4.4 ↑	44.2±7.7 ↓	39.8±7.4 ↑
Terminus-2-only	21.4 ↓	83.9 ↓	42.5 ↑	Goose-only	37.3±6.7 ↑	51.5±7.7 ↓	**64.6±8.1** ↑
3H	86.8 ↓	86.2 =	**43.2** ↑	3H	**55.3±7.9** ↑	**63.0±7.6** ↑	60.8±8.2 ↑

Table A14: **Held-out transfer depends on the benchmark.** Base policies, single-harness specialists, and mixtures evaluated on an interface excluded from training. Values are success (%) with 95% CIs where reported; arrows indicate changes from base.

	held-out		single-harness			
Benchmark	harness	base	(a)	(b)	(c)	3H
MedAgentBench	XML	49.0±11.1	43.8±10.5 ↓	32.9±8.0 ↓	58.4±7.2 ↑	**71.5±10.2** ↑
AppWorld	XML	4.5±3.8	9.5±4.3 ↑	3.3±2.2 ↓	14.1±5.0 ↑	**21.3±7.6** ↑
BFCL multi-turn	XML	20.6±3.8	2.5±1.1 ↓	**20.0±4.1** ↓	18.8±3.9 ↓	19.9±4.2 ↓
BFCL single-turn	XML	86.2	21.7 ↓	**88.5** ↑	21.4 ↓	87.0 ↑
Reasoning Gym Easy	Aider	20.8±6.8	2.4±2.3 ↓	0.0 ↓	0.0 ↓	**19.0±6.0** ↓
Reasoning Gym Hard	Aider	17.6±6.8	0.5±0.7 ↓	0.0 ↓	0.2±0.2 ↓	**9.8±4.8** ↓

Specialist order (a/b/c): MedAgentBench, PARITY/Terminus-2/FC; AppWorld, CODE/FC/mini-swe; BFCL multi-turn, FC/JSON/PYCALL; BFCL single-turn, FC/JSON/Terminus-2; Reasoning Gym, Terminus-2/mini-swe/Goose. Bold compares trained policies.

Table A15: **Qwen gains on MedAgentBench depend on both the training pool and the evaluation harness.** Success (%) with 95% task-bootstrap CIs across distillation, rehearsal, and RL conditions. XML is held out; bold marks the highest mean per column and dashes mark unevaluated cells.

Pool	PARITY	Terminus-2	FC	XML (held out)
none (base)	36.1±9.2	3.7±2.3	24.3±8.4	49.0±11.1
PARITY-only	47.3±11.7	4.1±2.7	21.8±8.6	43.8±10.5
Terminus-2-only	26.4±7.6	21.7±6.4	16.1±5.2	32.9±8.0
FC-only	28.7±11.3	0.0	45.1±9.4	58.4±7.2
PARITY+FC-only	63.5±10.0	2.0±1.6	47.1±10.0	63.9±9.0
rehearsal-only	40.4±10.2	9.5±5.7	38.5±10.9	38.3±8.1
rehearsal-fail	40.9±11.0	2.1±1.8	32.5±10.5	37.0±9.0
PARITY+FC+rehearsal	52.5±9.0	34.4±8.2†	57.0±9.0	49.0±8.6
PARITY+FC+32B-rehearsal	**70.2±9.5**	21.7±7.6	50.4±8.6	63.8±10.8
PARITY+Terminus-2-only	46.7±9.8	18.0±6.8	25.0±8.4	47.1±10.9
FC-rehearsal-only	26.6±11.3	0.0‡	39.6±9.6	39.8±9.4
PARITY+Terminus-2+FC-rehearsal	61.3±10.2	23.0±8.2	36.6±9.8	42.6±9.6
3H	67.2±10.2	16.9±5.9	55.3±8.2	71.5±10.0
3H+GRPO	65.8±9.4	39.7±9.6	**58.4±8.1**	**72.9±9.0**

†: eight-attempt replication, 34.2%. ‡: early termination after 105 trials.

Table A16: **Gemma reproduces cross-harness interference and recovery on MedAgentBench.** Gemma-3-4B-it evaluated across distillation, rehearsal, and RL conditions. Values are MedAgentBench success (%) with 95% task-bootstrap CIs; XML is held out and bold marks the highest mean per column.

Pool	PARITY	Terminus-2	FC	XML (held out)
none (base)	20.6‡	3.7±2.7	26.2±11.5	15.2±8.0
PARITY-only	25.7±11.5	0.0	26.5±11.5	23.0±8.8
Terminus-2-only	26.2±11.5	21.1±7.6	26.7±11.5	14.2±6.0
FC-only	26.2±11.5	0.0	58.8±9.2	35.0±8.2
rehearsal-only	26.1±11.5	7.8±4.1	24.8±11.5	20.2±8.6
rehearsal-fail	20.0±9.2	2.5±2.3	25.7±11.5	16.2±7.2
PARITY+rehearsal	26.2±11.5	14.1±6.1	25.2±11.5	23.4±8.2
PARITY+Terminus-2	**27.2±11.5**	20.5±7.2	25.9±11.5	22.6±7.5
FC-rehearsal-only	26.2±11.5	0.0	17.6±6.3	24.8±9.1
PARITY+Terminus-2+FC-rehearsal	26.2±11.5	**28.8±8.3**	42.6±8.1	40.7±9.4
3H	26.8±11.5	20.2±6.6	**73.5±8.0**	44.3±9.4
3H+GRPO	26.2±11.5	28.0±8.2	70.2±7.8	**50.4±9.4**

‡: earlier single evaluation, without an interval.

Table A17: **Gemma improves on Reasoning Gym training harnesses while losing held-out performance.** Gemma-3-4B-it on Reasoning Gym Easy after single- or multi-harness distillation. Values are success (%) with 95% task-bootstrap CIs; Aider is excluded from training.

Pool	Terminus-2	mini-swe	Goose	Aider (held out)
none (base)	11.7±4.4	0.0	0.0	**16.4±6.5**
Terminus-2-only	26.1±7.0	0.0	0.0	8.3±5.1
mini-swe-only	0.8±0.8	30.1±7.5	0.0	1.0±1.6
Goose-only	3.2±1.8	1.6±1.8	0.8±0.9	1.0±1.6
3H	**27.6±6.5**	**32.0±7.5**	**19.9±5.8**	0.0

Table A18: **Harness rehearsal complements cross-harness task supervision on BFCL multi-turn.** Success (%) on the target FC interface and the two interfaces supplying task supervision, with 95% task-bootstrap CIs. Conditions reproduce the BFCL multi-turn block of Table 4.

Pool	FC (target)	JSON	PYCALL
base	44.6±5.9	33.5±5.4	28.2±5.1
task supervision only	8.5±2.8	47.4±5.8	**52.5±5.3**
harness rehearsal only	32.5±5.1	0.0	0.1±0.2
task supervision + harness rehearsal	**58.8±5.7**	**50.8±5.9**	50.9±5.0
native target-only	55.2±5.3	0.4±0.6	19.5±3.9
3H	59.0±5.5	49.5±5.4	49.1±5.6

Task supervision: JSON+PYCALL traces; harness rehearsal: 268 FC traces. Values and bolding follow Table 4.

Table A19: **GRPO gains vary across harnesses.** 3H SFT policies before and after 3H GRPO. Values are success (%) with 95% task-bootstrap CIs; shading marks held-out interfaces and Δ is the change in percentage points.

Setting	Harness	SFT	GRPO	Δ
MedAgentBench Qwen3-4B	PARITY	67.2±10.2	65.8±9.4	−1.4
	Terminus-2	16.9±5.9	39.7±9.6	+22.8
	FC	55.3±8.2	58.4±8.1	+3.1
	XML (held out)	71.5±10.0	72.9±9.0	+1.4
MedAgentBench Gemma-3-4B-it	PARITY	26.8±11.5	26.2±11.5	−0.6
	Terminus-2	20.2±6.6	28.0±8.3	+7.8
	FC	73.5±8.1	70.2±7.9	−3.3
	XML (held out)	44.3±9.4	50.4±9.8	+6.1
Reasoning Gym Easy Qwen3-4B	Terminus-2	55.3±7.9	62.8±7.8	+7.5
	mini-swe	63.0±7.6	65.0±7.7	+2.0
	Goose	60.8±8.1	63.9±8.7	+3.1
	Aider (held out)	19.0±6.0	21.4±6.7	+2.4
Reasoning Gym Hard Qwen3-4B	Terminus-2	43.3±7.8	43.8±8.3	+0.5
	mini-swe	51.0±8.0	52.8±8.1	+1.8
	Goose	46.0±8.2	45.7±8.2	−0.3
	Aider (held out)	9.8±4.8	12.3±5.7	+2.5