HARNESS ANNEALING: LEARNING TO ACT WITH LESS EXTERNAL CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Language agents rely on external harnesses to track state, organize workflows, and verify answers. Beyond providing tools and information, these harnesses supply control decisions about what to investigate, whether to revise, and when to stop. Training on successful harness-supported trajectories can improve task performance while leaving these decisions dependent on runtime intervention. We ask whether harness-supported experience can also teach the model to make these decisions, allowing the division of control to change as the model learns. We call this objective ***harness internalization***: learning to assume specified control responsibilities while retaining task performance after the corresponding support is withdrawn. We introduce **HARNESS ANNEALING TRAINING (HAT)**, which combines explicit control supervision with a curriculum over teacher trajectories collected under progressively weaker harnesses. Experiments with 9B and 35B models on SWE-QA and SWE-QA-Pro evaluate every checkpoint under four deployment harnesses. Selected annealed checkpoints operating with tools alone achieve scores close to those of their respective starting checkpoints deployed with the full harness. The benefits vary with model scale and deployment configuration, and further annealing does not uniformly improve performance. These findings suggest that harness-supported experience can help reduce the runtime control required by a trained agent.

1 INTRODUCTION

LLM agents typically consist of a backbone model and an external execution framework, commonly referred to as a *harness* (Yang et al., 2024; Zhang et al., 2026). The harness supports task execution by maintaining state, organizing workflows, and checking candidate answers (Chen et al., 2026). These components supply control decisions about what to investigate, whether further evidence is needed, and when to stop. Successful harness-supported interactions therefore provide experience from which a model can learn both task execution and the judgments that guide it. However, training on these interactions can improve task performance while leaving the model dependent on external intervention for those judgments. The model may thus continue to require the same runtime guidance even after learning from the experience it helped produce. If these control responsibilities can be learned, support that was useful for generating training experience may become unnecessary at deployment. This motivates a central question: ***can harness-supported experience teach the model to assume specified control responsibilities while preserving task performance as the corresponding support is withdrawn?***

We call the corresponding learning objective ***Harness Internalization***: learning to assume specified control responsibilities while maintaining task performance after the associated external support is withdrawn. We focus on state tracking, workflow organization, and answer verification; tools and execution infrastructure remain external. Figure 1 illustrates this objective through verification: after external checking is removed, the model must assess whether the available evidence supports its answer and continue exploration when necessary. However, continued exploration in a successful trajectory may simply be a response to a verifier rejecting the model's draft. Learning the actions that follow this intervention does not guarantee that the model will independently recognize the need for further investigation. Successful execution under the full harness is therefore insufficient to establish internalization. Evaluation must also examine task performance and control-related behavior after the corresponding support is removed.

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.

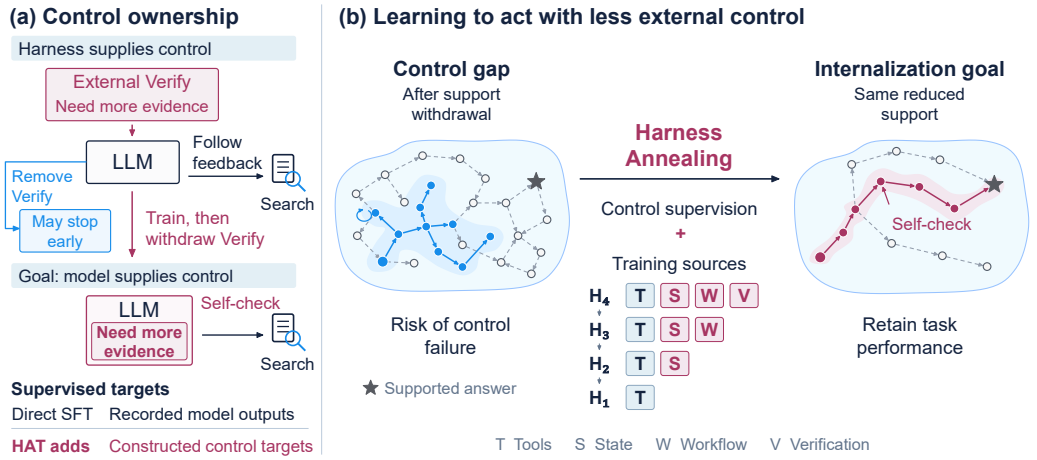

Figure 1: **Learning to retain task performance with less external control. (a) Control ownership.** Following verifier feedback does not ensure independent judgment after its removal; state, workflow, and tools remain. **(b) Harness annealing.** HAT combines explicit control supervision with a curriculum over progressively weaker source harnesses. Paths schematically contrast potential control gaps with intended behavior under the same reduced support.

Existing work improves model–harness collaboration in several ways. Adaptive harnesses optimize workflows, runtime code, or composable components to improve external support (Lee et al., 2026; Chen et al., 2026). Trajectory-based training instead updates the model using successful interactions (Chen et al., 2023; Zeng et al., 2023), but harness-generated prompts, states, and interventions typically remain conditioning inputs rather than explicit supervision targets. Related approaches, including skill distillation and workflow compilation, explore transferring external guidance into model weights. Our focus is the transfer of specified runtime control responsibilities, assessed through task performance and control-related behavior after support withdrawal. We study this transfer through a hierarchy of state, workflow, and verification support, evaluating each checkpoint both with and without the corresponding control layers.

We introduce **HARNESS ANNEALING TRAINING (HAT)**, which combines explicit control supervision with a stronger-to-weaker harness curriculum. Using a fixed bank of teacher trajectories, we retain interaction observations and teacher actions while constructing control annotations that distinguish externally supplied information from judgments the model must learn to produce. Starting from a checkpoint trained on full-harness trajectories, training progressively introduces records from source harnesses without verification, then workflow, and finally state support. Supervision covers both newly targeted responsibilities and those addressed at earlier stages, with replay and cue dropout supplementing the curriculum. We evaluate 9B and 35B models on SWE-QA and SWE-QA-Pro (Peng et al., 2026; Cai et al., 2026), testing every training checkpoint under all four deployment harnesses. Selected intermediate checkpoints achieve tools-only scores close to those of their respective starting checkpoints deployed with the full harness. For the 9B model, the H_2-targeted checkpoint improves tools-only scores over the H_4-trained starting checkpoint by 5.01 points on SWE-QA and 3.09 points on SWE-QA-Pro. The gains vary with model scale and deployment configuration, and further annealing does not uniformly improve performance.

Our contributions are threefold:

1. We formulate **Harness Internalization** as a learning objective: transferring specified process-control responsibilities to the model while retaining task performance after the corresponding external support is withdrawn. We assess progress through task performance and control-related behavior under predefined support removals.
2. We introduce **HARNESS ANNEALING TRAINING (HAT)**, which combines explicit control supervision with a curriculum over progressively weaker harness support, supplemented by replay and cue dropout, to train agents for deployment with less external guidance.
3. We evaluate the **full checkpoint-by-harness matrix** at two model scales on SWE-QA and SWE-QA-Pro, showing improvements under weak support and revealing how preferred checkpoints vary with model scale and deployment objective.

2 Related Work

Adapting the external harness. ADAS and Meta-Harness optimize agent programs, workflows, or runtime code (Hu et al., 2025; Lee et al., 2026). HarnessX extends this direction to composable processors and harness–model co-evolution, while reporting evaluation-shortcut exploitation during optimization (Chen et al., 2026). These approaches retain the optimized runtime at deployment. We instead use HarnessX's compositional substrate to define a fixed removal ladder and transfer selected control responsibilities to the policy.

Learning from interaction trajectories. FireAct and AgentTuning train agents on interaction trajectories (Chen et al., 2023; Zeng et al., 2023), while Agent-FLAN separates reasoning from format learning and ATLaS prioritizes critical decision steps (Chen et al., 2024; 2025). We focus on who supplies control decisions and which tokens receive supervision: harness guidance provided only as input is not directly supervised as an output. Our observation–reasoning–action records turn targeted control responsibilities into model outputs while retaining observed evidence as input. The Direct–Layered comparison evaluates this construction jointly, not control supervision alone, because segmentation also changes token weighting and optimizer exposure.

Distilling guidance and learning harness policies. Knowledge and policy distillation provide a teacher–student perspective (Hinton et al., 2015; Rusu et al., 2016). Prompt distillation, Skill-SD, and workflow compilation transfer document-, skill-, or workflow-based guidance into policies that operate without that guidance at inference (Kujanpää et al., 2024; Wang et al., 2026; Dennis et al., 2026). In contrast, Harness-1 retains environment-side search state (Jiang et al., 2026), and EvoHarness-RL learns selective access to Belief, Progress, and Experience, describing declining harness use as annealing (Ning et al., 2026). We combine explicit control-decision targets with predetermined layer removals and evaluate every checkpoint under every deployment harness, distinguishing reduced scaffold use from retained capability after its removal.

3 Method

3.1 Problem Formalization

3.1.1 Execution of Agentic Tasks

Given an environment E whose state is partially observable to the model, and an instruction q, a language model π_θ must interact with E over multiple steps through a tool set $Tools = \{T_1, \ldots, T_n\}$ to produce an outcome $\hat{y}$ (a predicted answer, patch or report). Each task can be regarded as a trace:

$$\tau = \big(s_0,\ a_1, o_1,\ a_2, o_2,\ \ldots,\ a_i, o_i,\ \hat{y}\big), \qquad a_t \sim \pi_\theta\big(\cdot \mid q,\ h_{t-1}\big), \quad o_t = E(a_t), \tag{1}$$

where s_0 is the initial environment state, a_t is usually an action in $Tools$, o_t is the observation returned by E, and $h_{t-1} = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ is the interaction history. Compared with single-turn generation, agentic tasks exhibit three structural properties that make them hard:

1. **Long horizon and partial observability.** The information required for solution is absent from the initial instruction and must be actively acquired by π_θ through a sequence of actions, each of which reveals only a local view of E.
2. **Sparse credit assignment.** Intermediate actions carry no labels; only the final outcome is evaluated. π_θ must therefore judge on its own whether the evidence gathered so far is sufficient to commit to an answer.
3. **Groundedness.** The predicted outcome $\hat{y}$ should be supported by relevant observed evidence. Final-answer correctness alone does not establish whether the agent gathered or used that evidence.

For a given question and reference answer, let $J(\tau)$ denote the score assigned to the final answer $\hat{y}$ in trajectory τ. We implement this score using a reference-based judge that receives the question, reference answer, and final answer (Appendix E). The judge does not receive intermediate observations and therefore does not directly assess whether $\hat{y}$ is grounded in the evidence gathered during the trajectory. We separately examine selected control behaviors using the diagnostics in Appendix F.

Table 1: Deployment hierarchy and targeted control responsibilities. Each level retains all preceding support; bold marks the added component. Tools, execution guards, and token budgets remain external throughout. The verifier has no access to the reference answer.

Nested configuration	Added runtime support	Targeted responsibility
$H_1 =$ **Tools**	Tool access and execution safeguards	Gather task evidence using external tools.
$H_2 = H_1 +$ **State**	Recent history, progress ledger, and repetition detection	Track evidence and gaps; avoid unproductive repetition.
$H_3 = H_2 +$ **Workflow**	Locate → Read → Connect → Answer	Organize investigation and answer construction; choose phase transitions.
$H_4 = H_3 +$ **Verification**	Heuristic draft checks and bounded repair	Check draft support; revise, gather evidence, or finish.

3.1.2 Formalizing Harness Operations

We define a harness as a set of external processors $H = \{P_1, \ldots, P_m\}$ usually acting on the model's context. At each step, the processors inject external content or state:

$$c_t \;=\; \mathcal{P}_H\big(c_{t-1}, a_{t-1}, o_{t-1}\big), \qquad a_t \sim \pi_\theta\big(\cdot \mid q,\; h_{t-1},\; c_t\big). \tag{2}$$

The harness maintains an *external state* c_t and supplies it to the policy at runtime. We study four *nested* harness configurations $H_1 \subset H_2 \subset H_3 \subset H_4$ (Table 1). Each H_ℓ denotes the complete configuration; for $\ell \geq 2$, Δ_ℓ denotes its added control layer. Thus H_3 includes tools, state, and workflow, whereas Δ_3 is workflow control.

This layering maps onto the above three difficulties: H_2 mitigates state forgetting over long horizons, H_3 mitigates disorganized exploration under sparse credit assignment, and H_4 directly targets groundedness by gating the terminal step. Because the levels are nested, a higher level H_i trace simultaneously contains the behaviors of lower levels $H_{\leq i}$, providing the foundation for the layer-wise annealing introduced later.

3.1.3 Harness Internalization Objective

For a policy π_θ and a deployment harness H, let $S(\pi_\theta, H)$ denote the score on a held-out evaluation set. We define the **harness-dependency gap** between two configurations as

$$HGap(\pi_\theta, i, j) = S(\pi_\theta, H_i) - S(\pi_\theta, H_j), \qquad i > j. \tag{3}$$

A positive gap indicates that the additional support in H_i improves performance; a negative gap indicates that it reduces performance.

We also measure variation across the full deployment ladder using

$$\Delta(\pi_\theta) = \max_{j \in \{1,2,3,4\}} S(\pi_\theta, H_j) - \min_{j \in \{1,2,3,4\}} S(\pi_\theta, H_j). \tag{4}$$

Our goal is to achieve strong performance under weak support while reducing sensitivity to the deployment harness. We therefore assess $S(\pi_\theta, H_1)$ together with $\Delta(\pi_\theta)$ and the full score matrix. A small range alone is insufficient, because a policy that performs poorly under every configuration would also have low cross-harness variation. The training procedure below pursues this goal through supervised fine-tuning.

3.2 Harness Annealing Training

A natural baseline is supervised fine-tuning on complete tool-use trajectories. When harness decisions appear only as conditioning context, this objective does not explicitly train the policy to generate them. We introduce two components to address this gap: *Layer-wise Reasoning Supervision*, which makes selected control decisions explicit targets, and *Harness-Dependency Annealing*, which progressively introduces demonstrations collected under weaker harnesses into continued training.

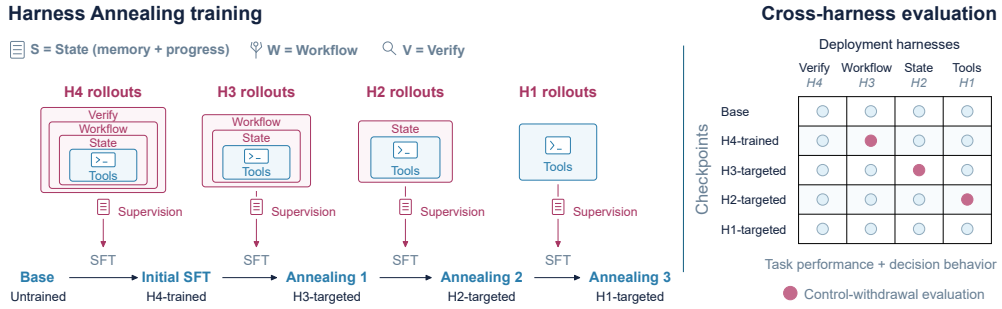

Figure 2: **Training and evaluation of HAT.** Initial SFT produces the Layered H_4-trained checkpoint. Three annealing stages then introduce records derived from H_3, H_2, and H_1 rollouts, supplemented by stronger-source replay and cue-dropout variants. The rollout labels identify source harnesses; targeted checkpoint labels identify cumulative training through the corresponding source level. Every checkpoint is evaluated under all four deployment harnesses. Red cells test removal of the control layer targeted at that stage: verification at H_3, workflow at H_2, and state at H_1. Related cue-removal patterns may occur in training through dropout.

3.2.1 Layer-wise Reasoning Supervision

We construct supervision targets for control decisions at each harness level and use them to train the policy to act with weaker external support.

Trajectory collection. For training tasks $\mathcal{D}_{\text{train}} = \{(q_n, E_n, y_n)\}_{n=1}^{N}$, we collect teacher trajectories under each source harness:

$$\mathcal{B} = \bigcup_{\ell=1}^{4} \mathcal{B}_\ell, \qquad \mathcal{B}_\ell = \left\{\tau_{n,r}^{(H_\ell, \pi_{\text{tea}})}\right\}_{n,r}, \tag{5}$$

where r indexes rollout rounds. The source bank $\mathcal{B}_\ell$ contains raw trajectories collected under H_ℓ; $\mathcal{D}_\ell$ denotes the training records derived from them. The source index is preserved after rewriting or cue dropout. Trajectories for the same task across harnesses support comparisons of recorded decisions and interventions, although their differences also reflect subsequent action choices and sampling variability. Control annotations are constructed deterministically from recorded interactions and level-specific rules, with no additional model calls.

Triplet rewriting. At model decision boundaries, we organize the trajectory into Obs, Think, and Act components:

- **Obs** contains the task, interaction history, observed evidence, and harness cues retained by the record construction. Its available support is specified separately from the source index.
- **Think** contains control-oriented supervision, such as workflow organization or answer checking, that the model is trained to produce. Related cues may also remain in the input.
- **Act** contains the recorded teacher action or answer.

For a record $x \in \mathcal{D}_\ell$, let $\mathcal{J}(x)$ denote its supervised decision boundaries. The loss covers Think and Act tokens, with the system prompt and designated special-token block masked out:

$$\mathcal{L}_{\text{LSFT}}^{(\ell)}(\theta) = -\mathbb{E}_{x\sim\mathcal{D}_\ell} \sum_{t\in\mathcal{J}(x)} \log \pi_\theta\big(\text{Think}_t, \text{Act}_t \mid \text{Obs}_t\big). \tag{6}$$

3.3 Harness-Dependency Annealing

Checkpoint notation. An H_ℓ-trained checkpoint, θ_ℓ^{ind}, is trained independently from Base using only H_ℓ-source data; Direct and Layered specify the supervision construction. An H_ℓ-targeted checkpoint, θ_ℓ^{ann}, continues the cumulative curriculum through source level ℓ. Its index identifies the latest source level introduced, not the deployment harness. Training starts from Layered H_4-trained, so $\theta_4^{\text{ann}} = \theta_4^{\text{ind,Layered}}$.

Training curriculum. The three annealing stages introduce $\mathcal{D}_3$, $\mathcal{D}_2$, and $\mathcal{D}_1$ in sequence. At stage ℓ, the mixture $\mathcal{M}_\ell$ combines current-source records, stronger-source replay, and cue-dropout variants where applicable (Appendix B.2). Let $\mathcal{L}^{(\ell)}_{\text{mix}}$ denote the loss in Equation 6 evaluated over this mixture. Training proceeds as

$$\theta_0 \xrightarrow{\mathcal{L}^{(4)}_{\text{LSFT}}} \theta^{\text{ann}}_4 \xrightarrow{\mathcal{L}^{(3)}_{\text{mix}}} \theta^{\text{ann}}_3 \xrightarrow{\mathcal{L}^{(2)}_{\text{mix}}} \theta^{\text{ann}}_2 \xrightarrow{\mathcal{L}^{(1)}_{\text{mix}}} \theta^{\text{ann}}_1. \tag{7}$$

For example, H_3-targeted continues from H_4-trained using H_3-source records and H_4-source replay. Standard H_3 records retain workflow support; cue dropout reduces selected input cues. The objective is to learn workflow control while retaining the verification behavior targeted earlier. We test this transfer under H_2, where neither workflow nor verification control is supplied externally.

The corresponding withdrawal evaluations are H_4-trained under H_3, H_3-targeted under H_2, and H_2-targeted under H_1. H_1-targeted continues training with tools-only source data; it does not target removal of the tools themselves. Every checkpoint is evaluated under all four harnesses. Appendix C summarizes the paths and distinguishes learning objectives from demonstrated internalization.

3.4 Design Rationale

Input support and supervised control. OBS supplies the conditions available at a decision boundary, whereas THINK supervises control judgments. Making a judgment an output target does not require removing every related input cue from every record. Standard examples provide supported demonstrations; cue dropout varies selected cues, and deployment evaluations test behavior after the corresponding runtime components are removed. Tools and observed evidence remain external throughout.

Direction. The stronger-to-weaker curriculum introduces source data with progressively less external control. It targets verification, then workflow, then state control, while replay revisits earlier supervision. This ordering lets each stage build on its predecessor rather than requiring the initial model to learn all control responsibilities from tools-only demonstrations. The final stage introduces tools-only source data, aligning its standard inputs with the weakest deployment interface. Replay still retains stronger support, and recorded histories may differ from those visited at deployment; the curriculum therefore does not assume that source records and deployment contexts are identical.

4 Experiments

In this section, experiments are conducted to answer the following research questions:

- RQ1: Can Harness Annealing internalize external control, while retaining performance under weak support after harness layers are removed?
- RQ2: Does Layered supervision transfer better under weak support than trajectory-level SFT?
- RQ3: Does Harness Annealing improve performance on out-of-distribution tasks?

4.1 Experimental Setup

Dataset. We conduct our experiments on repository-level question answering (repo-level QA), where an agent must answer questions by understanding an entire code repository rather than a single file or function. Agents may search and read the repository but may not modify it. We use the training split of SWE-QA-Pro (Cai et al., 2026) for training and evaluate on its test split and SWE-QA (Peng et al., 2026).

Table 2: Datasets and Rollout Configurations.
Evaluation yields 2,880 episodes per checkpoint on SWE-QA and 1,040 on SWE-QA-Pro.

Role	Dataset / split	Questions	Setup
Training	SWE-QA-Pro train	464	4 harness levels $\times$ 3 rollout rounds
Evaluation	SWE-QA	720	4 harness levels
Evaluation	SWE-QA-Pro test	260	4 harness levels

Harness Details We implement the 4 levels of harness with HarnessX (Chen et al., 2026), which represents prompts, memory, tools, and runtime control as composable processors (Darwin Agent Team, 2026a;b). We defer the full configuration details of HarnessX to Appendix D.

Implementation Details We fine-tune Qwen3.5-9B and Qwen3.5-35B-A3B (Qwen Team, 2026) on GPT-5.5 trajectories (OpenAI, 2026) collected under all four source harnesses. Base denotes the model before task-specific fine-tuning. Annealing starts from Layered H_4-trained, obtained by fine-tuning Base on records derived from $\mathcal{B}_4$ and also reported in Table 4. Each subsequent stage initializes from its predecessor and uses teacher-generated mixtures, without new student rollouts. Training examples follow Section 3, with reasoning tokens supervised unless otherwise stated. After quality filtering and removal of broken or duplicate records, 5,007 trajectories remain across the four source harnesses. Stage mixtures use current-source records, with stronger-source replay and cue-dropout variants where applicable. Table 5 reports mixture weights based on the original token proxy for 9B and tokenizer-counted supervised tokens for 35B.

Using LLaMA-Factory (Zheng et al., 2024b), each annealing stage runs for one epoch with bf16, rank-64 LoRA, and a per-device batch size of 1. The 65,536-token training cutoff exceeds the longest examples (approximately 58K–61K tokens), so none are truncated. Appendix B provides annealing configurations and the 35B checkpoint-selection procedure.

SGLang (Zheng et al., 2024a) serves rollout collection and evaluation. Both evaluation modes use a 65,536-token context window. No-thinking evaluation permits 4,096 completion tokens per call and 24 agent steps per episode; thinking-enabled evaluation permits 8,192 tokens and 32 steps. Results are not pooled across modes; main annealing results use thinking-enabled evaluation unless otherwise noted. Evaluation follows the reference-based SWE-QA rubric (Peng et al., 2026), summing five GPT-5.5 scores, each ranging from 1 to 20.

4.2 Main Results

Table 3 compares every checkpoint under all four deployment harnesses.

Annealing improves performance under weak support. For 9B, relative to the H_4-trained starting checkpoint, H_2-targeted improves H_1 scores from 71.99 to 77.00 on SWE-QA and from 74.64 to 77.73 on SWE-QA-Pro, gains of 5.01 and 3.09 points. Its cross-harness range Δ decreases from 5.48/3.61 to 0.57/1.32 on the two benchmarks. Thus, reduced variation accompanies stronger performance under weak support. For 35B, H_1 scores improve by 0.40 and 1.36 points, but Δ changes from 1.30 to 1.44 on SWE-QA and from 2.13 to 1.66 on SWE-QA-Pro. Better weak-support performance therefore does not always imply lower cross-harness variation.

Performance after removal of the targeted control layer. The red cells evaluate each checkpoint with the control layer targeted for internalization removed. For example, H_3-targeted is tested under H_2, with workflow and verification control both absent. For the 9B model, H_4-trained under H_3, H_3-targeted under H_2, and H_2-targeted under H_1 score 77.47, 77.15, and 77.00 on SWE-QA and 76.87, 77.96, and 77.73 on SWE-QA-Pro. Every red cell far exceeds Base under the *same* weak deployment, with the gap widening as support weakens. These scores show strong task performance after the targeted support is removed. The red cells vary both checkpoint and deployment harness, so they characterize the withdrawal evaluations rather than isolate the gain from each training stage. The trajectory comparison in Appendix G examines changes in model behavior after annealing.

The preferred checkpoint depends on model scale and deployment objective. For the 9B model, H_2-targeted achieves the highest mean across the two benchmarks and four deployment harnesses, at 77.17. For the 35B model, H_4-trained has the highest corresponding mean, at 81.59, compared with 81.39 for H_2-targeted. Nevertheless, H_2-targeted achieves the highest H_1 score on both benchmarks at 35B. Annealing therefore improves the weakest-support endpoint without necessarily maximizing performance averaged across all deployment configurations.

Control-related behavior under cue removal. In the paired behavioral probe (Appendix F), stop/continue macro-F1 under H_1 input cues rises from 64.43 at H_4-trained to 87.79 at H_1-targeted. This teacher-forced probe holds reference states fixed and varies input cues, measuring action selection conditioned on reference reasoning rather than executing the runtime harnesses.

4.3 Direct versus Layered Supervision

Layered supervision avoids the weak-support collapse of Direct SFT. Table 4 compares Direct and Layered supervision for each source-harness level. Direct supervision serializes complete trajectories, whereas Layered supervision constructs decision-level records as described in Section 3.2.1. All

Table 3: **Every training checkpoint is evaluated under every deployment configuration.** Scores are mean reference-based judge scores; higher is better. Red cells test the control responsibility targeted at each stage after its external support is removed: H_4-trained under H_3, H_3-targeted under H_2, and H_2-targeted under H_1. Related cue-removal patterns may occur in training through dropout. H_1 exposes the bare tool interface and each subsequent harness adds one layer of control, so that $H_1 \subset H_2 \subset H_3 \subset H_4$. Within each model–benchmark block, bold marks the best in each column. The starting row is Layered H_4-trained; each targeted checkpoint continues from the row above.

Model	Benchmark	Checkpoint	Deployment Harness				Row summary		
			H_1	H_2	H_3	H_4	Mean	Worst	Δ
Qwen3.5-9B	SWE-QA	Base	59.93	62.92	67.28	66.79	64.23	59.93	7.35
		H_4-trained	71.99	75.94	**77.47**	75.70	75.28	71.99	5.48
		H_3-targeted	**77.41**	**77.15**	75.10	75.66	76.33	75.10	2.31
		H_2-targeted	77.00	76.62	76.43	**76.45**	**76.63**	**76.43**	**0.57**
		H_1-targeted	76.30	75.12	74.43	74.16	75.00	74.16	2.14
	SWE-QA-Pro	Base	57.96	63.19	66.40	66.39	63.49	57.96	8.44
		H_4-trained	74.64	77.81	76.87	78.25	76.89	74.64	3.61
		H_3-targeted	77.06	**77.96**	75.97	77.54	77.13	75.97	1.99
		H_2-targeted	**77.73**	77.83	76.98	**78.30**	**77.71**	76.98	1.32
		H_1-targeted	77.12	77.40	**77.94**	77.47	77.48	**77.12**	**0.82**
Qwen3.5-35B	SWE-QA	Base	78.14	75.66	75.72	76.23	76.44	75.66	2.48
		H_4-trained	80.60	79.85	80.13	**81.15**	80.43	79.85	**1.30**
		H_3-targeted	80.33	**81.71**	79.61	79.71	80.34	79.61	2.10
		H_2-targeted	**81.00**	81.48	80.04	80.42	**80.74**	**80.04**	1.44
		H_1-targeted	80.51	81.23	79.62	81.03	80.60	79.62	1.61
	SWE-QA-Pro	Base	72.16	73.58	74.30	73.53	73.39	72.16	2.14
		H_4-trained	81.82	82.72	**82.48**	**83.95**	**82.74**	**81.82**	2.13
		H_3-targeted	79.88	**82.96**	81.61	82.03	81.62	79.88	3.08
		H_2-targeted	**83.18**	81.65	81.52	81.85	82.05	81.52	1.66
		H_1-targeted	81.06	82.42	82.20	81.77	81.86	81.06	**1.36**

adapted checkpoints in this table are trained independently from Base using only the indicated source level; Direct and Layered specify the construction. The Layered H_4-trained row is the starting checkpoint in Table 3.

The two constructions diverge sharply under the weakest deployment H_1. For every training source, on both benchmarks and at both model scales, Layered supervision outperforms Direct supervision at H_1. The gap is large: for the 9B model on SWE-QA, the four Direct checkpoints span 16.51–32.45 across sources, compared with 53.73–72.48 for their Layered counterparts. Direct supervision thus collapses precisely where external support is weakest, whereas Layered supervision retains usable performance. The H_1-source Direct checkpoint also scores only 32.45 under H_1, compared with 59.93 for Base, showing that the degradation is not confined to a mismatch between source and deployment harnesses. This advantage over Direct is uniform at H_1; relative to Base, however, it is source-dependent, as the strongest source (H_4-trained) exceeds Base under every deployment while some weaker sources fall below Base under certain configurations.

4.4 Cross-Task OOD Generalization

We evaluate Qwen3.5-9B checkpoints zero-shot on the 97 banking-knowledge tasks of τ^3-Bench (Barres et al., 2025; Shi et al., 2026). Fine-tuning uses only SWE-QA-Pro training trajectories. All checkpoints use the same banking harness, thinking mode, and $65{,}536$-token context window.

Table 4: **Direct and Layered supervision produce different cross-harness transfer profiles.** Scores are mean reference-based judge scores; higher is better. For each model scale, the table reports one Base row, four Direct-trained checkpoints, and four independently Layered-trained checkpoints. All adapted checkpoints are trained independently from Base using the indicated source level. Columns specify the deployment harness. Bold marks the highest score among the Layered checkpoints in each deployment column for each model size.

Model	Supervision	Independent checkpoint	SWE-QA inference harness				SWE-QA-Pro inference harness			
			H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4
9B		Base	59.93	62.92	67.28	66.79	57.96	63.19	66.40	66.39
	Direct	H_1-trained	32.45	65.83	62.29	59.65	34.74	65.69	66.28	64.02
		H_2-trained	16.76	57.83	58.98	61.30	17.88	66.95	62.58	62.63
		H_3-trained	17.91	64.15	64.60	61.71	18.43	66.89	66.95	66.88
		H_4-trained	16.51	60.21	61.28	62.19	16.69	68.88	67.23	64.02
	Layered	H_1-trained	**72.48**	75.18	75.78	72.91	**77.52**	**78.28**	**78.02**	76.72
		H_2-trained	53.73	69.23	68.56	66.18	59.75	71.53	68.43	66.21
		H_3-trained	62.27	67.52	72.08	55.14	68.70	73.21	68.97	61.54
		H_4-trained	71.99	**75.94**	**77.47**	**75.70**	74.64	77.81	76.87	**78.25**
35B		Base	78.14	75.66	75.72	76.23	72.16	73.58	74.30	73.53
	Direct	H_1-trained	30.62	63.35	62.27	60.12	31.57	65.63	66.10	60.74
		H_2-trained	21.14	68.32	68.74	65.09	23.48	75.14	73.85	74.48
		H_3-trained	25.70	70.39	69.18	67.42	32.31	75.51	77.59	74.72
		H_4-trained	24.58	69.93	71.16	66.57	30.83	76.16	76.58	74.23
	Layered	H_1-trained	80.51	75.49	76.27	74.91	81.01	79.04	77.88	78.68
		H_2-trained	73.68	75.12	75.69	73.59	73.59	78.62	75.88	76.60
		H_3-trained	65.81	77.88	76.43	76.48	71.80	80.88	77.73	78.43
		H_4-trained	**80.60**	**79.85**	**80.13**	**81.15**	**81.82**	**82.72**	**82.48**	**83.95**

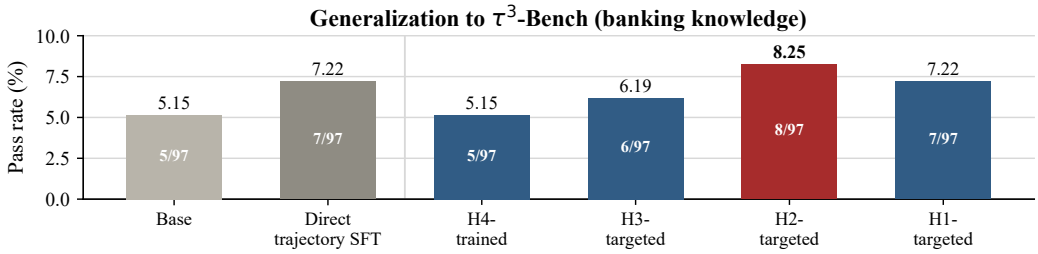

Figure 3: **Cross-task evaluation on the banking-knowledge domain of τ^3-Bench.** Zero-shot pass rates for Qwen3.5-9B on 97 tasks under a fixed banking harness. Numbers inside bars indicate solved tasks.

Figure 3 suggests that harness annealing can improve transfer beyond repository QA without target-domain fine-tuning. The peak at an intermediate stage mirrors the in-domain trend, suggesting that further annealing does not necessarily improve transfer. However, the modest gain over Direct SFT and unmatched training budgets do not isolate control internalization as the source of this benefit.

5 Conclusion

We view model learning and harness development as a feedback loop: stronger models enable richer workflows, while harness-supported experience can train subsequent models. Yet learning from this experience does not automatically transfer the control decisions supplied by the harness. Harness Annealing Training studies this objective by constructing explicit control supervision and progressively introducing demonstrations collected under weaker harnesses. On SWE-QA and SWE-QA-Pro, selected annealed checkpoints improve performance under weak support, although gains depend on model scale and deployment configuration and further annealing is not uniformly beneficial. These findings suggest a path for model–harness co-evolution: training can reshape the division of responsibility between the model and its harness. When a model can reliably assume a control responsibility, the corresponding guidance need no longer be repeatedly supplied at runtime. Harness development can then focus on the model's remaining limitations, enabling the model–harness system to tackle increasingly demanding tasks.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Songcheng Cai, Zhiheng Lyu, Yuansheng Ni, Xiangchao Chen, Baichuan Zhou, Shenzhe Zhu, Yi Lu, Haozhe Wang, Chi Ruan, Benjamin Schneider, Weixu Zhang, Xiang Li, Andy Zheng, Yuyu Zhang, Ping Nie, and Wenhu Chen. SWE-QA-Pro: A representative benchmark and scalable training recipe for repository-level code understanding. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. URL https://arxiv.org/abs/2603.16124.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023. URL https://arxiv.org/abs/2310.05915.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, et al. Harnessx: A composable, adaptive, and evolvable agent harness foundry. *arXiv preprint arXiv:2606.14249*, 2026.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*, 2024. URL https://arxiv.org/abs/2403.12881.

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. ATLaS: Agent tuning via learning critical steps. *arXiv preprint arXiv:2503.02197*, 2025. URL https://arxiv.org/abs/2503.02197.

Darwin Agent Team. HarnessX: A composable, adaptive, and evolvable agent harness foundry. Technical report, Darwin Agent, 2026a.

Darwin Agent Team. HarnessX: A composable, self-evolving agent harness foundry, 2026b. URL https://github.com/Darwin-Agent/HarnessX. Version 0.1.0, MIT License.

Simon Dennis, Rivaan Patil, Kevin Shabahang, and Hao Guo. Compiling agentic workflows into LLM weights: Near-frontier quality at two orders of magnitude less cost. *arXiv preprint arXiv:2605.22502*, 2026. URL https://arxiv.org/abs/2605.22502.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/36b7acf6f6010652b3f2a433774a66fe-Paper-Conference.pdf.

Pengcheng Jiang, Zhiyi Shi, Kelly Hong, Xueqiang Xu, Jiashuo Sun, Jimeng Sun, Hammad Bashir, and Jiawei Han. Harness-1: Reinforcement learning for search agents with state-externalizing harnesses, 2026. URL https://arxiv.org/abs/2606.02373.

Kalle Kujanpää, Harri Valpola, and Alexander Ilin. Knowledge injection via prompt distillation. *arXiv preprint arXiv:2412.14964*, 2024. URL https://arxiv.org/abs/2412.14964.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Xuying Ning, Dongqi Fu, Tianxin Wei, Hanqing Zeng, Yuanchen Bei, Bingxuan Li, Zihao Li, Qifan Wang, Xiang Shen, Yifan Wu, Jiayi Liu, Hong Li, Yinglong Xia, Xiangjun Fan, Hanghang Tong, and Jingrui He. Evoharness-rl: Learning self-evolving runtime harness for long-horizon llm agents, 2026. URL https://arxiv.org/abs/2608.05446.

OpenAI. GPT-5.5 model. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.5. Accessed 2026-08-28.

Weihan Peng, Yuling Shi, Yuhang Wang, Xinyun Zhang, Beijun Shen, and Xiaodong Gu. SWE-QA: Can language models answer repository-level code questions? In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. URL https://arxiv.org/abs/2509.14635.

Qwen Team. Qwen3.5-9B model card. Hugging Face model repository, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B. Accessed 2026-08-28.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016. URL https://arxiv.org/abs/1511.06295.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-knowledge: Evaluating conversational agents over unstructured knowledge, 2026. URL https://arxiv.org/abs/2603.04370.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents. *arXiv preprint arXiv:2604.10674*, 2026. URL https://arxiv.org/abs/2604.10674.

John Yang, Carlos E. 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*, 2024. URL https://arxiv.org/abs/2405.15793.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. *arXiv preprint arXiv:2310.12823*, 2023. URL https://arxiv.org/abs/2310.12823.

Yunbei Zhang, Janet Wang, Yingqiang Ge, Weijie Xu, Jihun Hamm, and Chandan K. Reddy. Stop comparing LLM agents without disclosing the harness. *arXiv preprint arXiv:2605.23950*, 2026. URL https://arxiv.org/abs/2605.23950.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024a.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pp. 400–410, 2024b.

A LIMITATIONS

Our experiments use two Qwen3.5 models, one fixed harness hierarchy, and a single training seed. Evaluation focuses on repository question answering, with one banking-domain transfer experiment. This leaves open whether the same training recipe transfers to other model families, task domains, or harness designs. Later checkpoints receive more training and use different data mixtures. Without equal-budget controls, the observed differences cannot be attributed to the annealing order alone, nor can the contributions of replay and cue dropout be separated. Direct and Layered supervision are also not matched for supervised-token counts or optimizer updates; their comparison therefore evaluates two training recipes rather than trajectory segmentation alone.

B IMPLEMENTATION DETAILS

The nested runtime configurations are H_1 (tools), H_2 (tools and state), H_3 (tools, state, and workflow), and H_4 (all three plus verification). The curriculum starts from Layered H_4-trained and produces H_3-, H_2-, and H_1-targeted checkpoints. These labels identify cumulative training history (Appendix C).

B.1 TRAINING CONFIGURATION

Qwen3.5-9B. All three annealing stages continue the preceding LoRA adapter for one epoch on four GPUs. One example per GPU and four accumulation steps give a global batch size of 16. We use bf16, AdamW (betas 0.9/0.999, epsilon 10^{-8}), a peak learning rate of 10^{-6}, cosine decay, a 0.05 warmup ratio, and seed 42. The cutoff is 65,536 tokens. LoRA rank, alpha, and dropout are 64, 128, and 0.05, with no bias and the attention/MLP targets recorded in the adapter configurations. The three stages use 1,775, 2,049, and 2,291 records and 111, 129, and 144 optimizer updates. The recorded environment uses PEFT 0.19.1, Transformers 5.8.1, PyTorch 2.12.1+cu130, Datasets 4.8.4, and Tokenizers 0.22.2.

Qwen3.5-35B-A3B. We use six GPUs with FSDP full sharding and gradient checkpointing. One example per GPU and two accumulation steps give a global batch size of 12. Each stage trains for one epoch. Peak learning rates are 7.5×10^{-7}, 4.0×10^{-7}, and 2.5×10^{-7} for H_3-, H_2-, and H_1-targeted training, respectively. We use bf16, fused AdamW, cosine decay, a 0.03 warmup ratio, weight decay 0.01, gradient clipping at 1.0, and training/data seed 42. LoRA rank, alpha, and dropout are 64, 128, and 0.05. The configuration targets all supported linear modules and freezes the vision tower and multimodal projector. The `qwen3_5` template preserves reasoning; prompts are not supervised, assistant history is not masked, and packing is disabled. The cutoff is 65,536 tokens. There are 1,812, 1,870, and 1,442 training records and 151, 156, and 121 optimizer updates.

For 35B, 43 training-source question IDs are held out across levels; checkpoint selection uses their validation loss, not benchmark test scores. Validation runs approximately every quarter epoch. The selected updates are 151/151, 117/156, and 121/121: the H_2-targeted adapter is selected at 0.75 epoch, while the other two are selected at the final update. Each subsequent stage loads the selected predecessor.

B.2 TRAINING MIXTURES

Table 5 separates current-source examples from stronger-source replay. Every row sums to 100%; a dash means that source is not used. The 9B weights use the original token proxy, whereas 35B uses tokenizer-counted supervised tokens.

Table 5: **Current-source data and replay at each annealing stage (%).** Training proceeds down each block from H_3- to H_1-targeted. The first stage starts from H_4-trained; subsequent stages continue the preceding adapter. For H_ℓ-targeted, the current source is $\mathcal{D}_\ell$; replay columns identify data sources, not deployment harnesses.

		Current-source examples		Replay source		
Model	**Checkpoint**	Standard	Cue dropout	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
9B	H_3-targeted	70	10	–	–	20
	H_2-targeted	60	10	–	20	10
	H_1-targeted	50	–	25	15	10
35B	H_3-targeted	40	30	–	–	30
	H_2-targeted	20	30	–	10	40
	H_1-targeted	50	–	20	10	20

Cue dropout modifies current-source inputs while retaining their response labels. For 35B H_3 data, it removes workflow instructions and the additional workflow observation to form an H_2-style input. For H_2 data, it removes state instructions, the progress ledger, and appended harness state to form an H_1-style input. These variants retain their source index: $\mathcal{D}_3^{\text{drop}}$ and $\mathcal{D}_2^{\text{drop}}$ remain H_3- and H_2-source data, respectively. They preserve the recorded interaction history and are not new lower-harness rollouts. Whole-record sampling makes realized 35B proportions slightly different from the target percentages shown in the table.

C Training Paths and Control-Withdrawal Evaluation

Table 6 separates cumulative training history from deployment. An H_ℓ-trained checkpoint is trained independently from Base on one source level. An H_ℓ-targeted checkpoint continues the curriculum from Layered H_4-trained through source level ℓ, with the mixtures in Appendix B.2. Thus H_3-targeted has the sequential history $H_4 \to H_3$, whereas H_3-trained has only H_3-source training. The index identifies the latest source introduced, not the deployment harness.

Table 6: **Cumulative training paths and corresponding control-withdrawal evaluations.** Arrows indicate sequential introduction of source data through continued training. Table 5 gives the mixtures used at each stage.

Checkpoint	**Cumulative source path**	**Learning objective**	**Withdrawal evaluation**
H_4-trained Initial SFT	H_4	Learn verification control.	H_3: verification removed; tools, state, and workflow retained.
H_3-targeted Annealing stage 1	$H_4 \to H_3$	Learn workflow control; retain earlier verification behavior.	H_2: workflow and verification absent; tools and state retained.
H_2-targeted Annealing stage 2	$H_4 \to H_3 \to H_2$	Learn state control; retain verification and workflow behavior.	H_1: state, workflow, and verification absent; tools retained.
H_1-targeted Annealing stage 3	$H_4 \to H_3 \to H_2 \to H_1$	Consolidate autonomous execution using tools-only source data.	H_1: evaluate the tools-only endpoint; no further layer is removed.

Source labels survive rewriting and cue dropout. Standard H_3 records retain workflow cues even though H_3-targeted is tested under H_2: training inputs and deployment support are specified separately. The table states learning objectives; checkpoint names do not establish successful internalization. Every checkpoint is evaluated under all four harnesses. H_1-targeted remains a tools-only endpoint, not a step toward tool-free H_0 deployment.

D HARNESS DETAILS

Table 7 shows which support is available in each deployment. All four expose `Read`, `Grep`, `Glob`, and read-only `Bash`; higher levels add state, workflow, and answer-checking support. The question-only H_0 setting has no repository context or tools and is not part of the four-column evaluation.

Table 7: **Runtime support retained at each harness level.** A checkmark means the component is present; a dash means it is absent. Each higher level retains all lower-level support.

Runtime support	H_1	H_2	H_3	H_4
Repository tools	✓	✓	✓	✓
Tool guards and token budget	✓	✓	✓	✓
Recent history and progress ledger	–	✓	✓	✓
Repeated-action detection	–	✓	✓	✓
Locate → Read → Connect → Answer	–	–	✓	✓
Answer verification and repair	–	–	–	✓

Tool guards and token budgets remain active in all four configurations. H_2 bundles recent-history retrieval, a progress ledger, and loop detection; this bundle is evaluated as one layer. H_4 checks candidate answers and permits at most two keyword-based repair searches. It may reject short, uncertainty-marked, or ungrounded drafts, but never receives the reference answer.

E EVALUATION WITH THE LLM JUDGE

We evaluate final-answer quality on SWE-QA and SWE-QA-Pro using a reference-based LLM-as-a-Judge protocol with `gpt-5.5` as the judge. For each example, the judge receives the question q_i, reference answer r_i, and candidate final answer a_i, without the agent's interaction history or tool observations:

$$(q_i, r_i, a_i),$$

For each example, the judge assigns five equally weighted integer scores:

$$s_{i,\text{corr}},\ s_{i,\text{comp}},\ s_{i,\text{rel}},\ s_{i,\text{clar}},\ s_{i,\text{reas}} \in \{1, \ldots, 20\},$$

corresponding to correctness, completeness, relevance, clarity, and reasoning, respectively. The dimensions are defined as follows.

Table 8: **The five questions asked by the answer judge.** Each dimension contributes 1–20 points. The full rubric and score anchors are reproduced in the prompt below.

Dimension	What is checked?	What lowers the score?
Correctness	Do the claims agree with the reference?	Errors or contradictions
Completeness	Are the reference's key points covered?	Missing mechanisms or conditions
Relevance	Does the answer address the question?	Off-topic or irrelevant content
Clarity	Is the answer precise and easy to follow?	Ambiguity or poor organization
Reasoning	Do the stated claims support the conclusion?	Logical gaps or unjustified inferences

The total judge score for example i is the unweighted sum

$$S_i = s_{i,\text{corr}} + s_{i,\text{comp}} + s_{i,\text{rel}} + s_{i,\text{clar}} + s_{i,\text{reas}}, \qquad S_i \in [5, 100]. \tag{8}$$

No additional weighting, normalization, or learned calibration is applied, and the primary judge does not emit a binary pass/fail decision.

E.1 Complete Judge Prompt

The following is the complete prompt template used by the principal `gpt-5.5` judge. The placeholders {`question`}, {`reference`}, and {`candidate`} are replaced by the corresponding example fields.

Verbatim judge prompt

```
You are a STRICT and RIGOROUS evaluator. You must rate the candidate answer
  STRICTLY against the reference answer. Be CONSERVATIVE with high scores -
  only award high scores (16-20) when the candidate answer is truly excellent
  and closely matches the reference answer in quality and content.

CRITICAL EVALUATION PRINCIPLES:
1. Compare the candidate answer DIRECTLY with the reference answer point by
  point
2. Any deviation, omission, or inaccuracy should result in score reduction
3. High scores (16-20) should be RARE - reserve them only for answers that
  are nearly perfect
4. Be strict about factual accuracy - even minor errors should lower the
  correctness score
5. Missing key points from the reference answer should significantly reduce
  completeness score
6. Vague or imprecise language should lower clarity scores
7. When in doubt between two score ranges, choose the LOWER one

Evaluation Criteria and Scoring Guidelines (each scored 1 to 20, total score
  100):
       1. Correctness (STRICT - penalize any inaccuracies):
           20 | ONLY if completely correct with ALL core points and details
  accurate, matching reference answer precisely
           16-19 | Mostly correct but must have only TRIVIAL inaccuracies;
  any noticeable error reduces to 15 or below
           12-15 | Partially correct; has some errors or omissions that
  affect understanding; main points may be accurate but details are wrong
           8-11 | Several errors or ambiguities that significantly affect
  understanding of core information
           4-7 | Many errors; misleading or fails to convey key information
  correctly
           1-3 | Serious errors; completely wrong or misleading
       2. Completeness (STRICT - penalize missing information):
           20 | ONLY if covers ALL key points from reference answer without
  ANY omission; must match reference in depth
           16-19 | Covers most key points but missing some non-trivial
  information; minor omissions are acceptable
           12-15 | Missing several important key points; content is
  noticeably incomplete compared to reference
           8-11 | Important information largely missing; content is
  one-sided or superficial
           4-7 | Covers very little relevant information; seriously
  incomplete
           1-3 | Covers almost no relevant information; completely incomplete
       3. Relevance (STRICT - penalize off-topic content):
           20 | ONLY if content is fully focused on question topic with NO
  irrelevant information whatsoever
           16-19 | Mostly focused but may have minor peripheral information;
  any significant off-topic content reduces score
           12-15 | Generally on topic but contains some off-topic content
  that detracts from answer
           8-11 | Topic not sufficiently focused; contains considerable
  off-topic or tangential content
           4-7 | Content deviates from topic; includes excessive irrelevant
  information
           1-3 | Majority of content irrelevant to the question
```

Verbatim judge prompt (continued)

```
       4. Clarity (STRICT - penalize unclear expression):
           20 | ONLY if language is exceptionally fluent, clear, and
  precise; very easy to understand without any ambiguity
           16-19 | Mostly fluent and clear but may have minor unclear
  points; any significant ambiguity reduces score
           12-15 | Generally clear but some expressions are unclear or not
  concise; may require effort to understand
           8-11 | Expression somewhat awkward; has ambiguity or lacks
  fluency that hinders understanding
           4-7 | Language obscure; sentences are not smooth; significantly
  hinders understanding
           1-3 | Expression confusing; very difficult to understand
       5. Reasoning (STRICT - penalize weak logic):
           20 | ONLY if reasoning is exceptionally clear, logical, and
  well-structured; argumentation is excellent and matches reference quality
           16-19 | Reasoning is clear and logical with solid argumentation;
  minor logical gaps may exist
           12-15 | Reasoning generally reasonable but has noticeable logical
  jumps or organization issues
           8-11 | Reasoning is average; has logical jumps or organization
  problems that affect understanding
           4-7 | Reasoning unclear; lacks logical order; difficult to follow
           1-3 | No clear reasoning; logic is chaotic

INPUT:
    Question:{question}
    Reference Answer:{reference}
    Candidate Answer:{candidate}

OUTPUT:
    Please output ONLY a JSON object with 5 integer fields in the range
  [1,20], corresponding
    to the evaluation scores:
        {
        "correctness": <1-20>,
        "completeness": <1-20>,
        "relevance": <1-20>,
        "clarity": <1-20>,
        "reasoning": <1-20>
        }

SCORING INSTRUCTIONS:
- Read the reference answer carefully and identify ALL key points, details,
  and structure
- Compare the candidate answer systematically against the reference answer
- For each criterion, start with a conservative score and only increase if
  the candidate truly deserves it
- If the candidate answer is significantly shorter, less detailed, or less
  precise than the reference, reduce scores accordingly
- If the candidate answer contains information not in the reference (unless
  it's clearly relevant and accurate), consider reducing relevance score
- When scoring, ask yourself: "Does this candidate answer match the quality
  and completeness of the reference answer?" If not, reduce scores
- Average or mediocre answers should receive scores in the 8-15 range, not
  higher
- Only truly excellent answers that closely match the reference should
  receive 16-20 scores

REQUIREMENT:
    No explanation, no extra text, no formatting other than valid JSON. Be
  strict and conservative with your scores.
```

F Decisions after Removing Harness Cues

Task scores show whether an agent succeeds with less external support. This 9B probe asks a narrower question: with the decision history held fixed, does removing control information change the model's next action? The four checkpoints are scored on the same reference states, rather than on independently generated trajectories.

F.1 Reference states and paired inputs

We use archived H_4-trained evaluations under H_4 on SWE-QA and SWE-QA-Pro. Candidate runs must be completed with `done`, have zero verifier failures and repairs, lexical F1 of at least 0.10, reference-keyword recall of at least 0.25, and a code reference. These are quality filters, not a guarantee that the answer is correct. We match the recorded final answer to the prediction and resolve stored tool observations. The H_4 training source contains 1,279 trajectory records covering 464 distinct questions. A post-hoc audit of the 128 probe questions against the original question texts from all four source pools found no exact matches after lowercasing and whitespace normalization.

With seed 6542, we select 64 distinct questions per benchmark. For each trajectory we take the first, middle, and last tool decision and the final answer, removing duplicated boundaries and examples above a 220,000-character prefilter. This yields 432 candidate states. Keeping only states that fit 65,536 tokens in all 16 checkpoint–input combinations, without truncation, leaves 395 states from 128 questions: 204 SWE-QA and 191 Pro states. There are 319 tool decisions and 76 final-answer decisions; the late/final subset contains 84 tool decisions and 76 answers.

Each state supplies four input-cue conditions, denoted H_1–H_4 in Table 9. These labels refer to retained input cues here. H_4 keeps the original input; H_3 removes the verification instruction block; H_2 also removes the workflow instruction block; H_1 additionally removes the state instruction and messages marked as harness-generated. The question, tool schemas, tool observations, previous model messages, and target response are unchanged. These are nested *input-cue ablations*, not four reruns of the executable harness. In particular, prior model history can retain effects of the original harness.

F.2 What the three metrics measure

All scores use teacher forcing: the model sees the reference reasoning before the action, and each predicted token is conditioned on the reference token prefix. Thus the probe tests action selection *given that reasoning*, not the ability to generate it unaided. No tools are executed.

Stop/continue macro-F1 — 395 decision states

At the first action token, a top-1 prediction of the tool-call marker means continue; any other token is counted as stop. The reference label is continue for a tool action and stop for a final answer.

$$F_{\text{stop/continue}} = \tfrac{1}{2}(F_{1,\text{stop}} + F_{1,\text{continue}}), \qquad F_{1,c} = \frac{2TP_c}{2TP_c + FP_c + FN_c}.$$

This is a binary marker diagnostic; a non-tool token need not be a valid final answer.

Late/final decision accuracy — 160 late/final states

On the late/final subset, we report

$$A_{\text{late/final}} = \frac{1}{|S_{\text{lf}}|} \sum_{i \in S_{\text{lf}}} \mathbf{1}[\hat{c}_i = c_i].$$

This metric measures agreement with the reference continue/stop decision on the late/final subset. There is no independent annotation of evidence sufficiency. Because it uses a subset of the first metric's states, it is not independent corroboration of control transfer.

Next-tool accuracy 319 tool decisions

The first function name is correct only if every token in its name has the correct teacher-forced top-1 prediction:

$$A_{\text{tool}} = \frac{1}{319} \sum_{i} \mathbf{1}[\forall j, \hat{t}_{ij} = t_{ij}].$$

The metric does not require correct arguments or successful tool execution.

F.3 WHERE CONTROL DECISIONS IMPROVE

Table 9 reports all 48 behavioral values. From H_4-trained to H_1-targeted, stop/continue macro-F1 under H_1 input cues rises from 64.43 to 87.79 and late/final decision accuracy from 62.50 to 83.13. Next-tool accuracy changes much less, from 96.24 to 97.18. The main change is therefore in the conditional continue/stop prediction under stripped inputs, rather than recognition of the reference tool name.

Table 9: **Stopping decisions recover under H_1 inputs; tool-name prediction changes little.** Teacher-forced scores (%) on the same reference states. Columns change input cues, not executable harnesses. Bold marks the best value within each column and metric.

	Input-cue condition			
Checkpoint	$\boldsymbol{H_1}$	$\boldsymbol{H_2}$	$\boldsymbol{H_3}$	$\boldsymbol{H_4}$
Stop/continue macro-F1 ($N = 395$)				
H_4-trained	64.43	**94.31**	**95.75**	**95.71**
H_3-targeted	78.12	93.25	94.66	94.66
H_2-targeted	83.95	92.57	94.31	94.31
H_1-targeted	**87.79**	92.23	93.60	93.95
Late/final decision accuracy ($N = 160$)				
H_4-trained	62.50	**92.50**	**94.38**	**94.38**
H_3-targeted	73.13	91.25	92.50	92.50
H_2-targeted	78.75	90.00	91.88	91.88
H_1-targeted	**83.13**	89.38	90.63	91.25
Next-tool accuracy ($N = 319$)				
H_4-trained	96.24	97.18	**95.92**	95.92
H_3-targeted	96.55	97.49	95.61	**96.24**
H_2-targeted	96.55	**97.81**	95.61	95.61
H_1-targeted	**97.18**	**97.81**	95.61	95.92

For metric m, define the descriptive behavioral gap as $G_m(M) = m(M, H_4) - m(M, H_1)$. The stop/continue gap contracts from 31.28 to 6.16 percentage points; the late/final gap contracts from 31.88 to 8.12 points. This is not cost-free: H_4-input F1 falls from 95.71 to 93.95, and late/final accuracy from 94.38 to 91.25. The matrix separates weak-input improvement from deterioration under full cues rather than treating a smaller gap alone as successful transfer.

G WORKFLOW RECOVERY CASE

Figure 4 compares the H_4-trained and H_3-targeted checkpoints on the same task, both deployed under H_2. The pair illustrates recovery and stopping after workflow control is removed; it is not an aggregate effect or proof that the final answer is grounded.

Same task under H_2: how does `ensure_eol()` modify the output buffer?

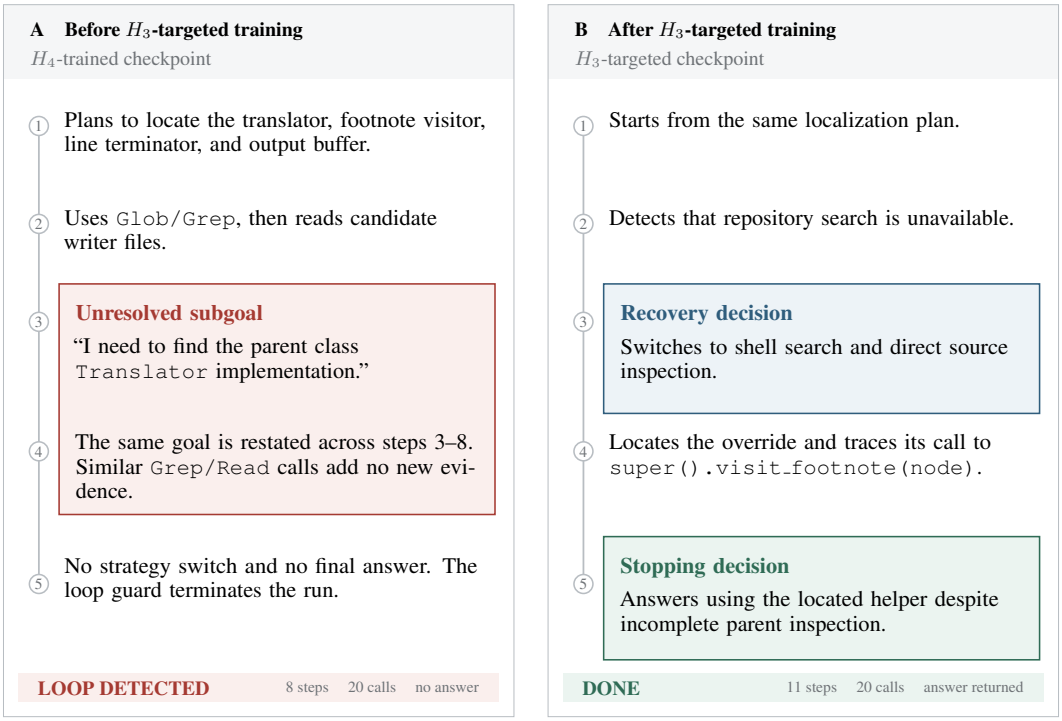

Figure 4: **Workflow recovery and stopping under H_2.** Both 9B runs use `sphinx-00045` at commit `6c9e320` and the same four-tool interface, with external workflow control absent. The question concerns the output buffer before the parent footnote visitor. The H_4-trained run repeats an unresolved search; the H_3-targeted run changes strategy and returns an answer after the same total of 20 tool calls. Text is abridged from the recorded reasoning. Both runs stop below their active step limits, and no displayed response is length-truncated. The returned answer cites a same-named helper in `texinfo.py` without verifying the inherited `docutils` implementation. The case therefore documents recovery and stopping behavior; it does not establish that the final answer is fully grounded.

H AI USE STATEMENT

We used generative AI tools for English-language editing, translation, and terminology checks, and for feedback on methodological descriptions and interpretation of the reported findings. Research uses of language models are documented in the Method and Experiments sections. The authors retain responsibility for the final manuscript and all scientific claims.