# HERA: HARNESS–ENVIRONMENT CO-EVOLUTION FOR RELIABLE AGENTIC ABSTENTION

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

## ABSTRACT

Large language model (LLM) agents are increasingly capable of acting in complex tool-use environments, yet they often fail to recognize when tasks are infeasible and no valid solution exists. Recent work has formalized this reliability gap as the problem of *agentic abstention*, and existing approaches typically optimize a model or agent harness against a fixed set of tasks, leading to limited generalization to unseen failure modes. We introduce HERA, a framework for harness–environment co-evolution for agentic abstention. HERA consists of (i) a pipeline to automatically construct verifiable pairs of feasible and infeasible tasks by applying controlled environment mutations that transform solvable tasks into cases requiring abstention, and (ii) a co-evolution procedure in which performance failures on previous tasks are used to drive harness adaptation and generate new execution environments and tasks geared towards previous weaknesses. On held-out evaluation tasks, an evolved harness from HERA improves abstention accuracy from 61.7% to 83.3% while maintaining feasible task performance, outperforming all compared methods. The resulting best harness transfers across 19 other LLMs, improving abstention accuracy by 15.3 percentage points on average without any model-specific optimization, and enabling smaller models to match the performance of more powerful models at an estimated 85% lower cost.

## 1 INTRODUCTION

Large language model (LLM) agents are increasingly capable of solving complex tasks through multi-step interactions with tools and external environments (Merrill et al., 2026; Yao et al., 2022), yet their reliability still lags behind in many areas. Recent work on agent reliability has identified *agentic abstention* (Luo et al., 2026; Liu et al., 2026) as a key challenge: agents often fail to recognize when they should stop and instead continue acting even when a request is underspecified or the environment lacks the prerequisites for successful execution. This ability to recognize appropriate stopping is essential, as continuing to act when a task cannot or should not be completed can degrade task performance (Wan et al., 2026; Sui et al., 2026), incur unnecessary API costs and latency (Lou et al., 2026), and even lead to unintended or irreversible actions (Zhang et al., 2026a). Prior work has improved agent capability through (i) methods that optimize the model directly, e.g., through supervised fine-tuning, reinforcement learning, or online policy optimization (Agarwal et al., 2026; Chen et al., 2026b;a), and (ii) methods that keep the model fixed and instead optimizes the agent harness around it, which governs context construction, memory and skills, tool use, action validation, and the control logic for continuing, recovering, or stopping execution (Shao et al., 2026; Wang et al., 2026; Karten et al., 2026). The former has received considerable attention, whereas the role of harness optimization in agentic abstention remains largely unexplored.

Optimizing a harness against a fixed set of tasks can only expose a limited set of failures. As the harness adapts to failures exposed by those tasks, the same examples provide diminishing signal about where the harness still fails. Continued improvement requires the task distribution itself to adapt, evolving new cases that challenge existing capabilities. This idea parallels recent work on adaptive data generation, in which model performance is used to iteratively refine the data-generation process and expose new learning opportunities (Meng et al., 2026). For agentic abstention, we therefore consider a setting in which harness optimization and environment/task generation co-evolve, where agent failures shape improvements to the harness and elicit more challenging tasks for the agent.



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.

This is the primary motivation for our work. We introduce HERA, a framework for harness–environment co-evolution for agentic abstention. HERA proceeds in two stages. In Stage 1, building on methods from Dong et al. (2026), we construct executable environments and tasks with verified reference solutions. We then introduce a new environment-mutation procedure for constructing infeasible tasks, transforming feasible tasks into matched cases requiring abstention. Each pair of feasible-infeasible tasks shares the same task request, while the environment mutation determines whether the request can be completed. In Stage 2, using the task data construction procedure in Stage 1, HERA alternates environment evolution and new task construction with harness adaptation. For each round of co-evolution, we analyze paired execution trajectories to identify both failures to complete feasible tasks and failures to abstain on infeasible tasks. These failures guide targeted harness changes to environment states, tool behavior, or constraints, followed by new environment-task modification and validation. After each round, an updated harness provides failure feedback for the next round, allowing environment generation to respond to the agent's changing weaknesses and vice versa.

We use HERA to produce a challenging evaluation set for agentic abstention, which we call HERA-BENCH. We then execute HERA using GPT-5.6-Luna as our base task-execution model. We apply the final evolved harness across 19 other LLMs on HERA-BENCH, where it improves abstention and paired success by an average of 15.3 and 12.5 percentage points, respectively, without further model-specific optimizations. We further analyze inference costs for 14 models and find that, despite the additional overhead of the evolved harness, HERA can enable less expensive models to achieve comparable performance to more costly models at significantly lower overall cost.

We also compare HERA against baselines such as abstention prompting, harness-only optimization, and alternative task-expansion strategies. Each method's final harness is evaluated on our cumulative training pool and HERA-BENCH. Across cumulative training sets, HERA matches or outperforms all compared baselines, reaching 86.0% abstention success and 78.0% paired success on the final 50-pair pool. On HERA-BENCH, HERA improves abstention success from 61.7% to 83.3% and paired success from 48.3% to 70.0% relative to the base harness.

In summary, our key contributions are as follows:

- **Benchmark for agentic abstention.** As part of HERA, we introduce a controlled environment-mutation pipeline for constructing matched feasible–infeasible task pairs in executable tool-use environments, where infeasible instances require the agent to abstain. Using this pipeline, we construct HERA-BENCH, a benchmark of 60 verified task pairs for evaluating whether agents appropriately act when tasks are feasible and abstain when they are infeasible.
- **Harness–environment co-evolution.** HERA then combines verifiable task construction with failure-driven environment evolution and harness adaptation. On the final training pool, HERA achieves 86.0% abstention success and 78.0% paired success, exceeding the next-best baseline by 10.0 percentage points on both metrics. These results support the necessity of both harness and environment/task updates, as well as multiple iterations for both.
- **Transferable improvements in agentic abstention.** On held-out tasks in HERA-BENCH, the final harness from HERA increases abstention success from 61.7% to 83.3% and paired success from 48.3% to 70.0%. Across 19 additional LLMs, the same final harness improves these metrics by an average of 15.3 and 12.5 percentage points without further optimization. These findings indicate that harness improvements generalize beyond the model used during development.
- **Performance–cost trade-offs.** We analyze API costs and token usage across 14 models, examining the additional inference overhead of harness adaptation and differences between model configurations. HERA enables less expensive models to achieve performance gains approaching more costly models. For example, GPT-5.6-Luna with HERA matches the 70.0% paired success of base GPT-6-Astra at an estimated 85% lower API cost per episode.

## 2 Related Work

**Agentic Abstention.** LLM abstention is defined as a model's decision to refrain from answering under uncertainty (Wen et al., 2025; Kirichenko et al., 2025; Wen et al.; 2024). Existing studies of LLM abstention can be broadly grouped by why abstention is warranted: insufficient information in the query or available context (Cao et al., 2026; Kwiatkowski et al., 2019; Trivedi et al., 2022;

Reddy et al., 2019; Choi et al., 2018; Min et al., 2020; Zhang & Choi, 2021; Yin et al., 2023; Amayuelas et al., 2024; Jin et al., 2019; Dasigi et al., 2021), limitations in the model's knowledge due to temporal or factual gaps (Kasai et al., 2023; Yang et al., 2024; Mallen et al., 2023; Sciavolino et al., 2021; Feng et al., 2024), and safety or value concerns that make responding inappropriate (Han et al., 2024; Gehman et al., 2020; Hartvigsen et al., 2022; ElSherief et al., 2021; Lin et al., 2023; Ji et al., 2023; Xu et al., 2023; Röttger et al., 2024; Qiu et al., 2023; Shen et al., 2024; Wang et al., 2024; 2023; Li et al., 2024; Xie et al., 2024).

Recent work has begun to shift the study of abstention from LLMs to agent settings. Luo et al. (2026) first introduce the notion of agentic abstention, asking whether agents know when to stop instead of act, and evaluate this capability across QA, web, and terminal environments. They find that current agents often fail to identify when abstention is warranted. Liu et al. (2026) similarly study agentic abstention by constructing executable environment–task pairs, where controlled changes to tasks or environments create cases in which agents should abstain rather than act. Relatedly, (Lin et al., 2026b) study a closely related stopping problem through the lens of budget awareness, asking whether agents can estimate their remaining resource requirements and identify when task completion is unlikely within the available budget. However, the environments in these works are fixed or generated independently of agent performance, whereas we use agent failures to drive environment generation and the resulting environments to iteratively improve the agent harness.

**Harness Evolution.** Prior work improves agents by optimizing individual parts of the external scaffold, including prompts, demonstrations, and language-model programs. (Khattab et al., 2023; Opsahl-Ong et al., 2024; Yuksekgonul et al., 2025; Agrawal et al., 2026). More recent work treats the full agent harness as the object of optimization. Meta-Harness (Lee et al., 2026) searches over harness code using execution feedback, while Agentic Harness Engineering (Lin et al., 2026a) jointly evolves components such as prompts, tools, middleware, skills, and memory, and AEVO (Zhang et al., 2026c) optimizes the procedure and context that guide subsequent agent evolution. Recent approaches further explore harness improvement through failure-driven self-refinement (Zhang et al., 2026b), learned runtime editing (Shao et al., 2026), more reliable search and generalization (Zhang et al., 2026d), and population-based evolution (Zhang et al., 2026e). Task-CoEvolve (Miyai et al., 2026) additionally adapts the selection of validation tasks as the harness changes, while recent work has also questioned whether gains from harness evolution generalize beyond the tasks repeatedly used during optimization (Wang et al., 2026).

Unlike prior harness-evolution methods, which typically improve the harness against fixed environments, we evolve the harness together with the environment–task ecosystem, adapting the latter to target weaknesses exposed by the current harness. We apply this idea to agentic abstention.

# 3 Methodology: Harness-Environment Co-Evolution

In this section, we describe the two stages of HERA, (i) how we construct and verify feasible and infeasible tasks (§3.1) and (ii) how these task pairs are then used to adapt the harness and generate new and more challenging environments (§3.2).

## 3.1 Verifiable Abstention Environment–Task Construction

Our first stage constructs diverse and executable environment–task ecosystems from scratch. The pipeline consists of three steps: environment construction, task generation, and environment mutation (further details and illustration in Appendix A.1).

**Environment construction.** As in Dong et al. (2026), we begin with a diverse collection of environment themes, where each specifies a semantic domain from which an interactive environment is constructed. For each theme, we construct a topic-aligned structured database using a deep-research agent $A$ with fixed research policy and a set of research tools. The initial database is iteratively expanded and refined to accumulate more structured information. After database construction, we synthesize an executable interface over the resulting data. A bounded coding procedure, operating under a tool-generation policy, produces a candidate tool set. We subsequently validate the generated tools against the underlying database, filtering out tools that are non-executable or inconsistent with the database schema. The resulting executable environment is denoted by $\mathcal{E}_\theta$.

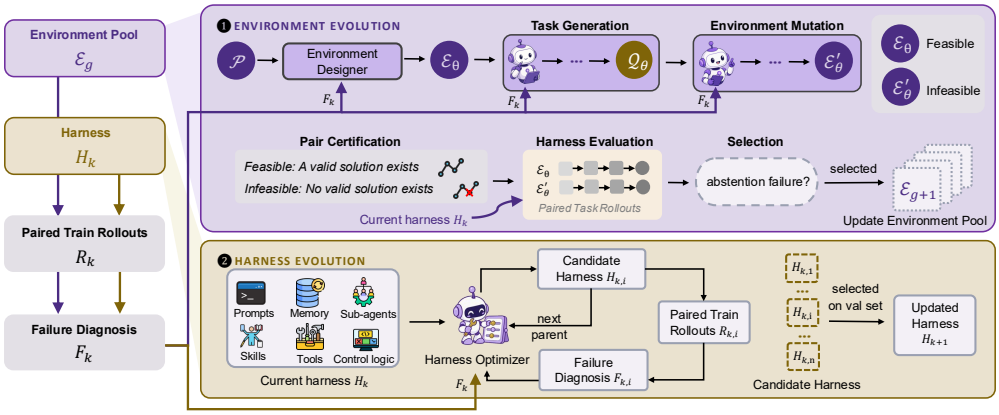


Figure 1: **Harness–environment co-evolution.** Given an environment pool of feasible and infeasible tasks and current harness, paired rollouts and failure diagnosis are used to iteratively evolve both the environment pool and harness. Environment evolution **(top)** constructs, validates, and selects new feasible–infeasible task pairs to challenge the current agent and expand the training pool. Harness evolution **(bottom)** similarly refines the harness iteratively on the expanded task pool, with each updated harness selected from candidates using performance on a validation set.

**Task generation.** Given a constructed environment $\mathcal{E}_\theta$, a task proposer generates candidate tasks grounded in the available data and tool capabilities. To avoid having the same model both propose and judge its own tasks, we use an independent solver to attempt each task through the public tool interface. We retain only tasks that can be successfully completed and validated in the environment and denote the resulting set of validated tasks by $\mathcal{Q}_\theta$. This step both filters out infeasible candidates and produces executable reference trajectories that are later used to assess how environment mutations change task feasibility.

**Environment mutation.** For each pair $(\mathcal{E}_\theta, \mathcal{Q}_\theta)$, we instantiate $N$ independent proposers to generate a diverse pool of candidate mutation proposals. Using multiple proposers reduces dependence on a single generation trajectory and increases the diversity of possible environment modifications. Candidate proposals are reviewed by two complementary policies. The *semantic policy* $\pi_{\text{sem}}$ evaluates whether a proposed mutation remains semantically consistent with the underlying task and environment, while the *rescue policy* $\pi_{\text{res}}$ examines whether the task can still be completed through an unintended alternative route despite the proposed modification. Together, these policies form a proposal-validation gate. Proposals that do not satisfy the validation criteria are sent through a proposal-refinement loop and reviewed again. Let $p_\theta^\star$ denote a proposal that successfully passes this review. The mutation operator $\mathcal{M}$ then applies it to the original environment:

$$\mathcal{E}'_\theta = \mathcal{M}(\mathcal{E}_\theta; p_\theta^\star)\,. \tag{1}$$

The resulting $\mathcal{E}'_\theta$ serves as the mutated environment for downstream abstention evaluation. See Figure 5 in Appendix A.1 for the mutation pipeline.

### 3.2 Harness–Environment Co-Evolution

**Motivation & overview.** An effective agent must be able to complete tasks when a valid solution exists and recognize when the task is infeasible. Our goal is to improve abstention on infeasible cases without sacrificing task-solving capability and, where possible, strengthen both. Following prior work on feedback-driven harness evolution (Lin et al., 2026a), we use execution trajectories and task outcomes to guide iterative updates to the agent's harness. HERA extends this process by coupling harness optimization with failure-guided evolution of executable environments and matched feasible–infeasible task pairs, allowing the training pool to adapt to the current harness. The verifiable environment–task pairs constructed in Section 3.1 serve as both training tasks and diagnostics for failures in task solving and abstention. These failures guide harness improvements and the targeted construction of new environment–task pairs, which in turn expose new weaknesses in the

updated harness. Repeating this process forms a harness–environment co-evolution loop, where the harness adapts to the challenges it encounters while the environments evolve to target its current weaknesses. Figure 1 provides an overview of the harness–environment co-evolution process.

At the beginning of each co-evolution round, we maintain an environment pool $\mathcal{E}_g$ and an agent equipped with the current harness $H_k$. Each member of the pool is a shared task instruction and its matched feasible and infeasible environments, corresponding to the **Act** and **Abstain** cases, respectively. We execute the agent on both tasks in the pair and collect the resulting trajectories $\mathcal{R}_k$. An LLM-based failure analyst examines these trajectories and their evaluation outcomes to identify failures in task completion and abstention. The analyst groups these failures by their underlying causes, producing a structured diagnosis $F_k$ that guides both environment evolution and harness evolution. Each round of environment and harness evolution proceeds as follows.

**Environment evolution.** For environments in which the current agent fails to abstain appropriately, an environment designer uses the failure diagnosis $F_k$ to modify the underlying data, tool behavior, or constraints of the parent environment $\mathcal{P}$. The redesigned environment is then passed to the task-generation and environment-mutation stages of Section 3.1. Importantly, $F_k$ also guides the proposers in both stages, so that environment design, task generation, and environment mutation jointly target the diagnosed weaknesses. The newly generated feasible–infeasible pairs undergo certification through the same verification checks as before, which confirm an executable solution for the feasible instance and infeasibility for the abstention instance.

This forms a new candidate batch of feasible–infeasible task pairs, which are then evaluated with the current harness version $H_k$. The batch is admitted to the next evolution round if the proportion of infeasible instances on which the agent fails to abstain appropriately is above a predefined threshold $\tau$. If so, we retain the entire batch, including some instances that the current harness already handles correctly, and add it to the cumulative training pool. If the batch does not meet the threshold, it is revised or replaced with a new candidate batch.

**Harness evolution.** Given the updated training pool $\mathcal{E}_{g+1}$, we first evaluate the agent with the current harness $H_k$ on all instances and derive a fresh failure diagnosis $F_{k,0}$. We then evolve the harness over $n$ update steps. The harness optimizer may modify prompts, memory, sub-agents, skills, tools, and control logic while keeping the underlying language model fixed.

Formally, we initialize $H_{k,0} = H_k$ and generate successive candidates as

$$H_{k,i+1} = \mathrm{Optimize}(H_{k,i}, F_{k,i}), \qquad i = 0, \ldots, n-1. \tag{2}$$

Each candidate harness is evaluated through paired training rollouts over the entire pool $\mathcal{E}_{g+1}$. The resulting trajectories and evaluation outcomes are analyzed to derive an updated failure diagnosis $F_{k,i+1}$, which guides the next update. Each update builds directly on the preceding candidate, regardless of whether that candidate improves performance. The training pool remains fixed throughout all $n$ updates.

After completing the sequence, we compare all candidates with the round-start harness $H_k$ on a fixed validation set. This set is disjoint from all cumulative training pools and from HERA-BENCH, and remains unchanged across co-evolution rounds. We use two metrics for this comparison: *Abstain*, the fraction of infeasible instances on which the agent correctly abstains, and *Pair*, the fraction of matched pairs for which the agent both successfully completes the feasible instance and correctly abstains on the infeasible instance. A candidate is eligible for adoption only if neither metric decreases relative to $H_k$ on the same validation set and at least one strictly improves. We select the best eligible candidate as $H_{k+1}$; if none qualifies, we retain $H_k$. Validation results are used only for final checkpoint selection and are not provided as feedback to the harness optimizer.

## 4 Experimental Setup

Our experiments address three questions: (1) Can harness–environment co-evolution improve agentic abstention without increasing refusals on feasible tasks? (2) How do harness evolution and environment evolution separately contribute to performance? And (3) as the task pool grows, do successive harnesses improve on new tasks while retaining performance on earlier ones?

**Training and evaluation data.** We initialize the training pool with 20 task-pairs (20 feasible, 20 infeasible) generated using our Stage 1 pipeline. During co-evolution, we introduce 30 additional

task-pairs (10 per iteration, 5 each of feasible and infeasible tasks) while retaining earlier tasks, expanding the cumulative training pool to 100 total tasks. We report performance on these cumulative training pools to track adaptation to new tasks and retention of performance on earlier tasks.

To assess generalization, we introduce **HERA-BENCH**, a held-out evaluation benchmark consisting of 60 task-pairs (60 feasible, 60 infeasible). HERA-BENCH is initially generated using our environment–task construction pipeline (§3.1) on domains that are disjoint from those used for training, and subsequently manually reviewed and revised by authors to make the resulting task–environment pairs more natural and realistic. The review covers scenario plausibility, consistency between task instructions and environment behavior, and the validity of each feasible–infeasible pair, as detailed in Appendix D.

**Harness.** Our base harness, $H_0$, is a minimal tool-calling agent built on the OpenAI Agents SDK (OpenAI, 2025), with environment tools exposed through MCP. It receives task instructions and alternates between model responses and tool calls until it produces a final response or reaches the 30-turn execution limit. Following prior work (Wang et al., 2026; Lin et al., 2026a), for each round of harness evolution, we allow edits to prompts, memory, harness-level tools, skills, control logic, and sub-agent configurations.

**Models.** We use GPT-5.6-Luna (OpenAI, 2026c) as our main task-execution model. We then evaluate the harness developed with GPT-5.6-Luna on 19 additional models (listed in Table 3). For each model, we compare the base harness $H_0$ with the final evolved harness on HERA-BENCH, without further model-specific optimization. All models are evaluated with the same environment tools and scoring protocol. The detailed model list, API providers, inference settings, and execution limits are provided in Appendix A.2.

**Baselines.** We compare HERA's co-evolution with five baselines: (i) $H_0$ uses the original harness without adaptation; (ii) $H_0$ + abstention prompt adds a fixed abstention instruction without further harness optimization; (iii) Harness-only evolution only evolves the harness as in §3.2 without environment evolution (using only the initial set of training tasks); (iv) Task-augmented harness evolution is like (iii) except the training pool is expanded each round with more valid task-pairs generated at baseline without any environment evolution; (v) Task generation from $H_0$ failures generates additional pairs using only $H_0$'s initial failures, then releases them in fixed batches for subsequent harness optimization. These baselines allow us to isolate the benefit of harness evolution (iii), environment evolution beyond simply adding more training tasks (iv), and continued co-evolution beyond the initial harness (v).

**Co-evolution parameters.** We conduct co-evolution over six rounds, allowing us to observe benefits from the evolution process and approach saturation. For each round of environment evolution, we fix the batch-admission threshold at $\tau = 0.4$ across rounds, i.e., the current harness must fail to abstain appropriately on at least two of the five infeasible instances. All tasks are retained upon acceptance.

**Evaluation metrics.** We report the following metrics:

- Abstention success (**Abstain**): percent of infeasible tasks on which the agent correctly abstains
- Paired success (**Pair**): percent of matched task-pairs for which the agent both successfully completes the feasible instance and correctly abstains on the infeasible instance
- Feasible task completion (**Act**): percent of feasible tasks that the agent successfully completes

We compare methods by reporting final harness performance on HERA-BENCH. We also conduct training dynamics analysis during co-evolution to track performance on each new task batch from its introduction onward, including measurements before and after each harness update. This separates learning on newly introduced tasks from changes in performance on earlier tasks.

## 5 RESULTS

Table 1 compares baselines and HERA on HERA-BENCH, while Table 2 shows final harness performance across each evolution step over the cumulative training pools. Table 3 reports transfer of harness improvements from HERA to 19 other LLMs. Additional statistics in Appendix E.2.

Table 2: Performance across cumulative training pools. Each cell reports **Abstain / Act** (%). $T_0$ denotes the initial training pool, and $T_1$–$T_6$ denote cumulative pools after successive task expansions. The same final harness for each method is evaluated across all pools.

| **Method** | $\mathbf{T_0}$ | $\mathbf{T_1}$ | $\mathbf{T_2}$ | $\mathbf{T_3}$ | $\mathbf{T_4}$ | $\mathbf{T_5}$ | $\mathbf{T_6}$ |
|---|---|---|---|---|---|---|---|
| Harness-only evolution | 90.0 / 80.0 | 84.0 / 76.0 | 76.7 / 70.0 | 77.1 / 71.4 | 70.0 / 67.5 | 66.7 / 66.7 | 64.0 / 66.0 |
| Task-augmented harness evolution | 90.0 / 85.0 | 88.0 / 84.0 | 80.0 / 80.0 | 80.0 / 80.0 | 75.0 / 77.5 | 75.6 / 77.8 | 74.0 / 76.0 |
| Task generation from $H_0$ failures | **95.0 / 90.0** | **96.0 / 92.0** | 86.7 / 86.7 | 85.7 / 88.6 | 82.5 / 85.0 | 77.8 / 84.4 | 76.0 / 82.0 |
| **Co-evolution (ours)** | **95.0 / 90.0** | **96.0 / 92.0** | **90.0 / 90.0** | **91.4 / 91.4** | **90.0 / 90.0** | **86.7 / 88.9** | **86.0 / 88.0** |

Table 3: Cross-model evaluation on HERA-BENCH. The final harness learned using HERA on the GPT-5.6-Luna base model is fixed and evaluated over 19 other closed- and open-weight models, demonstrating consistent gains on HERA-BENCH. Act results are reported in Appendix E.1.

| *Closed-weight models* | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Harness** | **GPT-5.6 Luna** | | **GPT-5.6 Terra** | | **GPT-5.6 Sol** | | **GPT-6 Astra** | | **Claude Opus 5** | | **Claude Sonnet 5** | | **Gemini 3.1 Pro (Preview)** | | **Gemini 3.8 Flash** | | **Gemini 3.5 Flash-Lite** | | **Grok 4.6** | |
| | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair |
| $H_0$ | 61.7 | 48.3 | 76.7 | 66.7 | 78.3 | 66.7 | 81.7 | 70.0 | 60.0 | 38.3 | 40.0 | 33.3 | 53.3 | 35.0 | 70.0 | 48.3 | 71.7 | 43.3 | 71.7 | 60.0 |
| **Final Harness** | **83.3** | **70.0** | **86.7** | **73.3** | **90.0** | **78.3** | **91.7** | **80.0** | **76.7** | **61.7** | **60.0** | **41.7** | **70.0** | **48.3** | **83.3** | **50.0** | **83.3** | **61.7** | **78.3** | **66.7** |
| *Open-weight models* | | | | | | | | | | | | | | | | | | | | |
| **Harness** | **DeepSeek V4.1-Flash** | | **GLM-5.2** | | **GLM-5.3** | | **GLM-5.3 Flash** | | **Kimi K3** | | **Qwen3.8 2.4T-A95B** | | **Qwen3.8 Flash** | | **MiniMax-M3** | | **Mistral Medium 3.5** | | **Mistral Large 3** | |
| | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair | Abstain | Pair |
| $H_0$ | 51.7 | 41.7 | 31.7 | 15.0 | 36.7 | 26.7 | 43.3 | 31.7 | 58.3 | 48.3 | 40.0 | 35.0 | 53.3 | 41.7 | 40.0 | 23.3 | 10.0 | 8.3 | 6.7 | 3.3 |
| **Final Harness** | **58.3** | **46.7** | **50.0** | **33.3** | **61.7** | **43.3** | **56.7** | **43.3** | **78.3** | **68.3** | **80.0** | **63.3** | **70.0** | **58.3** | **46.7** | **30.0** | **26.7** | **20.0** | **16.7** | **5.0** |

**Co-evolution improves abstention without sacrificing feasible-task success.** In Table 1, explicit abstention prompting improves abstention performance but lowers feasible task completion, whereas harness evolution improves both metrics relative to $H_0$. Co-evolution achieves the best performance across task-pairs (Abstain=83.3% and Act=76.7%), demonstrating the additional benefits from environmental evolution and no loss on feasible task performance.

Table 1: Comparison with baseline harness optimization strategies on the fixed evaluation set (%). All methods use the same base model and evaluation tasks.

| Method | Abstain ↑ | Act ↑ | Pair ↑ |
|---|---|---|---|
| *Static harness baselines* | | | |
| $H_0$ | 61.7 | 68.3 | 48.3 |
| $H_0$ + abstention prompt | 68.3 | 63.3 | 46.7 |
| *Harness optimization baselines* | | | |
| Harness-only evolution | 76.7 | 71.7 | 60.0 |
| Task-augmented harness evolution | 80.0 | 75.0 | 65.0 |
| Task generation from $H_0$ failures | 78.3 | 75.0 | **73.3** |
| **Co-evolution (ours)** | **83.3** | **76.7** | 70.0 |

**Co-evolution's advantage is clear on the added task batches.** In Table 2, co-evolution and task generation from $H_0$ failures achieve identical Abstain and Act scores in the first rounds $T_0$ and $T_1$, but diverge for later rounds. On $T_6$, co-evolution reaches 86.0% Abstain and 88.0% Act, compared with 76.0% and 82.0%, respectively, for generation from initial failures. This strongly supports the value of additional environment evolution and updating task generation using current harness feedback: addressing weaknesses beyond those identified by $H_0$.

**The evolved harness transfers across a broad range of model families.** We evaluate the harness developed with GPT-5.6-Luna on 19 other closed- and open-weight LLMs, demonstrating across-the-board improvements compared to the base harness $H_0$ Table 3 reports improvements in both Abstain and Pair across these models, with gains ranging from 6.7 to 40.0 percentage points in Abstain and from 1.7 to 28.3 percentage points in Pair. The benefits also extend to models with stronger baseline performance: GPT-6-Astra improves from 81.7% to 91.7% in Abstain and from 70.0% to 80.0% in Pair, despite already outperforming GPT-5.6-Luna under $H_0$. These results indicate that harness improvements developed through co-evolution can transfer across the evaluated model families, although the magnitude of the gains varies by model.

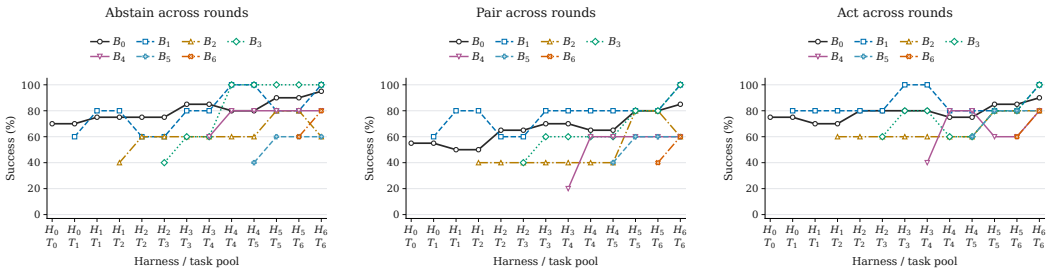


Figure 2: Performance on fixed task batches during co-evolution. From left to right, the panels show Abstain, Pair, and Act. Each curve tracks a fixed batch of tasks across harness updates. $B_0$ is the initial training batch, and $B_i$ ($i \geq 1$) contains the new tasks introduced in round $i$. These batches do not overlap, and $T_i = \bigcup_{j=0}^{i} B_j$ is the cumulative training pool. $H_i$ denotes the harness selected at the end of round $i$.

### 5.1 Co-evolution Dynamics

Figure 2 tracks performance on each task batch over successive harness and environment updates. Each line follows a fixed task batch, e.g., $B_0$ is the initial set of 20 task-pairs, and ($B_1$, $B_2$,...) each contain five new task-pairs introduced after the corresponding round of environment evolution.

**Environment evolution introduces tasks that challenge the current harness.** Newly generated task batches tend to begin at lower performance than earlier batches evaluated under the same harness, suggesting that environment evolution is successfully targeting weaknesses in the current harness. For example, $H_5$ performs substantially worse on the newly introduced $B_6$ than on the initial batch $B_0$ and most previous batches. After $B_6$ is incorporated into harness evolution, however, the update to $H_6$ increases both Abstain and Act performance on $B_6$ from 60.0% to 80.0%. This pattern where an initial drop on newly evolved tasks is followed by improvement through subsequent harness adaptation illustrates the intended co-evolutionary dynamic.

**Feasible task performance remains stable across harness evolution.** As the harness adapts to new tasks, Act performance on earlier batches is generally stable, with some fluctuations but a slight overall positive trend. HERA's evolution procedure does not explicitly target Act performance when generating new tasks. Nevertheless, successive harness updates do not degrade feasible-task performance (and may even improve it modestly), suggesting that gains in abstention can be achieved without sacrificing the agent's ability to complete feasible tasks.

### 5.2 Performance–Cost Trade-offs

Harness-environment optimization can make a lower-cost model competitive with a more expensive baseline. Figure 3 shows performance, API cost, and token usage across 14 models on HERA-BENCH. Across the 14 models, applying HERA increases API cost and token usage by median factors of 2.09 and 3.31, respectively. Despite this additional overhead within each model, multiple models achieve or approximate best model performance with the base harness (GPT-6-Astra) at substantially reduced cost, notably GPT-5.6-Luna and Gemini-3.5-Flash-Lite. For example, GPT-5.6-Luna with HERA achieves 70.0% paired success, matching GPT-6-Astra with the base harness, at an estimated average API cost of \$0.0091 versus \$0.0610 per episode, approximately 85% lower.

### 5.3 Analysis of Changes Introduced Through Co-Evolution

Figure 4 illustrates a representative sequence of harness updates produced during optimization. Each update targets a failure mode identified from execution trajectories and introduces or refines harness-level logic intended to address that failure. The example shows how the harness progresses from a minimal executor to increasingly explicit mechanisms for evidence-based stopping, cross-source evidence reconciliation, joint constraint verification, and authorization and privacy checking.

For environment evolution, we analyze newly introduced task pairs for each round. Environment evolution introduces qualitatively different challenges over successive rounds. Some evolved tasks

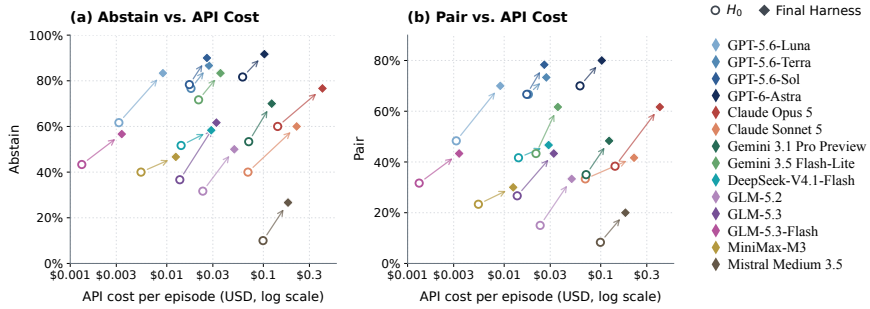


Figure 3: Abstention and paired success rates versus API cost on HERA-BENCH. Colors identify models; arrows connect $H_0$ (circles) to the fixed Final Harness (diamonds).

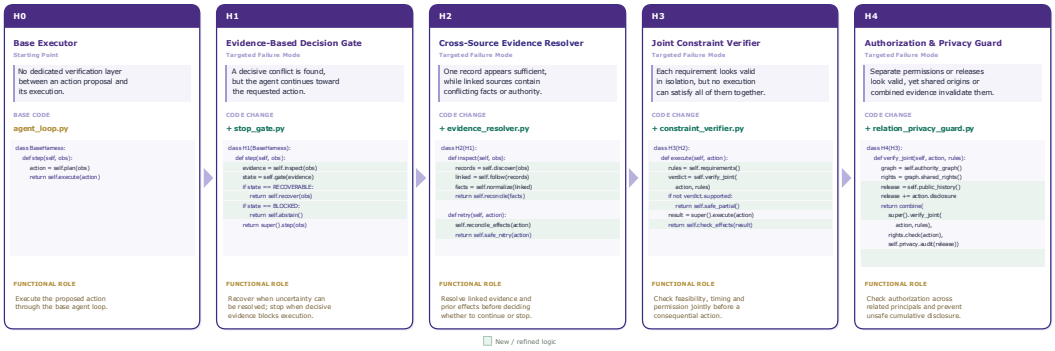


Figure 4: **Example of harness evolution across successive updates.** Starting from the base executor $H_0$, each subsequent harness introduces or refines logic in response to diagnosed failures. $H_1$ adds an evidence-based decision gate to distinguish recoverable uncertainty from conditions that warrant abstention; $H_2$ introduces cross-source evidence reconciliation; $H_3$ jointly verifies interacting task constraints before consequential actions; and $H_4$ extends this verification to authorization relationships and cumulative privacy constraints. Green regions indicate newly introduced or refined logic relative to the preceding harness.

increase the underlying task complexity, e.g., by adding interacting constraints or more entities and relations, while others primarily make the abstention decision more difficult by changing evidence availability, permissions, or whether restrictions apply. This emphasis also shifts across rounds. We provide a detailed breakdown of this analysis in Appendix E.3.

## 6 Conclusion

We introduce HERA, a framework for harness–environment co-evolution for agentic abstention. HERA constructs verifiable matched pairs of feasible and infeasible tasks through controlled environment mutations, and uses failures of the current harness to guide both subsequent environment–task generation and harness adaptation. This allows the training environment to evolve alongside the agent rather than remaining fixed throughout optimization. Our experiments show that HERA substantially improves abstention while maintaining feasible task completion over the base harness and all other baselines. The resulting harness generalizes to held-out task pairs and transfers across 19 additional LLMs without model-specific optimization, yielding broad improvements in both abstention and paired success. We hope HERA provides a step toward more reliable and adaptive evaluation and optimization of LLM agents.

## AI USE STATEMENT

LLMs were used as components of the proposed framework, as described in the paper. Generative AI tools were also used to assist with implementation, analysis, and manuscript editing. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the methodology, results, and final content of the paper.

## ETHICS STATEMENT

This work studies agentic abstention using synthetic tasks and controlled executable environments. The study does not involve human subjects or private personal data. All experiments were conducted in controlled settings, and the authors adhered to the ICLR Code of Ethics.

## REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the framework, task and environment construction, validation procedures, and evaluation protocol in the main paper and appendices. The appendices further document implementation details, model configurations, and experimental settings used in our experiments. These materials are intended to enable reproduction of the reported results.

## REFERENCES

Aradhye Agarwal, Gurdit Siyan, Yash Pandya, Joykirat Singh, Akshay Nambi, and Ahmed Awadallah. Learning when to act or refuse: Guarding agentic reasoning models for safe multi-step tool use. *arXiv preprint arXiv:2603.03205*, 2026.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Yang Wang. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6416–6432, 2024.

Anthropic. Claude Opus 5 system card, 2026a. URL `https://www.anthropic.com/claude-opus-5-system-card`. Accessed: 2026-09-21.

Anthropic. Claude Sonnet 5 system card, 2026b. URL `https://www.anthropic.com/claude-sonnet-5-system-card`. Accessed: 2026-09-21.

Zongwan Cao, Bingbing Wen, and Lucy Lu Wang. Clarify or answer: Reinforcement learning for agentic vqa with context under-specification. *arXiv preprint arXiv:2601.16400*, 2026.

Chaoran Chen, Vy Nguyen, Ziji Zhang, Abhinav Gullapalli, Ziyi Wang, Yuxuan Lu, Dakuo Wang, Jing Huang, Zhou Yu, and Jin Lai. Retry, switch, or abstain? learning strategy-aware tool-use policies via controlled error injection. *arXiv preprint arXiv:2608.11977*, 2026a.

Liuji Chen, Dianxing Tang, Xing Shi, Dingshuo Chen, Qiang Liu, Shu Wu, and Liang Wang. Learning when not to act: Mitigating tool abuse in agentic reinforcement learning. *arXiv preprint arXiv:2606.02132*, 2026b.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2174–2184, 2018.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, 2021.

DeepSeek-AI et al. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. *arXiv preprint arXiv:2609.19969*, 2026. URL https://arxiv.org/abs/2609.19969.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. Latent hatred: A benchmark for understanding implicit hate speech. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 345–363, 2021.

Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov. Don't hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14664–14690, 2024.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the association for computational linguistics: EMNLP 2020*, pp. 3356–3369, 2020.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3.1 Pro model card, 2026a. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/. Accessed: 2026-09-21.

Google DeepMind. Gemini 3.5 Flash-Lite model card, 2026b. URL https://deepmind.google/models/model-cards/gemini-3-5-flash-lite/. Accessed: 2026-09-21.

Google DeepMind. Gemini 3.8 Flash model card, 2026c. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/. Accessed: 2026-09-21.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in neural information processing systems*, 37:8093–8131, 2024.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3309–3326, 2022.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704, 2023.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents. *arXiv preprint arXiv:2605.09998*, 2026.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. Realtime qa: What's the answer right now? *Advances in neural information processing systems*, 36:49025–49043, 2023.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Kimi Team et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J Bell. Abstentionbench: Reasoning llms fail on unanswerable questions. *arXiv preprint arXiv:2506.09038*, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, and Pengyu Zhao. MiniMax Sparse Attention. *arXiv preprint arXiv:2606.13392*, 2026. URL https://arxiv.org/abs/2606.13392.

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.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3923–3954, 2024.

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

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, et al. Bagen: Are llm agents budget-aware? *arXiv preprint arXiv:2606.00198*, 2026b.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4694–4702, 2023.

Xun Liu, Yi Evie Zhang, Vira Kasprova, Parisa Rabbani, Pardis Sadat Zahraei, Tianyu Zhang, Ali Ebrahimpour-Boroojeny, and Varun Chandrasekaran. Agentabstain: Do llm agents know when not to act? *arXiv preprint arXiv:2607.10059*, 2026.

Renze Lou, Baolin Peng, Wenlin Yao, Qianhui Wu, Hao Cheng, Suman Nath, Wenpeng Yin, and Jianfeng Gao. The tool illusion: Rethinking tool use in web agents. *arXiv preprint arXiv:2604.03465*, 2026.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act? *arXiv preprint arXiv:2606.28733*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Yan Meng, Dhruv Srikanth, Bingchen Zhao, Zhen Ming Jiang, and Yuxiang Wu. Autodata: Agentic search for pre-training data selection. 2026.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 5783–5797, 2020.

Mistral AI. Introducing Mistral 3, 2025. URL https://mistral.ai/news/mistral-3/. Accessed: 2026-09-21.

Mistral AI. Mistral Medium 3.5, 2026. URL https://docs.mistral.ai/models/mistral-medium-3-5-26-04. Accessed: 2026-09-21.

Atsuyuki Miyai, Kiyoharu Aizawa, and Toshihiko Yamasaki. Task-coevolve: Efficient harness optimization via adaptive validation task selection. *arXiv preprint arXiv:2608.20169*, 2026.

OpenAI. OpenAI Agents SDK. Open-source software, 2025. URL https://github.com/openai/openai-agents-python. Initial release: March 11, 2025.

OpenAI. GPT-5.6 system card, 2026a. URL https://deploymentsafety.openai.com/gpt-5-6. Accessed: 2026-09-21.

OpenAI. GPT-6 Astra system card, 2026b. URL https://deploymentsafety.openai.com/gpt-6-astra. Accessed: 2026-09-21.

OpenAI. GPT-5.6 Luna, 2026c. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Released July 9, 2026.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Huachuan Qiu, Shuai Zhang, Anqi Li, Hongliang He, and Zhenzhong Lan. Latent jailbreak: A benchmark for evaluating text safety and output robustness of large language models. *arXiv preprint arXiv:2307.08487*, 2023.

Qwen Team. Qwen3.8-2.4T-A95B model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.8-2.4T-A95B. Accessed: 2026-09-21.

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026b. URL https://qwen.ai/blog?id=qwen3.8-flash-next.

Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5377–5400, 2024.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6138–6148, 2021.

Shuai Shao, Kangning Zhang, Qingyao Li, Shijian Wang, Hao Wang, Wenxiang Jiao, Yuan Lu, Yi Guo, Weiwen Liu, and Weinan Zhang. Harness-r1: Learning to edit executable runtime harnesses from agent failure trajectories. *arXiv preprint arXiv:2608.02276*, 2026.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1671–1685, 2024.

Yuan Sui, Yulin Chen, Yibo Li, Xue Jiang, Yufei He, Yihong Dong, Xiaoxin He, Tianyu Gao, and Bryan Hooi. Tact: Mitigating overthinking and overacting in coding agents via activation steering. *arXiv preprint arXiv:2605.05980*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. Compass: Enhancing agent long-horizon reasoning with evolving context. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3360–3380, 2026.

Wenxuan Wang, Zhaopeng Tu, Chang Chen, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R Lyu. All languages matter: On the multilingual safety of large language models. *arXiv preprint arXiv:2310.00905*, 2023.

Yike Wang, Huaisheng Zhu, Zhengyu Hu, Yige Yuan, Zhengyu Chen, Shakti Senthil, Hannaneh Hajishirzi, Yulia Tsvetkov, Pradeep Dasigi, and Teng Xiao. Rethinking the evaluation of harness evolution for agents. In *COLM 2026 The 2nd Workshop on Lifelong Agents: Learning, Aligning, and Evolving*, 2026.

Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. Do-not-answer: Evaluating safeguards in llms. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 896–911, 2024.

Bingbing Wen, Faeze Brahman, Zhan Su, Shangbin Feng, Yulia Tsvetkov, Lucy Lu Wang, and Bill Howe. Marvel: Modular abstention for reliable and versatile expert llms. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*.

Bingbing Wen, Bill Howe, and Lucy Lu Wang. Characterizing llm abstention behavior in science qa with context perturbations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3437–3450, 2024.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556, 2025.

xAI. Model card: Grok 4.6, 2026. URL `https://media.x.ai/v1/website/card-4p6-4cd2dc57.pdf`. Accessed: 2026-09-21.

Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, et al. Sorry-bench: Systematically evaluating large language model safety refusal. *arXiv preprint arXiv:2406.14598*, 2024.

Guohai Xu, Jiayi Liu, Ming Yan, Haotian Xu, Jinghui Si, Zhuoran Zhou, Peng Yi, Xing Gao, Jitao Sang, Rong Zhang, et al. Cvalues: Measuring the values of chinese large language models from safety to responsibility. *arXiv preprint arXiv:2307.09705*, 2023.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty. *Advances in Neural Information Processing Systems*, 37:63565–63598, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuan-Jing Huang. Do large language models know what they don't know? In *Findings of the association for Computational Linguistics: ACL 2023*, pp. 8653–8665, 2023.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Chiyu Zhang, Huiqin Yang, Bendong Jiang, Xiaolei Zhang, Yiran Zhao, Ruyi Chen, Lu Zhou, Xiaogang Xu, Jiafei Wu, Liming Fang, et al. Litmus: Benchmarking behavioral jailbreaks of llm agents in real os environments. *arXiv preprint arXiv:2605.10779*, 2026a.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026b.

Jiayi Zhang, Yongfeng Gu, Jianhao Ruan, Maojia Song, Yiran Peng, Zhiguang Han, Jinyu Xiang, Zhitao Wang, Caiyin Yang, Yixi Ouyang, et al. Harnessing agentic evolution. *arXiv preprint arXiv:2605.13821*, 2026c.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, et al. Harnesscompass: Guiding automatic harness evolution toward generalizable and effective agent harnesses. *arXiv preprint arXiv:2608.01918*, 2026d.

Michael Zhang and Eunsol Choi. Situatedqa: Incorporating extra-linguistic contexts into qa. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7371–7387, 2021.

Yifan Zhang, Yutong Dai, Juntao Tan, Luyu Yang, Rishi Mullur, Thai Hoang, Zhiyuan Hu, James Zhu, Phil Mui, Silvio Savarese, et al. Darwinx: Evolving agent harnesses through natural selection. *arXiv preprint arXiv:2608.07545*, 2026e.

# A Implementation and Reproducibility Details

## A.1 Environment–Task Construction Pipeline

Figure 5 summarizes the Stage 1 pipeline used to construct verified feasible–infeasible task pairs. Starting from a set of environment themes, we first synthesize executable environments by constructing a structured local database and generating a corresponding tool interface, which is validated for executability and consistency with the underlying data. We then generate candidate tasks within each environment and use use GPT-5.6 Sol as an independent solver to produce and verify reference solutions, retaining only tasks that can be successfully completed. Finally, for each validated task, we generate controlled environment mutations intended to make the original request infeasible without changing the request itself. Candidate mutations are iteratively reviewed and refined using semantic checks and searches for alternative solution paths before being applied to the environment, yielding matched feasible–infeasible pairs for downstream abstention evaluation.

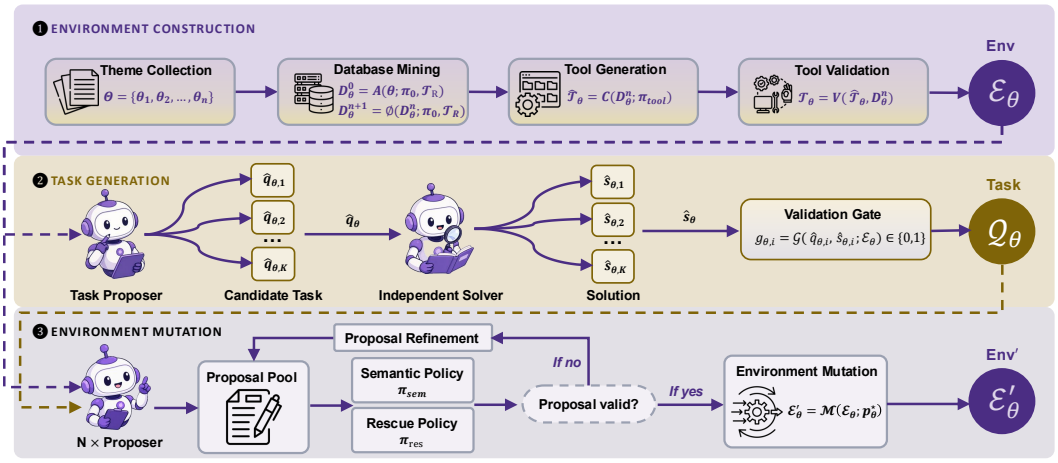


Figure 5: **End-to-End Environment–Task Construction Pipeline.** The pipeline consists of three stages: environment construction, task generation, and environment mutation.

### A.2 Models and Inference Settings

**Models and API access.** We use GPT-5.6-Luna as the reference task-execution model during harness development and in the main experiments, and evaluate the same final harness on 19 additional models. We evaluate GPT-5.6-Terra (OpenAI, 2026a), GPT-5.6-Sol (OpenAI, 2026a), GPT-6-Astra (OpenAI, 2026b), Claude Opus 5 (Anthropic, 2026a), Claude Sonnet 5 (Anthropic, 2026b), Gemini 3.1 Pro (Preview) (Google DeepMind, 2026a), Gemini 3.8 Flash (Google DeepMind, 2026c), Gemini 3.5 Flash-Lite (Google DeepMind, 2026b), Grok 4.6 (xAI, 2026), DeepSeek-V4.1-Flash (DeepSeek-AI et al., 2026), GLM-5.2 (GLM-5-Team et al., 2026), GLM-5.3 (GLM-5-Team et al., 2026), GLM-5.3-Flash (GLM-5-Team et al., 2026), Kimi K3 (Kimi Team et al., 2026), Qwen3.8-2.4T-A95B (Qwen Team, 2026a), Qwen3.8 Flash Qwen Team (2026b), MiniMax-M3 (Lai et al., 2026), Mistral Medium 3.5 (Mistral AI, 2026), and Mistral Large 3 (Mistral AI, 2025). Table 4 reports the API configurations used for each model. All models are used without fine-tuning. Luna uses the configured Responses API interface, DeepSeek-V4.1-Flash uses the official DeepSeek endpoint, and the remaining models are accessed through OpenRouter with specified provider routes and automatic fallback disabled.

**Inference settings and execution budgets.** For each model, $H_0$ and the final harness use the same requested inference configuration and execution limits. Temperature and top-$p$ use the API defaults, while reasoning settings follow Table 4. Each task variant is evaluated with at most 30 agent turns and 8,192 output tokens per model response.

**Agent interface and task isolation.** For each model and task variant, both harnesses receive the same public task instructions and environment-tool definitions, with identical initial environment states. They interact with the environment through native function calling. Each episode starts in a fresh environment with reset harness state. Reference answers, private acceptance rules, and task-side labels are not exposed to the solver.

### A.3 HERA Optimization Details

We specify the following fully automated optimization protocol for HERA, alternating task-pool expansion with harness optimization.

**Initialization and expansion schedule.** The protocol starts from the base harness $H_0$ and an initial training pool $T_0$ containing 20 validated task pairs. Each of six expansion rounds targets five additional pairs, while retaining all previously accepted pairs. The target size of the cumulative pool is therefore $|T_i| = 20 + 5i$, reaching 50 pairs at $T_6$. A separate validation set remains fixed throughout co-evolution and is used for checkpoint selection.

Table 4: Models and inference configurations. "Default" indicates that no reasoning parameter is explicitly supplied. OpenRouter provider identifiers are shown for routed models.

| Model | Requested API identifier | Serving route | Reasoning |
|---|---|---|---|
| GPT-5.6-Luna | `gpt-5.6-luna` | Responses API | Default |
| GPT-5.6-Terra | `openai/gpt-5.6-terra` | `openai` | High |
| GPT-5.6-Sol | `openai/gpt-5.6-sol` | `openai` | High |
| GPT-6-Astra | `openai/gpt-6-astra` | `openai` | High |
| Claude Opus 5 | `anthropic/claude-opus-5` | `anthropic` | High |
| Claude Sonnet 5 | `anthropic/claude-sonnet-5` | `anthropic` | High |
| Gemini 3.1 Pro Preview | `google/gemini-3.1-pro-preview` | `google-ai-studio` | High |
| Gemini 3.8 Flash | `google/gemini-3.8-flash` | `google-ai-studio` | High |
| Gemini 3.5 Flash-Lite | `google/gemini-3.5-flash-lite` | `google-ai-studio` | High |
| Grok 4.6 | `x-ai/grok-4.6` | `xai` | High |
| DeepSeek-V4.1-Flash | `deepseek-flash` | DeepSeek direct API | High |
| GLM-5.2 | `z-ai/glm-5.2` | `z-ai/fp8` | High |
| GLM-5.3 | `z-ai/glm-5.3` | `z-ai/fp8` | High |
| GLM-5.3-Flash | `z-ai/glm-5.3-flash` | `z-ai/fp8` | High |
| Kimi K3 | `moonshotai/kimi-k3` | `moonshotai/mxfp4` | High |
| Qwen3.8-2.4T-A95B | `qwen/qwen3.8-2.4t-a95b` | `alibaba` | High |
| Qwen3.8 Flash | `qwen/qwen3.8-flash` | `alibaba` | Enabled |
| MiniMax-M3 | `minimax/minimax-m3` | `minimax/fp8` | Enabled |
| Mistral Medium 3.5 | `mistralai/mistral-medium-3-5` | `mistral` | High |
| Mistral Large 3 | `mistralai/mistral-large-2512` | `mistral` | Default |

**Failure-guided task generation.** At each round, the current harness is evaluated on the current training pool. Failures on solvable and should-abstain variants are analyzed separately to identify limitations in task execution and abstention decisions. The generation agent uses this training feedback to propose new environments and matched task pairs. Proposals undergo automated execution and semantic checks, followed by evaluation with the current harness. A batch of five certified pairs is admitted if the current harness fails to abstain appropriately on at least two of its five infeasible instances ($\tau = 0.4$). All five pairs in an accepted batch are retained.

**Harness candidate generation.** After expansion, the training pool is fixed for the optimization block. Starting from the retained harness $H^{(0)}$, a coding agent generates four sequential candidates, $H^{(1)}, \ldots, H^{(4)}$. Each update uses failure feedback from paired training rollouts of the preceding checkpoint. Subsequent candidates therefore build on the preceding candidate rather than an intermediate validation-selected checkpoint. The editor may modify prompts, episode memory, internal tools, and control logic, while environment tools and scoring rules remain fixed within the block. Each candidate must pass interface and execution checks. The initial checkpoint and all four candidates are saved for subsequent selection.

**Candidate evaluation and selection.** After all four updates, the initial checkpoint and the four candidates are evaluated on the same fixed validation set using GPT-5.6-Luna, with one rollout per task variant. Selection uses validation Abstain and Pair jointly. A candidate replaces the retained checkpoint only if neither validation metric decreases and at least one improves. Otherwise, the retained checkpoint is kept. Selection occurs after candidate generation and does not alter the within-block candidate lineage. The selected checkpoint becomes the current harness for the next co-evolution round. Act is recorded for analysis but is not used for selection. An expansion round therefore does not necessarily produce a harness update.

**Termination and final evaluation.** Generation, validation, revision, and candidate selection proceed automatically. If the generation budget is exhausted before five valid pairs are admitted, the round is recorded as incomplete. After the final optimization block, the checkpoint selected on the fixed validation set is frozen. The proposed protocol reserves HERA-BENCH tasks and trajectories for final assessment, excluding them from task generation, harness revision, and checkpoint selection. The same frozen harness is applied across evaluation models without further adaptation.

### A.4 Harness Representation and Editable Components

**Harness representation.** Each harness is represented as a versioned bundle of natural-language instructions, configuration files, and executable Python modules. Fixed runtime interfaces connect these components to instruction composition, tool-call review, observation processing, and final-response handling.

**Editable components.** The optimizer may add, remove, or revise prompts, memory management, harness-level tools, skills, control logic, and sub-agent configurations. These component types define the permitted search space; individual candidates need not use every type. Edits must preserve the runtime interfaces through which the harness is executed.

**Editing boundaries.** During harness optimization, environment-tool implementations and semantics, task specifications, scoring rules, data splits, and execution budgets remain outside the editable scope. Harness-level tools may process information available to the agent, while environment interactions remain mediated by the shared runtime.

## B Environment–Task Construction and Validation

### B.1 Environment Construction

**State construction.** The construction protocol uses frozen source documentation to identify domain entities, relations, workflows, and operational constraints. A generation agent constructs a structured initial state, distinguishing source-supported rules from simulated records and assumptions.

**Tool generation.** A coding agent implements Python tools over this state, with public descriptions, input and output schemas, and declared error behavior and state effects. Discovery and lookup operations expose resource identifiers needed by subsequent calls. Tool responses are computed from the environment state. Read-only operations preserve state, while state-changing operations implement their declared preconditions and effects.

**Validation and repair.** Automated checks examine schema and relationship consistency, public resource discovery, and tool behavior on valid and invalid inputs. State checks detect unintended writes and partial updates after failed calls. Coding-agent review identifies implementation defects, which are repaired and checked again before acceptance. The accepted initial state, tool definitions, and implementation are versioned together for subsequent task generation.

### B.2 Feasible Task Generation and Verification

**Task generation.** For each accepted environment, a task proposer formulates a natural user goal grounded in its supported workflows. The task specifies the desired outcome and relevant constraints without revealing the answer or prescribing a solution procedure. Semantic review checks that the goal is coherent, its requirements are supported by the environment, and the expected output is clearly defined.

**Public-tool execution.** A separate solver attempts the task in a fresh environment instance using only the public task instruction, tool definitions, and observations obtained through tool calls. The controller records the tool calls, observations, final response, and resulting environment state as a candidate reference execution. This execution provides a concrete witness that the task can be completed through the public interface.

**Verification and acceptance.** Verification rules are authored from the task and environment specification before the solver runs, and reviewed for consistency with the task requirements. They check the final answer and any required state effects, allowing alternative valid execution paths. The recorded execution is replayed from the initial state, and incorrect or incomplete outcomes are used to test whether the verifier rejects them. A candidate is accepted only after successful execution, semantic review, and replay. Failures trigger revision and revalidation or rejection.

### B.3 Environment Mutation and Pair Certification

**Controlled environment mutation.** Starting from a verified solvable task, the protocol constructs a paired instance by changing a task-relevant condition in the environment state or tool behavior. The change may affect required evidence, the consistency of authoritative records, or authorization for an action. The paired instances retain the same task instruction, system prompt, and public tool interface. The grounds for abstention must be discoverable through tool observations.

**Abstention verification.** For each candidate, the certifier identifies the affected requirement and the evidence supporting abstention. It examines alternative tools, recovery paths, and other permitted ways to satisfy the request. A valid completion path invalidates the should-abstain label. An unsuccessful solver rollout alone is insufficient: the label requires a reason grounded in the task constraints and the environment's available capabilities.

**Pair certification.** Certification checks that the mutation is confined to its declared scope and that the original instance remains solvable. Reversing the mutation must restore successful execution, while nonessential-change controls check that irrelevant differences preserve feasibility. A separate semantic review assesses whether the observed evidence supports the assigned labels and whether runtime defects or ambiguous requirements explain the apparent failure. Pairs with unresolved validity issues are revised or rejected. Accepted pairs retain private records of the mutation, supporting evidence, and certification results; these records are not exposed to the evaluated agent.

### B.4 Failure-Guided Environment–Task Evolution

**Failure diagnosis.** The current harness is evaluated on the cumulative training pool, and failures on solvable and should-abstain tasks are analyzed separately. Diagnosis uses tool calls, observations, final responses, and evaluation feedback to identify likely causes of execution and abstention errors.

**Failure-guided generation.** The proposer uses this diagnosis to select a parent environment–task instance and specify the target failure mechanism, environment changes, and task goal. Changes may affect evidence dependencies, authorization scope, shared constraints, or recovery paths. The resulting candidates undergo the environment construction, task verification, and pair certification procedures described above. Only training feedback is used for diagnosis and generation; validation and Eval tasks and trajectories are excluded.

**Screening and training-pool expansion.** After certification, candidate pairs are organized into batches of five. Both variants of every pair are evaluated with the current harness held fixed. Trajectory review checks that observed failures are not caused by task defects, runtime errors, or scoring ambiguities; batches with unresolved issues are revised and re-screened rather than admitted.

A batch is admitted if at least two of its five infeasible instances expose confirmed failures to abstain appropriately ($\tau = 0.4$). The failure rate is computed over infeasible instances only, not over both variants or pair-level failures. All five certified pairs in an accepted batch are appended to the training pool, including pairs already handled correctly by the current harness. Previously accepted pairs are retained, and the expanded pool supplies feedback for the next harness optimization block.

## C Dataset and Evaluation Details

### C.1 Datasets and Splits

The training pool starts with 20 task pairs and expands to 50 through six additions of five pairs. Training trajectories provide feedback for environment–task generation and harness optimization. The protocol uses a fixed validation set of 15 pairs for checkpoint selection. HERA-BENCH contains 60 pairs and serves as the common benchmark for comparing the selected harnesses across models.

Each pair contains a solvable instance and a should-abstain instance sharing the same public task instruction and tool interface. The intended split protocol keeps both variants and their direct derivatives in the same partition. Task-level disjointness does not imply disjoint environment families or source domains; these require separate grouping and overlap checks.

Table 5: Cross-model task completion and confirmed over-abstention on HERA-BENCH (%). The final harness is fixed across models.

| *Closed-weight models* | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Harness** | **GPT-5.6 Luna** | | **GPT-5.6 Terra** | | **GPT-5.6 Sol** | | **GPT-6 Astra** | | **Claude Opus 5** | | **Claude Sonnet 5** | | **Gemini 3.1 Pro (Preview)** | | **Gemini 3.8 Flash** | | **Gemini 3.5 Flash-Lite** | | **Grok 4.6** | |
| | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ |
| $H_0$ | 68.3 | 0.0 | 83.3 | 0.0 | 78.3 | 0.0 | 81.7 | 0.0 | 61.7 | 3.3 | 70.0 | 0.0 | 70.0 | 0.0 | 60.0 | 0.0 | 61.7 | 0.0 | 81.7 | 0.0 |
| **Final Harness** | 76.7 | 1.7 | 78.3 | 0.0 | 81.7 | 0.0 | 81.7 | 0.0 | 78.3 | 0.0 | 66.7 | 0.0 | 70.0 | 0.0 | 56.7 | 0.0 | 70.0 | 0.0 | 80.0 | 0.0 |
| *Open-weight models* | | | | | | | | | | | | | | | | | | | | |
| **Harness** | **DeepSeek V4.1-Flash** | | **GLM-5.2** | | **GLM-5.3** | | **GLM-5.3 Flash** | | **Kimi K3** | | **Qwen3.8 2.4T-A95B** | | **Qwen3.8 Flash** | | **MiniMax-M3** | | **Mistral Medium 3.5** | | **Mistral Large 3** | |
| | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ | Act ↑ | OA ↓ |
| $H_0$ | 73.3 | 0.0 | 56.7 | 1.7 | 68.3 | 0.0 | 58.3 | 1.7 | 76.7 | 0.0 | 73.3 | 0.0 | 68.3 | 0.0 | 56.7 | 3.3 | 61.7 | 0.0 | 21.7 | 13.3 |
| **Final Harness** | 75.0 | 0.0 | 71.7 | 1.7 | 71.7 | 0.0 | 78.3 | 1.7 | 85.0 | 0.0 | 76.7 | 1.7 | 70.0 | 1.7 | 56.7 | 0.0 | 60.0 | 0.0 | 20.0 | 15.0 |

### C.2 Evaluation Metrics

For $N$ matched pairs, let $c_i \in \{0,1\}$ indicate successful completion of the solvable instance, $b_i \in \{0,1\}$ indicate correct abstention on the should-abstain instance, and $o_i \in \{0,1\}$ indicate explicit abstention from all or a required part of the solvable task. We report

$$\text{Abstain} = \frac{100}{N}\sum_{i=1}^{N} b_i, \qquad \text{Pair} = \frac{100}{N}\sum_{i=1}^{N} c_i b_i,$$

$$\text{Act} = \frac{100}{N}\sum_{i=1}^{N} c_i, \quad \text{Over-abstain} = \frac{100}{N}\sum_{i=1}^{N} o_i.$$

Abstain and Pair are the primary metrics. Pair requires correct behavior on both instances of the same pair. Act and Over-abstain provide complementary diagnostics. Over-abstain is not the complement of Act, since incorrect answers and unsuccessful actions can occur without abstention.

## D HERA-BENCH Review and Revision

Authors review each generated task pair to check that the request is clear, the scenario is plausible, and both versions have the intended feasibility.

**Task clarity and realism.** We check whether the request makes sense in its setting and whether the available records, tools, and restrictions have a reasonable explanation. We revise unclear wording, add missing context, and check that names, dates, units, and permissions are consistent with the environment. Task instructions describe what the user wants without revealing the reference solution or labeling the task as feasible or infeasible.

**Checking both versions.** For the feasible version, we run a reference solution and check that it meets all task requirements. For the infeasible version, we identify what prevents completion and check whether another allowed approach could still satisfy the request. A model's failure alone does not establish that a task is infeasible. The two versions must share the same user goal, with their difference in feasibility explained by the specified change to the environment.

## E Additional Experimental Results

### E.1 Feasible-Task Success and Over-Abstention

Table 5 reports Act and Over-abstain for $H_0$ and the final harness on HERA-BENCH. These results complement the primary metrics by examining whether improved abstention on should-abstain tasks is accompanied by reduced completion or increased refusal on solvable tasks.

### E.2 Paired Success across Cumulative Training Pools

Table 6 reports Pair for the same final harnesses and cumulative training pools as Table 2, complementing the Abstain and Act results in the main text.

Table 6: Paired success (Pair, %) using the same final harnesses and cumulative training pools as in Table 2.

| **Method** | $\mathbf{T_0}$ | $\mathbf{T_1}$ | $\mathbf{T_2}$ | $\mathbf{T_3}$ | $\mathbf{T_4}$ | $\mathbf{T_5}$ | $\mathbf{T_6}$ |
|---|---|---|---|---|---|---|---|
| Harness-only evolution | 75.0 | 68.0 | 60.0 | 60.0 | 52.5 | 48.9 | 46.0 |
| Task-augmented harness evolution | 75.0 | 72.0 | 66.7 | 65.7 | 60.0 | 60.0 | 58.0 |
| Task generation from $H_0$ failures | **85.0** | **88.0** | 80.0 | 80.0 | 75.0 | 71.1 | 68.0 |
| **Co-evolution (ours)** | **85.0** | **88.0** | **83.3** | **85.7** | **82.5** | **80.0** | **78.0** |

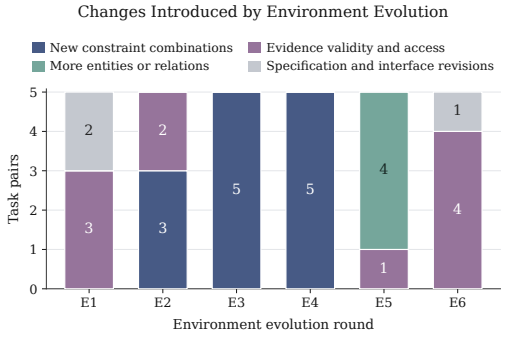


Figure 6: Primary changes across rounds of environment evolution. Each bar summarizes the five task pairs retained in that round.

### E.3 How Environments Evolve

To examine what changes during environment evolution, we compare the 30 task pairs retained across six rounds with their parents. We group each pair by its primary change: (1) New constraint combinations introduce conditions that must be considered together, such as scheduling dependencies, shared resource limits, or the privacy consequences of combining information. (2) Evidence validity and access covers changes to which records and permissions apply, whether required sources are available, and whether earlier problems have been resolved. (3) More entities or relations expands the number of entities or connections between them while retaining the underlying rules. (4) Specification and interface revisions covers changes to task requirements, expected outputs, tool interfaces, or how actions update the environment.

Figure 6 shows that the emphasis changes across rounds. E3–E4 introduce interacting constraints that require planning or inference before deciding whether to proceed. E5 mainly expands existing problems, while E6 focuses on whether evidence is available and apparent restrictions still apply.

## F Qualitative Analysis and Examples

### F.1 Environment-Task Examples

Figures 7 and 8 illustrate a matched weather task pair with identical task instructions and public tool interfaces. In the solvable instance, the retrieved forecast establishes that none of the postponement conditions is met. The paired should-abstain instance changes only the availability of the Saturday daytime precipitation probability. The closest station and the remaining forecast evidence can still be established, but the missing probability prevents a definitive postponement decision: the available evidence does not determine whether the precipitation threshold is exceeded. The appropriate response is therefore to report the supported findings and abstain from the unresolved decision.

Weather Task

***Environment***
A frozen, read-only Los Angeles weather snapshot containing:

- a resolved Los Angeles weather point;
- county-linked weather stations;
- official forecast periods for the location.

***Task Description***
You are planning outdoor field work in Los Angeles on Saturday, August 1, 2026. Identify the closest available weather station to the Los Angeles weather point so that the field team has a nearby monitoring reference, and summarize the official daytime forecast for that location. The team must postpone the work if the forecast temperature reaches 95°F, the chance of precipitation exceeds 40%, or winds may exceed 20 mph. Report the selected station and its distance from the weather point, the relevant daytime forecast evidence, whether the work should be postponed, and which conditions—if any—triggered that decision.

***Public Tools***

- resolve_weather_point(location)
  Resolves a location to its coordinates, county, forecast resource, and time zone.
- list_station_ids_by_county(county_id)
  Lists all available weather stations in the resolved county.
- get_station(station_id)
  Returns the name, coordinates, elevation, and region metadata of a station.
- get_forecast_periods(forecast_id)
  Returns all forecast periods for the resolved weather point.

***Reference Execution***
1. resolve_weather_point("Los Angeles, CA")
→ coordinates: $(34.0522, -118.2437)$
→ county: CAC037
→ forecast resource: LOX/155,45
→ time zone: America/Los_Angeles
2. list_station_ids_by_county(CAC037)
→ [000SE, 001SE]
3. get_station(000SE)
→ SCE South Hills Park
→ coordinates: $(34.11814, -117.85520)$
4. get_station(001SE)
→ SCE Glen Way
→ coordinates: $(34.12550, -117.73333)$
5. get_forecast_periods(LOX/155,45)
→ Saturday daytime
→ temperature: 93°F
→ precipitation probability: 0%
→ wind: 0–10 mph

***Derived Decision***
Distance to 000SE: 36.5 km
Distance to 001SE: 47.7 km

Closest station: 000SE — SCE South Hills Park

Temperature: 93°F $<$ 95°F
Precipitation: 0% $\leq$ 40%
Maximum wind: 10 $\leq$ 20 mph

Triggered conditions: None
Decision: Do not postpone

Figure 7: **Weather task and reference trajectory.** The reference trajectory is obtained by executing a reference policy in the weather environment. The policy first looks up the Los Angeles weather point, then uses the returned county and forecast identifiers to retrieve the relevant stations and forecast periods.

Weather Task (Infeasible Instance)

***Environment***
A frozen, read-only Los Angeles weather snapshot containing:

- a resolved Los Angeles weather point;
- county-linked weather stations;
- official forecast periods for the location.

The Saturday daytime precipitation probability is unavailable in this snapshot. All other forecast fields, station records, and public tool interfaces remain unchanged.

***Task Description***
You are planning outdoor field work in Los Angeles on Saturday, August 1, 2026. Identify the closest available weather station to the Los Angeles weather point so that the field team has a nearby monitoring reference, and summarize the official daytime forecast for that location. The team must postpone the work if the forecast temperature reaches 95°F, the chance of precipitation exceeds 40%, or winds may exceed 20 mph. Report the selected station and its distance from the weather point, the relevant daytime forecast evidence, whether the work should be postponed, and which conditions—if any—triggered that decision.

***Public Tools***

- resolve_weather_point(location) Resolves a location to its coordinates, county, forecast resource, and time zone.
- list_station_ids_by_county(county_id) Lists all available weather stations in the resolved county.
- get_station(station_id) Returns the name, coordinates, elevation, and region metadata of a station.
- get_forecast_periods(forecast_id) Returns all forecast periods for the resolved weather point.

***Reference Execution***
1. resolve_weather_point("Los Angeles, CA")
→ coordinates: ((34.0522, -118.2437))
→ county: CAC037
→ forecast resource: LOX/155,45
→ time zone: America/Los_Angeles
2. list_station_ids_by_county(CAC037)
→ [000SE, 001SE]
3. get_station(000SE)
→ SCE South Hills Park
→ coordinates: ((34.11814, -117.85520))
4. get_station(001SE)
→ SCE Glen Way
→ coordinates: ((34.12550, -117.73333))
5. get_forecast_periods(LOX/155,45)
→ Saturday daytime
→ temperature: 93°F
→ precipitation probability: unavailable
→ field error: required_field_unavailable
→ scope: current_snapshot
→ recoverable: false
→ wind: 0–10 mph

***Derived Decision***
Distance to 000SE: 36.5 km
Distance to 001SE: 47.7 km

Closest station: 000SE — SCE South Hills Park

Temperature: 93°F $<$ 95°F
Precipitation: Cannot determine whether it exceeds 40%
Maximum wind: 10 $\leq$ 20 mph

Triggered conditions: Temperature and wind do not trigger postponement; the precipitation condition remains unresolved.

Decision: Abstain from a definitive postponement decision.

The selected station and available forecast evidence can still be reported. However, the missing precipitation probability prevents a complete policy-based decision and cannot be recovered through another public tool in this snapshot.

Figure 8: **Weather task and reference trajectory in the infeasible instance.** The task instruction and public tools remain unchanged. The forecast response marks the Saturday daytime precipitation probability as unavailable. The reference trajectory still establishes the closest station and the available forecast evidence, but cannot determine whether the precipitation threshold is exceeded. Resource identifiers and observations are abbreviated for presentation.