BAD GENIUS: COUNTERFACTUAL-GUIDED HARNESS EVOLUTION BEYOND TASK-SPECIFIC SHORTCUTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable agent evaluation is complicated by automatic harness optimization, which repeatedly uses a released benchmark B_{rel} to guide a Proposer that edits prompts, memory, retrieval, tools, and control code around a fixed foundation model. Task holdout is commonly used to guard against harness overfitting. It varies semantic tasks but leaves the benchmark protocol fixed, so a "bad genius" Proposer can produce a cheating harness whose improvement over the initial harness on B_{rel} depends on a benchmark-wide shortcut. We introduce **C**ounterfactual **Ha**rness **S**earch and **E**volution (**CHASE**), which casts harness evolution as constraint generation over valid counterfactual benchmarks. After each Proposer update, a *Challenger* searches for an executable protocol transformation with large gain destruction. A validity firewall checks that task semantics are preserved, while a held-out confirmation set determines whether the counterfactual enters a finite archive. We formalize an ideal shortcut-neutralized benchmark B_0 and establish theoretical guarantees linking finite counterfactual archives to B_0 and characterizing sequential Challenger search. We evaluate CHASE on Syn-Ledger and OfficeQA, where CHASE retains strong released-benchmark gains while substantially reducing gain destruction under valid protocol transformations.

1 INTRODUCTION

Reliable agent evaluation is central to progress in agentic AI, with benchmarks playing a key role in determining which systems appear capable, safe, and ready for deployment. A reported score, however, does not reflect the foundation model alone, but also depends on the harness through which it operates and the benchmark protocol under which it is evaluated. A harness specifies the executable context around the model, including what information is stored and retrieved, how tools and workspace are exposed, and how outputs are handled. In the literature, the evaluated agent consists of the foundation model with its harness. Evaluation is increasingly concerned not only with performance under a fixed harness, but also with the performance attainable after optimizing the harness around a fixed foundation model. Meta-Harness uses a coding agent (the Proposer) to revise the harness based on previous code, evaluation scores, and execution traces (Lee et al., 2026). Subsequent work studies held-out evaluation, optimizer quality, priority ranking, and reliable harness selection (Wang et al., 2026b; Ong et al., 2026; Ursekar et al., 2026; Zhao et al., 2026). Harness-Bench further shows that performance and failure modes vary materially across model–harness configurations under shared tasks (Yao et al., 2026). Together, these results make iterative harness optimization, which we call harness evolution, a substantive evaluation target (Zhu et al., 2026a;b).

Repeated use of benchmark feedback creates a distinct generalization problem. The "Bad genius"[1] Proposer is optimized for benchmark score, not for preserving what that score is intended to measure, and can modify the harness using benchmark feedback and execution traces. Classical shortcut learning concerns predictive rules induced by data or task artifacts (Geirhos et al., 2020; Ribeiro et al., 2020). Harness evolution adds an executable route: the Proposer can write the shortcut into the harness. Existing methods ban task identifiers, filenames, and per-task repair recipes, then evaluate on held-out tasks (Wang et al., 2026b; Zhang et al., 2026). These methods limit a *task-specific shortcut*—for example, detecting one question ID and opening a known document—but not a ***benchmark-wide shortcut***, such as searching one document channel first because the released

[1]The title refers to the film *Bad Genius*, where a gifted student helps others cheat through clever schemes.

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.

protocol places supporting evidence there more often. As illustrated in Figure 1(a), task holdout therefore leaves benchmark-wide shortcuts intact: tasks change, but the protocol correlation does not. We examine this problem in OfficeQA Full, a benchmark in which agents answer questions using U.S. Treasury Bulletin documents (Opsahl-Ong et al., 2026). For example, 58.1% of questions in OfficeQA Full mention numerical scales such as millions. In its 697-document corpus, the next nonblank line after 95.2% of unit statements begins a table (Supplementary C.1). The Proposer may propose a harness that induces the model to inspect the line before a table when looking for units.

This is distinct from overfitting to particular tasks: the shortcut can persist on held-out tasks. The natural next step is therefore to retain held-out tasks while varying the protocol as well. Figure 1(b)–(c) illustrates this comparison. Under the released protocol shown in Figure 1(b), the shortcut "search channel A first" (e.g., inspect the text immediately preceding a table to identify its unit) still places the supporting evidence first. Figure 1(c) then reassigns the same documents across channels, removing this shortcut advantage while keeping the question, answer, and documents fixed. Changing the protocol may also affect the initial harness H_0. We therefore compare the evolved harness's gain over H_0 before and after the change. A large loss of gain reveals benchmark dependence that held-out tasks alone cannot expose. This raises a question: *can we construct a benchmark that neutralizes all benchmark-wide shortcuts while preserving the underlying tasks?*

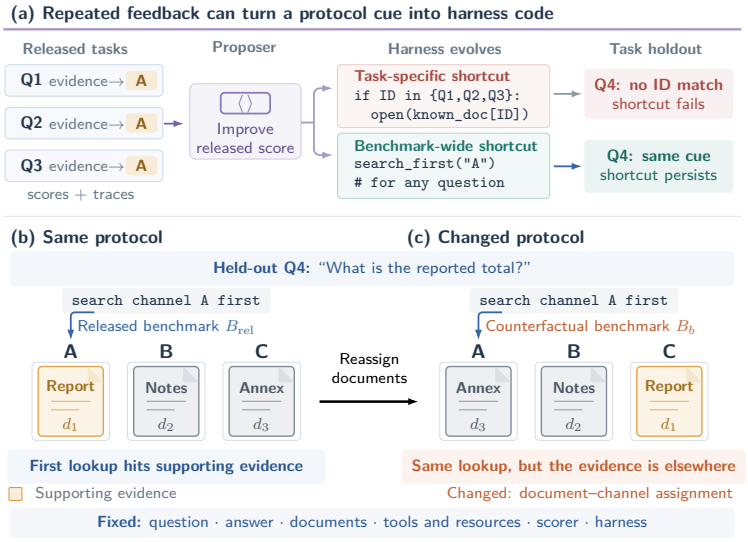

Figure 1: **Motivation: Why task holdout can miss benchmark-wide shortcuts.**

If every shortcut mechanism were known, one could construct an ideal shortcut-neutralized benchmark B_0 and optimize the harness on it. In realistic benchmarks, however, benchmark-wide shortcuts can arise from many parts of the protocol and remain hidden from both manual inspection and automated checks, making them difficult to pre-specify with fixed rules. We therefore introduce a coding agent named ***Challenger*** that searches online, after each harness update, for executable protocol transformations, each producing a counterfactual benchmark designed to expose shortcuts in the updated harness. Whereas prior work introduces auxiliary agents as *debuggers* or *critics* to guide harness repair (Lin et al., 2026; Li et al., 2026), our Challenger attacks shortcuts at their source—the benchmark protocol. Each valid counterfactual confirmed on held-out tasks becomes a constraint on harness evolution, limiting how much future gains can depend on the same protocol correlation.

Our contributions are fourfold. First, we distinguish task-specific shortcuts from benchmark-wide shortcuts and define a gain-destruction estimand that assesses whether observed harness gains remain reliable under valid counterfactual benchmark. Second, we propose **Counterfactual Harness Search and Evolution (CHASE)**, in which a Challenger searches for counterfactual benchmarks. Third, we establish theoretical guarantees that certify how much harness gain survives shortcut neutralization through counterfactual-guided evolution and when Challenger search can stop. Fourth, across the synthetic benchmark Syn-Ledger and OfficeQA, we show that CHASE yields confirmed counterfactuals and better generalization to counterfactual benchmarks than other baselines.

2 Related Work

Harness optimization. Earlier work automates large language model (LLM) system optimization through metric-driven pipeline compilation in *DSPy* (Khattab et al., 2024), prompt search in *OPRO* (Yang et al., 2024), and feedback propagation in *TextGrad* (Yuksekgonul et al., 2025). *Meta-Harness* optimizes prompts, memory, tools, and control code using evaluation feedback (Lee et al., 2026). *HarnessOpt-Bench* evaluates the optimizer under a fixed budget (Ursekar et al., 2026), while *Priority ranking* tests whether optimizers can identify promising harness components (Ong et al., 2026). *Harness-Bench* isolates harness effects across model backends (Yao et al., 2026). *HarnessLens* reduces verification cost by selecting tasks for each candidate and confirming promising edits (Xu et al., 2026). *AutoSaddler* further combines failure-trace diagnosis with validation-based harness updates (Park et al., 2026). These works characterize the optimization object, optimizer quality, and verification efficiency. CHASE addresses a different question: whether feedback from the released benchmark B_{rel} selects a harness whose gain depends on protocol correlations shared across tasks.

Harness generalization. Held-out-task evaluation shows that optimizing and evaluating harnesses on the same tasks can substantially overstate their gains (Wang et al., 2026b; Esakkiraja et al., 2026). *HarnessCompass* uses a fixed generalization gate to allow only task-agnostic modifications, together with component-wise feedback to guide harness updates (Zhang et al., 2026). *HarnessEvolve* uses reference trajectories, quality and performance gates, and held-out validation to filter and select harness updates (Jiang et al., 2026a). *EvoSafeHarness* uses a separate critic to check whether proposed safety rules still work when tool names or attack wording change (Li et al., 2026). *Harness continual learning* evaluates new harness updates for retention of previously acquired behavior (Kang et al., 2026). More broadly, continually updated benchmarks, lifelong test sets, and overfitting alarms reduce repeated reliance on a fixed evaluation set (Prabhu et al., 2024; Ishida et al., 2026). Overall, existing methods mainly vary tasks, apply fixed filters to harness edits, or validate harness candidate updates before adoption. Task variation leaves the benchmark protocol unchanged, while the fixed filters can only cover anticipated shortcut patterns. Pre-adoption validation assesses candidates under selected test conditions. CHASE instead searches for counterfactual benchmarks against the updated harness while preserving task semantics, without relying on a fixed filter over harness edits.

Benchmark validity. Agent scores are properties of a model–harness–environment–protocol configuration rather than the foundation model alone (Zhu et al., 2026a;b; Yao et al., 2026). Recent benchmark audits show that agent scores can be distorted by protocol flaws, search-time contamination, broken tasks, and scoring errors (Shao et al., 2026; Wang et al., 2026c; Dong et al., 2026). Contamination tests and refreshed benchmarks address exposure of benchmark content (Oren et al., 2024; White et al., 2025; Wu et al., 2025). *Auditing Harness Tampering* studies a related problem in self-improving agents, where harness edits can produce apparent performance gains without genuine capability improvement, and develops audits to detect and locate such edits (Wang et al., 2026a). *HackProbe* similarly detects reward hacking during self-evolution and uses the resulting signal for harness reselection (Yang et al., 2026). These works diagnose or reduce specific sources of invalid benchmark gains. CHASE instead tests whether harness gains survive valid protocol transformations, without pre-specifying the shortcut.

Counterfactual-guided optimization. Counterfactual-guided model repair repeatedly finds a counterfactual to the current model and updates the model to remove it (Bauer-Marquart et al., 2022; Boetius et al., 2023). *Model-written evaluations* and *automated red teaming* use one model to generate tests that expose failures of another (Perez et al., 2022; 2023). *Metamorphic testing* evaluates a system under transformations that keep the task semantics unchanged (Hyun et al., 2024; Cho et al., 2025). CHASE combines these ideas for harness evolution: after each Proposer update, the Challenger searches for a valid counterfactual benchmark that destroys the current harness's gain, and a confirmed counterfactual constrains subsequent harness updates.

3 Counterfactual Harness Search and Evolution

3.1 Problem Setup

Fix a foundation model A, and recall from Section 1 that B_{rel} denotes the released benchmark used for harness optimization. To describe B_{rel}, let U denote the underlying task instance and let $Y = \psi(U)$ be the response, where $\psi(\cdot)$ is the target rule. The released protocol V_{rel} contains file

names, directory layout, metadata, tool aliases, demonstration order, feedback format, and other benchmark-specific details, while Q_{rel} specifies how the agent interacts with these components. Thus, B_{rel} combines a distribution P over semantic tasks U and their responses $Y = \psi(U)$, the released protocol $(V_{\text{rel}}, Q_{\text{rel}})$, the available tools and resource budgets, and a scorer. For simplicity, we write $B_{\text{rel}} = (P, V_{\text{rel}}, Q_{\text{rel}}, \psi)$. For any benchmark variant B sharing the semantic-task distribution P and any harness H, let $\tau^{H,B} \sim p_{\mathsf{A}}(\cdot \mid H, B, U)$ denote the resulting execution trajectory and let $r_B(\tau^{H,B}, U) \in [0, 1]$ denote the score assigned by benchmark B.

Suppose harness optimization runs for T rounds. Let H_0 be the initial harness. At each round $t = 1, \ldots, T$, a Proposer—for example, GPT-5.6 Sol configured as a coding agent—observes the current harness, previous scores, and execution traces, edits H_{t-1}, and returns H_t. Let $D_s = \{U_i\}_{i=1}^{n_s}$ denote the search-task set. If $\tau_i^{H_t,B}$ is the rollout of (A, H_t) on U_i, define the empirical score $\widehat{R}_{B,D_s}(\mathsf{A}, H_t)$ and the population score $R_B(\mathsf{A}, H)$:

$$\widehat{R}_{B,D_s}(\mathsf{A}, H_t) := \frac{1}{n_s} \sum_{i=1}^{n_s} r_B(\tau_i^{H_t,B}, U_i), \quad R_B(\mathsf{A}, H) := \mathbb{E}_{U \sim P} \mathbb{E}_{\tau^{H,B} \sim p_{\mathsf{A}}(\cdot \mid H,B,U)}[r_B(\tau^{H,B}, U)].$$

The outer loop therefore selects $H_0, H_1, \ldots, H_T$ using the empirical scores. After breaking ties by a fixed rule, the search-optimal harness among the candidates is:

$$H^\star_{\text{rel}} \in \left\{ H_t : \widehat{R}_{B_{\text{rel}},D_s}(\mathsf{A}, H_t) = \max_{0 \le j \le T} \widehat{R}_{B_{\text{rel}},D_s}(\mathsf{A}, H_j) \right\}. \tag{1}$$

The Proposer's direct objective is to increase $\widehat{R}_{B_{\text{rel}},D_s}(\mathsf{A}, H_t)$ by modifying the harness, rather than to ensure that each task is solved through the intended capability. Because it can inspect prompts, memory, retrieval logic, code, tool calls, and execution traces, the resulting harness may achieve a higher released-benchmark score on D_s by exploiting a benchmark-wide shortcut.

If a shortcut mechanism is known, an ideal transformation may construct a neutralized benchmark $B_0 := \Phi_0(B_{\text{rel}})$ through $\Phi_0(B_{\text{rel}}) = (P, V_0, Q_0, \psi)$, where V_0 and Q_0 are the shortcut-neutralized counterparts of V_{rel} and Q_{rel}, respectively. A valid Φ_0 removes the specified protocol correlation while leaving $U, Y = \psi(U)$, and the semantic-task distribution P unchanged. Write $R_0 := R_{B_0}$ and $R_{\text{rel}} := R_{B_{\text{rel}}}$. Define $G_{\text{rel}}(H; H_0) = R_{\text{rel}}(\mathsf{A}, H) - R_{\text{rel}}(\mathsf{A}, H_0)$ and $G_0(H; H_0) = R_0(\mathsf{A}, H) - R_0(\mathsf{A}, H_0)$. Then, we define $\Delta_{\text{BS}}(H; H_0)$ with:

$$G_{\text{rel}}(H; H_0) = G_0(H; H_0) + \Delta_{\text{BS}}(H; H_0),$$

where $G_{\text{rel}}(H; H_0)$ is the released gain during harness optimization, while $G_0(H; H_0)$ is the gain that survives shortcut neutralization. The remaining $\Delta_{\text{BS}}(H; H_0)$ is the signed gain difference associated with the "bad genius" Proposer's changes in reliance on the benchmark-wide shortcut rather than improved task-solving capability.

Prior work commonly uses task holdout to guard against shortcuts, but it addresses only task-specific shortcuts (Wang et al., 2026b). Let D_h be a test-task set disjoint from D_s, and let $R_{B,D}(\mathsf{A}, H) := \mathbb{E}_{U \sim \widehat{P}_D} \mathbb{E}_{\tau^{H,B} \sim p_{\mathsf{A}}(\cdot \mid H,B,U)}[r_B(\tau, U)]$ denote expected average score on a finite task set D, where P is replaced by the empirical distribution $\widehat{P}_D$. The excess search-set gain is:

$$\Delta_{\text{TS}}(H; H_0) = \{R_{B_{\text{rel}},D_s}(\mathsf{A}, H) - R_{B_{\text{rel}},D_s}(\mathsf{A}, H_0)\} - \{R_{B_{\text{rel}},D_h}(\mathsf{A}, H) - R_{B_{\text{rel}},D_h}(\mathsf{A}, H_0)\}.$$

A task-specific shortcut can make $\Delta_{\text{TS}}(H; H_0) > 0$. A benchmark-wide shortcut can persist in both D_s and D_h because both use the same $(V_{\text{rel}}, Q_{\text{rel}})$. Therefore $\Delta_{\text{TS}}(H; H_0) \approx 0$ does not imply $\Delta_{\text{BS}}(H; H_0) \approx 0$. Our goal is to achieve a substantial positive $G_{\text{rel}}(H; H_0)$ while keeping $\Delta_{\text{BS}}(H; H_0)$ below a small tolerance.

3.2 From Counterfactuals to Harness Evolution

The ideal B_0 above defines the target estimand when all shortcut mechanisms are known. In practice, however, the complete set of shortcut mechanisms is unknown, so B_0 cannot be constructed. CHASE therefore replaces the single B_0 with a family of valid counterfactual benchmarks. Index each protocol transformation by b. Each Φ_b acts on benchmark configurations; its action on the released benchmark preserves the same semantic task and maps $B_b := \Phi_b(B_{\text{rel}}) = (P, V_b, Q_b, \psi)$. Write $\text{Valid}(\Phi_b) = 1$ when the transformation preserves the semantic task, target, evidence, and scoring

semantics while changing only the permitted components of the protocol. The valid family is $\mathcal{B}_{\text{val}} = \{B_b = \Phi_b(B_{\text{rel}}) : \text{Valid}(\Phi_b) = 1\}$, and the identity transformation includes B_{rel} in $\mathcal{B}_{\text{val}}$.

Validity Firewall. The benchmark-specific firewall sets $\text{Valid}(\Phi_b) = 1$ only when all executable checks comparing B_{rel} with $\Phi_b(B_{\text{rel}})$ pass; otherwise, Φ_b is rejected. These checks include preserving the semantic task and ground-truth answer and restricting changes to the permitted components. The complete set of checks is given in Supplementary B.3.

For any candidate H and $B_b \in \mathcal{B}_{\text{val}}$, let $R_b = R_{B_b}$ and define $G_b(H; H_0) = R_b(\mathsf{A}, H) - R_b(\mathsf{A}, H_0)$. Taking $B_b = B_{\text{rel}}$ or $B_b = B_0$ gives $G_{\text{rel}}(H; H_0)$ or $G_0(H; H_0)$, respectively. The released gain destroyed by B_b is $\Delta_b(H; H_0) = G_{\text{rel}}(H; H_0) - G_b(H; H_0)$. When $B_b = B_0$, $\Delta_b(H; H_0)$ reduces to $\Delta_{\text{BS}}(H; H_0)$, so the valid family $\mathcal{B}_{\text{val}}$ directly extends ideal neutralization B_0. Negative Δ_b means that the harness gain increases rather than decreases under the counterfactual.

CHASE then alternates between a Proposer and a Challenger. Since $\mathcal{B}_{\text{val}}$ is an unknown infinite family, before round t, the Proposer has access to a finite archive $\mathcal{A}_{t-1} \subset \mathcal{B}_{\text{val}}$ containing B_{rel} and the counterfactuals confirmed in earlier Challenger rounds. At the population level, the Proposer solves:

$$H_t \in \arg\max_H G_{\text{rel}}(H; H_0) \quad \text{s.t.} \quad \Delta_b(H; H_0) \le \varepsilon, \quad \forall B_b \in \mathcal{A}_{t-1}, \tag{2}$$

where $\varepsilon \ge 0$ controls how much of the released-benchmark gain may disappear under any archived counterfactuals. The Challenger targets the resulting H_t by searching:

$$B_{b_t} \in \arg\max_{B_b \in \mathcal{B}_{\text{val}}} \Delta_b(H_t; H_0), \tag{3}$$

and returns a typed, executable specification of Φ_{b_t}. The resulting counterfactual B_{b_t} is then considered for inclusion in $\mathcal{A}_t$, subject to the validity and confirmation checks below (Section 3.3).

Finally, the selection criterion in equation 1 is inadequate for CHASE, since a high-scoring harness may violate counterfactual constraints discovered in later rounds. After updating the archive, we therefore select the final harness:

$$H^\star \in \arg\max_{H \in \{H_0, H_1, \ldots, H_T\}} G_{\text{rel}}(H; H_0) \quad \text{s.t.} \quad \Delta_b(H; H_0) \le \varepsilon, \quad \forall B_b \in \mathcal{A}_T. \tag{4}$$

3.3 Finite Archive and Confirmation

The objectives in equations 2–4 above are written at the population level; implementation and the analysis below use their empirical counterparts. For any finite task set D, define $\widehat{R}_{B,D}(\mathsf{A}, H) := \frac{1}{|D|}\sum_{U_i \in D} r_B(\tau_i^{H,B}, U_i)$, $\widehat{G}_{b,D}(H; H_0) := \widehat{R}_{B_b,D}(\mathsf{A}, H) - \widehat{R}_{B_b,D}(\mathsf{A}, H_0)$, and $\widehat{\Delta}_{b,D}(H; H_0) := \widehat{G}_{\text{rel},D}(H; H_0) - \widehat{G}_{b,D}(H; H_0)$. When a task is evaluated with multiple rollouts, the rollout scores are first averaged within task. To separate search, confirmation, and final certification, the evaluation tasks are assigned disjoint roles:

$$D_{\text{evo}},\ D_{\text{disc}},\ D_{\text{conf},1}, \ldots, D_{\text{conf,T}},\ D_{\text{cert}} \quad \text{are mutually disjoint.}$$

Here D_{evo} supplies the feedback used by the Proposer to search H_t in equation 2, D_{disc} supplies the feedback used by the Challenger to search for Φ_{b_t} in equation 3, $D_{\text{conf,t}}$ is used to confirm Φ_{b_t} after it has been fixed, and D_{cert} remains sealed until final certification. This separation reserves fresh tasks for confirmation and certification, extending the search–test distinction (D_s, D_h) in Section 3.1.

Since $\mathcal{B}_{\text{val}}$ is unknown and cannot be exhaustively searched, CHASE maintains the sequence of finite archives $\{\mathcal{A}_t\}_{t=0}^T$. Initialize $\mathcal{A}_0 = \{B_{\text{rel}}\}$. Once the Challenger proposes Φ_{b_t}, its executable code is fixed and $\Delta_{b_t}(H_t; H_0)$ is re-estimated on $D_{\text{conf,t}}$. For a confirmation threshold $\eta_{\text{conf,t}} > 0$, let $\mathsf{Conf}_t := \mathbf{1}\left\{\widehat{\Delta}_{b_t, D_{\text{conf,t}}}(H_t; H_0) \ge \eta_{\text{conf,t}}\right\}$. The archive is then updated by:

$$\mathcal{A}_t = \begin{cases} \mathcal{A}_{t-1} \cup \{B_{b_t}\}, & \text{Valid}(\Phi_{b_t}) = 1 \text{ and } \mathsf{Conf}_t = 1, \\ \mathcal{A}_{t-1}, & \text{otherwise.} \end{cases} \tag{5}$$

In practice, each archived B_{b_t} is stored together with the executable code for Φ_{b_t}. Thus a Challenger proposal becomes a constraint on subsequent harness evolution only when it is valid and its effect is confirmed on $D_{\text{conf,t}}$.

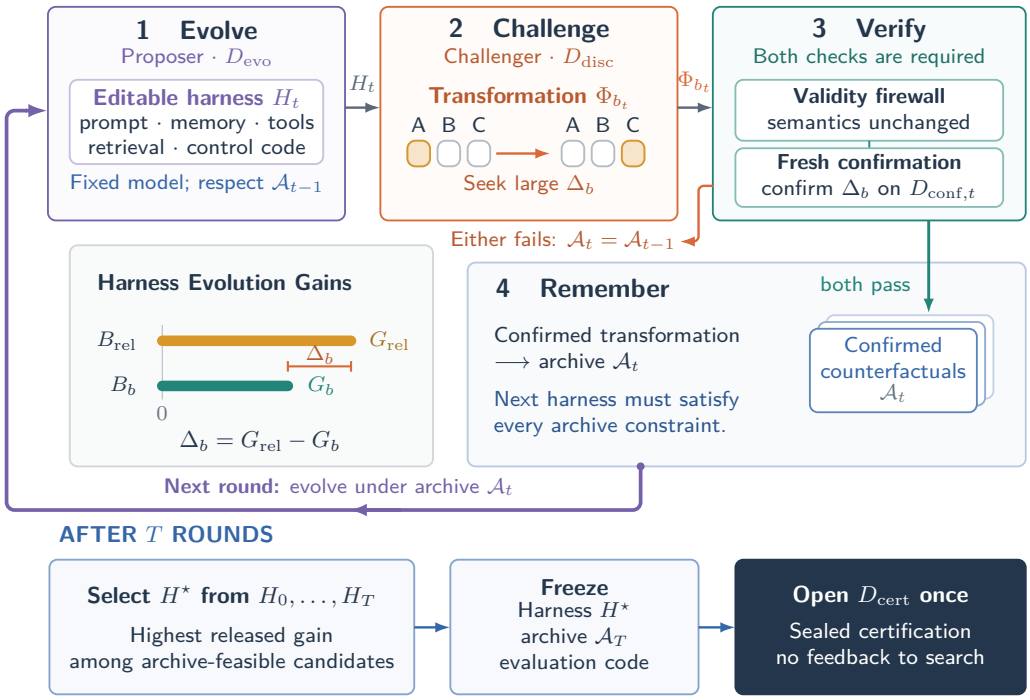

Figure 2: An overview of **CHASE**.

To summarize performance over any finite counterfactual archive $\mathcal{A} \subset \mathcal{B}_{\text{val}}$, define:

$$\Gamma_{\mathcal{A}}(H; H_0) := \max_{B_b \in \mathcal{A}} \Delta_b(H; H_0), \qquad G_{\min,\mathcal{A}}(H; H_0) := \min_{B_b \in \mathcal{A}} G_b(H; H_0),$$

with empirical counterparts on a task set D,

$$\widehat{\Gamma}_{\mathcal{A},D}(H; H_0) := \max_{B_b \in \mathcal{A}} \widehat{\Delta}_{b,D}(H; H_0), \qquad \widehat{G}_{\min,\mathcal{A},D}(H; H_0) := \min_{B_b \in \mathcal{A}} \widehat{G}_{b,D}(H; H_0).$$

Here, $\Gamma_{\mathcal{A}}(H; H_0)$ is the largest gain destruction, $G_{\min,\mathcal{A}}(H; H_0)$ is the smallest surviving gain over $\mathcal{A}$, and we have $G_{\min,\mathcal{A}}(H; H_0) + \Gamma_{\mathcal{A}}(H; H_0) = G_{\text{rel}}(H; H_0)$. Figure 2 summarizes the complete CHASE workflow.

4 Theoretical Guarantees

Theoretical analyses of harness evolution remain limited. Recent work studies how to determine from finite evaluation data whether a harness update improves performance without degrading prior behavior under a fixed task distribution (Cai et al., 2026). It does not model protocol correlation or counterfactual discovery. We therefore ask what a finite counterfactual archive can certify about B_0 and, when it does not yet recover B_0, what can be concluded from sequential Challenger search.

For valid $\Phi_b, \Phi_{b'}$, define $B_{b' \circ b} := \Phi_{b'}(\Phi_b(B_{\text{rel}}))$ by applying Φ_b first and $\Phi_{b'}$ second. This composition maps $(P, V_{\text{rel}}, Q_{\text{rel}}, \psi)$ to $(P, V_{b' \circ b}, Q_{b' \circ b}, \psi)$, and we assume $\mathcal{B}_{\text{val}}$ is closed under it. For nonempty finite $\mathcal{A} \subset \mathcal{B}_{\text{val}}$, define:

$$K(\mathcal{A}) := \min\left\{k \in \mathbb{N}^+ : \exists\, B_{b_1}, \ldots, B_{b_k} \in \mathcal{A}, B_{b_k \circ \cdots \circ b_1} = B_0\right\}.$$

Thus $K(\mathcal{A}) = \infty$ if no composition of transformations in $\mathcal{A}$ produces B_0. Also define:

$$\rho := \sup_{H} \sup_{B_b, B_{b'} \in \mathcal{B}_{\text{val}}} \left[\Delta_{b' \circ b}(H; H_0) - \Delta_b(H; H_0) - \Delta_{b'}(H; H_0)\right]_+.$$

Thus $K(\mathcal{A})$ is the smallest number of archived transformations whose composition produces B_0, while ρ measures the worst-case excess gain destruction under composition. Equivalently, $\Delta_{b' \circ b}(H; H_0) \le \Delta_b(H; H_0) + \Delta_{b'}(H; H_0) + \rho$. Fix $\alpha \in (0, 1)$. We obtain the following results.

Theorem 1 *Let $\mathcal{A} \subset \mathcal{B}_{\mathrm{val}}$ be nonempty and finite. If $\mathcal{A}$ and $H^\star$ are fixed before D_{cert} is opened and $K(\mathcal{A}) < \infty$, then, under the conditions in Supplementary A, with probability at least $1 - \alpha/2$:*

$$\Delta_{\mathrm{BS}}(H^\star; H_0) \le K(\mathcal{A}) \left\{ \widehat{\Gamma}_{\mathcal{A}, D_{\mathrm{cert}}}(H^\star; H_0) + \sqrt{8 \log(4|\mathcal{A}|/\alpha)/|D_{\mathrm{cert}}|} \right\} + (K(\mathcal{A}) - 1)\rho,$$

$$G_0(H^\star; H_0) \ge \widehat{G}_{\min, \mathcal{A}, D_{\mathrm{cert}}}(H^\star; H_0) - \sqrt{2 \log(4|\mathcal{A}|/\alpha)/|D_{\mathrm{cert}}|} - (K(\mathcal{A}) - 1) \left\{ \widehat{\Gamma}_{\mathcal{A}, D_{\mathrm{cert}}}(H^\star; H_0) + \sqrt{8 \log(4|\mathcal{A}|/\alpha)/|D_{\mathrm{cert}}|} + \rho \right\}.$$

Theorem 1 converts a finite archive into guarantees for B_0. The bounds tighten as $K(\mathcal{A})$ and ρ decrease. If $B_0 \in \mathcal{A}$, then $K(\mathcal{A}) = 1$ and every term containing ρ disappears.

For each search set $D \in \{D_{\mathrm{evo}}, D_{\mathrm{disc}}\}$, let $\mathfrak{R}_D$ denote the Rademacher complexity term defined in Supplementary A.1, which typically decreases as $|D|$ grows. It controls uniform estimation of $R_{\mathrm{rel}}(\mathsf{A}, H)$ over harnesses H and of $\Delta_b(H; H_0)$ over harness–counterfactual pairs (H, B_b). We use $r = \max_{D \in \{D_{\mathrm{evo}}, D_{\mathrm{disc}}\}} \left\{ 2\mathfrak{R}_D + \sqrt{8 \log(8/\alpha)/|D|} \right\}$ which bounds the empirical-to-population deviation. For confirmation, set $\eta_{\mathrm{conf},t} = \varepsilon + \gamma + \sqrt{8 \log(8T/\alpha)/|D_{\mathrm{conf},t}|}, 0 < \gamma \le 1$. Supplementary A discusses $\mathfrak{R}_D$ and γ.

Theorem 2 *Suppose $\mathrm{Valid}(\Phi_{b_t}) = 1$ and $\widehat{\Delta}_{b, D_{\mathrm{evo}}}(H_t; H_0) \le \varepsilon$ for every $B_b \in \mathcal{A}_{t-1}$ at each round. Assume that, for every $0 < q \le 1$, any m harnesses with $\sup_{B_b \in \mathcal{B}_{\mathrm{val}}} |\Delta_b(H^{(j)}; H_0) - \Delta_b(H^{(k)}; H_0)| \ge q$, for all $j \ne k$, obey $m \le (C/q)^d$, for constants $C \ge 1$ and $d > 0$. If $r < \gamma$, then, with probability at least $1 - \alpha/2$,*

$$\#\{t \le T : \mathsf{Conf}_t = 1\} \le \left(\frac{C}{\gamma - r} \right)^d.$$

Consequently, if $T > \left(\frac{C}{\gamma - r} \right)^d$, then $\mathsf{Conf}_t = 0$ for at least one $t \le T$.

Theorem 2 shows that, under the packing condition, only finitely many counterfactuals can be confirmed. In particular, for a sufficiently large T, the process must reach a round with $\mathsf{Conf}_t = 0$. We next characterize what can be concluded at such a round.

Theorem 3 *Suppose H_t and B_{b_t} solve the empirical counterparts of equation 2 and equation 3. On the common evolution event defined in Supplementary Lemma 1, which has probability at least $1 - \alpha/2$, every round t for which $\mathrm{Valid}(\Phi_{b_t}) = 1$ and $\mathsf{Conf}_t = 0$ satisfies:*

$$\sup_{B_b \in \mathcal{B}_{\mathrm{val}}} \Delta_b(H_t; H_0) < \varepsilon + \gamma + 2r + 2\sqrt{8 \log(8T/\alpha)/|D_{\mathrm{conf},t}|},$$

$$G_{\mathrm{rel}}(H_t; H_0) \ge \sup_{\widetilde{H}:\ \Delta_b(\widetilde{H}; H_0) \le \varepsilon - r, \forall B_b \in \mathcal{B}_{\mathrm{val}}} G_{\mathrm{rel}}(\widetilde{H}; H_0) - 2r.$$

Under these conditions, Theorem 3 gives the complementary conclusion: a non-confirmed round certifies small gain destruction over the valid family $\mathcal{B}_{\mathrm{val}}$, up to estimation error, while retaining near-optimal released gain among harnesses satisfying the constraints. Thus, confirmed rounds expand the archive, whereas a non-confirmed round suggests stopping with the current harness.

5 Experiments

5.1 Benchmarks and experimental setup

OfficeQA is a benchmark suite for question answering over U.S. government financial documents (Opsahl-Ong et al., 2026). Its questions require document discovery, text and table retrieval, numerical reasoning, and exact answer extraction. We use two OfficeQA releases: Full, which contains 246 questions, and Pro V2, which contains 90 questions over a separate receipts-and-expenditures corpus (Databricks, 2026). We use Full for harness optimization and Pro V2 for cross-corpus evaluation. OfficeQA is sensitive to harness design because search, evidence handling, tool use, computation, and answer formatting are harness-controlled. At a fixed model, *EnvHarness*

improved OfficeQA exact-match accuracy from 54.40% to 56.20% over its original-environment baseline (Huang et al., 2026). We retain all released questions and answers and let the foundation model A search the full 697-document transformed-text corpus for OfficeQA Full. Supplementary C gives release, corpus, and scoring details.

Taking $T = 3$, we divide the 246 questions into D_{evo} (49 questions), D_{disc} (49), D_{cert} (76), and three round-specific confirmation task sets $D_{\text{conf},t}$, each containing 24 questions. Pro V2 is reported separately as a cross-corpus analysis. Our use of task splits follows prior OfficeQA evaluations (Alzubi et al., 2026; Ursekar et al., 2026). We additionally report Pro V2 results, which were not reported in those studies.

To isolate benchmark-wide shortcuts in this type of multi-document numerical reasoning, we also construct Syn-Ledger with 320 synthetic ledger tasks. Controlled shortcuts allow us to construct B_0 and directly measure how much harness gain survives their neutralization (Supplementary D).

5.2 Methods and Evaluation

On OfficeQA, we compare five methods. **RawHarness** keeps the initial harness H_0 unchanged and provides the reference for all gain and gain-destruction metrics. **Meta-Harness** optimizes H_0 for released-benchmark performance (Lee et al., 2026). **HarnessCompass** optimizes H_0 for released-benchmark performance under the fixed generalization gate of HarnessCompass (Zhang et al., 2026). **HarnessEvolve** uses reference trajectories and quality, performance, and held-out validation gates to select harness updates (Jiang et al., 2026a). **CHASE** uses HarnessCompass's Proposer backbone without the fixed generalization gate. The Challenger supplies confirmed counterfactuals that enter $\mathcal{A}_t$ and constrain subsequent Proposer rounds and final harness selection. Syn-Ledger includes these five methods and adds B_0**-Access** with access to B_0. It uses HarnessCompass's three-round schedule and fixed generalization gate, but selects candidates by their B_0 performance. Within each benchmark, all methods share A, H_0. All optimized methods except HarnessEvolve (Supplementary B.6) use three Proposer rounds. Section 4 provides guidance on threshold calibration when budgets are sufficiently large. Under our limited budget, we fix $\eta_{\text{conf},t}$ and ε (Supplementary B).

On OfficeQA, the primary evaluation applies each method's final harness H to the 76 certification questions under every benchmark in the final CHASE archive $\mathcal{A}_3$. We report the released-benchmark score $\widehat{R}_{B_{\text{rel}},D_{\text{cert}}}(\mathsf{A}, H)$, the average and worst-case scores over $\mathcal{A}_3$. For a finite archive $\mathcal{A}$, define $\widehat{R}_{\text{avg},\mathcal{A},D_{\text{cert}}}(\mathsf{A}, H) := |\mathcal{A}|^{-1} \sum_{B_b \in \mathcal{A}} \widehat{R}_{B_b,D_{\text{cert}}}(\mathsf{A}, H)$ and $\widehat{R}_{\min,\mathcal{A},D_{\text{cert}}}(\mathsf{A}, H) := \min_{B_b \in \mathcal{A}} \widehat{R}_{B_b,D_{\text{cert}}}(\mathsf{A}, H)$. For brevity, we write the three certification metrics as $\widehat{R}_{\text{rel}}$, $\widehat{R}_{\text{avg},\mathcal{A}_3}$ and $\widehat{R}_{\min,\mathcal{A}_3}$. Separately, $\widehat{R}_{\text{ProV2}}$ denotes the released-benchmark score on the 90-question Pro V2 release. Details are given in Supplementary C. We also compare token use for both pre-certification optimization and final certification, with further details in Supplementary C.4.

On Syn-Ledger, we evaluate all six methods on the 208 certification tasks under B_{rel} and B_0. We report $\widehat{R}_{\text{rel}}$, $\widehat{R}_0$, $\widehat{G}_{\text{rel}}$, $\widehat{G}_0$, and $\widehat{\Delta}_{\text{BS}}$, with the certification-set and harness arguments suppressed. Supplementary D describes the task construction, shortcut generation, scoring, and task allocation.

5.3 Results

Table 1 reports OfficeQA results on Pro V2 and Full certification results. On D_{cert}, CHASE achieves a competitive released-benchmark score and the highest average and worst-case scores over the final archive $\mathcal{A}_3$. In contrast, HarnessCompass performs below the initial harness, suggesting that its fixed generalization gate does not ensure improved held-out performance on OfficeQA. OfficeQA is a new benchmark for HarnessCompass, which focuses on SWE-bench Verified (Chowdhury et al., 2024); our implementation of its generalization gate is detailed in Supplementary B.5. Notably, the advantage of CHASE carries over to the Pro V2 corpus, where it scores 30.37%.

The first-round Challenger proposes collecting a table's associated context before the table and supplies an executable transformation specification. It hypothesizes that the evolved harness H_1, instructed to "keep an explicit unit for every operand," may rely on customary locations of units and notes. Figure 3 illustrates a schematic visualization of the proposal; the host executes the generated specification by re-encoding retrieved text as table context in the agent-visible JSON search results (Supplementary C.3). On D_{evo}, H_1's gain over H_0 decreases from 8.16% under B_{rel} to -5.10%

Table 1: OfficeQA results. All evaluations use three rollouts per question.

Method	OfficeQA Pro V2	Certification of OfficeQA Full		
	$\widehat{R}_{\mathrm{ProV2}}(\uparrow)$	$\widehat{R}_{\mathrm{rel}}(\uparrow)$	$\widehat{R}_{\mathrm{avg},\mathcal{A}_3}(\uparrow)$	$\widehat{R}_{\mathrm{min},\mathcal{A}_3}(\uparrow)$
RawHarness	27.04%	67.98%	66.23%	64.47%
Meta-Harness	29.26%	63.60%	64.04%	63.60%
HarnessCompass	26.30%	64.04%	63.16%	62.28%
HarnessEvolve	24.07%	**69.30%**	68.20%	67.11%
CHASE	**30.37%**	68.86%	**68.42%**	**67.98%**

under B_{b_1}. This reversal shows that H_1's released-benchmark gain depends on how retrieved evidence is represented. Once B_{b_1} enters the archive, H_1 and all second-round Proposer candidates violate the $\varepsilon = 0.05$ constraint, so CHASE sets $H_2 = H_0$. In the third round, one candidate recovers a 4.08% released gain while satisfying the constraint and is selected as H_3.

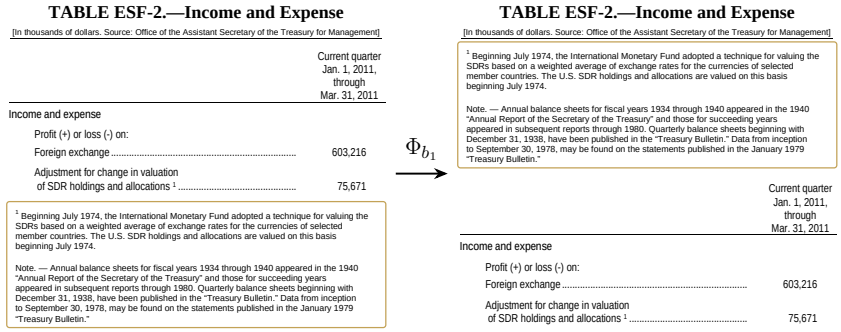

Figure 3: Illustration of the Challenger's proposal using a real OfficeQA excerpt.

We next turn to Syn-Ledger. Table 2 shows that, although CHASE trails HarnessEvolve on B_{rel}, it achieves the highest score and gain under B_0. Its gain is larger under B_0 than under the released benchmark, yielding a negative $\widehat{\Delta}_{\mathrm{BS}}$. This does not indicate that B_0 is easier: scores remain substantially lower under B_0, while the larger gain means that the selected harness of CHASE improves more over H_0 under B_0 than under the released benchmark.

Table 2: Syn-Ledger results. All final evaluations use three rollouts per task.

Method	$\widehat{R}_{\mathrm{rel}}$ (↑)	$\widehat{R}_0$ (↑)	$\widehat{G}_{\mathrm{rel}}$ (↑)	$\widehat{G}_0$ (↑)	$\widehat{\Delta}_{\mathrm{BS}}$ (↓)
RawHarness	81.89%	15.22%	–	–	–
Meta-Harness	82.05%	19.87%	+0.16%	+4.65%	-4.49%
HarnessCompass	80.45%	14.74%	-1.44%	-0.48%	-0.96%
HarnessEvolve	**93.59%**	18.27%	**+11.70%**	+3.04%	+8.65%
B_0-Access	83.01%	32.69%	+1.12%	+17.47%	-16.35%
CHASE	87.18%	**38.30%**	+5.29%	**+23.08%**	**-17.79%**

6 Discussion

Harness optimization changes what must generalize. Unlike parameter optimization, where shortcuts typically arise from data or task artifacts (Geirhos et al., 2020; Jiang et al., 2026b), harness evolution searches over executable programs. As a result, harness optimization can introduce benchmark-wide shortcuts. CHASE makes this distinction explicit: task generalization asks whether a final harness works across tasks, whereas CHASE asks whether the harness-evolution gain persists under valid $B_b \in \mathcal{B}_{\mathrm{val}}$. Our benchmarks use ground-truth-based scoring, while open-ended benchmarks often rely on imperfect LLM judges (Feng et al., 2026; Lai et al., 2026). CHASE could also be extended to LLM-as-a-Judge settings, offering a practical direction for testing judge validity.

AI USE STATEMENT

Generative AI tools assisted with language polishing, code scaffolding, and related-work discovery; the authors independently verified all definitions, citations, transformations, analyses, and claims.

REPRODUCIBILITY STATEMENT

Supplementary Sections A–D provide complete proofs, implementation details, OfficeQA experimental procedures, and the construction of Syn-Ledger. The anonymous supplementary code contains the executable protocol transformations, validity checks, archive records, and scripts for reproducing the reported results from task-level outputs.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Fabian Bauer-Marquart, David Boetius, Stefan Leue, and Christian Schilling. SpecRepair: Counter-example guided safety repair of deep neural networks. In *Model Checking Software*, volume 13255 of *Lecture Notes in Computer Science*, pp. 79–96. Springer, 2022. doi: 10.1007/978-3-031-15077-7_5.

David Boetius, Stefan Leue, and Tobias Sutter. A robust optimisation perspective on counterexample-guided repair of neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2712–2737. PMLR, 2023.

Qianshu Cai, Yonggang Zhang, Jun Nie, Maohao Ran, Huajiang Zheng, Jun Song, Xinmei Tian, Yike Guo, and Wei Xue. Safe harness self-evolution: A theoretical analysis of feasibility and limits. *arXiv preprint arXiv:2609.08175*, 2026.

Steven Cho, Stefano Ruberto, and Valerio Terragni. Metamorphic testing of large language models for natural language processing. In *Proceedings of the 41st IEEE International Conference on Software Maintenance and Evolution*, pp. 174–186. IEEE, 2025. doi: 10.1109/ICSME64153.2025.00025.

Neil Chowdhury, James Aung, Jun Shern Chan, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified. OpenAI, August 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Databricks. OfficeQA: A grounded reasoning benchmark suite. https://github.com/databricks/officeqa, 2026. Accessed 2026-09-01.

Zihan Dong, Zhiyuan Ma, Zekun Wang, Yunqing Li, Zirou Liu, Ruixuan Deng, Qishi Zhan, and Rui Qian. How benchmarks mis-score computer-use agents. *arXiv preprint arXiv:2607.28367*, 2026.

Esakkivel Esakkiraja, Denis Akhiyarov, Vikas Yadav, Sai Rajeswar, Patrice Bechard, Sridhar Nemala, and Sagar Davasam. StarHarness: Evolving harnesses with stratified search for enterprise environments. *arXiv preprint arXiv:2608.24804*, 2026.

Chen Feng, Minghe Shen, Ananth Balashankar, Carsten Gerner-Beuerle, and Miguel R. D. Rodrigues. Noisy but valid: Robust statistical evaluation of LLMs with imperfect judges. In *International Conference on Learning Representations*, 2026.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, Yifei Ming, Jiaqi Pan, Bhavana Dalvi Mishra, Jiaxin Huang, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. EnvHarness: Awakening static worlds for agent learning. *arXiv preprint arXiv:2608.19880*, 2026.

Sangwon Hyun, Mingyu Guo, and M. Ali Babar. METAL: Metamorphic testing framework for analyzing large-language model qualities. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pp. 117–128. IEEE, 2024. doi: 10.1109/ICST60714.2024.00019.

Takashi Ishida, Thanawat Lodkaew, and Ikko Yamane. CapBencher: Give your LLM benchmark a built-in alarm for test-set overfitting. In *International Conference on Machine Learning*, 2026.

Wen Jiang, Mingmin Chu, Yimeng Tian, Qianxin Zhang, Haofei Yang, Rui Yang, Yang Liu, Tao Lv, and Fangming Li. HarnessEvolve: Learning from reference trajectories for reliable agent self-evolution. *arXiv preprint arXiv:2609.00829*, 2026a.

Yuhang Jiang, Fengchuan Zhang, Sanguo Zhang, and Guojun Zhu. Chaos is a LADDER: Domain generalization beyond invariance via reweighting. *arXiv preprint arXiv:2607.26458*, 2026b.

Borui Kang, Jinrui Gu, Junhan Lv, Wenbin Li, Lei Wang, and Yang Gao. Harness continual learning: Continual adaptation beyond model parameters. *arXiv preprint arXiv:2608.19013*, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Peng Lai, Zhihao Ou, Yong Wang, Longyue Wang, Jian Yang, Yun Chen, and Guanhua Chen. BiasScope: Towards automated detection of bias in LLM-as-a-judge evaluation. In *International Conference on Learning Representations*, 2026.

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.

Nanxi Li, Yingzi Ma, Yulong Cao, Edward Suh, Bo Li, Dawn Song, and Chaowei Xiao. EvoSafe-Harness: Evolving model- and domain-specific harnesses for securing agents. *arXiv preprint arXiv:2609.05903*, 2026.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic Harness Engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

Kai Tzu-iunn Ong, Minseok Kang, Dongwook Choi, Junhee Cho, Seungju Kim, Seungwon Lim, Geunha Jang, Minwoo Oh, Bogyung Jeong, Sunghwan Kim, Taeyoon Kwon, and Jinyoung Yeo. Towards direct evaluation of harness optimizers via priority ranking. *arXiv preprint arXiv:2605.22505*, 2026.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning. *arXiv preprint arXiv:2603.08655*, 2026.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B. Hashimoto. Proving test set contamination in black-box language models. In *International Conference on Learning Representations*, 2024.

Sungho Park, Wonjoong Kim, Rongyuan Tan, Jue Zhang, Wook-Shin Han, Pengfei Gao, Chanyoung Park, Yongqiang Yao, Rao Fu, Elsie Nallipogu, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. AutoSaddler: Automatic harness optimization with durable updates from agent execution traces. *arXiv preprint arXiv:2608.23041*, 2026.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.225.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13387–13434. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.847.

Ameya Prabhu, Vishaal Udandarao, Philip H. S. Torr, Matthias Bethge, Adel Bibi, and Samuel Albanie. Efficient lifelong model evaluation in an era of rapid progress. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.442.

Jiaqi Shao, Hanck Chen, Wei Zhang, Maxm Pan, and Bing Luo. Do agent benchmarks measure capability? protocol validity in the age of agentic AI. *arXiv preprint arXiv:2607.22368*, 2026.

Varun Ursekar, Apaar Shanker, Yash Maurya, Shehab Yasser, Vijay S. Kalmath, Veronica Chatrath, and Yuan Xue. HarnessOpt-Bench: Evaluating LLMs at harness optimization. *arXiv preprint arXiv:2608.06301*, 2026.

Xing Wang, Xiaoyi Zhang, and Jie Shao. Auditing harness tampering in self-improving agents. *arXiv preprint arXiv:2609.00069*, 2026a.

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. *arXiv preprint arXiv:2607.12227*, 2026b.

Yongjie Wang, Xinyue Zhang, Kunhong Yao, Zhiwei Zeng, Kaisong Song, Jun Lin, and Zhiqi Shen. Search-time contamination in deep research agents: Measuring performance inflation in public benchmark evaluation. *arXiv preprint arXiv:2606.05241*, 2026c.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A challenging, contamination-limited LLM benchmark. In *International Conference on Learning Representations*, 2025.

Xiaobao Wu, Liangming Pan, Yuxi Xie, Ruiwen Zhou, Shuai Zhao, Yubo Ma, Mingzhe Du, Rui Mao, Anh Tuan Luu, and William Yang Wang. AntiLeakBench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18403–18419. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.901.

Jinghan Xu, Yikai Zhang, Aili Chen, Weiyuan Li, Jiaqing Liang, and Deqing Yang. Verify smarter, evolve further: Efficient harness evolution through behavior-aware verification. *arXiv preprint arXiv:2608.27311*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Rongxin Yang, Yang Liu, Shang Luo, Haoxuan Jia, Chongyang Zhang, Hao Zheng, Yingguang Yang, Yulin Huang, Jianshen Zhang, Yongzhi Qi, Kefu Xu, Congjing Ran, and Bin Chong. Harness-agnostic detection and immunization of reward hacking in self-evolving language models. *arXiv preprint arXiv:2609.04665*, 2026.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-Bench: Measuring harness effects across models in realistic agent workflows. *arXiv preprint arXiv:2605.27922*, 2026.

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. doi: 10.1038/s41586-025-08661-4.

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

Cen Mia Zhao, Haibo Ruan, Wenjie Chen, Pei-fen Tu, Usman Abbasi, and Joel Hesch. Beyond prompts: Measuring and optimizing LLM tool-agent harnesses. *arXiv preprint arXiv:2609.05736*, 2026.

Pengyu Zhu, Lijun Li, Yaxing Lyu, Qianxin Luo, Jingyi Yang, Yi Liu, Tingfeng Hui, Xinyu Yuan, Li Sun, Sen Su, and Jing Shao. UniACE: A unified framework for evaluating LLM agentic capabilities. *arXiv preprint arXiv:2605.27898*, 2026a.

Pengyu Zhu, Li Sun, Philip S. Yu, and Sen Su. "LLM Agent Performance" is not a single evaluation target. *arXiv preprint arXiv:2602.03238*, 2026b.

SUPPLEMENTARY MATERIAL

Section A contains the proofs, and Section B describes the method implementations. Sections C and D give the OfficeQA experimental details and the Syn-Ledger benchmark construction and evaluation, respectively.

A THEORY AND PROOFS

Section A.1 establishes the common evolution event. Sections A.2–A.4 prove Theorems 1–3, respectively, covering finite-archive certification, the number of confirmed rounds, and guarantees for a round with $\mathsf{Conf}_t = 0$.

Condition S1 *All task sets have positive sizes fixed before they are opened. For each $D \in \{D_{\mathrm{evo}}, D_{\mathrm{disc}}, D_{\mathrm{conf},1}, \dots, D_{\mathrm{conf},T}, D_{\mathrm{cert}}\}$, let $\mathcal{F}_D^-$ denote the information available before D is opened. The tasks $U_i \in D$ satisfy $U_i \overset{\mathrm{iid}}{\sim} P$. Conditional on $\mathcal{F}_D^-$, their task-level evaluation vectors are independent, with each vector containing all rollout scores for one task. For every harness–benchmark pair (H, B) evaluated on D,*

$$0 \le r_B(\tau_i^{H,B}, U_i) \le 1, \qquad \mathbb{E}\Big[r_B(\tau_i^{H,B}, U_i) \mid \mathcal{F}_D^-, U_i\Big] = \mathbb{E}_{\tau \sim p_{\mathsf{A}}(\cdot \mid H,B,U_i)}[r_B(\tau, U_i)].$$

For D_{evo} and D_{disc}, the classes of harnesses and harness–counterfactual pairs over which the displayed suprema are taken are fixed before the corresponding task set is observed. Moreover, (H_t, b_t) is $\mathcal{F}^-_{D_{\mathrm{conf},t}}$-measurable, and $(H^\star, \mathcal{A})$ is $\mathcal{F}^-_{D_{\mathrm{cert}}}$-measurable. Dependence among evaluations of the same task is unrestricted.

Condition S1 is only a technical condition for the concentration analysis; it allows within-task dependence and evolution across rounds.

A.1 AUXILIARY LEMMA AND ITS PROOF

For $U_i \in D$ and $B_b \in \mathcal{B}_{\mathrm{val}}$, write the task-level gain destruction as:

$$\delta_{b,i}(H) := r_{B_{\mathrm{rel}}}(\tau_i^{H,B_{\mathrm{rel}}}, U_i) - r_{B_{\mathrm{rel}}}(\tau_i^{H_0,B_{\mathrm{rel}}}, U_i) - r_{B_b}(\tau_i^{H,B_b}, U_i) + r_{B_b}(\tau_i^{H_0,B_b}, U_i) \in [-2, 2].$$

For $D \in \{D_{\mathrm{evo}}, D_{\mathrm{disc}}\}$, let σ_i be independent Rademacher signs and define:

$$\mathfrak{R}_D := \mathbb{E} \max\left\{ \sup_H \left| \frac{1}{|D|} \sum_{U_i \in D} \sigma_i r_{B_{\mathrm{rel}}}(\tau_i^{H,B_{\mathrm{rel}}}, U_i) \right|, \sup_H \sup_{B_b \in \mathcal{B}_{\mathrm{val}}} \left| \frac{1}{|D|} \sum_{U_i \in D} \sigma_i \delta_{b,i}(H) \right| \right\},$$

where the expectation is over the task-level evaluations and the Rademacher signs. Here, $\mathfrak{R}_D$ is the expected maximum of two worst-case Rademacher averages: one over released-benchmark scores as H varies and the other over gain destruction as both H and B_b vary. It measures the complexity cost of searching over harnesses and counterfactuals using D. For fixed-complexity bounded classes, $\mathfrak{R}_D$ is typically of order $|D|^{-1/2}$. Hence, sufficiently large search sets allow a positive margin $\gamma > r$, as required below.

Lemma 1 *Under Condition S1, define:*

$$r := \max_{D \in \{D_{\mathrm{evo}}, D_{\mathrm{disc}}\}} \left\{ 2\mathfrak{R}_D + \sqrt{\frac{8\log(8/\alpha)}{|D|}} \right\}, \qquad x_t := \sqrt{\frac{8\log(8T/\alpha)}{|D_{\mathrm{conf},t}|}}. \tag{A.1}$$

Then, with probability at least $1 - \alpha/2$, simultaneously,

$$\begin{aligned} \sup_H \left| \widehat{R}_{B_{\mathrm{rel}}, D_{\mathrm{evo}}}(\mathsf{A}, H) - R_{\mathrm{rel}}(\mathsf{A}, H) \right| &\le r, \\ \sup_H \sup_{\substack{B_b \in \mathcal{B}_{\mathrm{val}} \\ D \in \{D_{\mathrm{evo}}, D_{\mathrm{disc}}\}}} \left| \widehat{\Delta}_{b,D}(H; H_0) - \Delta_b(H; H_0) \right| &\le r, \\ \left| \widehat{\Delta}_{b_t, D_{\mathrm{conf},t}}(H_t; H_0) - \Delta_{b_t}(H_t; H_0) \right| &\le x_t \end{aligned} \tag{A.2}$$

for every round t in which a proposal passes the validity firewall and is evaluated on $D_{\mathrm{conf},t}$.

Proof. For either search set D with $n = |D|$, standard symmetrization (Zhang, 2023) and the bounded-differences inequality give:

$$\Pr\Big\{\max\Big[\sup_H \Big|\widehat{R}_{B_{\mathrm{rel}},D}(\mathsf{A},H) - R_{\mathrm{rel}}(\mathsf{A},H)\Big|, \sup_H \sup_{B_b\in\mathcal{B}_{\mathrm{val}}} \Big|\widehat{\Delta}_{b,D}(H;H_0) - \Delta_b(H;H_0)\Big|\Big] > 2\mathfrak{R}_D + z\Big\} \le e^{-nz^2/8},$$

since all task-level quantities above lie in $[-2,2]$. Taking $z = \sqrt{8\log(8/\alpha)/n}$ and a union bound over D_{evo} and D_{disc} gives total failure probability at most $\alpha/4$.

For each round, conditional on $\mathcal{F}^-_{D_{\mathrm{conf},t}}$, $\widehat{\Delta}_{b_t,D_{\mathrm{conf},t}}(H_t;H_0)$ is an average of independent $[-2,2]$ variables with mean $\Delta_{b_t}(H_t;H_0)$. Hence, Hoeffding's inequality gives:

$$\Pr\Big\{\Big|\widehat{\Delta}_{b_t,D_{\mathrm{conf},t}}(H_t;H_0) - \Delta_{b_t}(H_t;H_0)\Big| > x_t \,\Big|\, \mathcal{F}^-_{D_{\mathrm{conf},t}}\Big\} \le 2e^{-|D_{\mathrm{conf},t}|x_t^2/8} = \frac{\alpha}{4T}.$$

Taking expectations and a union bound over $t = 1,\dots,T$ contributes at most another $\alpha/4$.

A.2 Proof of Theorem 1

Let $K = K(\mathcal{A}) < \infty$. By definition, there exist $B_{b_1},\dots,B_{b_K} \in \mathcal{A}$ such that $B_{b_K\circ\cdots\circ b_1} = B_0$. Closure of $\mathcal{B}_{\mathrm{val}}$ gives $B_{b_j\circ\cdots\circ b_1} \in \mathcal{B}_{\mathrm{val}}$ for every $j = 1,\dots,K$.

For any H and $B_b, B_{b'} \in \mathcal{B}_{\mathrm{val}}$, the definition of ρ gives:

$$\begin{aligned}\Delta_{b'\circ b}(H;H_0) - \Delta_b(H;H_0) - \Delta_{b'}(H;H_0)\\ \le [\Delta_{b'\circ b}(H;H_0) - \Delta_b(H;H_0) - \Delta_{b'}(H;H_0)]_+ \le \rho.\end{aligned}$$

Taking $b = b_{j-1}\circ\cdots\circ b_1$ and $b' = b_j$ therefore yields, for $j = 2,\dots,K$,

$$\Delta_{b_j\circ\cdots\circ b_1}(H;H_0) \le \Delta_{b_{j-1}\circ\cdots\circ b_1}(H;H_0) + \Delta_{b_j}(H;H_0) + \rho.$$

The base case is:

$$\Delta_{b_1}(H;H_0) \le \Gamma_{\mathcal{A}}(H;H_0).$$

If

$$\Delta_{b_{j-1}\circ\cdots\circ b_1}(H;H_0) \le (j-1)\Gamma_{\mathcal{A}}(H;H_0) + (j-2)\rho,$$

then the preceding recurrence and $B_{b_j} \in \mathcal{A}$ give:

$$\begin{aligned}\Delta_{b_j\circ\cdots\circ b_1}(H;H_0) &\le (j-1)\Gamma_{\mathcal{A}}(H;H_0) + (j-2)\rho + \Gamma_{\mathcal{A}}(H;H_0) + \rho\\ &= j\Gamma_{\mathcal{A}}(H;H_0) + (j-1)\rho.\end{aligned}$$

Thus induction and $B_{b_K\circ\cdots\circ b_1} = B_0$ give:

$$\Delta_{\mathrm{BS}}(H;H_0) \le K\Gamma_{\mathcal{A}}(H;H_0) + (K-1)\rho. \tag{A.3}$$

Since $G_{\min,\mathcal{A}} = G_{\mathrm{rel}} - \Gamma_{\mathcal{A}}$ and $G_0 = G_{\mathrm{rel}} - \Delta_{\mathrm{BS}}$, the preceding bound also gives:

$$G_0(H;H_0) \ge G_{\min,\mathcal{A}}(H;H_0) - (K-1)\{\Gamma_{\mathcal{A}}(H;H_0) + \rho\}.$$

For certification, condition on $\mathcal{F}^-_{D_{\mathrm{cert}}}$. Then $H^\star$ and $\mathcal{A}$ are fixed. Let $m = |\mathcal{A}|$ and $n = |D_{\mathrm{cert}}|$. For each $B_b \in \mathcal{A}$, the task-level summands defining $\widehat{G}_{b,D_{\mathrm{cert}}}(H^\star;H_0)$ are

$$r_{B_b}(\tau_i^{H^\star,B_b},U_i) - r_{B_b}(\tau_i^{H_0,B_b},U_i) \in [-1,1],$$

while those defining $\widehat{\Delta}_{b,D_{\mathrm{cert}}}(H^\star;H_0)$ are

$$r_{B_{\mathrm{rel}}}(\tau_i^{H^\star,B_{\mathrm{rel}}},U_i) - r_{B_{\mathrm{rel}}}(\tau_i^{H_0,B_{\mathrm{rel}}},U_i) - r_{B_b}(\tau_i^{H^\star,B_b},U_i) + r_{B_b}(\tau_i^{H_0,B_b},U_i) \in [-2,2].$$

By the sampling conditions, these summands are conditionally independent across tasks and have conditional means $G_b(H^\star;H_0)$ and $\Delta_b(H^\star;H_0)$, respectively. Hence the one-sided Hoeffding inequalities give:

$$\begin{aligned}\Pr\Big\{G_b(H^\star;H_0) < \widehat{G}_{b,D_{\mathrm{cert}}}(H^\star;H_0) - s_G \,\Big|\, \mathcal{F}^-_{D_{\mathrm{cert}}}\Big\} &\le e^{-ns_G^2/2},\\ \Pr\Big\{\Delta_b(H^\star;H_0) > \widehat{\Delta}_{b,D_{\mathrm{cert}}}(H^\star;H_0) + s_\Delta \,\Big|\, \mathcal{F}^-_{D_{\mathrm{cert}}}\Big\} &\le e^{-ns_\Delta^2/8},\end{aligned}$$

where we set $s_G = \sqrt{\frac{2\log(4m/\alpha)}{n}}$, $s_\Delta = \sqrt{\frac{8\log(4m/\alpha)}{n}}$. A union bound over all $B_b \in \mathcal{A}$ then gives, with conditional probability at least $1 - \alpha/2$,

$$G_b(H^\star; H_0) \geq \widehat{G}_{b,D_{\text{cert}}}(H^\star; H_0) - s_G, \qquad \Delta_b(H^\star; H_0) \leq \widehat{\Delta}_{b,D_{\text{cert}}}(H^\star; H_0) + s_\Delta,$$

simultaneously for all $B_b \in \mathcal{A}$. Therefore,

$$G_{\min,\mathcal{A}}(H^\star; H_0) \geq \widehat{G}_{\min,\mathcal{A},D_{\text{cert}}}(H^\star; H_0) - s_G, \qquad \Gamma_{\mathcal{A}}(H^\star; H_0) \leq \widehat{\Gamma}_{\mathcal{A},D_{\text{cert}}}(H^\star; H_0) + s_\Delta.$$

Substituting these bounds into equation A.3 and the corresponding lower bound for G_0 gives the two inequalities in Theorem 1. The same probability bound holds unconditionally by the tower property.

A.3 Proof of Theorem 2

Work on the common evolution event $\mathcal{E}$ of Lemma 1. At every confirmed round,

$$\Delta_{b_t}(H_t; H_0) \geq \widehat{\Delta}_{b_t,D_{\text{conf},t}}(H_t; H_0) - x_t \geq \varepsilon + \gamma. \tag{A.4}$$

Let $t_1 < \cdots < t_m$ list the confirmed rounds. If $m = 0$, the claim is immediate. For $j < k$, archive nesting gives $B_{b_{t_j}} \in \mathcal{A}_{t_k-1}$. Empirical feasibility at the later round and the uniform estimation bound in Lemma 1 give:

$$\Delta_{b_{t_j}}(H_{t_k}; H_0) \leq \widehat{\Delta}_{b_{t_j},D_{\text{evo}}}(H_{t_k}; H_0) + r \leq \varepsilon + r. \tag{A.5}$$

Combining this inequality with confirmation at the earlier round yields:

$$\sup_{B_b \in \mathcal{B}_{\text{val}}} |\Delta_b(H_{t_j}; H_0) - \Delta_b(H_{t_k}; H_0)| \geq \Delta_{b_{t_j}}(H_{t_j}; H_0) - \Delta_{b_{t_j}}(H_{t_k}; H_0) \geq \gamma - r.$$

Because $0 < \gamma - r \leq 1$, the packing condition in Theorem 2 applies and gives $m \leq (C/(\gamma - r))^d$. If the round budget exceeds this bound and a valid proposal is tested every round, at least one round is not confirmed.

A.4 Proof of Theorem 3

Work on the event $\mathcal{E}$ of Lemma 1. If $\mathsf{Conf}_t = 0$, then:

$$\Delta_{b_t}(H_t; H_0) \leq \widehat{\Delta}_{b_t,D_{\text{conf},t}}(H_t; H_0) + x_t < \varepsilon + \gamma + 2x_t.$$

Because B_{b_t} solves the empirical Challenger problem, the D_{disc} deviation bound in equation A.2 gives:

$$\begin{aligned}\sup_{B_b \in \mathcal{B}_{\text{val}}} \Delta_b(H_t; H_0) &\leq \sup_{B_b \in \mathcal{B}_{\text{val}}} \widehat{\Delta}_{b,D_{\text{disc}}}(H_t; H_0) + r \\ &= \widehat{\Delta}_{b_t,D_{\text{disc}}}(H_t; H_0) + r \\ &\leq \Delta_{b_t}(H_t; H_0) + 2r < \varepsilon + \gamma + 2x_t + 2r.\end{aligned}$$

Now take any $\widetilde{H}$ satisfying $\Delta_b(\widetilde{H}; H_0) \leq \varepsilon - r$ for all $B_b \in \mathcal{B}_{\text{val}}$. Since $\mathcal{A}_{t-1} \subseteq \mathcal{B}_{\text{val}}$,

$$\widehat{\Delta}_{b,D_{\text{evo}}}(\widetilde{H}; H_0) \leq \Delta_b(\widetilde{H}; H_0) + r \leq \varepsilon, \qquad B_b \in \mathcal{A}_{t-1},$$

so $\widetilde{H}$ is feasible for the empirical Proposer. Hence,

$$\begin{aligned}R_{\text{rel}}(\mathsf{A}, H_t) &\geq \widehat{R}_{B_{\text{rel}},D_{\text{evo}}}(\mathsf{A}, H_t) - r \\ &\geq \widehat{R}_{B_{\text{rel}},D_{\text{evo}}}(\mathsf{A}, \widetilde{H}) - r \\ &\geq R_{\text{rel}}(\mathsf{A}, \widetilde{H}) - 2r.\end{aligned}$$

Subtracting $R_{\text{rel}}(\mathsf{A}, H_0)$ and taking the supremum over $\widetilde{H}$ gives the second claim. If $\varepsilon \geq r$, the comparison set contains H_0.

B Method Implementation Details

Section B.1 defines the comparison methods and shared settings. Section B.2 describes the Proposer and Challenger updates. Section B.3 specifies the protocol transformations and validity firewall.

Section B.4 summarizes the initial harness H_0, the fixed Proposer and Challenger harnesses , and the CHASE pseudocode. Section B.5 details the HarnessCompass generalization gate used in OfficeQA and Syn-Ledger. Section B.6 describes HarnessEvolve and our budget-constrained adaptation.

B.1 Comparison Methods and Shared Settings

Within OfficeQA or Syn-Ledger, all methods use the same foundation model and reasoning setting, initial harness H_0, request schema, sampling parameters, corpus, retrieval indexes, tools, context policy, step limit, time limit, retry policy, and scorer. For OfficeQA, all methods use `gpt-5.6-sol with high reasoning effort` for every model role, including the foundation model, Proposer, and Challenger. The original HarnessCompass configuration uses `gpt-5.4 in the non-thinking setting` (Zhang et al., 2026). We use a reasoning-enabled Proposer in OfficeQA so that any observed shortcut behavior cannot be simply attributed to insufficient reasoning. For Syn-Ledger, we instead use `gpt-5.6-sol in the non-thinking setting` for all model calls to reduce experimental cost. The foundation model A is not told which split each task belongs to. Each optimized method runs three Proposer rounds, except HarnessEvolve (Supplementary B.6). Table B.1 summarizes the six methods. RawHarness keeps H_0 unchanged to measure gains from optimization. The fixed generalization gate applies to HarnessCompass and B_0-Access; CHASE uses archive constraints without the generalization gate. HarnessEvolve bounds regression across recent task batches, whereas CHASE bounds gain destruction across archived counterfactual benchmarks.

Table B.1: Method comparison in this study. "Shortcut-aware" indicates that harness evolution explicitly accounts for shortcut behavior. "Worst-case constraints" bound the maximum regression across recent task batches (HarnessEvolve) or the maximum gain destruction across archived counterfactual benchmarks (CHASE) during optimization. "B_0-free" indicates no access to B_0 during optimization.

Method	Proposer	Challenger	Shortcut-aware	Worst-case constraints	B_0-free
RawHarness	✗	✗	✗	✗	✓
Meta-Harness	✓	✗	✗	✗	✓
HarnessCompass	✓	✗	✓	✗	✓
HarnessEvolve	✓	✗	✓	✓	✓
B_0-Access	✓	✗	✓	✗	✗
CHASE	✓	✓	✓	✓	✓

For cost control, we use two rollouts per harness–question pair on D_{evo}, two on D_{disc}, five on each $D_{\text{conf},t}$, and three on D_{cert}. We set $(\eta_{\text{conf},t}, \varepsilon) = (0.05, 0.05)$ on Syn-Ledger and $(0.075, 0.05)$ on OfficeQA.

B.2 Proposer and Challenger Updates

For controlled comparison, CHASE uses the same Proposer backbone as HarnessCompass, including its feedback procedure and separate structural and guidance tracks (Zhang et al., 2026). For HarnessCompass and B_0-Access, the generalization gate is specified once in the system prompt, fixed before optimization, and applied to every candidate edit. CHASE does not use the generalization gate.

At round t, HarnessCompass and CHASE each begin with their current harness H_{t-1} and evaluate it on the full D_{evo}. Following the history interface introduced by Meta-Harness (Lee et al., 2026), the Proposer can then inspect the method's complete optimization history through a controlled read-only file interface. This history contains the code of the current harness and all previously evaluated candidates, together with their scores and execution traces, including prompts, tool calls, model outputs, and state changes. The files retain their original names and relative paths so that the Proposer can navigate the history directly. HarnessCompass and CHASE have the same access to their own histories, but neither method can inspect the other method's rollouts. To reduce evaluation cost, we reuse previously computed estimates.

Proposer Update. Following HarnessCompass, the Proposer analyzes this history through two complementary feedback passes (Zhang et al., 2026). Proactive feedback describes how the current

harness affects the model's behavior and suggests possible improvements, while hindsight feedback uses the observed outcomes to identify behaviors associated with success or failure. The Proposer then develops one candidate for structural components and another for guidance components. Finally, it applies R^3—Revision, Recombination, and Refinement—to revise the two candidates and combine compatible changes into a third, integrated candidate (Zhang et al., 2026).

HarnessCompass selects a new harness only when one of the three candidates achieves a higher released-benchmark score; otherwise, it retains the current harness. In CHASE, before candidate generation, the Proposer is also given the current archive $\mathcal{A}_{t-1}$ and the corresponding D_{evo} scores and trajectories. The archive constraints are included in the Proposer prompt during candidate generation and are also enforced during candidate selection. After evaluation, CHASE forms a selection pool containing the three new candidates, H_{t-1}, and H_0. It removes every harness that violates $\widehat{\Delta}_{b,D_{\text{evo}}}(H; H_0) \leq \varepsilon$ for at least one $B_b \in \mathcal{A}_{t-1}$, then selects the remaining harness with the highest released-benchmark score on D_{evo} as H_t. Thus, an H_{t-1} that violates an archive constraint is excluded from the score comparison, while H_0 always provides a feasible fallback.

Challenger Update. After CHASE selects H_t, the Challenger searches for protocol transformations Φ_b under which the gain of H_t over H_0 may disappear. In each round, it initially generates eight proposals, each containing a natural-language description and a typed transformation specification. On OfficeQA, these specifications modify protocol-level features while preserving the underlying question and answer; see Supplementary B.3 for details.

Before any model rollout, the host applies the validity firewall and duplicate screen to all eight proposals. A proposal is discarded if it fails any validity check in Table B.2 or repeats a transformation already stored in the archive. To detect such repeats, the host represents each transformation as an ordered sequence of allowlisted API operators and their normalized arguments. A new proposal is treated as a duplicate when this representation matches that of an archived transformation. A hash of the representation is stored with each archived transformation and used for subsequent comparisons.

Among the proposals that pass both screens, CHASE selects two before any model rollout on D_{disc}, by a deterministic pre-evaluation ranking rule. The rule favors proposals that produce a larger observable change when applied to a fixed example, appear relevant to more discovery questions, and target more of the suspected shortcut mechanisms. When possible, the two selected proposals represent different kinds of protocol transformation. The retained proposals are then evaluated on D_{disc}, and CHASE selects the proposal with the largest $\widehat{\Delta}_{b,D_{\text{disc}}}(H_t; H_0)$.

Since the Challenger searches over D_{disc}, a large value of $\widehat{\Delta}_{b,D_{\text{disc}}}(H_t; H_0)$ may be inflated by selection. For OfficeQA, the selected transformation is fixed before confirmation and evaluated on a 24-question set $D_{\text{conf},t}$, with five rollouts per question. We evaluate both H_t and H_0 under the released benchmark B_{rel} and the proposed counterfactual benchmark B_{b_t}. For each harness–benchmark pair, the five rollout scores are first averaged within each of the 24 questions and then across questions. The counterfactual benchmark B_{b_t} enters the archive when the resulting task-level gain destruction is at least $\eta_{\text{conf},t} = 0.075$. After three rounds, CHASE selects its final harness from $H_0, \ldots, H_3$ using the constrained selection in Section 3. For fixed task and rollout budgets, the cost of evaluating each harness scales linearly with the number of archived benchmarks $|\mathcal{A}_{t-1}|$.

B.3 Protocol Transformations and Validity Firewall

OfficeQA permits reversible changes to identifiers, document layout, tool-interface schemas, and enumeration order, including valid compositions. For Syn-Ledger, the Challenger proposes transformations within the permitted protocol components; the canonical neutralizations and placebo transformations in Section D.2 are used only to validate the benchmark construction, not Challenger inputs. Across both benchmarks, transformations preserve the semantic task, target, available evidence, and scoring semantics (Ribeiro et al., 2020).

Table B.2 lists the seven checks evaluated by the host-side validity firewall. Each check returns a Boolean value, and a proposal passes the validity firewall only when all seven values are true. The Challenger's natural-language validity claims are retained in the audit record but are not used as substitutes for these executable checks.

Table B.2: Validity firewall: every check must pass.

Invariant	Required check
Task and answer	The question, requested target, and ground-truth answer are unchanged.
Available information	Document contents and membership match after undoing identifier mappings; no content is added, removed, altered, or moved between documents.
Access and resources	The same documents and tool operations remain accessible, with unchanged tool-call, context, and step limits.
Scoring	Answer normalization, numeric tolerance, and scorer verdicts on fixed correct and incorrect outputs are unchanged.
Declared changes	Only the protocol metadata declared in the proposal are modified.
Task independence	Rules do not depend on questions, answers, gold sources, or dataset splits.
Replay and inversion	Identical inputs and settings reproduce the transformation; reversible changes recover the original state.

Table B.3: Host-side pseudocode for the Syn-Ledger validity firewall.

Syn-Ledger validity firewall: host-side pseudocode.

1. Parse the Challenger output as a typed transformation specification s; reject any operator, argument, or modified field outside the allowlisted protocol API.
2. Canonicalize s into an ordered operator–argument representation. Reject s if the same canonical representation already occurs in the archive.
3. For every task U in the fixed validity-check set, execute $Z_1 \leftarrow \mathrm{Render}(U, s, \mathrm{seed})$ and $Z_2 \leftarrow \mathrm{Render}(U, s, \mathrm{seed})$, and parse Z_1 back into typed document records.
4. *Task and answer:* verify that the question and requested target are unchanged and that recomputing the answer from the parsed records returns the original ground-truth answer.
5. *Available information:* verify equality of document membership and of the canonical record multisets before and after transformation.
6. *Access, resources, and scoring:* verify equality of the accessible tool operations, resource limits, scorer hash, and scorer verdicts on fixed correct and incorrect output fixtures.
7. *Declared changes and task independence:* compute the modified fields and require them to be a subset of the fields declared by s; reject any reference to a question identifier, answer, gold source, or dataset split.
8. *Replay and inversion:* require $Z_1 = Z_2$ and require the typed document records recovered from Z_1 to equal those of U.
9. Return `PASS` if and only if all seven checks in Table B.2 pass; otherwise return the failed check names and reject the proposal before D_{disc} evaluation.

Table B.3 gives the Syn-Ledger implementation of the checks in Table B.2. The canonical parser reconstructs the typed records from each rendered document, after which the benchmark generator independently recomputes the ground-truth answer. Identity, valid single-field transformations, and valid compositions serve as positive controls; transformations that modify the answer, introduce undeclared task-dependent fields, or use a non-allowlisted operator serve as negative controls. OfficeQA applies the same requirement that all seven checks pass with benchmark-specific JSON canonicalization and tool-output fixtures; we omit the analogous OfficeQA pseudocode.

OfficeQA firewall example. In the first OfficeQA round, the Challenger first returned eight typed transformation proposals. The validity firewall and the duplicate screen accepted seven and rejected one before any model rollout on D_{disc}. The rejected proposal specified a seeded permutation of the search results. When executed on the validity firewall, however, it left the result order unchanged, so the duplicate screen excluded it as an inactive transformation and removed it from the candidate pool.

B.4 Initial Harness, Proposer Harness, Challenger Harness, and Pseudocode

Tables B.4–B.6 summarize the three harness designs, including their accessible context, tools, control, outputs, and condensed core instructions. The harness evolves from H_0; the Proposer and Challenger use fixed harness configurations. The instructions below are illustrative templates; benchmark-specific tool schemas and run histories are supplied separately.

Table B.4: Initial harness H_0.

Accessibility	**OfficeQA:** the evaluated release's full transformed-text corpus. **Syn-Ledger:** the current task's twelve documents. No gold annotations or generation metadata.
Memory	No task-specific long-term memory.
Tools / retrieval	**OfficeQA:** content-only passage BM25; conjunctive `all` matching on the unmodified query, with native BM25 ranking. **Syn-Ledger:** task-local tools and result order (Section D.1).
Control	Agent-directed tool use within the shared budget; no harness-side query rewriting, fallback retrieval, or reranking.
Core prompt. *Use the available tools to answer the question and return the answer in the required format.*	

Table B.5: Proposer harness: generating harness candidates.

Accessibility	Harness code, scores, and complete D_{evo} histories through read-only files; CHASE also receives $\mathcal{A}_{t-1}$ and its D_{evo} feedback.
Edit target	Foundation model prompts, memory, retrieval, tool wrappers, and control code; the foundation model, corpus, available tool operations, scorer, and budget stay fixed.
Control	Diagnose failures and regression risks; develop structural and guidance candidates, then integrate compatible changes through R^3 (Section B.2).
Output	Executable edits, targeted failures, activation and stopping conditions, expected tool cost, and regression risks.
Selection	**HarnessCompass:** released-benchmark score with the generalization gate. B_0**-Access:** $\widehat{R}_{B_0,D_{\text{evo}}}$ with the generalization gate. **CHASE:** released-benchmark score with $\widehat{\Delta}_{b,D_{\text{evo}}}(H;H_0) \le \varepsilon$ for every $B_b \in \mathcal{A}_{t-1}$.
Core prompt. *Improve scores on D_{evo}. Inspect every task–rollout result; distinguish stable successes, unstable outcomes, and stable failures. Separate observations from hypotheses. Use distinct mechanisms and concise conditional guidance; preserve behavior outside each edit's activation condition. Do not hard-code question identifiers, dataset membership, answers, or source filenames learned as labels.*	

Table B.6: Challenger harness: generating protocol transformations.

Accessibility	H_t, H_0, the archive, and scores and execution traces for H_t and H_0 on D_{disc} through read-only history files.
Edit target	Benchmark protocol transformations from Section B.3; both H_t and H_0 remain unchanged.
Control	Generate distinct proposals. The host rejects invalid or archived duplicates, ranks valid proposals on D_{disc}, and fixes the selected transformation before evaluation on $D_{\text{conf},t}$.
Output	Typed transformation specification, declared changes, suspected shortcut, supporting execution traces, and replay settings.
Core prompt. *Maximize $\widehat{\Delta}_{b,D_{\text{disc}}}(H_t;H_0)$: destroy gain over H_0, not merely raw score. Preserve questions, answers, document contents and membership, tool access, budgets, and scorer. Apply the same question-independent rule to both harnesses.*	

Algorithm 1 summarizes how the fixed Proposer and Challenger harnesses above interact with archive-constrained selection, the validity firewall, and confirmation on $D_{\text{conf},t}$.

Algorithm 1 CHASE evolution procedure

Require: initial harness H_0, foundation model A, Proposer, Challenger, released benchmark B_{rel}, task sets D_{evo}, D_{disc}, and $\{D_{\mathrm{conf},t}\}_{t=1}^{T}$, rounds T, tolerance ε, confirmation thresholds $\{\eta_{\mathrm{conf},t}\}_{t=1}^{T}$
Ensure: final harness $H^\star$ and counterfactual archive $\mathcal{A}_T$

$\mathcal{A}_0 \leftarrow \{B_{\mathrm{rel}}\}$
for $t = 1, \ldots, T$ **do**
  Evaluate H_{t-1} under every $B_b \in \mathcal{A}_{t-1}$ on D_{evo}, and update the Proposer history
  $(H_t^{\mathrm{str}}, H_t^{\mathrm{gui}}, H_t^{\mathrm{int}}) \leftarrow \textsc{ProposerUpdate}(H_{t-1}, \mathcal{A}_{t-1}, \text{Proposer history})$
  $\mathcal{S}_t \leftarrow \{H_t^{\mathrm{str}}, H_t^{\mathrm{gui}}, H_t^{\mathrm{int}}, H_{t-1}, H_0\}$
  Evaluate every $H \in \mathcal{S}_t$ under every $B_b \in \mathcal{A}_{t-1}$ on D_{evo}
  $\mathcal{F}_t \leftarrow \left\{ H \in \mathcal{S}_t : \widehat{\Delta}_{b,D_{\mathrm{evo}}}(H; H_0) \le \varepsilon \text{ for all } B_b \in \mathcal{A}_{t-1} \right\}$
  $H_t \leftarrow \arg\max_{H \in \mathcal{F}_t} \widehat{R}_{B_{\mathrm{rel}},D_{\mathrm{evo}}}(\mathsf{A}, H)$ ▷ constrained selection
  Evaluate H_t and H_0 under B_{rel} on D_{disc}, and update the Challenger history
  if $H_t = H_0$ **then**
    $\mathcal{A}_t \leftarrow \mathcal{A}_{t-1}$
    continue to the next round
  end if
  $\mathcal{P}_t \leftarrow \textsc{ChallengerUpdate}(H_t, H_0, \mathcal{A}_{t-1}, \text{Challenger history})$ ▷ generate typed proposals
  $\mathcal{P}_t^{\mathrm{sel}} \leftarrow \textsc{ScreenAndSelect}(\mathcal{P}_t, \mathcal{A}_{t-1})$ ▷ retain at most two preselected proposals
  $\mathcal{A}_t \leftarrow \mathcal{A}_{t-1}$
  if $\mathcal{P}_t^{\mathrm{sel}} \neq \varnothing$ **then**
    Evaluate H_t and H_0 under each B_b, $b \in \mathcal{P}_t^{\mathrm{sel}}$, on D_{disc}
    $b_t \leftarrow \arg\max_{b \in \mathcal{P}_t^{\mathrm{sel}}} \widehat{\Delta}_{b,D_{\mathrm{disc}}}(H_t; H_0)$
    Fix Φ_{b_t} and set $B_{b_t} \leftarrow \Phi_{b_t}(B_{\mathrm{rel}})$
    Evaluate H_t and H_0 under B_{rel} and B_{b_t} on $D_{\mathrm{conf},t}$ ▷ confirmation on $D_{\mathrm{conf},t}$
    if $\mathrm{Valid}(\Phi_{b_t}) = 1$ and $\widehat{\Delta}_{b_t,D_{\mathrm{conf},t}}(H_t; H_0) \ge \eta_{\mathrm{conf},t}$ **then**
      $\mathcal{A}_t \leftarrow \mathcal{A}_{t-1} \cup \{B_{b_t}\}$ ▷ add the confirmed counterfactual
    end if
  end if
end for
Evaluate $H_0, \ldots, H_T$ under every$B_b \in \mathcal{A}_T$ on D_{evo}
$\mathcal{F}^\star \leftarrow \left\{ H_t : t = 0, \ldots, T,\ \widehat{\Delta}_{b,D_{\mathrm{evo}}}(H_t; H_0) \le \varepsilon \text{ for all } B_b \in \mathcal{A}_T \right\}$
$H^\star \leftarrow \arg\max_{H \in \mathcal{F}^\star} \widehat{R}_{B_{\mathrm{rel}},D_{\mathrm{evo}}}(\mathsf{A}, H)$
return $H^\star, \mathcal{A}_T$

B.5 Generalization Gate Contracts and Enforcement

Our HarnessCompass baseline retains the optimization components reported by Zhang et al. (2026): proactive and hindsight feedback, separate structural and guidance tracks, R^3 integration, a fixed generalization gate, and released-score-based updates. The shared OfficeQA settings for HarnessCompass and CHASE are specified in Section B.1.

HarnessCompass places a fixed generalization gate between candidate generation and evaluation (Zhang et al., 2026). Its content requirement permits transferable decision rules with explicit applicability conditions and excludes rules tied to a task instance, test, private symbol, path, or task-specific token. Its placement requirement assigns executable capability changes to structural components and behavioral guidance to the system prompt or memory. We fixed a benchmark-specific realization of these requirements before optimization and applied it to every HarnessCompass and B_0-Access candidate in every round. The Proposer received the fixed contract in its controlled context: the full OfficeQA generalization-gate contract was available through the read-only interface, and the Syn-Ledger contract was supplied as `generalization_gate.txt`. Table B.7 records the Proposer-facing contract, and Table B.8 summarizes the host-side checks applied before foundation model evaluation.

B.6 Budget-Constrained HarnessEvolve Adaptation

All optimized methods in our experiments, except HarnessEvolve, use three outer optimization rounds. HarnessEvolve is organized on two nested scales: within each epoch, it processes a sequence

Table B.7: Fixed generalization gate contract supplied to the Proposer.

Common contract. Propose only rules that use information observable at inference time and can apply to unseen tasks. State the condition under which each rule applies. Do not encode a task identifier, split role, answer, gold source, private symbol or path, exact observed question, or data-derived lookup table. Put executable retrieval or control changes in structural fields; put behavioral advice in the system prompt, long-term memory, or guidance modules.
OfficeQA realization. Apply the contract jointly to the system prompt, long-term memory, retrieval policy, tool policy, policy graph, and guidance modules. Do not use question identifiers, labels revealing a task's split assignment or whether it is evaluated under the released or a counterfactual benchmark, observed-question text, ground-truth answers or source names, question-specific date–number pairs, or branches on task membership.
Syn-Ledger realization. Base candidate logic only on observable current inputs. Do not use task IDs, split roles, answers, gold sources, exact question lists, private filenames or paths, labels revealing which observable-feature levels are favorable, data-derived lookup tables, or branches on hidden roles, answers, sources, or feature assignments.

Table B.8: Host-side enforcement of the fixed generalization gate before model evaluation.

OfficeQA. The host validates the declared schemas and track boundaries, then scans all six editable harness components. It rejects identifiers and role or gold labels; any shared eight-token span with an observed question; any gold source filename; and any pairing of a year from an observed question with a sufficiently long number from its answer. A structural candidate must change a declared retrieval or control-policy field without changing any guidance components; a guidance candidate must leave structural policy unchanged; and an integrated candidate must contain both types of change. Candidate harnesses that make no effective change or duplicate a previously evaluated harness are also rejected.
Syn-Ledger. The host requires an integer-only answer schema, 1–12 search results, 1–6 opened files, and at most four search-query templates using only `entity`, `period`, `periods`, and `question` as placeholders. It enforces the structural and guidance track boundaries and requires an integrated candidate to combine both types of change from the structural and guidance candidates. It rejects overlap with exact task strings withheld from the Proposer; hidden role labels, answers, gold-source names, and terms used only by the hidden generator; experiment or task identity strings; and rules based on filename length, directory depth, search rank, or serialization format. A candidate proceeds only after its static audit returns `PASS`.

of batches, and each batch can propose a harness update, which is accepted only if it passes the quality and performance gates (Jiang et al., 2026). Held-out validation then selects a harness at the end of the epoch. A HarnessEvolve epoch can contain multiple potential harness updates and does not correspond to one round in our experiments. In our adaptation, round t is one batch-level update opportunity from H_{t-1} to H_t. At the start of each epoch, we independently repartition $D_{\text{evo}} \cup \left(\bigcup_{t=1}^{3} D_{\text{conf},t}\right)$ into three disjoint batches, with one batch processed in each round. Each task therefore serves as a current-batch task exactly once per epoch and three times over the full three-epoch run, although it may be reevaluated when the performance gate revisits an earlier batch.

For each task, the foundation model A receives the ground-truth answer only to generate a candidate reference trajectory. The *trajectory verifier* admits the trajectory only if its actions are grounded in observations and tool outputs and do not directly restate the supplied answer. If no reference is admitted within the allowed number of attempts, the task uses the failed-trajectory fallback. In round t, H_{t-1} is executed on the current batch without access to the ground-truth answer. For each failed execution, the Proposer compares the failed trajectory with its verified reference, when available, and locates their first action divergence. It then diagnoses the causes of the failures, groups failures with similar causes, and uses the resulting groups to propose an edit to H_{t-1}.

The candidate edit is evaluated by a quality gate and a performance gate, following (Jiang et al., 2026). The quality gate screens for task leakage and prompt bloat and may return the edit for revision. The performance gate requires the candidate to perform at least as well as H_{t-1} on the current batch and limits its score decrease on each of up to R previous-round batches to ϵ_{HE}. An accepted candidate defines H_t and enters the cumulative candidate pool; if the candidate is rejected, $H_t = H_{t-1}$. Table B.9 lists the settings adjusted for this three-epoch (nine-round) implementation.

Table B.9: HarnessEvolve settings adjusted for the nine-round outer optimization budget.

Budget item	HarnessEvolve (Jiang et al., 2026)	Budget-constrained adaptation
Optimization schedule	20 epochs with batch size 40. Each batch provides one gated update opportunity.	3 epochs, each with a three-batch partition of $D_{\text{evo}} \cup \left(\bigcup_{t=1}^{3} D_{\text{conf},t}\right)$: $40/40/41$ tasks for OfficeQA and $27/27/26$ for Syn-Ledger.
Performance gate	$R = 2$ and $\epsilon_{\text{HE}} = 0.025$.	$R = 2$ and $\epsilon_{\text{HE}} = 0.05$.

After each of our three epochs, the score on D_{disc} selects one harness from the pool containing H_0 and all distinct accepted harnesses obtained so far. The selections after epochs 1 and 2 initialize the next epoch, whereas the selection after epoch 3 is the final harness used for certification. Within each benchmark, our budget-constrained adaptation of HarnessEvolve uses the same initial harness, foundation model, tools, scorer, and final certification procedure as the other methods.

C Additional Results of OfficeQA

Section C.1 describes corpus preprocessing and data allocation, and Section C.2 specifies evaluation and reporting. Section C.3 presents the OfficeQA counterfactual and its implementation. Section C.4 reports token use during pre-certification optimization and certification.

C.1 Preprocessing and Task Allocation

OfficeQA Full contains 246 questions[2], which use the same corpus of 697 documents from the U.S. Treasury Bulletin collection spanning 1939–2025 (Opsahl-Ong et al., 2026; Databricks, 2026). Each question record includes an identifier, question, answer, source URLs, corresponding source files, and difficulty label. The corpus is released as original PDFs, parsed JSON, and transformed text with tables represented in Markdown. We use the transformed-text corpus throughout; the original PDFs and parsed JSON are not used. Question identifiers and texts are unique within the release.

We examine the question texts and corpus layout to quantify the formatting regularity described in Section 1 (Table C.1). A question mentions a numerical scale if it contains the whole word *thousand*, *million*, *billion*, or *trillion*, allowing plural forms and ignoring case. In the corpus, we count tables with at least two columns, a header, a delimiter row, and a data row. This yields 94,303 tables, including contents tables, repeated tables across bulletin editions, and separately rendered parts of longer tables. We identify unit statements by matching non-table lines containing explicit numerical-scale or unit expressions involving hundreds, thousands, millions, billions, trillions, dollar(s), cent(s), percent/percentage(s), basis points, ounces, or units. Matching ignores case and allows leading heading markers, an opening bracket, and an Amounts, Figures, or Dollar Amounts prefix, optionally preceded by All. This rule identifies 66,951 unit statements. A Note/Source line begins with either label and a colon, period, or dash, allowing plurals and leading heading markers. Adjacency ignores blank lines only.

Table C.1: Descriptive layout regularities in OfficeQA Full and its 697-document text corpus.

Criterion	Count / denominator	Percentage
Questions mentioning numerical scales	143 / 246	58.1%
Identified unit statements immediately before a table	63,747 / 66,951	95.2%
Table blocks preceded by an identified unit statement	63,747 / 94,303	67.6%
Table blocks followed by a Note/Source line	25,253 / 94,303	26.8%

All 697 documents contain examples of units before tables and Note/Source lines after tables. These counts describe marked text and its position; notes may apply to a table group or an entire bulletin. For a concrete example, Table FFO-7 in the April 1980 bulletin places its million-dollar unit above the table and explains below it that interfund payments are excluded when calculating trust-fund receipts and outlays. Reading near the table boundaries can therefore supply both the numerical scale

[2]OfficeQA Full: https://huggingface.co/datasets/databricks/officeqa

and an aggregation rule. The question count records explicit scale mentions; it does not identify which questions require a particular adjacent note.

OfficeQA Pro V2 contains 90 questions over a separate corpus of 1,435 parsed documents (Databricks, 2026)[3]. We apply the released conversion functions to all parsed inputs and freeze the resulting text before evaluation. The conversion does not load questions, answers, gold sources, or model outputs. Pro V2 results are analyzed separately from the primary OfficeQA experiment.

For each question, we collect all annotated source filenames, including those listed directly in the record and those recovered from its official source URLs. This yields 296 unique source filenames across the 246 questions in OfficeQA Full. We use shared source files to group related questions into 100 source components. The questions are then divided into D_{evo} (49 questions), D_{disc} (49 questions), three round-specific confirmation sets $D_{\text{conf},t}$ (24 questions each), and D_{cert} (76 questions). The source components assigned to D_{evo} and D_{cert} do not appear in any other split. By design, D_{disc} and the confirmation sets may share source components: each confirmation set contains 11 questions from components that also appear in D_{disc} and 13 questions from components that do not appear in D_{disc}. Questions never repeat across splits. Each confirmation set contains 11 easy and 13 hard questions. The allocation uses only question identifiers, difficulty labels, and source membership, without using model outcomes. The certification set D_{cert} is accessed only after the final harnesses, $\mathcal{A}_3$, and the analysis code are fixed.

A purely random question-level split would ignore the fact that multiple OfficeQA questions can rely on the same source files, making the comparison sensitive to accidental source overlap. We therefore use source components to control how related questions are distributed. We test whether a discovered benchmark-wide shortcut reproduces on new questions involving related sources, and whether the same shortcut extends beyond the sources used for discovery.

C.2 Evaluation and Reporting

We compute all OfficeQA correctness indicators using the released `fuzzy_match_answer` function at 0% numerical tolerance, following the scoring procedure in Alzubi et al. (2026). The same answer normalization and scoring rule are held fixed across B_{rel} and every counterfactual benchmark B_b. Gold source annotations are not exposed to the foundation model and do not enter the score.

The primary certification evaluates the final harness H from each method in the main comparison—RawHarness, Meta-Harness, HarnessCompass, HarnessEvolve and CHASE—on the 76 certification questions under every benchmark in $\mathcal{A}_3$. The archive contains B_{rel} and the counterfactuals confirmed during CHASE optimization. This evaluation contains $5 \times 76 \times |\mathcal{A}_3|$ method–question–benchmark combinations, with three rollouts per combination. Rollout scores are averaged within question and then across questions. OfficeQA Pro V2 is evaluated separately under its released protocol, with three rollouts per method–question pair; its score is denoted by $\widehat{R}_{\text{ProV2}}$. We also report the archive summaries $\widehat{R}_{\text{avg},\mathcal{A}_3}$ and $\widehat{R}_{\text{min},\mathcal{A}_3}$, using the shorthand defined in the main text.

C.3 Tracing Counterfactual Harness Search and Evolution

We examine H_1, B_{b_1}, and the harness candidates proposed in rounds 2–3 of the OfficeQA experiment.

The first-round Proposer and a possible shortcut. The initial harness H_0 is specified in Table B.4. Table C.2 reports the changes in H_1. Its tool-call budget remains unchanged from H_0. H_1 only adds model instructions and a conditional search-query rewrite.

H_1 broadens retrieval by allowing any query term to match and encourages the model to verify or calculate once the needed values appear to have been found. Its numerical guidance emphasizes units and precision, but does not explicitly tie each value to its table context, such as headers or nearby notes. We therefore hypothesize that the regular placement of such context in OfficeQA may itself signal that enough evidence has been found. Relocating that context could then change which values the model uses or whether it continues reading.

The first-round Challenger and the counterfactual. The Challenger returns a natural-language proposal together with a typed transformation specification. As illustrated in Figure C.1, the proposal

[3] OfficeQA Pro V2: https://huggingface.co/datasets/databricks/officeqa-pro-v2

Table C.2: Changes from H_0 to H_1.

Component	Changes from H_0
Prompt and memory	Adds reminders for recovery and numerical checking. Once the needed values are found, the model is encouraged to verify or calculate rather than continue searching.
Conditional guidance	Adds instructions for tracking units, converting scales, preserving numerical precision, and returning answers in the required order and format.
Search-query rewrite	When at least four tool calls remain, removes stopwords, keeps up to eight terms of at least three characters, and changes search from requiring all terms to allowing any term.
Tool-use guidance	Tells the model when to retry, to reserve two reads, and to stop once sufficient evidence has been found.

is to collect a table's title, unit line, footnotes, source notes, scope explanations, and surrounding explanatory prose before the table, targeting H_1's possible reliance on their customary locations.

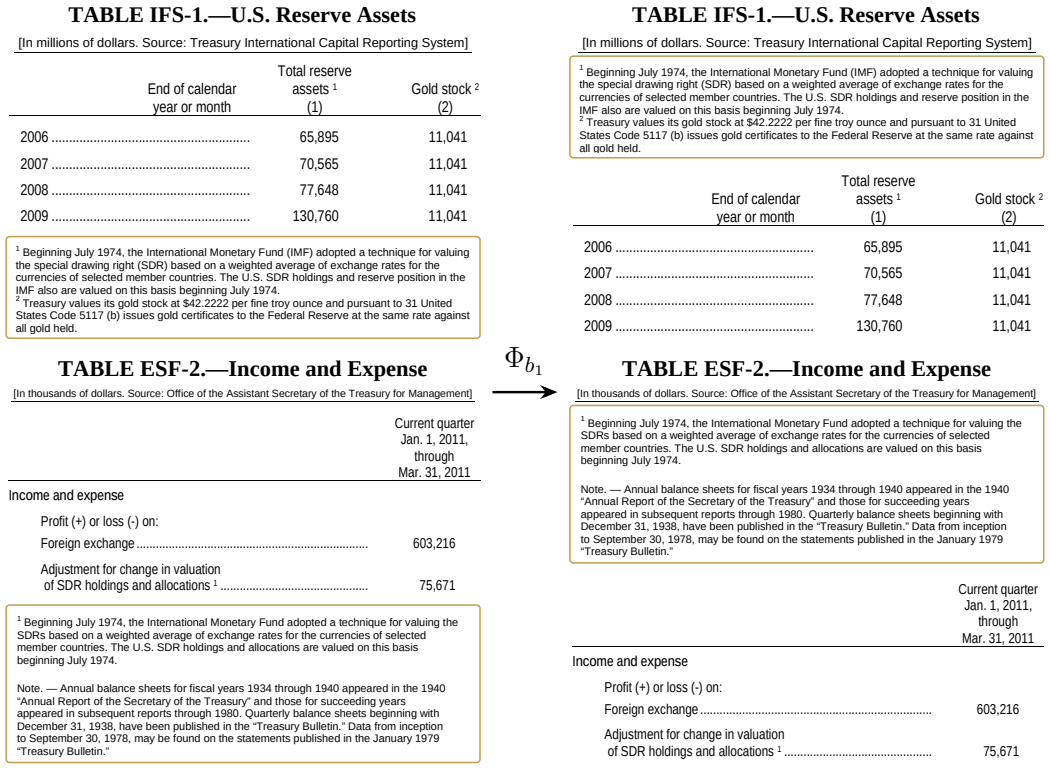

Figure C.1: Schematic illustration of the Challenger's JSON table-context re-encoding proposal on two OfficeQA excerpts from the September 2011 *Treasury Bulletin*: IFS-1 (top) and ESF-2 (bottom). Highlighted notes are shown above their tables only to visualize the `placement="before"` field; the evaluated transformation preserves the complete text string and does not reorder content within it.

For evaluation, we execute the Challenger-generated transformation Φ_{b_1} on the model-visible JSON search results. For each result containing a `text` field, the code moves the complete string into `table_context.content`, sets `table_context.placement` to `"before"`, and removes the original `text` field. The string's internal order, document identifier, and line references are preserved; no table parsing or within-string reordering is performed. This transformation applies to every text-bearing search result, changing its field names and nesting while leaving the underlying documents, result counts, read limits, tool-call budgets, questions, and scoring rule unchanged.

On D_{disc}, the gain destruction under Φ_{b_1} is 7.14%, compared with 4.08% for the other candidate transformation, a ten-line passage-boundary shift, which moves each eligible internal passage bound-

ary ten lines later while preserving complete document coverage and the original line order. We therefore select Φ_{b_1} for evaluation on $D_{\mathrm{conf},1}$. Across five rollouts for each of its 24 questions under B_{b_1}, the estimated gain destruction is 14%, exceeding the 7.5% confirmation threshold, and B_{b_1} is added to the archive.

Second-round candidates and fallback to H_0. In the second round, the Proposer generates three revisions of H_1 (Table C.3). The guidance candidate most directly responds to this concern by instructing the model to track the table context of each value it uses. The structural and integrated candidates instead emphasize combining evidence across periods and documents. All three retain H_1's prompt, memory, retrieval settings, and tool-use guidance.

Table C.3: Candidate pool of the second-round Proposer.

Candidate	Main changes
Structural	Decompose multi-period queries, merge results in round-robin order, promote document diversity, and combine needed values across documents.
Guidance	Bind each value to its title, date, unit, row label, and column header. Read adjacent headers or footnotes before using totals, subtotals, or year-to-date columns.
Integrated	Track each needed value by its fiscal year, calendar year, publication date, period, and whether it is cumulative; add rules for numerical and multi-period questions.

For the selection analysis, we apply $\varepsilon = 0.05$. Each harness is evaluated on D_{evo} under both B_{rel} and B_{b_1}. We compute $\widehat{\Delta}_{b_1} = \widehat{G}_{\mathrm{rel}} - \widehat{G}_{b_1}$ and require $\widehat{\Delta}_{b_1} \leq 0.05$. For the second-round and third-round selections, the archive contains B_{rel} and B_{b_1}, so B_{b_1} provides the only nontrivial constraint, as shown in Table C.4. All three candidates and H_1 fail, giving $H_2 = H_0$. The structural and guidance candidates come closest, but each loses 6.12%. Since the second-round selection returned $H_2 = H_0$, CHASE skipped Challenger generation, discovery, and confirmation in this round.

Table C.4: Second-round selection at $\varepsilon = 0.05$.

Harness	$\widehat{G}_{\mathrm{rel}}$	$\widehat{G}_{b_1}$	$\widehat{\Delta}_{b_1}$	Pass
H_1	+8.16%	−5.10%	13.27%	No
Structural	+15.31%	+9.18%	6.12%	No
Guidance	+9.18%	+3.06%	6.12%	No
Integrated	+15.31%	−2.04%	17.35%	No

Third-round candidates after failure feedback. The third-round Proposer starts from $H_2 = H_0$. Its input reports the second-round constraint failures and fallback, and explicitly states that a higher released-benchmark score alone is insufficient. Table C.5 summarizes the candidates. All three retain H_0's generic prompt and empty memory, rather than inheriting H_1's instruction to stop repeated searching once the needed values appear to have been found.

Table C.5: Candidate pool of the third-round Proposer.

Candidate	Main changes
Structural	Instructs the model to search separately for different periods, combine search results, and assemble evidence across documents, without changing the underlying retrieval procedure.
Guidance	Adds conditional instructions to convert source values to the requested units, preserve full precision during calculation, and follow the requested rounding order and answer format.
Integrated	Combines these types of numerical guidance with revised retrieval: retain all query terms except stopwords, first require all retained terms to match, and allow matches on any term only when the initial search has low coverage.

Because all second-round candidates violate the archive constraint, the third round starts from $H_2 = H_0$ while retaining feedback from the first-round counterfactual and the second-round constraint failures. The integrated candidate makes the clearest change to the earlier retrieval strategy: it preserves the query's specificity and uses broad matching as a recovery step rather than at the outset.

It combines this change with explicit numerical guidance, without inheriting H_1's instruction to shift away from repeated searching once the needed values appear to have been found. The resulting design therefore revisits both how evidence is retrieved and how the model is instructed to proceed after finding values, rather than simply adding another reminder about table context. This sequence illustrates how counterfactual feedback can inform subsequent harness design, not merely reject candidates with high released-benchmark scores.

As shown in Table C.6, the integrated candidate achieves the highest released-benchmark score among the candidates and is therefore selected as H_3. In contrast to the larger gains that violated the archive constraint in round two, the third-round revision yields a smaller released gain that satisfies the constraint.

Table C.6: Third-round selection at $\varepsilon = 0.05$.

Harness	$\widehat{G}_{\mathrm{rel}}$	$\widehat{G}_{b_1}$	$\widehat{\Delta}_{b_1}$	Pass
Structural	-1.02%	-1.02%	0.00%	Yes
Guidance	-2.04%	-6.12%	4.08%	Yes
Integrated	$+4.08\%$	0.00%	4.08%	Yes

Figure C.2 illustrates how the archived counterfactual shapes candidate selection. The second-round candidates achieve larger released gains but violate the archive constraint, whereas the third-round integrated candidate provides a smaller released gain within the same tolerance.

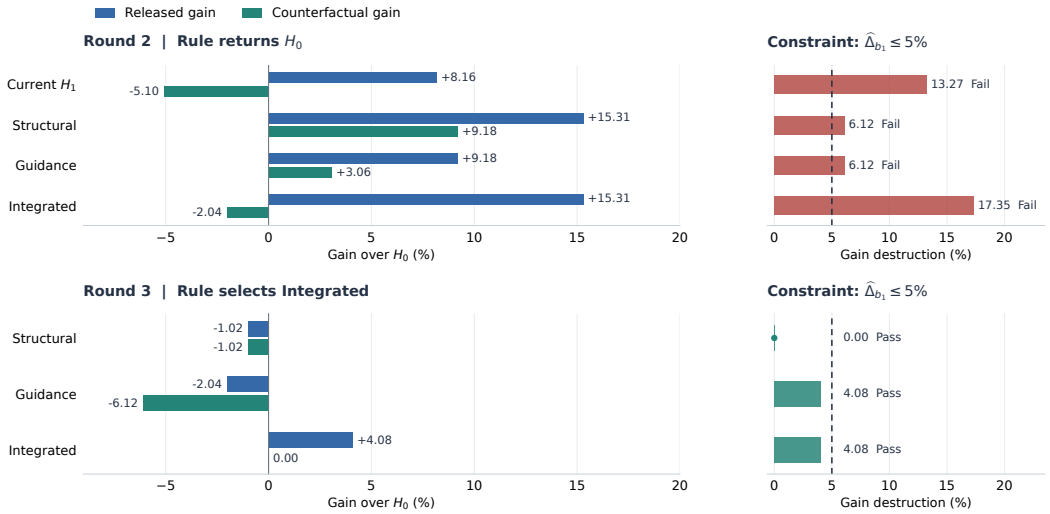

Figure C.2: OfficeQA candidate selection in rounds two and three. Left: released and counterfactual gains over H_0. Right: gain destruction and the archive constraint $\varepsilon = 0.05$.

After selecting H_3, the third-round Challenger evaluates two transformations on D_{disc}. A ten-line passage-boundary shift mentioned above produces a gain destruction of 10.20% and is selected for evaluation on $D_{\mathrm{conf},3}$. Across five rollouts on each of its 24 questions, the estimated gain destruction is 4.17%, which is below the 7.5% confirmation threshold. The transformation is therefore not added to the archive.

C.4 Token Use

We compare certification token use over 456 evaluation rollouts per method, with three rollouts per evaluated question–benchmark pair. Table C.7 reports total token use (input plus output tokens) and token use excluding cached input. The latter counts only uncached input and output tokens. Among the five methods, RawHarness used the fewest total tokens (93.82 million), whereas Meta-Harness used the fewest tokens excluding cached input (20.68 million).

Table C.7: Token use during certification with three rollouts per evaluated question–benchmark pair.

	Token use (millions)	
Method	Total	Excluding cached input
RawHarness	93.82	24.30
Meta-Harness	106.71	20.68
HarnessCompass	118.50	24.96
HarnessEvolve	132.60	23.36
CHASE	98.15	22.39

We report certification separately because RawHarness has no optimization stage, and the four optimized methods run different numbers and types of optimization episodes. An episode is one end-to-end execution of one model role on one assigned input; an episode may contain multiple model requests. Table C.8 includes every episode in the reported optimization and selection procedures with recorded token use, whether or not its candidate was retained. These totals describe the distinct optimization procedures rather than an episode-matched comparison. Meta-Harness used 224.31 million total tokens, whereas HarnessEvolve used 574.16 million. Compared with HarnessCompass, CHASE used 2.43 times as many total tokens; its optimization procedure also included Challenger discovery and confirmation.

Table C.8: Token use during optimization and selection. All episodes with recorded token use are included, whether or not their candidates are retained.

		Token use (millions)	
Method	Episodes	Total	Excluding cached input
Meta-Harness	983	224.31	45.74
HarnessCompass	1,390	283.02	58.59
HarnessEvolve	2,214	574.16	106.75
CHASE	3,439	688.96	154.42

D Additional Results of Synthetic Benchmark Syn-Ledger

Syn-Ledger contains 320 multi-document arithmetic tasks with controlled benchmark-wide shortcuts. Section D.1 describes task construction, the agent interface, and scoring; Section D.2 describes benchmark-wide shortcut construction and validation; Section D.3 gives the task allocation.

D.1 Task Construction, Agent Interface, and Scoring

A task consists of a question, twelve ledger documents, and an arithmetic program that determines the answer. Each ledger record specifies an entity, period, accounting category, approval status, version, accounting basis, and amount. The question identifies the requested entity and relevant periods and restricts the calculation to approved records on the enacted basis. Two evidence documents supply the required amounts. The ten distractors comprise two documents for periods not required by the task, two for an incorrect entity, two with superseded records, two with cancelled or draft records, one with an incorrect accounting basis, and one with an irrelevant category. Each document has six records, including contextual rows outside the requested entity or period.

Table D.1 lists the five task families and shows how their required amounts are distributed across the two evidence documents. Each program contributes 64 tasks and uses the same set of 64 distinct two-digit positive answers. The generator first fixes the answer and then samples amounts satisfying the corresponding arithmetic relation. Combined-total tasks use signed adjustments, and integer-share tasks satisfy $100x/y \in \mathbb{Z}$. Distractor amounts are sampled without using the required amounts. A task is rejected if its answer appears as a complete integer token anywhere in the visible task inputs. The correctness of Syn-Ledger is verified through a separate validation procedure described in Section D.2.

Table D.1: Syn-Ledger task families. The symbols x, y, z denote the amounts required by each calculation. All answers are integers.

Task family	Calculation	First evidence document	Second evidence document
Period difference	$x - y$	Current appropriation x	Previous appropriation y
Combined total	$x + y$	Capital adjustment x	Operations adjustment y
Net balance	$x + y - z$	Appropriation x, adjustment y	Obligation z
Integer share	$100x/y$	Allocated amount x	Reference total y
Reconciliation gap	$z - (x + y)$	Reported total z	Component amounts x, y

The model A retrieves evidence through `list_files`, `search`, and `open_file`, and can evaluate arithmetic with `calculator`. Each task allows at most twelve tool calls: four search/list calls in total, six file-opening calls, and two calculator calls. A file-opening call returns three records; another call reads the next three. Search returns four results by default and permits up to twelve.

Scoring uses normalized integer exact match. The scorer normalizes surrounding whitespace, signs, and valid comma grouping, then compares the submitted integer with the ground-truth answer. Explanations, multiple numbers, and decimal outputs receive zero. All benchmark variants B_b use the same tools, budgets, and scorer.

Each document is generated from a designated candidate-record block and additional contextual rows, for a total of six records. In an evidence document, the candidate block contains the one or two records required by the arithmetic program. In a distractor, the corresponding block contains structurally matched decoy records that are excluded by the entity, period, approval-status, version, accounting-basis, or category conditions in the question. The remaining records provide context but do not satisfy the requested conditions.

D.2 Benchmark-wide Shortcut Construction and Validation

To construct benchmark-wide shortcuts without changing task semantics, we vary five observable features of each document: filename, directory depth, search rank, candidate-record position, and serialization format (Table D.2). For each feature, we define a favorable and an unfavorable level. Under the released benchmark, required evidence is more often assigned favorable levels, creating associations that a harness can exploit without identifying evidence from document contents. For example, evidence documents may tend to have shorter filenames, shallower directories, or earlier search ranks. These assignments are stored separately from the ledger contents in a rendering manifest and applied by a deterministic renderer. Changing the manifest therefore changes only these observable features while preserving the question, ground-truth answer, and underlying ledger records.

Table D.2: Observable document features used to construct benchmark-wide shortcuts in Syn-Ledger. Search-rank manipulations change only the ordering of broad-query results and do not change document accessibility.

Feature	Favorable level	Unfavorable level
Filename	Short, regular, lexically early	Longer, lexically late
Directory depth	Shallow path	Deeper path
Search rank	Earlier in broad-query results	Later in broad-query results
Candidate-record position	Designated candidate-record block in the first `open_file` window	Designated candidate-record block in the continuation
Serialization	Fixed-order Markdown table	Equivalent CSV or key–value records

Under B_0, the five observable features are balanced with respect to evidence membership. In each task, one evidence document and five distractors share one feature configuration, while the other evidence document and five distractors take the opposite level of all five features. Across each 32-task block, the feature configuration assigned to one evidence document cycles through all 2^5 possible combinations. Consequently, every feature configuration occurs twice among the 64 evidence

documents and ten times among the 320 distractors. Thus neither any individual feature nor any interaction among the five features is associated with evidence membership under B_0.

To construct B_{rel}, we modify the feature levels assigned to the two evidence documents while keeping all distractor assignments fixed. For each observable feature, both evidence documents take the favorable level in 24 of the 32 tasks, while the remaining 8 tasks retain one favorable and one unfavorable evidence document. Hence, $56/64 = 7/8$ of the evidence documents have the favorable level for each feature. The eight tasks with one favorable and one unfavorable evidence document are chosen differently for the five features, preventing their favorable assignments from always occurring on the same tasks.

For each observable feature $j \in \{1, \ldots, 5\}$, we define a canonical neutralization Φ_j that replaces its assignment under B_{rel} with the corresponding assignment under B_0, while leaving the other four features unchanged. Each Φ_j is deterministic and idempotent, the five canonical neutralizations mutually commute, and applying all five recovers:

$$B_0 = (\Phi_5 \circ \cdots \circ \Phi_1)(B_{\text{rel}}).$$

We additionally define five placebo transformations that alter irrelevant presentation details for construction audits: line endings, metadata order, trailing whitespace, low-rank distractor order, and section labels. The distractor-ordering placebo leaves the rank of every evidence document unchanged. The five canonical neutralizations and five placebo transformations are used only to audit the benchmark construction; they are not revealed to the Challenger, which proposes transformations dynamically.

Validation of Syn-Ledger. We validate Syn-Ledger independently of the optimization and evaluation rollouts. For every task, the ground-truth answer is recomputed from both the typed document records and the rendered documents under B_{rel}, B_0, the five canonical neutralizations, and the five placebo transformations, and the two computations must agree. We also verify that both designated evidence documents are necessary: replacing either one while keeping the question, the remaining documents, and all observable-feature assignments fixed must change the resulting ground-truth answer. Additional checks verify the intended feature balance under B_{rel} and B_0, the composition of the five canonical neutralizations into B_0, deterministic rendering and replay, and scorer behavior on valid and malformed outputs.

D.3 Preprocessing and Task Allocation

The preceding construction gives each task three distinct attributes: its *task family* specifies what is computed, its documents carry *observable features* that create benchmark-wide shortcuts, and its *task set* specifies when the task is used in the experiment. The 32-task blocks are only a construction device for balancing the five observable features.

The 320 tasks are organized into ten complete 32-task blocks, indexed 0–9. We partition them into D_{evo}, D_{disc}, three round-specific confirmation sets $D_{\text{conf},t}$, and D_{cert}, with sizes $32 + 32 + 3 \times 16 + 208 = 320$ (Table D.3). Without using model scores or trajectories, a deterministic allocator selects

Table D.3: Syn-Ledger data allocation. Each task appears in exactly one set.

Task set	Size	Construction
D_{evo}	32	Block 0
D_{disc}	32	Block 1
$D_{\text{conf},t}$	16 per round	16 tasks selected from Blocks 2–7
D_{cert}	208	The remaining 24 tasks from each of Blocks 2–7, plus all tasks from Blocks 8 and 9

one eight-task subset from each of Blocks 2–7. For $t = 1, 2, 3$, the subsets selected from Blocks $2t$ and $2t + 1$ form $D_{\text{conf},t}$; the 24 unselected tasks in each of these six blocks enter D_{cert}. Blocks 0 and 1 are assigned intact to D_{evo} and D_{disc}, respectively, while Blocks 8 and 9 enter D_{cert} intact.

Each selected subset contains one or two tasks from every task family. For each observable feature, exactly 14 of the subset's $2 \times 8 = 16$ evidence documents take the favorable level under B_{rel}. Each confirmation set therefore has $28/32 = 7/8$ favorable evidence documents for every feature. Because a complete block has 56 favorable evidence documents out of 64 for every feature, the 24 tasks left

after selecting a valid subset have $56 - 14 = 42$ favorable evidence documents out of 48. It follows that D_{cert} has: $\frac{6\times42+2\times56}{6\times48+2\times64} = \frac{364}{416} = \frac{7}{8}$ favorable evidence documents for every feature. Figure D.1 summarizes the task construction and deterministic allocation.

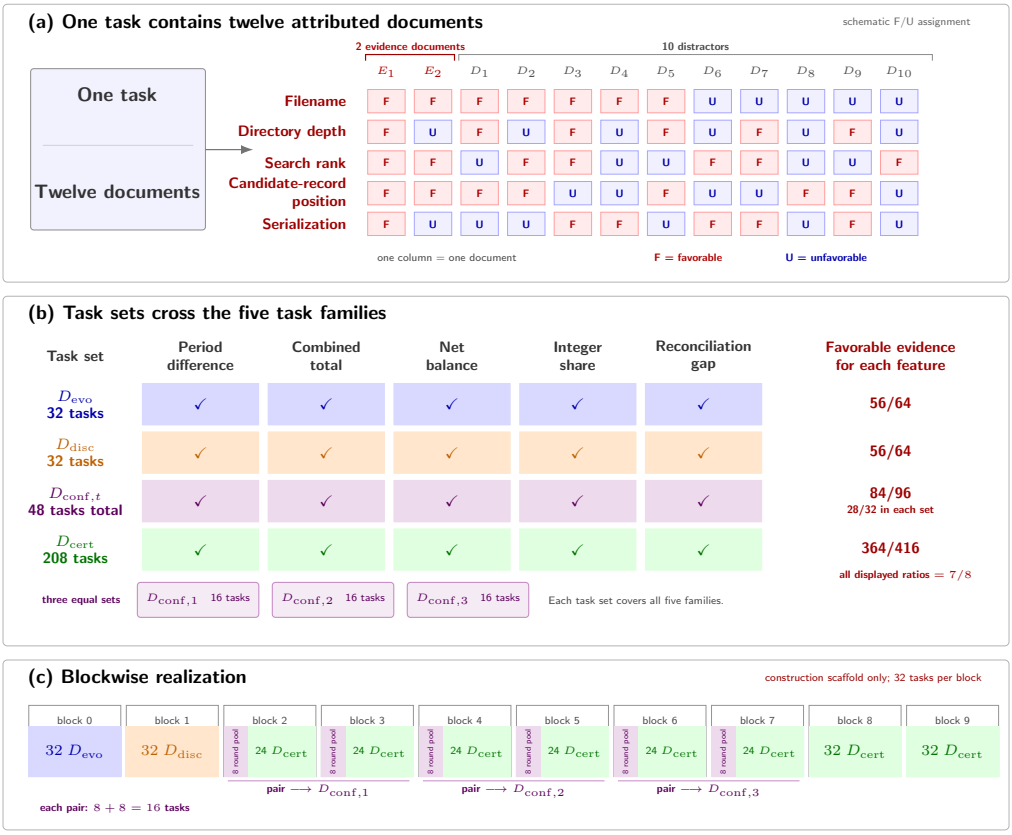

Figure D.1: Syn-Ledger task construction and deterministic allocation.

All task sets are required to cover the five task families. Exact equality of task-family counts is neither possible nor required because 32, 16, and 208 are not all divisible by five. Among allocations satisfying the preceding allocation constraints, we minimize the range of task-family counts within each task set and then balance the joint favorable/unfavorable patterns of each pair of observable features as evenly as possible; ties are resolved deterministically using a seeded hash of task identifiers. Before model evaluation, we verify set sizes, disjointness, task-family counts, the favorable rate for each feature, pairwise feature balance, and deterministic reconstruction.

Supplementary References

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Databricks. OfficeQA: A grounded reasoning benchmark suite. https://github.com/databricks/officeqa, 2026. Accessed 2026-09-01.

Wen Jiang, Mingmin Chu, Yimeng Tian, Qianxin Zhang, Haofei Yang, Rui Yang, Yang Liu, Tao Lv, and Fangming Li. HarnessEvolve: Learning from reference trajectories for reliable agent self-evolution. *arXiv preprint arXiv:2609.00829*, 2026.

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.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning. *arXiv preprint arXiv:2603.08655*, 2026.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.442.

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

Tong Zhang. *Mathematical Analysis of Machine Learning Algorithms*. Cambridge University Press, 2023. doi: 10.1017/9781009093057.