HARNESS EVOLUTION AS LEARNING: APPROXIMATION, GENERALIZATION, AND OPTIMIZATION LIMITS OF SELF-IMPROVING PERSONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

As the capabilities of large language models (LLMs) continue to advance, increasing attention is turning to how to translate their abilities into useful behavior. Personal agents bring this question into everyday settings, where models are expected to serve individual users and continually adapt to their preferences. With the underlying model held fixed, such adaptation relies on harness engineering: designing and evolving the surrounding layer that manages context, memory, tools, and execution. Despite rapid progress, the factors governing effective harness evolution remain insufficiently understood. To narrow this gap, we investigate three central questions concerning harness architecture, harness scale, and self-evolution algorithms through complementary empirical and theoretical analyses. Empirically, we introduce a preference-oriented benchmark and systematically characterize the capabilities and limitations of personal agents associated with these three dimensions. Theoretically, we formulate harness evolution as a learning problem and explain these phenomena through approximation, generalization, and optimization errors. Analyses of reachable policies, capacity under finite interaction evidence, and biased update dynamics provide theoretical accounts of the observed phenomena. Together, these results offer a unified perspective on the limits of personalization through harness evolution and inform future harness design. We open-source our code through an anonymous GitHub repository at https://anonymous.4open.science/r/harness-as-learning.

1 INTRODUCTION

As large language models (LLMs) become more capable, research attention increasingly extends from how to train a good model to how to use one to accomplish useful tasks (Wang et al., 2024a). LLM agents embody this direction by coupling language-model reasoning with tools, memory, and repeated interaction with an environment (Yao et al., 2023; Yang et al., 2024). The resulting applications bring a concrete promise of everyday assistance: an agent might locate a receipt across applications, organize the files needed for a project, or diagnose and repair a failing software test. Though extensive works turn parts of this promise into executable tasks, they also expose the difficulty of translating model capabilities into reliable actions (Trivedi et al., 2024; Jimenez et al., 2024). With the growing demand for commercial and open-source assistants, the design of more reliable and advanced agent systems has become an increasingly practical research question.

A particularly relevant setting is the *personal agent*: an assistant that works within a user's persistent workspace and is accessed through their own computers, servers, or everyday communication tools. OpenClaw (Peter Steinberger, 2026) and Hermes Agent (Nous Research, 2026) connect assistance to messaging and recurring daily workflows, while Claude Code (Anthropic, 2025c) and Codex (OpenAI, 2025) emphasize software development within the user's working environment. Personal agents must accommodate requirements that vary across users and persist across tasks, for example, a preferred message sign-off or conventions for payment notes, while following an API-backed deployment regime. In this regime, the agent runtime and personalization state are locally controlled, while the foundation model is remotely served and its parameters are not updated by the local agent. Hence, adapting behavior through external state and procedures becomes a practical alternative to modifying

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.



(a) Agent Fails in Complex Tasks (b) Agent Gets Lost in Long Context (c) Misunderstanding during Evolution

Figure 1: Schematic examples of three practical tensions in personal agent harness evolution: (a) effectiveness varies with preference requirements; (b) agents can overlook preferences stored in long memories; and (c) misinterpreted feedback can produce inaccurate or unnecessary memory updates.

the model itself (Zhao et al., 2024). One practical attempt is to make the model behave appropriately for an individual user over continued interaction by constructing an extra surrounding layer.

The surrounding layer that constructs model inputs, manages persistent state, exposes tools, and controls execution is commonly called an *agent harness*. Designing this layer is the subject of *harness engineering* (Li et al., 2026). Existing approaches span interaction interfaces, context retrieval and compaction, reusable skills, execution checks, and recovery procedures (Yang et al., 2024; Anthropic, 2025a;b). For a personal agent, this layer also provides a place to encode user-specific requirements and revise them as new evidence arrives. Such *harness evolution* has become an active research direction, with particular interest in self-evolution: using an LLM to interpret interaction feedback and update the memory, skills, or code that govern subsequent behavior (Zhang et al., 2026; Yang et al., 2026; Pan et al., 2026). The prospect is appealing: an assistant could become better adapted to its user simply through accumulated experience, without retraining its foundation model.

Yet the diversity of harness designs and evolution algorithms leaves three practical tensions, illustrated schematically in Figure 1. **First, the effectiveness of a harness depends on what the preference requires.** Panel (a) contrasts two everyday requests: a remembered instruction to use a friendly tone may suffice to shape an email draft, whereas honoring "leave at least 30 minutes between my meetings" requires checking calendar state and reasoning about time constraints before booking. This contrast is consistent with evidence that memory designs transfer unevenly across tasks (Pan et al., 2026; Zhou et al., 2026b). **Second, increasing harness size does not necessarily improve performance.** Panel (b) illustrates how an agent can omit a stored preference despite maintaining an increasingly comprehensive memory document. Accumulated instructions can interact with one another and interfere with the agent's output, so their benefits need not increase with their number (Zhao et al., 2025; Zhou et al., 2026a; Anthropic, 2025a). **Third, repeated self-evolution does not guarantee sustained improvement.** Panel (c) illustrates one potential source of this difficulty: a correction about a missing signature is turned into an unsupported full-name requirement and an additional politeness rule, suggesting that additional iterations alone may not overcome the underlying bottlenecks (Zhang et al., 2026; Nie et al., 2026; Liu et al., 2026). Although existing studies document individual failures and partial theoretical accounts have begun to emerge (Gan et al., 2026a), a common basis for standardized, quantitative empirical study of all three tensions, alongside a convincing theoretical account of how they arise and relate to one another in personal-agent harness evolution, remains lacking. This gap makes it difficult to determine when to change the harness mechanism, adjust its scale, or improve the evolution algorithm.

To address this gap, we study harness evolution from complementary empirical and theoretical perspectives. Empirically, we introduce a preference-oriented benchmark that makes the three tensions measurable and enables their systematic characterization. Theoretically, we cast harness evolution as a learning problem and organize its risk gap into *approximation*, *generalization*, and *optimization* errors, following the classical learning-theoretic separation. Latent-concept, information-theoretic, and fixed-point analyses then provide conditional explanations for the empirical phenomena.

Our contributions are twofold. **Empirically**, we provide a common testbed and controlled comparisons showing that explicit context reliably handles some local preferences while computational mechanisms improve more demanding ones, that memory expansion can yield non-monotonic compliance under the evaluated construction, and that several self-evolving recipes retain substantial oracle gaps. **Theoretically**, we introduce a unified learning-theoretic framework for harness evolution, linking architecture, memory scale, and update dynamics to approximation, generalization, and optimization errors, with analyses of reachability limits, memory-capacity and estimation bounds, and residual error under biased updates. The combined perspective links observed failures to distinct questions about harness design and learning. The remainder of the paper reviews related work in Section 2, presents the formulation in Section 3, and develops the empirical and theoretical analyses in Section 4. Finally, Section 5 draws a brief conclusion.

2 Related Works

Personal Agents. Personal agents increasingly operate in persistent user environments, from messaging-based assistants such as OpenClaw (Peter Steinberger, 2026) to workspace-based coding assistants such as Codex (OpenAI, 2025) and Claude Code (Anthropic, 2025c). Earlier research established memory as a substrate for persistent agent behavior (Park et al., 2023; Packer et al., 2024). Personalization also requires applying memory appropriately. Zhao et al. (2025) evaluate whether models infer and follow user preferences across extended conversations. Wang et al. (2026) take a complementary route by converting subsequent user and environment responses into training signals for online reinforcement learning. Our setting fixes the foundation model and studies how a local harness learns to express user preferences through the available interaction and feedback channels.

Harness Engineering. An agent's behavior depends on how the harness system constructs context, exposes actions, maintains state, and checks outcomes. Yao et al. (2023) first interleave reasoning with environment actions, Yang et al. (2024) further demonstrate the importance of interfaces for navigating, editing, and executing code. Wang et al. (2024b) then use executable code as a compositional action representation. Instead, Xia et al. (2025) structure software repair into localization, repair, and patch validation. Industrial accounts similarly emphasize context curation and compaction, persistent progress artifacts, and testing across sessions (Anthropic, 2025a;b). Verification and evaluation provide complementary evidence: Jimenez et al. (2024) test repository-level repairs, while Trivedi et al. (2024) evaluate tool-mediated tasks through environment outcomes.

Self-Evolving Harness. The community has begun to apply automated adaptation to modify several components of a harness. Shinn et al. (2023) store verbal feedback in an episodic buffer, and Zhao et al. (2024) extract reusable insights from task experience. Agrawal et al. (2026) search over prompts using trajectory reflection and evaluated candidate selection. For persistent context, Zhang et al. (2026) use structured playbooks and a grow-and-refine process. Zhou et al. (2026a) track the validity of keyed precedents and revoke stale memories. More recent approaches consider targeting dynamic user adaptation (Yang et al., 2026; Zhou et al., 2026b). Evolution can also change the mechanism that uses memory or controls actions (Zhang et al., 2025; Pan et al., 2026; Lou et al., 2026; Nie et al., 2026; Gan et al., 2026b). Analytical treatments are also emerging. Liu et al. (2026) decompose an oracle-harness gap into evolution and adaptation losses to guide harness construction.

3 Preliminary & Problem Formulation

We formulate personalization as learning a local harness around a frozen foundation model, and use a three-way risk decomposition to organize the questions studied in this paper.

3.1 Personal Agents and Local Harness

Let $z = (x, e)$ denote a task, where x is the natural-language instruction and e contains the application state and available tools. A frozen foundation model f induces a distribution over interaction traces $\tau = (a_1, o_1, \ldots, a_T, o_T)$, which we write as $\tau \sim \pi_f(\cdot \mid z)$. Here a_t is an attempted action (for example, an API call) and o_t is the observation returned by the environment. The notation emphasizes that f is held fixed throughout the paper; all adaptation happens outside its parameters.

Harnesses are local stateful transducers that mediate between the model and the environment. For a local state s_t and model history h_t, a harness c can construct the next model input, maintain auxiliary state, and optionally transform or reject an action: $(\bar{h}_t, \bar{a}_t, s_{t+1}) = c(h_t, a_t, s_t, e)$. The composition $c \circ f$ therefore defines an induced policy $\pi_{c \circ f}$. This notation covers common personalization mechanisms: contextual memory and system-prompt edits change $\bar{h}_t$; checkers, routers, statistics, and autofill components can additionally change $\bar{a}_t$ or the available action set.

Specifically, we distinguish two broad hypothesis classes. A *context harness* has $\bar{a}_t = a_t$ and acts only through an injected token sequence (memory, skills, or prompt text). A *control harness* may compute over the interaction state or intervene in the action path, for example by returning a derived value, maintaining a counter, or rewriting a preference-owned field.

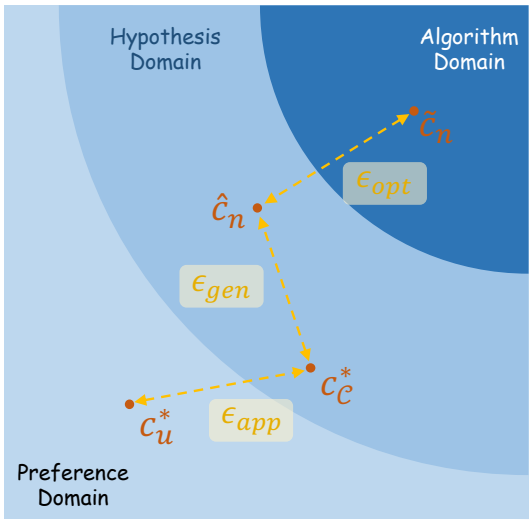


Figure 2: Harness evolution as learning. Approximation, generalization, and optimization errors separate the user optimum, class optimum, empirical optimum, and learned harness, respectively.

3.2 Harness Evolution as Learning

Let f^*_u denote the ideal policy that would express the user's preferences if the model and the local harness were unrestricted. We evaluate a policy under the user's task distribution $\mathcal{P}_u$ with loss ℓ_u and risk $R(\pi) = \mathbb{E}_{z \sim \mathcal{P}_u,\, \tau \sim \pi(\cdot|z)}\left[\ell_u(\tau, z)\right]$. The personalization objective is consequently $\min_{c \in \mathcal{C}} R(\pi_{c \circ f})$, ideally approaching $R(\pi_{f^*_u})$.

The user-optimal policy is not observed directly. During interaction i, the agent produces a trace τ_i and receives weak user or environment feedback y_i. The learner observes the stream $S_n = \{(z_i, \tau_i, y_i)\}_{i=1}^n$, where y_i may be only acceptance/dissatisfaction, an artefact-level complaint, or a more informative correction.

For a fixed harness class $\mathcal{C}$, define the population-optimal harness and the empirical-risk minimizer by $c^*_{\mathcal{C}} \in \arg\min_{c \in \mathcal{C}} R(\pi_{c \circ f})$ and $\hat{c}_n \in \arg\min_{c \in \mathcal{C}} \widehat{R}_n(\pi_{c \circ f}; S_n)$ respectively, where $\widehat{R}_n$ is the empirical risk induced by the observed feedback channel. An evolution algorithm is a model- and channel-dependent update operator $c_i = U_f(c_{i-1}; z_i, \tau_i, y_i)$, and $\tilde{c}_n = U_f(S_n)$, with $\tilde{c}_n$ the harness actually deployed after n interactions.

3.3 Error Decomposition: Approximation, Generalization and Optimization

Consider the real risk gap between $\pi_{\tilde{c}_n \circ f}$ and $\pi_{f^*_u}$ during the harness evolution. By adding and subtracting the risks of $\pi_{c^*_{\mathcal{C}} \circ f}$ and $\pi_{\hat{c}_n \circ f}$, we obtain the following risk decomposition:

$$R(\pi_{\tilde{c}_n \circ f}) - R(\pi_{f^*_u}) = \underbrace{R(\pi_{c^*_{\mathcal{C}} \circ f}) - R(\pi_{f^*_u})}_{\epsilon_{\text{app}}(\mathcal{C}, f^*_u)} + \underbrace{R(\pi_{\hat{c}_n \circ f}) - R(\pi_{c^*_{\mathcal{C}} \circ f})}_{\epsilon_{\text{gen}}(\mathcal{C}, n)} + \underbrace{R(\pi_{\tilde{c}_n \circ f}) - R(\pi_{\hat{c}_n \circ f})}_{\epsilon_{\text{opt}}(U_f; \mathcal{C}, S_n)}.$$

The first term (ϵ_{app}) is known as **Approximation Error**, and is a property of the reachable policy class. The second term (ϵ_{gen}) is known as **Generalization Error**, related to data and its effective capacity. And the third term (ϵ_{opt}) is often called **Optimization Error**, which is related to the evolution algorithm. As illustrated in Figure 2, the global optimal harness solution c^*_u lies within the preference domain, while the optimal solution $c^*_{\mathcal{C}}$ in the hypothesis set $\mathcal{C}$ lies within the hypothesis domain. The gap between c^*_u and $c^*_{\mathcal{C}}$ constitutes ϵ_{app}. The empirically optimal solution $\hat{c}_n$ also lies within the hypothesis domain, but is constrained by the limited interaction data, thus the gap between $\hat{c}_n$ and $c^*_{\mathcal{C}}$ constitutes ϵ_{gen}. Moreover, the realistic solution $\tilde{c}_n$ obtained during the evolution lies within the algorithm domain, the gap between $\tilde{c}_n$ and $\hat{c}_n$ constitutes ϵ_{opt}.

These three types of errors characterize gaps from different perspectives within the harness evolution process. Approximation error suggests that the specific implementation of the harness inherently determines the upper limit of the evolutionary outcome. Generalization error considers the finite interaction and indicates that the size of the harness matters. Moreover, optimization error is closely linked to the specific evolutionary algorithm employed. The limitations of the algorithm itself will affect the quality of the solution. This decomposition provides the theoretical basis for the remainder of the paper, and we further use them to analyze the following proposed research questions:

Q1 (harness architecture, approximation error). Which user preferences can context harnesses reliably express, and when does additional computation or action-level control improve compliance? We compare harness mechanisms while fixing the model, actuator, and task distribution.

Q2 (harness scale, generalization error). How does the scale of a memory harness affect preference compliance when its contents are constructed from finite interaction evidence? We study this relationship through memory-length sweeps under a fixed model and preference set.

Q3 (self-evolving algorithm, optimization error). To what extent can repeated harness updates close the gap to an oracle context? We compare update recipes under a common feedback protocol and track their held-out performance across interaction checkpoints.

We study the proposed three research questions through complementary empirical and theoretical analyses. For each question, we first conduct controlled experiments on *AppWorld-P*, a preference-oriented benchmark built on AppWorld (Trivedi et al., 2024) that adds executable user-preference checks to the original tasks. The empirical analysis examines whether the phenomenon of interest arises in practice and characterizes when and how it occurs as we vary the harness mechanism, memory scale, or update procedure while keeping the foundation model fixed. We then analyze each question theoretically, identifying conditions and mechanisms that can explain the observed behavior under explicit modeling assumptions. This empirical-to-theoretical progression organizes section 4: each subsection first presents the experimental evidence and then develops a theoretical interpretation through the lens of approximation, generalization, or optimization error.

4 Main Results

In this section, we present the main results for Q1—Q3 in sections 4.1 to 4.3, respectively. In each subsection, we first briefly introduce the problem settings and then present our empirical observations. Finally, we provide the theoretical analysis for understanding the empirical observations. A brief introduction to our experimental environment is as below.

Benchmark and evaluation. We conduct our experiments on AppWorld-P, a preference-oriented extension of AppWorld (Trivedi et al., 2024), an interactive environment in which agents perform everyday tasks through application APIs. Across all experiments, the foundation model is `claude-haiku-4-5-20251001` (Anthropic, 2025d). AppWorld-P preserves the original task environments and adds a preference-evaluation layer. Each user persona consists of executable rules specifying when a preference applies and how compliance is checked from the agent's actions and the resulting state. For example, a rule may require text messages to end with the user's first name or payment notes to follow a particular format. These rules make preference compliance programmatically measurable, while access to preference information is controlled by the experimental condition. For experiments involving harness evolution, the agent receives feedback after training interactions and updates its local harness; at each checkpoint, the harness is frozen and evaluated on held-out tasks. Static harness configurations are evaluated directly. This framework supports all three questions, with preferences and task pools tailored to each comparison. Our primary metric is the *preference violation rate*: the fraction of applicable episode–rule pairs that violate the corresponding preference. Lower values indicate better compliance with the applicable preferences. Appendix A provides full persona definitions, task splits, feedback channels, and implementation details, including the rule-specific checks and preference information available to each harness during interaction and evaluation.

4.1 Approximation: On the Capability Boundaries of Context Harness

For Q1, we examine whether contextual specifications suffice for reliable preference compliance, or whether implementing a preference requires additional computation or action-level control.

Problem setting. Two preference groups probe the theoretical support distinction. **(i) In-Support:** local instructions the model is expected to follow once specified, including *Private note format* (format payment notes using a prescribed template) and *SMS sign-off* (end messages with the user's first name). **(ii) Out-of-Support:** preferences requiring history-based inference or exact computation, including *SMS character checksum* (compute a character-count checksum for an outgoing message), *Habitual card* (infer the user's habitual payment card from past interactions), and *Running spend total* (maintain the cumulative amount paid to a recipient). Fixing the foundation model, constrained function-call actuator, and evaluation tasks, we compare five conditions: **(i) Context:** *No memory* (no personalization context), *Stated* (explicit golden preference description), and *From history* (history-derived context); **(ii) Control:** *Checker rejects* (a light-weight control implementation, verification and retry without computed answers) and *Harness computes* (well designed control harness for specific preferences, including external computation or action rewriting). We report per-preference violation rates on applicable episodes. Detailed protocols are provided in Appendix A.1.

Empirical observation. The results show that explicit context is sufficient for the tested local instruction-following preferences, whereas history-dependent and computational preferences benefit substantially from additional harness mechanisms. As shown in Figure 3, stating the preference reduces violations of *Private note format* and *SMS sign-off* from 1.00 to 0.07 and 0.00, respectively. For *SMS character checksum*, *Habitual card*, and *Running spend total*, stated context leaves rates of 0.50, 0.78, and 0.44. History-derived context improves habitual-card inference and reduces running-total violations to 0.22, without reliably resolving all three preferences. Lightweight *Checker rejects* achieves zero observed running-total violations through verification and retry, and also reduces violation rates to 0.38 on checksum and 0.39 on habitual card. *Harness computes* further reduces these two rates to 0.17 and 0.00, while also achieving 0.00 on running totals. Thus, Q1 calls for matching the mechanism to the preference: textual specifications can activate familiar behaviors, but compliant executions and computational preferences need a control-based guidance. We next analyze how these mechanisms shape reachable policies through a latent-concept model.

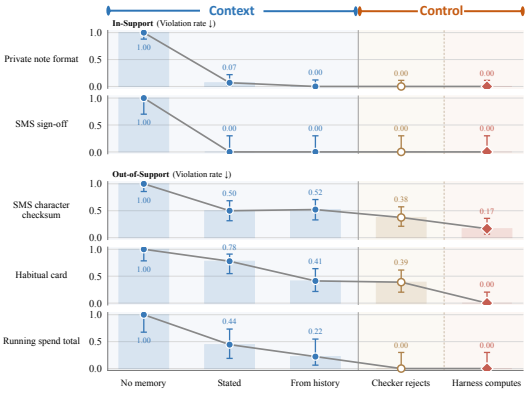


Figure 3: Preference violations by harness type. Context largely resolves in-support preferences; computation or action rewriting substantially improves compliance on out-of-support preferences.

Theoretical analysis. We analyze this separation through a latent-concept model of the frozen model's reachable policies. Let $P_\theta := p(\cdot \mid \cdot, \theta)$ be the trace policy for behavioral concept θ, and $\mathcal{M}_f := \overline{\mathrm{conv}}\{P_\theta : \theta \in \operatorname{supp} p(\theta)\}$ the closed convex hull of policies supported by the pretrained prior $p(\theta)$. Under Assumption B.1, context reweights these concepts independently of the test task, while leaving each P_θ fixed. The context- and control-reachable sets are $\mathcal{R}_{\mathrm{ctx}} := \{\pi_{cof} : c \in \mathcal{C}_{\mathrm{ctx}}\}$ and $\mathcal{R}_{\mathrm{ctrl}} := \{\mathcal{T}_c \pi_f : c \in \mathcal{C}_{\mathrm{ctrl}}\}$, where $\mathcal{T}_c$ transforms model-proposed traces into executed traces, for example by computing an action argument or rewriting a preference-owned field. The following theorem links these reachable sets to approximation error, identifying when control can attain the user optimum while the context class retains a strictly positive risk gap (proof in Appendix B.1).

Theorem 4.1 (Approximation gap of context and control harnesses)**.** *Under Assumption B.1,* $\mathcal{R}_{\mathrm{ctx}} \subseteq \mathcal{M}_f$*, whereas* $\mathcal{R}_{\mathrm{ctrl}}$ *need not be contained in* $\mathcal{M}_f$*. If the user-optimal policy* $\pi_{f_u^*} \in \mathcal{R}_{\mathrm{ctrl}} \setminus \mathcal{R}_{\mathrm{ctx}}$ *and* $\Delta_u := \inf_{\pi \in \mathcal{R}_{\mathrm{ctx}}} R(\pi) - R(\pi_{f_u^*}) > 0$*, then* $\epsilon_{\mathrm{app}}(\mathcal{C}_{\mathrm{ctrl}}, f_u^*) = 0 < \epsilon_{\mathrm{app}}(\mathcal{C}_{\mathrm{ctx}}, f_u^*) = \Delta_u$.

Messages from Theorem 4.1. The theorem answers Q1 through policy reachability: *context* selects among existing behaviors, while *control* can transform their execution. Under the stated risk separation, refining context alone cannot eliminate the approximation gap, whereas a suitable control harness can reach the user-optimal policy. This perspective explains the contrast in Figure 3: in-support formatting and sign-off requirements activate familiar instruction-following behaviors, while out-of-support habitual-card inference and running totals benefit from maintained statistics, and checksums from explicit computation. For harness design, the implication is to match the mechanism

to the preference: use context to express supported behaviors, and introduce targeted state tracking, computation, or action rewriting where reliable execution requires these operations.

4.2 Generalization: On the Impact of Harness Scale

Having compared harness mechanisms, we turn to Q2: whether adding more learned statements to a context harness consistently improves preference compliance under finite interaction evidence.

Problem setting. We use eight preferences from three applications: **(i) SMS:** *SMS greeting* (start with a greeting), *SMS sign-off* (end with the user's first name), and *SMS terseness* (use at most five words); **(ii) Venmo:** *Venmo private* (mark transactions private), *Payment has note* (include a non-empty payment note), *Payment-note lowercase* (write notes entirely in lowercase), and *Single-word payment note* (use exactly one word per note); **(iii) Spotify:** *Playlist over like* (save songs to playlists rather than liking them individually). Memory blocks draw from a fixed pool of statements learned from interaction trajectories and instance-level complaints, rather than hand-written oracle specifications. We select statements round-robin across preference groups, reusing entries for longer blocks; all concern scored preferences, with no unrelated content added. Holding the foundation model, actuator, preference set, and evaluation tasks fixed, we test nine lengths, $L \in \{0, 1, 2, 3, 5, 10, 20, 60, 150\}$, including the no-memory reference $L = 0$. Each block is shuffled separately for each (L, seed) and frozen for evaluation over three seeds, 12 tasks, and 10 rollouts per task. We report eight-rule violation rates and assess scoring-set sensitivity by re-scoring the same episodes from the full eight-preference harness on six rules: we exclude *SMS terseness*, which can conflict with task-required message content and formatting, and *Payment has note*, which is implied by *Single-word payment note*. See Appendix A.2 for memory-construction and evaluation details.

Empirical observation. Increasing memory scale improves preference compliance initially, but does not produce sustained gains under the evaluated memory construction. Figure 4 shows that the pooled violation rate decreases from approximately 0.77 at $L = 0$ to 0.20 at $L = 10$, consistent with improved coverage as more preferences are represented in memory. Further expansion reverses part of this improvement: violations rise to approximately 0.25–0.27 at $L = 20$–150, even though every injected statement concerns a scored preference. The six-rule re-scoring retains the same overall decline-and-rise pattern at lower absolute rates, indicating that the pattern is not solely an artifact of including the two excluded rules in the aggregate metric. These results answer Q2 by showing that a larger collection of relevant, learner-generated statements is not necessarily a more effective harness. The following theoretical analysis examines the competing contributions of preference coverage and estimation error under finite interaction, providing a stylized account of why increasing memory scale can have both benefits and costs.

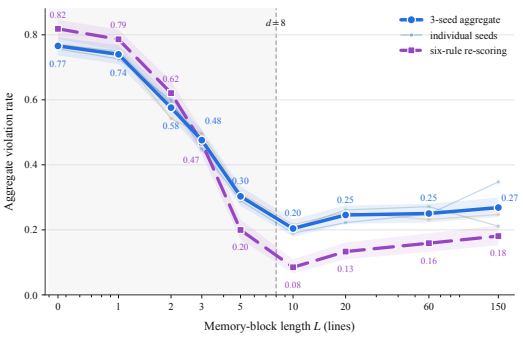


Figure 4: Non-monotonic violations across memory lengths. Solid: three-seed pooling over eight rules; faint: individual seeds; dashed: the same evaluation episodes re-scored on six preferences.

Theoretical analysis. To explain the observed reversal in performance, we combine an information-theoretic memory model with concentration bounds for noisy feedback. Consider d independent, unbiased binary preferences, evaluated uniformly and executable once correctly specified. Let $\mathcal{C}_L$ contain context harnesses with at most L statements, with population optimum $c^*_{\mathcal{C}_L}$. The learned harness $\hat{c}_{n,L}$ minimizes statement-wise feedback disagreement by majority vote, using $m = \lfloor n/L \rfloor$ observations per statement from a budget of n observations. We assume that recovering all target statements reproduces class-optimal preference behavior, linking statement-inference errors to the generalization gap. The following theorem bounds the resulting generalization error and shows how this bound increases with memory length L when the feedback budget n is fixed (detailed proofs are presented in Appendices B.2–B.3).

Theorem 4.2 (Generalization cost of harness scale)**.** *Under the above model, let* $1 \le L \le n$ *and suppose each feedback observation independently reports its associated preference correctly with probability* $1/2 + \gamma$*, where* $0 < \gamma \le 1/2$*. Then*

$$\mathbb{E}_{S_n}[\epsilon_{\text{gen}}(\mathcal{C}_L, n)] \le L \exp\left(-2\gamma^2 \left\lfloor \frac{n}{L} \right\rfloor\right).$$

For fixed n and γ, this generalization bound is strictly increasing in integer L. Storing $\min\{L, d\}$ correct preferences gives the coverage bound $\epsilon_{\text{app}}(\mathcal{C}_L, f_u^*) \le (d-L)_+/d$, where $(x)_+ := \max\{x, 0\}$. Combining the two bounds yields the following excess-risk guarantee.

Corollary 4.3 (Coverage–estimation tradeoff)**.** *Under the same model and feedback assumptions,*

$$\mathbb{E}_{S_n,U}\left[R(\pi_{\hat{c}_{n,L}\circ f}) - R(\pi_{f_u^*})\right] \le \frac{(d-L)_+}{d} + L \exp\left(-2\gamma^2 \left\lfloor \frac{n}{L} \right\rfloor\right).$$

Messages from Theorem 4.2 and Corollary 4.3. Theorem 4.2 identifies a growing statistical burden behind memory expansion: a fixed feedback budget must support more statements, leaving fewer observations per statement and more opportunities for incorrect inference. Corollary 4.3 combines this increasing estimation bound with a decreasing coverage bound. Once coverage saturates, further expansion no longer reduces the approximation bound, while the estimation bound continues to increase. These competing effects can yield a U-shaped upper bound, consistent with the initial improvement and subsequent degradation in Figure 4. Q2 therefore reflects a coverage–estimation tradeoff: relevant statements expand what a harness can represent, but their reliability depends on the evidence supporting them. For harness design, memory growth should track both the amount and reliability of feedback: retain statements that add supported preference coverage, consolidate redundant descriptions, and gather further evidence before expanding uncertain parts of the memory.

4.3 Optimization: On the Limits of Self-Evolving Harness

Beyond harness architecture and scale, Q3 concerns whether repeated feedback-driven updates approach oracle compliance and how this progress depends on the update procedure.

Problem setting. We fix a persona with five preferences: *SMS greeting* (begin messages with a greeting word), *SMS sign-off* (end with the user's first name), *Venmo private* (mark transactions private), *Payment-note category* (begin notes with the category tag), and *Payment-note initials* (end notes with parenthesized dotted initials). Training feedback consists of instance-level complaints that specify the desired behavior for the first three preferences but identify only the defective field for the latter two, withholding the required category tag or exact initials format. This tests the use of available feedback. We compare **(i) Static references:** *No memory* (no personalization memory) and *Oracle context* (explicit descriptions of all five preferences); and **(ii) Self-evolving harnesses:** adaptations of ACE (Zhang et al., 2026) (incrementally updated itemized memory), TEPA (Zhou et al., 2026a) (keyed precedents with replacement and revocation), TRACE (Zhou et al., 2026b) (learned runtime checks with retries), and Reflexion (Shinn et al., 2023) (a bounded buffer of verbal reflections). Diagnostic arms use full-memory rewriting with instance-level or fully specified corrective feedback. With the foundation model, actuator, persona, and feedback protocol fixed, each of the four main methods updates from its own outcomes over 24 training interactions and three seeds. At $n \in \{0, 6, 12, 18, 24\}$, we freeze each harness and evaluate it on six held-out tasks with three rollouts per task. We compare self-evolving harnesses with the static references to characterize improvement and the remaining oracle gap. Detailed protocols are provided in Appendix A.3.

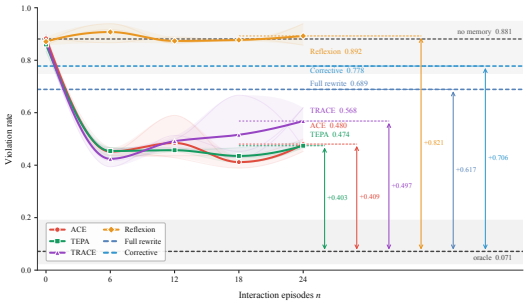


Figure 5: Self-evolution trajectories and remaining oracle gaps. Curves show three-seed means with min–max bands for the self-evolving harnesses.

Empirical observation. Repeated harness updates narrow the oracle gap for some recipes, but none of the tested procedures closes it within the evaluated interaction budget. As shown in Figure 5, ACE, TEPA, and TRACE reduce violations to approximately 0.4–0.5 after six interactions, compared with the no-memory reference of 0.881. Their subsequent trajectories differ. ACE and TEPA fluctuate around similar levels and finish near 0.48, leaving a gap of approximately 0.40 to the oracle rate of 0.071. TRACE loses part of its early improvement and ends near 0.57, while Reflexion remains close to the no-memory reference throughout. The full-rewrite and corrective-feedback diagnostics also finish above the oracle, at approximately 0.69 and 0.78, respectively. The answer to Q3 is therefore that self-evolution can improve personalization, but additional interactions do not necessarily translate into continued progress toward oracle compliance. The similar ACE and TEPA endpoints suggest that the persistent gap is not specific to one memory-update recipe, while the other trajectories demonstrate the importance of the update procedure. Because the feedback withholds some target details, this gap must also be interpreted in light of the information available to the learner. We next examine conditions under which biased update dynamics can sustain residual error, providing a theoretical perspective on the limits of repeated self-evolution.

Theoretical analysis. We interpret these trajectories through a time-homogeneous Markov model of online harness updates and a Wasserstein fixed-point analysis. For a fixed model f, capacity L, and feedback protocol, let U denote the update rule. Write $\nu_{n,L}$ and $\widehat{\nu}_{n,L}$ for the distributions of the deployed harness $\tilde{c}_{n,L}$ and empirical minimizer $\hat{c}_{n,L}$. The operator $\mathcal{T}_U$ maps a harness distribution to its distribution after one interaction and update. With W_L denoting the first Wasserstein distance under a harness metric d_L, define $\beta_{U,n,L} := W_L(\mathcal{T}_U\widehat{\nu}_{n,L}, \widehat{\nu}_{n,L})$, the displacement induced by updating the distribution of empirical minimizers. Assumptions B.3–B.5 quantify contraction by $\rho < 1$, quadratic empirical-risk growth by $\mu > 0$, and empirical-to-population estimation risk by $\epsilon_{n,L} \to 0$ as $n \to \infty$. The following theorem links a persistent update residual to a positive optimization floor (a detailed proof is provided in Appendix B.4).

Theorem 4.4 (Persistent optimization floor under biased updates)**.** *Under Assumptions B.3–B.5, $\nu_{n,L}$ converges to a unique invariant distribution and its expected risk converges, while*

$$\liminf_{n\to\infty} \mathbb{E}[\epsilon_{\mathrm{opt}}(U; \mathcal{C}_L, S_n)] \geq \frac{\mu\beta_0^2}{2(1+\rho)^2} > 0.$$

If a family of update rules satisfies these assumptions with common constants ρ, μ, β_0, the same positive lower bound holds for every rule in the family.

Messages from Theorem 4.4. The theorem answers Q3 by showing how online harness updates can stabilize while retaining a positive optimization gap. Even as risk-estimation error vanishes, continued iterations need not escape a suboptimal stationary regime. This obstruction can arise under different update recipes satisfying the stated assumptions, while allowing different trajectories and limiting risks. Such dynamics provide a mechanism consistent with the performance plateaus and persistent oracle gaps in Figure 5. The lower bound highlights how changing the update recipe can leave a common source of error unresolved, revealing the inherent bottleneck of self-evolving algorithms. Harness design should therefore target these sources of update bias through more informative feedback, grounded correction signals, and independently validated memory edits, aligning the update dynamics with the preference objective.

5 Conclusion

We studied how harness design and evolution shape personalization around a frozen foundation model. Experiments on AppWorld-P reveal that preference compliance depends on the harness mechanisms available, does not improve monotonically with memory size, and can remain below oracle performance despite repeated updates. Our learning-theoretic formulation connects these observations to distinct limitations in approximation, generalization, and optimization, providing conditional explanations for when and why adaptation falls short. These findings suggest that reliable personalization requires jointly considering what a harness can express, how its capacity matches the available evidence, and how feedback guides its updates. This perspective provides a unified account of diverse harness-engineering practices by identifying which error components they seek to reduce, supporting both failure diagnosis and more principled design in future research.

AI USE STATEMENT

We used OpenAI Codex and Anthropic Claude Code to assist with core code implementation, data construction and processing, evaluation, interpretation of results, translation, and the checking and refinement of mathematical statements and proofs. Additionally, we used these tools for literature search and synthesis, manuscript drafting and editing, and figure preparation. The initial research ideas, conceptual framework, hypotheses, proof strategies, and experimental design were developed by the authors without AI assistance. The authors reviewed all AI-assisted work, including the text, references, mathematical arguments, code, data and evaluation procedures, and reported results and figures. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies personal-agent adaptation in the simulated application environments of AppWorld-P using explicitly specified preference rules. The evaluated messaging, payment, and other application actions take place within the benchmark environment. Our aim is to understand limitations that can undermine reliable personalization. In practical deployments, persistent memories may contain sensitive information, and incorrectly inferred preferences may lead to unintended actions. Responsible deployment therefore requires user consent, protection of stored preferences and interaction logs, and appropriate oversight of consequential actions. Our findings motivate assessing preference compliance alongside privacy, safety, and user control.

REPRODUCIBILITY STATEMENT

Section 4 describes the benchmark, evaluation metric, and experimental comparisons. Appendix A details the preference rules, task selection, memory construction, feedback channels, model configuration, seeds, and training and evaluation protocols. The assumptions and proofs supporting our theoretical claims are provided in Appendices B.1–B.4. The code is open-sourced via an anonymous GitHub repository listed at the end of the abstract.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning, 2026. URL https://arxiv.org/abs/2507.19457.

Anthropic. Effective context engineering for ai agents, 2025a. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Anthropic. Effective harnesses for long-running agents, 2025b. URL https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents.

Anthropic. Claude code: Anthropic's agentic coding system, 2025c. URL https://www.anthropic.com/product/claude-code.

Anthropic. Claude haiku 4.5, 2025d. URL https://www.anthropic.com/claude/haiku.

Zeyu Gan, Ruifeng Ren, Wei Yao, Xiaolin Hu, Gengze Xu, Chen Qian, Huayi Tang, Zixuan Gong, Xinhao Yao, Pengwei Tang, Zhenxing Dou, and Yong Liu. Beyond the black box: A survey on the theory and mechanism of large language models, 2026a. URL https://arxiv.org/abs/2601.02907.

Zeyu Gan, Huayi Tang, and Yong Liu. Statistical priors for implicit preferences: Decoupling skill selection as a local harness in personal agents, 2026b. URL https://arxiv.org/abs/2606.05828.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Yingqiang Ge, Weijie Xu, Xi Fang, Xiang Xu, Tianchen Zhao, Youngeun Kim, Jihun Hamm, Tianyang Wang, and Chandan Reddy. Agent harness engineering: A survey, 2026. URL https://openreview.net/pdf?id=eONq7FdiHa.

Zewen Liu, Zhan Shi, Yisi Sang, Bing He, Minhua Lin, Tianxin Wei, Dakuo Wang, Benoit Dumoulin, Wei Jin, and Hanqing Lu. Adaptive auto-harness: Sustained self-improvement for agentic system deployment on open-ended task streams, 2026. URL https://arxiv.org/abs/2606.01770.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. Autoharness: improving llm agents by automatically synthesizing a code harness, 2026. URL https://arxiv.org/abs/2603.03329.

Jun Nie, Yonggang Zhang, Jun Song, Qianshu Cai, Dahai Yu, Yike Guo, Xinmei Tian, and Bo Han. Tthe: Test-time harness evolution, 2026. URL https://arxiv.org/abs/2607.08124.

Nous Research. Hermes agent — the agent that grows with you, 2026. URL https://hermes-agent.nousresearch.com/.

OpenAI. Codex: Ai coding partner from openai, 2025. URL https://openai.com/codex/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Wenbo Pan, Shujie Liu, Xiangyang Zhou, Shiwei Zhang, Wanlu Shi, Mirror Xu, and Xiaohua Jia. M*: Every task deserves its own memory harness, 2026. URL https://arxiv.org/abs/2604.11811.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701320. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Peter Steinberger. Openclaw — personal ai assistant, 2026. URL https://openclaw.ai/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of computer science*, 18(6):186345, 2024a.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/wang24h.html.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking, 2026. URL https://arxiv.org/abs/2603.10165.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying llm-based software engineering agents. *Proc. ACM Softw. Eng.*, 2(FSE), June 2025. doi: 10.1145/3715754. URL https://doi.org/10.1145/3715754.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026. URL https://arxiv.org/abs/2603.01145.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems, 2025. URL https://arxiv.org/abs/2512.18746.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8a94ff6f922d995d7d3f4ebf4143e442-Paper-Conference.pdf.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 15888–15931, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/28a46044775d97a4efcbcf14e7f13209-Paper-Conference.pdf.

Yan Zhou, Yue Ouyang, Kaiyang Zheng, and Suncheng Xiang. Tepa: Revoking stale memories for conflict-robust language agents, 2026a. URL https://arxiv.org/abs/2608.07429.

Yujun Zhou, Kehan Guo, Haomin Zhuang, Xiangqi Wang, Yue Huang, Zhenwen Liang, Pin-Yu Chen, Tian Gao, Nuno Moniz, Nitesh V. Chawla, and Xiangliang Zhang. Getting better at working with you: Compiling user corrections into runtime enforcement for coding agents, 2026b. URL https://arxiv.org/abs/2606.13174.

A Detailed Experimental Settings

This section provides the complete experimental protocol underlying the three main questions. All experiments use AppWorld-P, our preference-evaluation layer over AppWorld (Trivedi et al., 2024). AppWorld itself is left unchanged: each task is executed in its original application world, while AppWorld-P attaches a persona consisting of programmatic preference rules. A rule specifies (i) when it is applicable, (ii) whether the recorded API calls satisfy the preference, and (iii) the feedback exposed to the harness after a training episode. Rule semantics and checker state are never exposed to the agent except through the arm-specific channel described below.

Across all experiments, the foundation model is `claude-haiku-4-5-20251001` (Anthropic, 2025d). We use a constrained function-call actuator: at each agent step, the model may emit exactly one API call with literal arguments. This restriction prevents the model from outsourcing arithmetic or aggregation to an interpreter and makes the distinction between contextual and computational harnesses well defined. Unless stated otherwise, the model temperature is 0.7, the per-task step budget is 50, and evaluation uses a frozen task set. The primary metric is the micro-averaged preference violation rate,

$$\widehat{V} = \frac{\sum_{i,r} \mathbf{1}[r \text{ is applicable in episode } i]\mathbf{1}[r \text{ is violated in episode } i]}{\sum_{i,r} \mathbf{1}[r \text{ is applicable in episode } i]}. \tag{1}$$

Episodes in which the constrained action is not taken are not counted as either a success or a violation for that rule. Where confidence intervals are reported, we use Wilson 95% intervals.

Concretely, a persona is a small declarative bundle of rules rather than a free-form prompt. Each rule names the API action it constrains, the field or state variable it checks, and a natural-language statement used only by the corresponding context arm. For example, the Q1 persona can be represented as:

```
name: q1_format_checksum
rules: [private_note_format, sms_char_checksum]
private_note_format.trigger: venmo.create_transaction
sms_char_checksum.trigger: phone.send_text_message
```

The checker decides whether each rule is applicable from the recorded API calls and then evaluates the resulting note or message. Thus the same persona can be attached to different harness arms without changing the task world or the model.

A.1 Detailed Experimental Settings for Q1

Objective and preference selection. Q1 tests whether changing the harness implementation changes the set of preferences that the fixed model can satisfy. We first calibrate candidate rules using two static conditions: no memory and a single, maximally explicit natural-language description of the preference. A rule is labeled *in-support* when the stated description yields near-perfect compliance and the no-memory condition does not; it is labeled *out-of-support* when a substantial violation rate remains despite the perfect description. These labels are determined empirically rather than from the rule's source file or our prior judgment.

The final experiment contains five preferences (Table 1). The two in-support rules test formatting and sign-off instructions that the model can implement once stated. The three out-of-support rules cover complementary failure modes: exact computation on the current action, statistical inference from an interaction history, and exact aggregation across both past and within-episode actions.

Harness conditions. The five columns in Figure 3 compare contextual specifications with two forms of execution control:

1. **No memory** injects no personalization context and provides the unpersonalized reference.
2. **Stated** injects the rule's clearest natural-language description. This removes preference-induction error and provides the principal context-class reachability reference.
3. **From history** asks the model to use interaction-derived context. For the four rules in the main AppWorld streams, a top-L assertion updater selects three statements from a bank

Table 1: Preferences used in Q1. The family assignment is determined by the perfect-description calibration.

Preference	Family	Trigger	Requirement
Private note format	In-support	Venmo payment	The note follows a private bracketed format; its code is determined by the payment amount.
SMS sign-off	In-support	Text message	The message ends with the user's first name.
SMS character checksum	Out-of-support	Text message	The message ends with a checksum equal to its character count modulo seven.
Habitual card	Out-of-support	Venmo payment	The first attempted payment card is the user's latent habitual bank card.
Running spend total	Out-of-support	Venmo payment	The note ends with the cumulative amount successfully sent to that recipient, including the current payment and earlier successful ones.

containing the two persona rules and six distractors, using corrective feedback over the training stream specified below. For the habitual-card rule, this condition contains the adoptable rule statement and the complete raw habit log; no statistic is precomputed.

4. **Checker rejects** combines the stated rule with up to three whole-episode attempts. A programmatic checker accepts the first attempt with an applicable action and no preference violation; otherwise it supplies rejection feedback for the next attempt. This channel restates the constraint without returning the computed target value. For habitual card, it also identifies the rejected card and accumulates exclusions across attempts.
5. **Harness computes** supplies preference-specific external computation. For character checksum, the checker returns the expected value for the previous message to the model, with up to three fresh attempts; it does not directly rewrite the suffix. The habitual-card harness maintains per-card acceptance rates and injects their argmax. The running-total harness maintains a ledger, writes the computed total into the API's actual `description` field before execution, and updates the ledger only after successful payment. For the two in-support rules, the value-returning configuration adds no target computation.

The first three conditions are context harnesses. The last two are control harnesses: *Checker rejects* adds a lightweight verification-and-retry gate, whereas *Harness computes* supplies derived values or rewrites action arguments. Although rejection feedback is textual, the external checker also controls retry and trajectory selection. These conditions compare mechanisms, not equal inference budgets.

The stated arm inserts the rule's `oracle_text` verbatim. A representative prompt fragment is:

Stated Harness: Oracle Preference Context

```
My preferences for how you should do things:
1.  Venmo payment descriptions must be exactly [<code>|<date>|<summary>]
where <code> is K7 if the whole-dollar part of the amount is even and
Q3 if it is odd, <date> is today's date as YYYY-MM-DD, and <summary>
is at most 6 words.
2.  Text messages you send for me must end with my first name as a
sign-off.
```

The reject-only arm instead returns a bounded failure message. For a habitual-card attempt, an illustrative instance is:

Checker-Rejection Example

```
Automated constraint checker REJECTED the previous attempt.
You paid with the Chase card, which is not the one - pick a different
card next time.
Redo the task so the constraint passes.
```

This reports the card just attempted, but never the latent habitual bank. Other rules use the same reject-only principle: the checker may identify the failed attempt or restate the constraint, but it does not return the value that must be computed.

Personas, streams, and sample sizes. The five panels combine three experiments because several rules claim incompatible fields and cannot coexist in one persona. The main persona contains private note format and SMS character checksum, with eight training tasks, seven frozen evaluation tasks, six

seeds, and a 50-step budget per attempt. The crossover persona contains SMS sign-off and running spend total, reversing the easy/hard assignment across the two applications; it uses six training tasks, six frozen evaluation tasks, three seeds, and a 60-step budget per attempt. Static arms are evaluated once per seed; history-derived results use the final checkpoint, at $n = 8$ and $n = 6$, respectively. The habitual-card experiment uses a synthetic stream of binary-feedback micro-interactions followed by six fixed AppWorld evaluation tasks. Each micro-interaction records the card used and whether the user accepted it, but not which card the user would have preferred. The latent preference is a categorical distribution whose mode is the habitual card. We evaluate after 0, 20, 60, and 120 observations and rotate the mode across different candidates. The headline figure reports $n = 20$, while the full schedule diagnoses the context method's sample efficiency. The per-attempt step budget is 30.

Scoring and controls. All arms within each experiment see the same evaluation tasks; applicable-episode counts can differ with the actions actually executed. An attempt with no applicable persona rule is not accepted by the verifier. If its retry budget is exhausted, the final attempt is retained unless it contains no applicable action, in which case the last acted attempt is preferred. For habitual card, the first card attempted is scored, since later cards may reflect an insufficient-balance fallback. The non-leaking reject-only arm is also compared with a card-cycling elimination baseline over the 4–5 available cards. For running total, the checker and computational harness start from the same read-only transaction ledger. Only successful payments contribute to the cumulative amount or make this rule applicable; failed calls do neither. Each successful payment is checked against the cumulative total including that payment, using the actual `description` field and validating the recorded outcome against the transaction identifier and database. For checksum, returning the expected value still leaves the model responsible for composing a valid final message.

A.2 Detailed Experimental Settings for Q2

Objective and scan set. Q2 studies the relationship between harness scale and preference compliance by varying the number L of injected memory lines. The adopted persona contains eight preferences spanning text messages, Venmo payments, and Spotify actions: SMS sign-off, SMS greeting, SMS terseness, payment-note presence, payment-note lowercasing, single-word payment notes, private Venmo transactions, and saving songs to a playlist rather than liking them individually. The details are listed in Table 2. Every rule is represented by text induced by the same learner from instance-level feedback; the memory is not constructed from hand-written oracle statements. The eight-rule set was chosen using induced-text fidelity: a candidate was retained only when the learner-generated statement described a behavior that could, in principle, affect the corresponding checker.

Table 2: Preferences administered in the Q2 memory-length sweep. All eight rules are included in the primary violation metric.

Preference	Trigger	Requirement
SMS sign-off	Text message	End with the user's first name.
SMS greeting	Text message	Begin with a greeting word.
SMS terseness	Text message	Use at most five words.
Payment has note	Venmo payment	Include a non-empty description.
Payment-note lowercase	Venmo payment	Write the note entirely in lowercase.
Single-word payment note	Venmo payment	Use exactly one word.
Venmo private	Venmo payment	Mark the transaction private.
Playlist over like	Spotify save	Add songs to a playlist instead of liking individually.

Learner-generated assertion pool. Before the length sweep, we collect assertions by running the learner for 32 episodes using instance-level complaints. During collection, the injected memory is empty, so every assertion is a function of the observed task trajectory and feedback rather than an earlier memory. We collect two independent pool seeds, deduplicate exact repeats, group assertions by their generating rule, and rank within each group by consensus with other learner paraphrases. Consensus ranking uses only the generated text and never the oracle rule statement. A round-robin construction then cycles across rule groups, which makes coverage increase as evenly as possible as L grows.

The collection prompt is instantiated once per violated rule. It receives the task, the write actions, and the episode-specific complaint, but not the rule name or its oracle statement:

Assertion-Pool Induction Prompt

```
You maintain a long-term memory of one user's personal preferences for
how their assistant should do things.  The user has just complained
about something the assistant did.  From that complaint, write the
standing preference you will store in memory.
Task the user asked for:  <instruction>
What the assistant did:  <API writes>
What the user said about it:  <complaint>
Output exactly one sentence as a general user preference; output only
the sentence.
```

For example, the complaint `sms not signed with first name: 'Please get on venmo.' - I don't want it done that way.` produced the learner assertion "Text messages should be signed with your first name." in one collection seed. A different seed produced "When sending text messages on behalf of the user, they should be signed with the user's first name." These paraphrases remain separate pool entries and are later selected by the length sweep.

Primary length sweep. The primary R-arm contains only assertions about the eight scored preferences. We sweep

$$L \in \{0, 1, 2, 3, 5, 10, 20, 60, 150\}. \tag{2}$$

When L exceeds the number of distinct learner assertions, the finite pool is recycled. For each (L, seed) cell, the selected lines are shuffled deterministically using the evaluation seed and L.

We run three evaluation seeds. Each cell contains 12 frozen tasks with 10 fresh rollouts per task, for 120 episodes per cell and 3,240 episodes over the 9×3 primary grid. The task pool is balanced so that each of the eight rules has four triggering tasks per cell before behavioral inapplicability.

A.3 Detailed Experimental Settings for Q3

Objective and persona construction. Q3 tests whether a self-evolving harness reaches the oracle context when the model, actuator, task distribution, and preference set are fixed. We use a persona containing five jointly satisfiable preferences over phone and Venmo actions (Table 3). The set is designed to separate preferences whose target is carried by the feedback channel from those whose target literal is withheld. This distinction is checked mechanically by executing each rule and comparing the quoted literals in its oracle statement with the checker detail available to the updater.

Table 3: Preferences used in Q3 and whether instance-level feedback carries the target needed to reconstruct the oracle statement.

Preference	Feedback role	Requirement
SMS greeting	Carried	Begin with a greeting word.
SMS sign-off	Carried	End a text message with the user's first name.
Venmo private	Carried	Mark a Venmo transaction private.
Payment-note category	Target withheld	Begin the note with the literal category tag `[personal]`.
Payment-note initials	Target withheld	End the note with parenthesized initials such as `(J.D.)`.

The category and initials complaints identify the defective field but never reveal the literals `[personal]` or `(J.D.)`. By contrast, the two SMS complaints identify the required form. Joint satisfiability is verified by executing all five checkers on a constructed episode, not only by comparing their prose descriptions.

For example, one training episode attempted the SMS "Done." and a Venmo note "Grocery Bill." Its instance-level feedback was:

Instance-Level Feedback Example

```
That's not how I like things done:
- note lacks the category tag:  'Grocery Bill' - I don't want it done
```

```
that way.
- note lacks initials suffix: 'Grocery Bill' - I don't want it done
that way.
- venmo transaction not marked private - I don't want it done that
way.
- sms lacks an opening greeting: 'Done.' - I don't want it done that
way.
- sms not signed with first name: 'Done.' - I don't want it done
that way.
```

The updater sees this text, while the correspondence between each complaint line and its checker rule is retained only as driver-side provenance for scoring and analysis. In particular, the first two complaints identify the defective note field but do not disclose the required category tag or the parenthesized initials format.

Interaction stream and evaluation split. Each evolving run contains 24 on-policy training episodes. The current harness is injected into the agent at the start of an episode; after task execution, the programmatic checkers produce instance-level feedback, the updater modifies the harness, and only the next episode observes the new state. The main checkpoints are $n \in \{0, 6, 12, 18, 24\}$. At each checkpoint, the harness is frozen and evaluated on six tasks with three independent rollouts per task, yielding 18 evaluation episodes. We use three seeds. The training and evaluation splits are disjoint at the task-template-family level. This design tests whether an induced preference transfers across task templates rather than whether the updater memorizes a specific task.

References and update recipes. The no-memory arm provides the initial behavioral ceiling. The oracle arm injects the five clearest rule statements. It is the reachability reference against which the final self-evolution gap is measured. The primary evolving method is an **ACE**-style updater with itemized bullets, model-proposed add/modify deltas, deterministic code-side merging, trigram-based semantic deduplication, and a capacity of at most 12 bullets. We compare the primary trajectory with three admissible self-evolution recipes under the same supervision: **TEPA** stores keyed precedents, replaces the active precedent under the same key, and retains revoked entries for audit; **TRACE** compiles feedback into constrained runtime predicates and permits up to three self-gated attempts; and **Reflexion** appends natural-language reflections to a FIFO buffer of width three. TRACE gates against its own learned checks, never the ground-truth persona checker, so it receives no privileged label. The Reflexion, TEPA, and TRACE updater prompts remove authentication-only calls and redact credential values from action trajectories. For mechanism diagnostics, we also run a naive full-memory rewrite updater and a corrective-feedback rewrite updater.

Baseline adaptations. We adapt the memory-update and enforcement mechanisms of four methods to the shared AppWorld-P protocol. All four use the same foundation model, actuator, persona, task-stream construction, feedback protocol, and evaluation checkpoints, but update from their own on-policy outcomes. These are mechanism-level adaptations rather than reproductions of the complete original systems. TEPA, TRACE, and Reflexion receive summaries of the current episode's API calls with authentication calls omitted and credential values redacted; ACE receives the current memory and feedback.

ACE. Our ACE-style updater (Zhang et al., 2026) maintains identified preference bullets and asks the model to propose JSON `add`/`modify` deltas, which are merged deterministically into the existing memory. After each update, character-trigram vectors identify near-duplicate entries using a cosine-similarity threshold of 0.85; if more than 12 bullets remain, the oldest are removed. All retained bullets are injected into subsequent episodes. This preserves itemized, incremental updating.

TEPA. Our TEPA adaptation (Zhou et al., 2026a) extracts keyed precedents from the current action summary and feedback, using a supplied attribute vocabulary such as `sms.ending` and permitting additional keys. Each key has at most one active precedent: a new, nonidentical entry immediately revokes the previous entry, while an exact match to an archived entry can reactivate it. Revoked entries remain archived, and all active entries are injected without task-dependent retrieval. We retain keyed replacement and explicit revocation, but omit evidence-counting, posterior-threshold lifecycle

transitions, and the trial-validation stage; the experiment therefore tests this simplified mechanism under fixed user preferences.

TRACE. Our TRACE adaptation (Zhou et al., 2026b) compiles complaints into checks over API arguments using string equality, prefix, suffix, and containment predicates, including negated prefix, suffix, and containment tests. It retains at most 12 checks and injects their natural-language instructions into context. After an episode, the learned checks inspect the recorded calls; failures produce feedback for a fresh-world retry, with at most three total attempts in both training and evaluation. Unlike the original event-level hooks, this gate operates after a complete episode and uses neither semantic verifiers nor the full rule-resolution lifecycle; it never accesses the ground-truth persona checker, and the final attempt is scored even if the learned checks still fail.

Reflexion. Our Reflexion adaptation (Shinn et al., 2023) generates a one- or two-sentence reflection from the current action summary and complaint after a rejected training episode. Reflections are appended to a FIFO buffer retaining the latest three entries, all of which are injected into subsequent tasks. Previous reflections inform task execution but are not inputs to the reflection-generation prompt, and entries are neither merged nor organized by preference. We thus use bounded verbal memory for cross-task adaptation, without a dedicated same-task reflection-and-retry loop.

Feedback and capacity controls. For each violated rule, the updater receives a complaint constructed uniformly from the checker's episode-specific detail. In particular, it may say that a payment note lacks a category tag while withholding which tag is required. Accepted episodes do not trigger an update, avoiding memory drift without new evidence. The corrective diagnostic supplies a fully quantified correction and therefore tests whether a plateau is caused by the information channel rather than the updater alone.

The four main methods share the feedback-generation protocol and evaluation checkpoints. Their state archives include memory size, update counts, parse failures, and mechanism-specific diagnostics such as ACE order fingerprints, TEPA revocations, TRACE checks fired/failed, and Reflexion evictions.

Taking the primary ACE as an example, the complaint is inserted into a delta-update prompt rather than a full-memory rewrite:

ACE Delta-Update Prompt

```
You maintain a memory of the user's preferences as itemized bullets.
Current memory:  <current bullets>
Latest feedback from the user:  <feedback>
Propose changes as a JSON array of objects with fields:
action = add, text = ..., reason = ...; or
action = modify, id = ..., text = ..., reason = ...  .
Each text is one sentence written as a user preference.  Output only
valid JSON.
```

For example, a valid delta may add "Text messages should end with my first name." The code-side updater assigns a stable identifier, deduplicates semantically similar bullets, and prunes only when the 12-bullet capacity is exceeded.

B PROOFS

B.1 PROOF OF THEOREM 4.1

Assumption B.1. *For any context harness c, (i) the trace distribution is conditionally invariant to context given a concept $p(\tau \mid z, c, \theta) = p(\tau \mid z, \theta)$, and (ii) the concept distribution is fixed by the context and independent of the test task $p(\theta \mid z, c) = p(\theta \mid c)$.*

Remark B.2. *Assumption B.1 formalizes the view that a context harness describes user-specific preferences through latent concepts already represented by the frozen model. Once a concept θ is specified, the context provides no additional effect on the resulting trace distribution. Meanwhile, the context determines the mixture over these latent concepts, while the test task z only governs how a selected concept is expressed in the current interaction.*

Proof. We introduce a latent-concept representation of the frozen model. Let θ denote a behavioral concept represented by f, with pretrained prior $p(\theta)$. The frozen policy is represented as

$$\pi_f(\tau \mid z) = \int p(\tau \mid z, \theta) p(\theta)\, d\theta. \tag{3}$$

Let $\mathcal{C}_{\text{ctx}}$ and $\mathcal{C}_{\text{ctrl}}$ denote the context and control harness classes, respectively. For any $c \in \mathcal{C}_{\text{ctx}}$, marginalizing over θ and applying Assumption B.1 gives

$$\pi_{c \circ f}(\tau \mid z) = \int p(\tau \mid z, c, \theta) p(\theta \mid z, c)\, d\theta = \int p(\tau \mid z, \theta) p(\theta \mid c)\, d\theta. \tag{4}$$

Define the pretrained policy hull as $\mathcal{M}_f := \overline{\text{conv}}\{p(\cdot \mid \cdot, \theta) : \theta \in \operatorname{supp} p(\theta)\}$. Since $p(\theta \mid c)$ is a probability distribution, $\pi_{c \circ f}$ is a convex mixture of the pretrained concept-conditioned trace distributions and therefore $\pi_{c \circ f} \in \mathcal{M}_f$ for every $c \in \mathcal{C}_{\text{ctx}}$. Hence, defining $\mathcal{R}_{\text{ctx}} := \{\pi_{c \circ f} : c \in \mathcal{C}_{\text{ctx}}\}$, we obtain $\mathcal{R}_{\text{ctx}} \subseteq \mathcal{M}_f$.

For a control harness $c \in \mathcal{C}_{\text{ctrl}}$, let $K_c(\bar{\tau} \mid \tau, z)$ denote the intervention kernel that maps a model-proposed trace τ to an executed trace $\bar{\tau}$. For any policy π, define the induced control operator as

$$(\mathcal{T}_c \pi)(\bar{\tau} \mid z) := \int K_c(\bar{\tau} \mid \tau, z) \pi(\tau \mid z)\, d\tau. \tag{5}$$

Since $K_c(\cdot \mid \tau, z)$ is a probability distribution for every (τ, z), $\mathcal{T}_c \pi$ is a valid policy distribution. For each pretrained concept θ, define the control-transformed concept-conditioned trace distribution as

$$q(\bar{\tau} \mid z, c, \theta) := \int K_c(\bar{\tau} \mid \tau, z) p(\tau \mid z, \theta)\, d\tau. \tag{6}$$

Applying the control operator directly to the frozen policy and substituting Equation 3 gives

$$(\mathcal{T}_c \pi_f)(\bar{\tau} \mid z) = \int K_c(\bar{\tau} \mid \tau, z) \left[\int p(\tau \mid z, \theta) p(\theta)\, d\theta \right] d\tau = \int q(\bar{\tau} \mid z, c, \theta) p(\theta)\, d\theta. \tag{7}$$

Thus, a context harness changes the mixture weights over fixed pretrained concept-conditioned trace distributions, while a control harness can transform the component distributions themselves from $p(\tau \mid z, \theta)$ to $q(\bar{\tau} \mid z, c, \theta)$ through action-path intervention.

Define the control-reachable policy set as $\mathcal{R}_{\text{ctrl}} := \{\mathcal{T}_c \pi_f : c \in \mathcal{C}_{\text{ctrl}}\}$. Unlike $\mathcal{R}_{\text{ctx}}$, the set $\mathcal{R}_{\text{ctrl}}$ is not restricted to convex reweightings of the pretrained concept-conditioned trace distributions. In particular, if there exists $c \in \mathcal{C}_{\text{ctrl}}$ such that $\mathcal{T}_c \pi_f \notin \mathcal{M}_f$, then $\mathcal{R}_{\text{ctrl}} \not\subseteq \mathcal{M}_f$, and $\mathcal{R}_{\text{ctx}} \subseteq \mathcal{M}_f$.

Finally, consider a user-optimal policy $\pi_{f_u^*} \in \mathcal{R}_{\text{ctrl}} \setminus \mathcal{R}_{\text{ctx}}$ whose risk is separated from the context-reachable class by

$$\Delta_u := \inf_{\pi \in \mathcal{R}_{\text{ctx}}} R(\pi) - R(\pi_{f_u^*}) > 0.$$

Then $\epsilon_{\text{app}}(\mathcal{C}_{\text{ctx}}, f_u^*) = \Delta_u > 0$. Since $\pi_{f_u^*} \in \mathcal{R}_{\text{ctrl}}$ and $\pi_{f_u^*}$ is user-optimal,

$$\epsilon_{\text{app}}(\mathcal{C}_{\text{ctrl}}, f_u^*) = \inf_{\pi \in \mathcal{R}_{\text{ctrl}}} R(\pi) - R(\pi_{f_u^*}) = 0.$$

Hence, for user-optimal policies that are unreachable by context-only reweighting but reachable through control intervention, the control class achieves a strictly smaller approximation error. □

B.2 Proof of Theorem 4.2

Let $U = (U_1, \ldots, U_d)$ denote the user's d independent binary preferences, with $U_j \sim \text{Bernoulli}(1/2)$. For a chosen harness-capacity budget L, let $\mathcal{C}_L$ denote the class of context harnesses containing at most L memory statements. Let $c^*_{\mathcal{C}_L} \in \arg\min_{c \in \mathcal{C}_L} R(\pi_{c \circ f})$ denote the population-optimal harness in $\mathcal{C}_L$, and let $\hat{c}_{n,L} \in \mathcal{C}_L$ denote the harness learned from the interaction stream S_n. For the theoretical analysis, each fresh task z activates a subset $J(z) \subseteq [d]$ of the preferences, and evaluation is over applicable task–preference pairs (z, j) with $j \in J(z)$, with each preference appearing with probability $1/d$. The loss ℓ_u is the indicator that preference j is violated.

Proof. We bound the generalization error induced by the finite interaction stream. Each memory statement is associated with a preference that it is intended to represent. Let $m = \lfloor n/L \rfloor$ denote the number of feedback observations allocated to each memory statement. For each statement $\ell \in [L]$, let $X_{\ell,1}, \ldots, X_{\ell,m} \in \{0, 1\}$ be independent binary feedback observations associated with the preference represented by statement ℓ, where $X_{\ell,i} = 1$ if the i-th observation agrees with the corresponding preference, and $\Pr(X_{\ell,i} = 1) = 1/2 + \gamma$ for some $0 < \gamma \leq 1/2$. Here, γ measures the advantage of the feedback over random guessing.

For each statement $\ell \in [L]$, let E_ℓ denote the event that the correct preference does not receive a strict majority among its associated feedback observations:

$$E_\ell := \left\{ \frac{1}{m} \sum_{i=1}^{m} X_{\ell,i} \leq \frac{1}{2} \right\}.$$

Since $\mathbb{E}[X_{\ell,i}] = 1/2 + \gamma$, we have

$$E_\ell = \left\{ \frac{1}{m} \sum_{i=1}^{m} X_{\ell,i} - \mathbb{E}[X_{\ell,i}] \leq -\gamma \right\}.$$

Thus, by Hoeffding's inequality,

$$\Pr(E_\ell) \leq \exp\bigl(-2m\gamma^2\bigr) = \exp\Bigl(-2\gamma^2 \left\lfloor \frac{n}{L} \right\rfloor\Bigr).$$

Let $E := \bigcup_{\ell=1}^{L} E_\ell$. By the union bound,

$$\Pr(E) \leq \sum_{\ell=1}^{L} \Pr(E_\ell) \leq L \exp\Bigl(-2\gamma^2 \left\lfloor \frac{n}{L} \right\rfloor\Bigr).$$

On the complement event E^c, all target statements are inferred correctly, so $\hat{c}_{n,L}$ and $c^*_{\mathcal{C}_L}$ induce identical preference behavior and hence incur the same loss. Therefore, any increase in risk of $\hat{c}_{n,L}$ relative to $c^*_{\mathcal{C}_L}$ can occur only on E. Since $\ell_u \in \{0, 1\}$, taking expectations over S_n gives

$$\begin{aligned} \mathbb{E}_{S_n}[\epsilon_{\text{gen}}(\mathcal{C}_L, n)] &= \mathbb{E}_{S_n}\bigl[R(\pi_{\hat{c}_{n,L} \circ f})\bigr] - R(\pi_{c^*_{\mathcal{C}_L} \circ f}) \\ &\leq \Pr(E) \leq L \exp\Bigl(-2\gamma^2 \left\lfloor \frac{n}{L} \right\rfloor\Bigr). \end{aligned} \tag{8}$$

For fixed n and γ, $\lfloor n/L \rfloor$ is nonincreasing in integer L, so the exponential factor is nondecreasing. Since the prefactor L is strictly increasing, the generalization bound is increasing in integer L. □

B.3 Proof of Corollary 4.3

Proof. Consider the information-theoretic framework, where $H(\cdot)$ denotes the information entropy and $I(\cdot;\cdot)$ denotes the mutual information. Since the d preference bits are independent and unbiased, $H(U) = d$. Since every harness in $\mathcal{C}_L$ contains at most L memory statements of at most b bits each, the population-optimal harness $c^*_{\mathcal{C}_L}$ satisfies

$$I(U; c^*_{\mathcal{C}_L}) \leq H(c^*_{\mathcal{C}_L}) \leq Lb.$$

Therefore,

$$H(U \mid c^*_{\mathcal{C}_L}) = H(U) - I(U; c^*_{\mathcal{C}_L}) \geq d - Lb.$$

For each $j \in [d]$, let $p_j^* := \inf_{\phi_j} \Pr\big[\phi_j(c^*_{\mathcal{C}_L}) \neq U_j\big]$ be the minimum probability of incorrectly recovering U_j from the population-optimal harness, and define $p_L^* := 1/d \cdot \sum_{j=1}^d p_j^*$. By binary Fano's inequality,

$$H(U_j \mid c^*_{\mathcal{C}_L}) \leq h_2(p_j^*),$$

where $h_2(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function. Using subadditivity of conditional entropy and concavity of h_2,

$$H(U \mid c^*_{\mathcal{C}_L}) \leq \sum_{j=1}^d H(U_j \mid c^*_{\mathcal{C}_L}) \leq \sum_{j=1}^d h_2(p_j^*) \leq d\, h_2(p_L^*).$$

Combining the lower and upper bounds on $H(U \mid c^*_{\mathcal{C}_L})$ gives $h_2(p_L^*) \geq 1 - Lb/d$. Since $h_2(p_L^*) \geq 0$, this yields

$$h_2(p_L^*) \geq \left[1 - \frac{Lb}{d}\right]_+.$$

Since an incorrectly represented applicable preference induces a violation, while the user-optimal policy incurs zero violation risk on these binary preferences,

$$\mathbb{E}_U\left[\epsilon_{\text{app}}(\mathcal{C}_L, f_u^*)\right] = \mathbb{E}_U\left[R(\pi_{c^*_{\mathcal{C}_L}\circ f}) - R(\pi_{f_u^*})\right] \geq p_L^*.$$

Since $p_L^* \in [0, 1/2]$ and h_2 is increasing on $[0, 1/2]$, we obtain

$$\mathbb{E}_U[\epsilon_{\text{app}}(\mathcal{C}_L, f_u^*)] \geq h_2^{-1}\left(\left[1 - \frac{Lb}{d}\right]_+\right). \tag{9}$$

We next upper bound the approximation error by constructing a harness in $\mathcal{C}_L$. Let $c_L^{\text{exp}} \in \mathcal{C}_L$ store the correct values of $\min\{L, d\}$ distinct preferences, one per memory statement. Since each of the d preferences is equally likely to appear in the evaluation, the probability that the evaluated preference is not represented in c_L^{exp} is

$$\frac{d - \min\{L, d\}}{d} = \frac{(d-L)_+}{d}.$$

Since covered preferences incur no violation and the user-optimal policy has zero violation risk,

$$R(\pi_{c_L^{\text{exp}}\circ f}) - R(\pi_{f_u^*}) \leq \frac{(d-L)_+}{d}.$$

By the population optimality of $c^*_{\mathcal{C}_L}$, $R(\pi_{c^*_{\mathcal{C}_L}\circ f}) \leq R(\pi_{c_L^{\text{exp}}\circ f})$. Therefore,

$$\epsilon_{\text{app}}(\mathcal{C}_L, f_u^*) = R(\pi_{c^*_{\mathcal{C}_L}\circ f}) - R(\pi_{f_u^*}) \leq \frac{(d-L)_+}{d}. \tag{10}$$

Together, eq. (9) and eq. (10) give

$$h_2^{-1}\left(\left[1 - \frac{Lb}{d}\right]_+\right) \leq \mathbb{E}_U[\epsilon_{\text{app}}(\mathcal{C}_L, f_u^*)] \leq \frac{(d-L)_+}{d}. \tag{11}$$

Combining the approximation upper bound with Theorem 4.2 yields

$$\begin{aligned}\mathbb{E}_{S_n,U}\left[R(\pi_{\hat{c}_{n,L}\circ f}) - R(\pi_{f_u^*})\right] &= \mathbb{E}_U\left[\epsilon_{\text{app}}(\mathcal{C}_L, f_u^*)\right] + \mathbb{E}_{S_n}[\epsilon_{\text{gen}}(\mathcal{C}_L, n)] \\ &\leq \frac{(d-L)_+}{d} + L\exp\left(-2\gamma^2\left\lfloor\frac{n}{L}\right\rfloor\right).\end{aligned}$$

□

B.4 Proof of Theorem 4.4

Let f be the frozen foundation model, and fix an update rule U and a feedback protocol throughout the analysis. For a chosen harness-capacity budget L, let $\mathcal{C}_L$ denote the class of context harnesses containing at most L memory statements. Let $\tilde{c}_{n,L} \in \mathcal{C}_L$ denote the harness produced by the online update procedure after n interactions, and let $\hat{c}_{n,L} \in \arg\min_{c \in \mathcal{C}_L} \widehat{R}_n(\pi_{c \circ f}; S_n)$ be a measurably selected empirical-risk minimizer based on the interaction stream S_n.

For the theoretical analysis, let $\mathsf{d}_L(c, c')$ measure the distance between two harnesses, and assume that $(\mathcal{C}_L, \mathsf{d}_L)$ is a complete separable metric space. Let $\mathcal{P}_1(\mathcal{C}_L)$ denote the probability distributions over harnesses with finite expected distance from a fixed reference harness $c_0 \in \mathcal{C}_L$. To compare two harness distributions $\nu, \lambda \in \mathcal{P}_1(\mathcal{C}_L)$, define their first Wasserstein distance by

$$W_L(\nu, \lambda) := \inf_{\Gamma \in \Pi(\nu, \lambda)} \int_{\mathcal{C}_L \times \mathcal{C}_L} \mathsf{d}_L(c, c')\, \Gamma(dc, dc'),$$

where $\Pi(\nu, \lambda)$ is the set of joint distributions with marginals ν and λ. Thus, d_L compares individual harnesses, while W_L compares their distributions through the smallest achievable expected harness distance.

Assumption B.3. *The online harness sequence under U is a Markov process with a time-independent transition kernel P_U. All harness distributions belong to $\mathcal{P}_1(\mathcal{C}_L)$, and the induced operator $\mathcal{T}_U : \mathcal{P}_1(\mathcal{C}_L) \to \mathcal{P}_1(\mathcal{C}_L)$ satisfies $W_L(\mathcal{T}_U \nu, \mathcal{T}_U \lambda) \le \rho W_L(\nu, \lambda)$ for some $0 \le \rho < 1$ and all $\nu, \lambda \in \mathcal{P}_1(\mathcal{C}_L)$.*

Assumption B.4. *The population risk $c \mapsto R(\pi_{c \circ f})$ is bounded and continuous under d_L. There exist $\mu > 0$ and a deterministic sequence $\epsilon_{n,L} \ge 0$ such that, for every $n \ge 1$, $\mathbb{E}\mathsf{d}_L(\tilde{c}_{n,L}, \hat{c}_{n,L})^2 < \infty$,*

$$\widehat{R}_n(\pi_{\tilde{c}_{n,L} \circ f}; S_n) - \widehat{R}_n(\pi_{\hat{c}_{n,L} \circ f}; S_n) \ge \frac{\mu}{2} \mathsf{d}_L(\tilde{c}_{n,L}, \hat{c}_{n,L})^2 \quad \textit{almost surely},$$

and

$$\mathbb{E}\Big[\Big|R(\pi_{\tilde{c}_{n,L} \circ f}) - \widehat{R}_n(\pi_{\tilde{c}_{n,L} \circ f}; S_n)\Big| + \Big|R(\pi_{\hat{c}_{n,L} \circ f}) - \widehat{R}_n(\pi_{\hat{c}_{n,L} \circ f}; S_n)\Big|\Big] \le 2\epsilon_{n,L}.$$

For each fixed L, the calibration error satisfies $\epsilon_{n,L} \to 0$ as $n \to \infty$.

Assumption B.5 (Persistent update bias)**.** *For $\beta_{f,n,L} := W_L(\mathcal{T}_f \widehat{\nu}_{n,L}, \widehat{\nu}_{n,L})$, there exists a constant $\beta_0 > 0$ such that $\liminf_{n \to \infty} \beta_{f,n,L} \ge \beta_0$.*

Proof. For a current harness $c \in \mathcal{C}_L$, let $P_U(c, \cdot)$ denote the conditional distribution of the next harness over $\mathcal{C}_L$ after a fresh on-policy interaction and update. Let $\nu_{n,L}$ and $\widehat{\nu}_{n,L}$ denote the distributions of the online output $\tilde{c}_{n,L}$ and the empirical-risk minimizer $\hat{c}_{n,L}$, respectively, over the random interaction stream and algorithmic randomness. Define the distribution update operator by

$$\mathcal{T}_U \nu := \int_{\mathcal{C}_L} P_U(c, \cdot)\, \nu(dc). \tag{12}$$

By the law of total probability, the online evolution satisfies $\nu_{n+1,L} = \mathcal{T}_U \nu_{n,L}$. Let

$$r_{U,n,L} := W_L(\nu_{n+1,L}, \nu_{n,L}), \quad \beta_{U,n,L} := W_L(\mathcal{T}_U \widehat{\nu}_{n,L}, \widehat{\nu}_{n,L}). \tag{13}$$

Here, $r_{U,n,L}$ measures the change in the online output distribution, while $\beta_{U,n,L}$ measures the change induced by applying the update operator to the distribution of the empirical-risk minimizer. The optimization gap is $\epsilon_{\mathrm{opt},n} := R(\pi_{\tilde{c}_{n,L} \circ f}) - R(\pi_{\hat{c}_{n,L} \circ f})$.

Since $(\mathcal{C}_L, \mathsf{d}_L)$ is complete and separable, the associated Wasserstein space $(\mathcal{P}_1(\mathcal{C}_L), W_L)$ is complete. By Assumption B.3, $\mathcal{T}_U$ is a contraction mapping. Banach's fixed-point theorem therefore gives a unique invariant distribution ν_∞ satisfying

$$\mathcal{T}_U \nu_\infty = \nu_\infty, \quad W_L(\nu_{n,L}, \nu_\infty) \le \rho^n W_L(\nu_{0,L}, \nu_\infty).$$

Thus $\nu_{n,L} \to \nu_\infty$ in W_L. By the boundedness and continuity of $c \mapsto R(\pi_{c \circ f})$ in Assumption B.4,

$$\mathbb{E}[R(\pi_{\tilde{c}_{n,L} \circ f})] = \int_{\mathcal{C}_L} R(\pi_{c \circ f})\, \nu_{n,L}(dc) \to \int_{\mathcal{C}_L} R(\pi_{c \circ f})\, \nu_\infty(dc). \tag{14}$$

Moreover, since $\nu_{n+1,L} = \mathcal{T}_U \nu_{n,L}$, the contraction condition gives

$$\begin{aligned} r_{U,n+1,L} &= W_L(\mathcal{T}_U \nu_{n+1,L}, \mathcal{T}_U \nu_{n,L}) \\ &\le \rho W_L(\nu_{n+1,L}, \nu_{n,L}) = \rho r_{U,n,L}. \end{aligned}$$

Iterating this inequality yields $r_{U,n,L} \le \rho^n r_{U,0,L} \to 0$.

Fix $n \ge 1$ and write $\nu = \nu_{n,L}$ and $\widehat{\nu} = \widehat{\nu}_{n,L}$. By the triangle inequality and Assumption B.3,

$$\begin{aligned} \beta_{U,n,L} &= W_L(\mathcal{T}_U \widehat{\nu}, \widehat{\nu}) \\ &\le W_L(\mathcal{T}_U \widehat{\nu}, \mathcal{T}_U \nu) + W_L(\mathcal{T}_U \nu, \nu) + W_L(\nu, \widehat{\nu}) \\ &= W_L(\mathcal{T}_U \widehat{\nu}, \mathcal{T}_U \nu) + r_{U,n,L} + W_L(\nu, \widehat{\nu}) \\ &\le (1+\rho) W_L(\nu, \widehat{\nu}) + r_{U,n,L}. \end{aligned}$$

Since $W_L(\nu, \widehat{\nu}) \ge 0$, this yields

$$W_L(\nu, \widehat{\nu}) \ge \frac{[\beta_{U,n,L} - r_{U,n,L}]_+}{1+\rho}.$$

The joint distribution of $(\tilde{c}_{n,L}, \hat{c}_{n,L})$ has marginals ν and $\widehat{\nu}$, so it belongs to $\Pi(\nu, \widehat{\nu})$. By Jensen's inequality and the definition of W_L,

$$\mathbb{E}\mathsf{d}_L(\tilde{c}_{n,L}, \hat{c}_{n,L})^2 \ge \left(\mathbb{E}\mathsf{d}_L(\tilde{c}_{n,L}, \hat{c}_{n,L})\right)^2 \ge W_L(\nu, \widehat{\nu})^2.$$

Adding and subtracting the empirical risks and using the risk calibration condition in Assumption B.4, we obtain

$$\begin{aligned} \mathbb{E}[\epsilon_{\mathrm{opt},n}] =& \mathbb{E}\Big[\widehat{R}_n(\pi_{\tilde{c}_{n,L}\circ f}; S_n) - \widehat{R}_n(\pi_{\hat{c}_{n,L}\circ f}; S_n)\Big] + \mathbb{E}\Big[R(\pi_{\tilde{c}_{n,L}\circ f}) - \widehat{R}_n(\pi_{\tilde{c}_{n,L}\circ f}; S_n)\Big] \\ &- \mathbb{E}\Big[R(\pi_{\hat{c}_{n,L}\circ f}) - \widehat{R}_n(\pi_{\hat{c}_{n,L}\circ f}; S_n)\Big] \\ \ge& \mathbb{E}\Big[\widehat{R}_n(\pi_{\tilde{c}_{n,L}\circ f}; S_n) - \widehat{R}_n(\pi_{\hat{c}_{n,L}\circ f}; S_n)\Big] - 2\epsilon_{n,L}. \end{aligned}$$

Together with the empirical risk growth condition in Assumption B.4,

$$\begin{aligned} \mathbb{E}[\epsilon_{\mathrm{opt},n}] &\ge \mathbb{E}\Big[\widehat{R}_n(\pi_{\tilde{c}_{n,L}\circ f}; S_n) - \widehat{R}_n(\pi_{\hat{c}_{n,L}\circ f}; S_n)\Big] - 2\epsilon_{n,L} \\ &\ge \frac{\mu}{2}\mathbb{E}\mathsf{d}_L(\tilde{c}_{n,L}, \hat{c}_{n,L})^2 - 2\epsilon_{n,L} \\ &\ge \frac{\mu}{2} W_L(\nu, \widehat{\nu})^2 - 2\epsilon_{n,L}. \end{aligned}$$

Combining these inequalities with $r_{U,n,L} \le \rho^n r_{U,0,L}$ gives

$$\begin{aligned} \mathbb{E}[\epsilon_{\mathrm{opt},n}] &\ge \frac{\mu}{2(1+\rho)^2}[\beta_{U,n,L} - r_{U,n,L}]_+^2 - 2\epsilon_{n,L} \\ &\ge \frac{\mu}{2(1+\rho)^2}[\beta_{U,n,L} - \rho^n r_{U,0,L}]_+^2 - 2\epsilon_{n,L}. \end{aligned}$$

Finally, since $\rho^n r_{U,0,L} \to 0$ and $\epsilon_{n,L} \to 0$ by Assumption B.4, taking the lower limit and applying Assumption B.5 yields

$$\liminf_{n\to\infty} \mathbb{E}[\epsilon_{\mathrm{opt},n}] \ge \frac{\mu \beta_0^2}{2(1+\rho)^2} > 0.$$

□