HARNESS CONTINUAL LEARNING: CONTINUAL ADAPTATION BEYOND MODEL PARAMETERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning has largely been model-centric, treating model parameters as the state that changes with sequential experience. Modern agents can also adapt through a harness of prompts, memories, tools, skills, and routing rules. Because these contents jointly shape later execution, a harness update can disrupt previously reliable behavior even when the model is frozen. This raises a new question: how can an agent continually improve its state outside the model while retaining behavior acquired earlier? We formulate *Harness Continual Learning* (HCL), a new continual learning paradigm in which the harness evolves around a frozen foundation model, and define the resulting loss of earlier behavior as *harness-level forgetting*. We instantiate HCL with four execution-facing components: the Task Interface, Experience Memory, Capability Map, and Adaptive Router. We further introduce *guarded harness evolution* to separate update generation from state commitment. A Continual Optimizer proposes candidate harnesses from post-execution feedback, and a Continual Evaluator commits the resulting candidate harness only after checking current improvement, historical retention, and validity. Experiments on open-world capability accumulation, textual reasoning, and multimodal perception demonstrate capability accumulation and failure recovery, with relative gains exceeding 10% over corresponding baselines in multiple settings. Additional experiments further examine knowledge transfer through harness evolution. Component ablations assess the contribution of each harness component, while controlled retention sweeps reveal measurable harness-level forgetting and demonstrate a controllable stability–plasticity trade-off.

1 INTRODUCTION

Continual learning studies how a system acquires capabilities from sequential experience while retaining previously learned behavior (Delange et al., 2022; Shi et al., 2025; Lu et al., 2026). Existing formulations realize this process mainly by changing model parameters, representations, or architectural components; we refer to this established view as *model-centric continual learning*.

The rise of agentic AI introduces another source of adaptation: an external *harness* that determines how a foundation model receives information, retrieves experience, and acts (Chen et al., 2025; Jimenez et al., 2024; Li et al., 2026; Meng et al., 2026; Xie et al., 2024; Xu et al., 2025). Prompts, memories, tool and skill specifications, and routing policies can persist and evolve across interactions even when the foundation model remains frozen. Agent adaptation is therefore no longer confined to model state: harness state can accumulate experience and reshape future behavior. This makes the harness a new object of continual learning research, as illustrated in Figure 1.

We formalize this setting as *Harness Continual Learning* (HCL), a continual learning paradigm that acquires and retains capabilities by sequentially updating harness state around a frozen foundation model. Conventional harness optimization typically searches for prompts, functions, or workflows that improve a current objective (Zhang et al., 2024; 2025); HCL instead studies a sequence of deployed harnesses. Because harness components are coupled in execution, a memory update can change evidence retrieved for an earlier query, a skill revision can alter tool use, and a routing edit can break a previously successful workflow. An update that helps recent cases can therefore turn an earlier correct answer, valid tool call, or successful action trajectory into a failure without changing

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

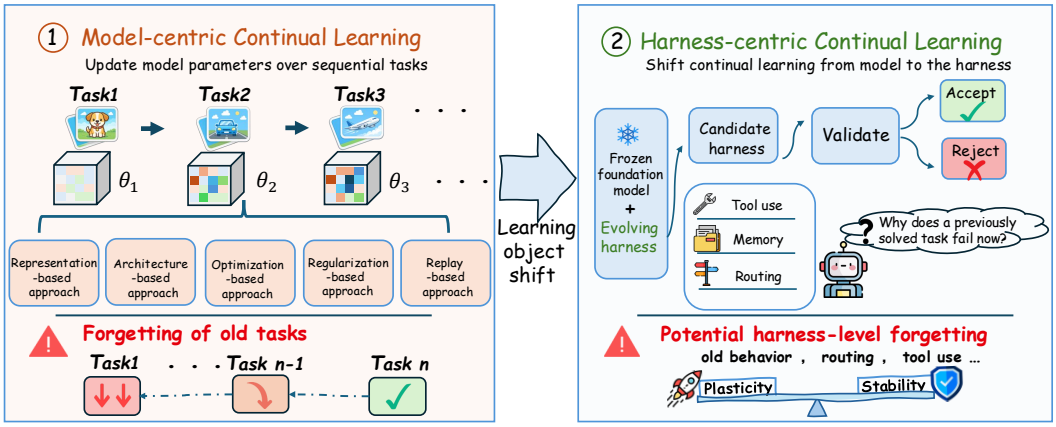


Figure 1: The shift in the object of continual learning. Model-centric methods update model parameters θ over sequential experience. HCL instead updates harness state around a frozen foundation model. In both settings, adaptation can improve performance on new capabilities, but may also degrade capabilities acquired earlier.

the foundation model. We call this phenomenon *harness-level forgetting*, extending the classical stability–plasticity problem from model state to harness state.

To study continual adaptation under this retention requirement, we develop an HCL framework with two parts. First, we define the Task Interface, Experience Memory, Capability Map, and Adaptive Router as the harness state and learning object of HCL. These components are jointly versioned and determine how the agent processes information, reuses experience and capabilities, and organizes execution. Second, guarded harness evolution governs state transitions through two modules: a Continual Optimizer that proposes candidate harnesses from post-execution feedback, and a Continual Evaluator that determines whether those candidates can be committed. Only a candidate harness that improves current validation performance while satisfying the historical-retention budget and validity constraints can be committed as the deployed state. This proposal–evaluation–commitment process makes retention an explicit condition of harness adaptation, mitigating harness-level forgetting while controlling the stability–plasticity trade-off.

We evaluate HCL across textual reasoning, multimodal perception, and open-world interaction, and further study retention control, harness-level distillation, and component contributions. In this work, our contributions are as follows:

- We propose and formalize *Harness Continual Learning*, as a new continual learning paradigm, shifting the continual learning object from model state to harness state around a frozen foundation model.
- We identify *harness-level forgetting* and introduce *guarded harness evolution*, which makes current improvement, historical retention, and validity explicit conditions for committing a harness update.
- We show across reasoning, multimodal, and interactive streams that harness evolution supports capability accumulation. Ablations and retention sweeps further reveal component contributions, measurable forgetting, and controllable stability–plasticity trade-offs, with exploratory evidence for knowledge transfer.

2 Related Work

2.1 Harness Engineering

Contemporary agent systems place a runtime harness around a foundation model to turn inference into task-directed execution (He et al., 2026; Li et al., 2026; Meng et al., 2026; Zhou et al., 2026). Harness state commonly includes input interfaces, memory, tool and skill registries, and routing

or workflow controllers (Chen et al., 2026b; Gu, 2026). Prior systems instantiate these functions through reasoning–action loops, tool coordination, persistent memory, reflection, and executable skills (Karpas et al., 2022; Packer et al., 2023; Schick et al., 2023; Shen et al., 2023; Shinn et al., 2023; Wang et al., 2024a; Yao et al., 2023). Harness engineering increasingly revises such contents from execution feedback, including prompts, memories, skills, workflows, and cross-component repairs (Abuzakuk et al., 2026; Chen et al., 2026a; Liu et al., 2026b; Yao et al., 2026; Zhang et al., 2026a;d; Zhong et al., 2026; Zhou et al., 2023). However, the main objective of these methods is usually the quality of a component or the next configuration on a current task or target distribution. Repeated improvement alone does not provide a general retention criterion for the full harness state (Lin et al., 2026). HCL instead treats the mutable harness as a unified continual learning state and evaluates retention across committed updates.

2.2 Model-Centric Continual Learning

Model-centric continual learning adapts a model to a non-stationary stream of tasks or data while seeking to retain capabilities acquired from earlier experience. Its central challenge is catastrophic forgetting, which arises when learning new knowledge disrupts knowledge acquired by the model (Delange et al., 2022; Kirkpatrick et al., 2017; Wang et al., 2024b). Model-centric continual learning addresses catastrophic forgetting through replay, regularization, architectural isolation, representation learning, and constrained optimization(Abbes et al., 2026; Bellitto et al., 2024; Kang et al., 2026; Lewandowski et al., 2025; Liu et al., 2026a; Lopez-Paz & Ranzato, 2017; Lu et al., 2024; Shang et al., 2025; Urettini & Carta, 2025; Wang et al., 2022a;b; 2025b; Yue et al., 2025). Recent work extends these families to large language models and broader knowledge streams, but the evolving state remains model knowledge, representations, architectures, or parameters (Zhao et al., 2026; Zhang et al., 2026b). HCL transfers the same acquisition of new capability and retention of old capability problem to the state outside a frozen model.

3 Harness Continual Learning

3.1 Definition and Problem Setting

Consider a fixed foundation model F_θ and a harness H_n deployed at interaction step n. The model parameters θ remain unchanged. We define *Harness Continual Learning* as as the problem of sequentially updating the deployed harness to acquire new behavior while retaining behavior that was reliable before the update. Such behavior may be a correct response, a valid tool call, or an action trajectory satisfying an environment goal. Retention requires such behavior to remain successful after later harness updates when evaluated under the same input and execution conditions.

At interaction step n, $\mathbf{u}_n$ denotes the raw interaction, such as an instruction, an observation, or a multimodal input. The harness transforms $\mathbf{u}_n$ into the structured interaction $\mathbf{i}_n$. Guided by the frozen foundation model, it then combines $\mathbf{i}_n$ with selected memory and capabilities to assemble the execution context $\mathbf{z}_n$. The model and external runtime execute $\mathbf{z}_n$ to produce the outcome $\mathbf{y}_n$. Post-execution feedback is denoted by $\mathbf{f}_n$. We collect these interaction-level objects as

$$\mathbf{e}_n = (\mathbf{u}_n, \mathbf{i}_n, \mathbf{z}_n, \mathbf{y}_n, \mathbf{f}_n). \tag{1}$$

The Optimizer provides the foundation model F_θ with an update rule, the deployed harness, and the available interaction evidence as context for generating a candidate harness,

$$\widetilde{H}_{n+1} = \mathcal{O}_{F_\theta}(H_n, \mathbf{e}_n), \tag{2}$$

The candidate remains separate from the deployed harness until a commitment decision is made. Let $G_n \in \{0, 1\}$ denote this decision. The deployed harness evolves as:

$$H_{n+1} = \begin{cases} \widetilde{H}_{n+1}, & G_n = 1, \\ H_n, & G_n = 0. \end{cases} \tag{3}$$

3.2 Harness State for Continual Learning

HCL organizes recurring execution functions into four jointly versioned components,

$$H_n = (I_n, M_n, C_n, R_n), \tag{4}$$

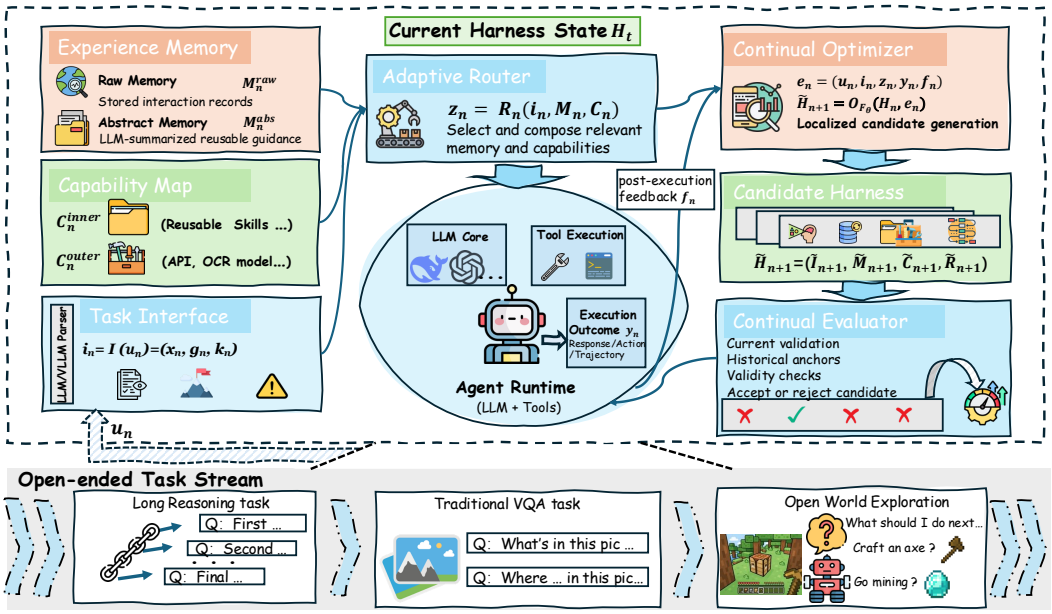


Figure 2: Overview of HCL. The harness H_n supports execution from interaction $\mathbf{u}_n$ to outcome $\mathbf{y}_n$. After execution, the Continual Optimizer proposes candidate harnesses, which the Continual Evaluator accepts or rejects based on current improvement, historical retention, and validity.

where I_n, M_n, C_n, and R_n denote the Task Interface, Experience Memory, Capability Map, and Adaptive Router. The four components are not meant as a new decomposition of agent systems; they collect recurring execution functions into a state that can be versioned, evaluated, and updated under explicit acquisition and retention constraints. Figure 2 shows how these components support execution and how the Continual Optimizer proposes candidate harnesses after execution, which are then evaluated by the Continual Evaluator before commitment.

Task Interface. The Task Interface transforms a raw interaction $\mathbf{u}_n$ into a structured representation of the task input, objective, and execution constraints:

$$\mathbf{i}_n = I_n(\mathbf{u}_n) = (\mathbf{x}_n, \mathbf{g}_n, \mathbf{k}_n), \tag{5}$$

where $\mathbf{x}_n$ denotes the available input, $\mathbf{g}_n$ the task objective, and $\mathbf{k}_n$ constraints such as output format, legal tool use, and environment restrictions. Internally, I_n includes the prompts, task templates, and parsing or normalization rules used to construct this representation. Since changes to these contents may alter how future tasks are interpreted, I_n is versioned as part of the evolving harness.

Experience Memory. Agent memory can take many forms, including episodic records, summaries, and reflections (Packer et al., 2023; Park et al., 2023; Shinn et al., 2023; Wang et al., 2025c; Zhong et al., 2024). HCL organizes accumulated experience as $M_n = (M_n^{\text{raw}}, M_n^{\text{abs}})$, where M_n^{raw} and M_n^{abs} denote Raw Memory and Abstract Memory, respectively.

Raw Memory M_n^{raw} stores the task input $\mathbf{u}_n$, the resulting response or action trajectory $\mathbf{y}_n$, and the corresponding environment or verifier feedback $\mathbf{f}_n$. To keep storage bounded, it retains a fixed number of interactions from each task in arrival order, preserving evidence of successful behavior and encountered failures. Abstract Memory M_n^{abs} summarizes Raw Memory into reusable, such as output conventions, reliable reasoning patterns, and common errors to avoid. As new interactions accumulate, abstract entries can be added or updated to support transfer across related tasks.

Together, Raw Memory preserves concrete experience for reuse and recovery, while Abstract Memory provides reusable knowledge for cross-task adaptation.

Capability Map. The Capability Map defines the operations and skills available to the agent during execution. We denote it as $C_n = (C_n^{\text{outer}}, C_n^{\text{inner}})$, where C_n^{outer} contains externally provided capabilities and C_n^{inner} contains skills acquired through continual interaction.

Outer capabilities connect the frozen model to external resources, such as APIs, retrieval services, calculators, and environment actions. Each entry specifies its function, invocation interface, applicable conditions, and limitations. Inner capabilities are reusable skills abstracted from M_n^{abs}. An LLM can consolidate related abstract memories into procedures with explicit inputs, outputs, execution steps, and applicable scopes. As Abstract Memory evolves, new inner skills can be added and existing ones revised. Unlike a static capability library, C_n can expand through experience, allowing the frozen-model agent to continually acquire and reuse executable skills across tasks.

Adaptive Router. The Adaptive Router connects the Task Interface, Experience Memory, and Capability Map to task execution. Given the structured interaction $\mathbf{i}_n$, it retrieves relevant experience from M_n, selects capabilities from C_n, and organizes them into an execution context:

$$\mathbf{z}_n = R_n(\mathbf{i}_n, M_n, C_n). \tag{6}$$

Its routing prompts, selection criteria, and workflow templates may evolve as memory and capabilities change. Detailed storage and update boundaries are given in Appendix B.

3.3 Guarded Harness Evolution

Continual Optimizer. Guarded harness evolution separates candidate generation from deployment. From H_n and $\mathbf{e}_n$, the Continual Optimizer diagnoses which harness contents may explain the observed outcome and proposes revisions to the Task Interface, Experience Memory, Capability Map, or Adaptive Router state. When several components are selected, we use a sequential component-wise search: components are considered in a predefined order, up to K alternatives are generated for the current component while the remaining state is fixed, and the highest-scoring admissible alternative becomes the basis for the next component. If no alternative passes the gate, that component is left unchanged. The deployed H_n is never modified during this search; only the final candidate can be committed.

Continual Evaluator. The Continual Evaluator E examines three complementary aspects: *current improvement* measures whether the candidate better solves the current task, *historical retention* checks whether previously reliable behavior is preserved, and *validity* ensures that the updated harness and its outputs remain usable. Let V_n be current-task validation cases and $P(H, V_n)$ the corresponding performance. Current improvement is

$$\Delta_n = P(\widetilde{H}_{n+1}, V_n) - P(H_n, V_n). \tag{7}$$

For historical retention, the Continual Evaluator maintains an evaluation-only anchor set A_n. Anchors preserve raw inputs and task-specific success criteria from previously observed examples chosen by an LLM-based selector (Appendix B.1). They are unavailable to execution and candidate generation, serving only as a retention test. For each anchor a, define

$$q(H, a) \in \{0, 1\}, \tag{8}$$

where success is task-specific. Historical loss counts anchors solved by the current harness that fail under the candidate:

$$D_n = \sum_{a \in A_n} \mathbf{1}\Big[q(H_n, a) = 1 \wedge q(\widetilde{H}_{n+1}, a) = 0\Big]. \tag{9}$$

The historical loss serves as an empirical proxy for forgetting by measuring performance regression on sampled historical anchors. Task-specific success criteria are listed in Appendix G. Let $v_n(\widetilde{H}) \in \{0, 1\}$ denote whether a candidate satisfies applicable syntax, output-schema, tool-use, task, and environment constraints. Candidate k is admissible if

$$G_n^{(k)} = \mathbf{1}\Big[\Delta_n^{(k)} \geq \delta_n \ \wedge \ D_n^{(k)} \leq B_n \ \wedge \ v_n(\widetilde{H}_{n+1}^{(k)}) = 1\Big]. \tag{10}$$

This gate separates local proposal generation from persistent state change. Under Eq. (10), a candidate is admissible only if it (i) improves current-task performance by at least δ_n, (ii) keeps historical

regression on the sampled anchors within B_n, and (iii) satisfies task-specific validity requirements. If multiple candidates are admissible, the Evaluator selects the candidate with the highest current-task performance, using smaller anchor-based historical regression as a tie-breaker. If no candidate is admissible, the currently deployed harness remains unchanged.

3.4 Connections to Model-Centric Continual Learning

HCL is related to several principles in model-centric continual learning, but realizes them through harness evolution rather than parameter updates (Delange et al., 2022; Wang et al., 2024b). Experience Memory plays a role analogous to replay by retaining past interactions, while the Capability Map corresponds to representation-based methods by abstracting accumulated experience into reusable skills. The Adaptive Router is analogous to architecture-based methods, as it dynamically selects and composes capabilities into workflows. Similarly, the Continual Optimizer and Continual Evaluator introduce an update-and-retention mechanism analogous to optimization and regularization: candidate harness changes are proposed from current feedback and committed only when they improve current performance without violating historical retention and validity constraints.

The key distinction is that HCL moves continual learning beyond the model parameters themselves. Rather than treating replay, representation, architecture, and update control as separate strategies for adapting a model (Kang et al., 2025; Liu et al., 2026c), HCL integrates their functions into a single evolving harness under the same acquisition–retention objective. This provides a system-level view of continual learning, where not only the model but also the surrounding agent infrastructure becomes a persistent carrier of continually acquired capabilities.

4 Experiments

We evaluate HCL on ALFWorld (Shridhar et al., 2021) and Minecraft (Wang et al., 2024a) to study long-horizon capability accumulation and reuse, and on textual and multimodal task streams to measure harness-level forgetting through repeated evaluation of earlier tasks. We use frozen Qwen-family models for interactive and multimodal experiments and frozen DeepSeek-family models for the main textual stream. Further analyses examine stability–plasticity control, stronger-model assistance for harness evolution, and individual component contributions.

Validation cases and historical anchors are used only for candidate selection, while disjoint test sets are reserved for reporting. The anchor set is updated after each task and remains fixed during candidate generation and evaluation for the next task. Stability-HCL ($B_n = 0$) and Plasticity-HCL ($B_n = \infty$) differ only in the historical-loss budget, with identical current-improvement and validity requirements. For stage-wise benchmarks, we report final average performance (*Final Avg.*) and average forgetting (*Avg. Fgt.*) following standard continual learning protocols (Delange et al., 2022; Wang et al., 2025a). Forgetting measures the mean decline from each earlier task's best observed performance to its final performance. Full experimental settings, including anchor-set sizes and proposal budgets, are provided in Appendix C.

4.1 Open-World Capability Accumulation

ALFWorld. We evaluate HCL on six sequential ALFWorld task categories using frozen Qwen3.5-9B, with 10 training episodes per category and 134 evaluation episodes. We compare against Static Harness, RAG, MemP (Fang et al., 2026), and MemRL (Zhang et al., 2026c), with the memory-based baselines reimplemented under the same execution framework for fair comparison. As shown in Table 1, all methods improve over the Static Harness, while HCL achieves the strongest overall performance. Plasticity-HCL obtains the highest final average success of 62.98%, compared with 55.56% for the best non-HCL baseline, but incurs 10.94% average forgetting. Stability-HCL achieves a comparable 61.74% final average while reducing average forgetting to 2.64%. Since the two HCL variants share the same frozen model and update mechanism and differ only in the historical-loss tolerance B_n, their contrast directly illustrates how the Continual Evaluator controls the stability–plasticity trade-off between capability acquisition and retention. The particularly large gain on the Two-object category reflects HCL's ability to reuse and compose accumulated experience, skills, and routing patterns for longer-horizon multi-object interactions. In contrast, the lower

Table 1: Final task-wise performance on ALFWorld. The best and second-best results in each metric column are marked in **bold** and underlined, respectively.

Method	Pick	Look	Clean	Heat	Cool	Two-object	Final Avg. ↑	Avg. Fgt. (%) ↓
Static Harness	95.80	66.70	25.80	26.10	9.50	58.80	47.12	–
RAG Baseline	95.80	83.30	41.90	39.10	14.30	58.80	55.56	**1.74**
MemP (Fang et al., 2026)	95.80	83.30	48.40	34.80	9.50	47.10	53.15	5.18
MemRL (Zhang et al., 2026c)	95.83	**88.89**	41.94	**39.13**	**57.14**	41.18	60.68	1.76
Stability-HCL (Ours)	**100.00**	83.30	**51.60**	30.40	28.60	76.50	61.74	2.64
Plasticity-HCL (Ours)	**100.00**	77.80	41.90	39.10	19.00	**100.00**	**62.98**	10.94

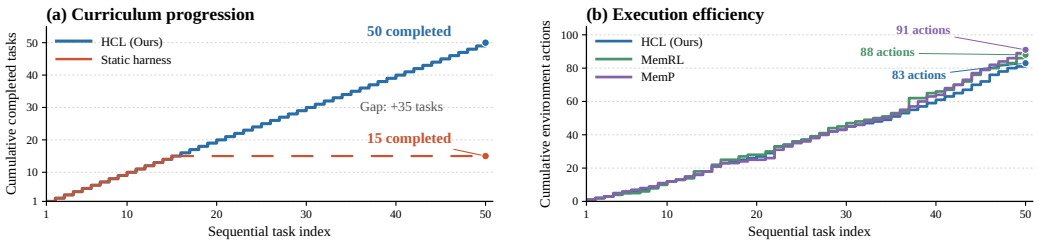


Figure 3: Minecraft curriculum progression and execution efficiency. (a) Plasticity-HCL completes all 50 tasks, while the Static Harness plateaus at 15. (b) Plasticity-HCL uses 83 cumulative environment actions, versus 88 for MemRL and 91 for MemP; lower is more efficient.

performance on Cool relative to MemRL suggests that some task-specific behaviors may benefit more from episodic memory, while harness-level updates can introduce task-dependent interference.

Minecraft. We further examine long-horizon evolution in Minecraft using deepseek-v4-flash-0731 on a 50-task curriculum spanning collection, crafting, mining, tool use, placement, smelting, and multi-step dependencies. Environment feedback is stored and can revise reusable capabilities and workflows, while retained skill tests serve as historical anchors. We use Plasticity-HCL in this setting, while MemRL and MemP are reproduced within the same harness stack as baselines. Figure 3 shows that the Static Harness follows HCL for 15 tasks and then plateaus, whereas HCL completes all 50. HCL uses 83 environment actions, compared with 88 for MemRL and 91 for MemP, indicating less redundant execution while retaining progression across the full curriculum. This setting complements the stage-wise benchmarks by testing capability accumulation and reuse over a substantially longer interaction sequence.

4.2 Controlled-Stream Continual Learning

Textual and multimodal streams. We evaluate HCL on two controlled sequential task streams. The textual stream follows the sequence of MuSiQue (Trivedi et al., 2022), ProofWriter (Tafjord et al., 2021), GSM8K (Cobbe et al., 2021), and HotpotQA (Yang et al., 2018). The multimodal stream consists of COCO detection, COCO captioning, RefCOCO grounding, and VQAv2, in that order. We use frozen DeepSeek-V4.1-Flash for the textual stream and frozen Qwen3.6-27B for the multimodal stream. Each task uses 250 adaptation examples, 50 validation examples, and 500 test examples. As shown in Table 2, HCL improves final average performance on both streams, while Stability-HCL and Plasticity-HCL exhibit different patterns of capability acquisition and retention.

On the textual stream, Plasticity-HCL achieves the highest final average of 76.10% with 0.40% average forgetting, while Stability-HCL ranks second with a final average of 70.65%. These results show that HCL achieves the strongest overall performance on the textual stream. On the multimodal stream, HCL also achieves the strongest overall balance between final performance and retention: Stability-HCL achieves the best final mean of 68.92 with only 0.22% average forgetting, while Plasticity-HCL reaches 67.96 with 0.81% forgetting. These results show that HCL can accumulate

Table 2: Final task-wise performance on the textual and multimodal streams.

Textual reasoning: MuSiQue → ProofWriter → GSM8K → HotpotQA						
Method	MuSiQue	ProofWriter	GSM8K	HotpotQA	Final Avg. ↑	Avg. Fgt. (%) ↓
DeepSeek-V4.1-Flash Zero-shot	48.00	79.20	65.20	62.20	63.65	–
MemP (Fang et al., 2026)	51.40	78.80	70.80	60.80	65.45	0.47
MemRL (Zhang et al., 2026c)	53.80	80.00	77.40	**64.20**	68.85	**0.07**
Plasticity-HCL (Ours)	**58.60**	**85.60**	**96.00**	**64.20**	**76.10**	0.40
Stability-HCL (Ours)	53.60	82.80	82.60	63.60	70.65	0.27
Multimodal perception: Detection → Caption → Grounding → VQAv2						
Method	Detection	Caption	Grounding	VQAv2	Final Mean ↑	Avg. Fgt. (%) ↓
Qwen3.6-27B Zero-shot	37.52	21.10	43.00	**85.47**	46.77	–
MemP (Fang et al., 2026)	58.67	31.89	90.80	79.27	65.16	**0.00**
MemRL (Zhang et al., 2026c)	59.33	30.66	91.40	77.60	64.75	**0.00**
Plasticity-HCL (Ours)	64.14	37.31	90.60	79.80	67.96	0.81
Stability-HCL (Ours)	**65.34**	**39.41**	**91.60**	79.33	**68.92**	0.22

Table 3: Effect of historical-loss tolerance b on final performance and forgetting.

b	MuSiQue	ProofWriter	GSM8K	HotpotQA	Final Avg. ↑	Avg. Fgt. (%) ↓
0	27.83	73.33	84.33	**59.50**	61.25	**0.39**
1	24.83	77.50	**92.33**	59.17	**63.46**	1.22
3	26.83	**79.83**	83.00	58.50	62.04	2.00
∞	**28.33**	71.00	82.00	59.17	60.13	3.45

capabilities across heterogeneous task streams, while the historical-retention constraint controls the stability–plasticity trade-off between adaptation and retention.

4.3 Harness Evolution in Harness Continual Learning

Stability–Plasticity Control To examine the effect of historical-retention constraints, we vary the historical-loss budget $B_n \equiv b$ over $b \in \{0, 1, 3, \infty\}$ using frozen DeepSeek-V4-Flash, while keeping the current-improvement and validity criteria fixed. Each previous task contributes 80 historical anchors. For each candidate update, b limits the total number of anchors that are solved by the deployed harness but fail under the candidate. Thus, $b = 0$ requires all currently solved anchors to remain successful, whereas $b = \infty$ removes this constraint. Additional experimental settings and stage-wise results are provided in Appendix D. As shown in Table 3, $b = 0$ yields the lowest average forgetting of 0.39%, whereas $b = 1$ achieves the highest final average accuracy of 63.46%, improving over $b = 0$ by 2.21 percentage points with average forgetting of 1.22%. Increasing b to 3 and ∞ reduces final average accuracy to 62.04% and 60.13%, while average forgetting rises to 2.00% and 3.45%, respectively. These results show that limited tolerance for historical regression improves final average accuracy in this setting, but further relaxation reduces both final accuracy and retention. The budget constrains regression on sampled anchors rather than all historical test cases, so $b = 0$ does not guarantee zero test-set forgetting.

External-Model Assistance We further examine whether a stronger external model can improve HCL without replacing the frozen model. Base-HCL uses Qwen3-4B throughout. Opt-HCL uses Qwen3.6-27B only to generate candidate harness updates, while Qwen3-4B handles online execution. Online-HCL instead uses Qwen3.6-27B for per-example task structuring, workflow planning, and memory, skill, and tool selection. All variants follow Stability-HCL ($B_n = 0$), with all other settings held fixed. As shown in Figure 4, Opt-HCL achieves a final average accuracy of 45.75%, improving over Base-HCL (35.10%) by 10.65 percentage points while maintaining comparable online latency (9.72 vs. 9.95 seconds per sample). Online-HCL also improves over Base-HCL, reaching 44.80% accuracy, but its online latency is substantially higher at 39.31 seconds per sample. In Opt-HCL, Qwen3.6-27B is used only to generate candidate harness updates; accepted updates are

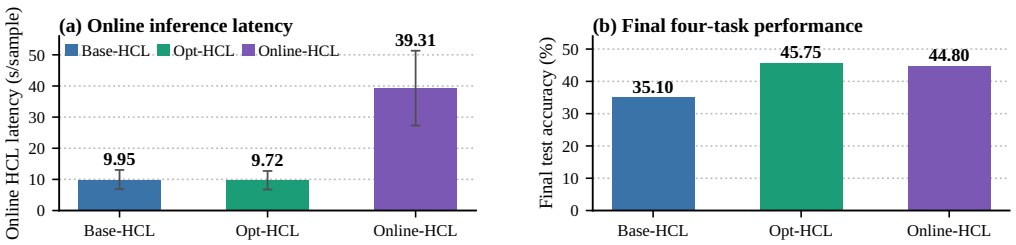


Figure 4: Effect of stronger-model placement in HCL. Base-HCL is fully driven by Qwen3-4B. Opt-HCL uses Qwen3.6-27B only for candidate harness generation while keeping Qwen3-4B for online execution. Online-HCL uses Qwen3.6-27B for online HCL operations. (a) Mean online processing time per matched sample with one standard deviation. (b) Final four-task average accuracy.

Table 4: Component ablation on the controlled multimodal stream. A component is kept available during execution but its persistent updates are frozen.

Component	I	M	C	R	Final Avg. ↑	Avg. Fgt. (%) ↓
Zero-shot	–	–	–	–	34.84	–
w/o Interface update	×	✓	✓	✓	62.37	<u>0.11</u>
w/o Memory update	✓	×	✓	✓	62.28	0.83
w/o Capability update	✓	✓	×	✓	<u>63.12</u>	**0.06**
w/o Router update	✓	✓	✓	×	62.77	0.14
Plasticity-HCL	✓	✓	✓	✓	**63.41**	0.45

retained for subsequent online execution with the frozen Qwen3-4B model. These results suggest that, in this setting, stronger-model assistance can improve HCL through persistent harness updates without requiring the stronger model during online execution.

4.4 Ablation Study

We conduct component ablations to examine the contribution of each HCL component. Specifically, on the controlled multimodal stream with frozen Qwen3.5-4B, we keep one component fixed at a time while allowing the other three to continue adapting. The fixed component remains available with its initialized contents, and all variants share the same data allocation, evaluation criteria, and update schedule. Interventions and full per-task results are provided in Appendix F. As shown in Table 4, Plasticity-HCL achieves the highest final average of 63.41%, compared with 62.28–63.12% for restricted variants. Fixing Memory causes the largest performance drop and increases average forgetting to 0.83%. Some restricted variants exhibit lower forgetting because they also acquire less new capability. These results indicate that all four components contribute to continual adaptation, and that forgetting should be interpreted together with capability acquisition rather than in isolation.

5 Conclusion

We formulate Harness Continual Learning (HCL) as a new continual learning paradigm in which the agent harness, rather than model parameters, evolves through sequential experience. HCL represents mutable harness components as a unified continual state and introduces a proposal–evaluation–commitment mechanism to balance capability acquisition and historical retention. Experiments across open-world interaction, textual reasoning, and multimodal perception show that frozen models can continually acquire and retain capabilities through the proposed harness evolution. Further analysis reveals the stability–plasticity trade-off of harness adaptation and shows that stronger models provide a more efficient strategy when used for candidate generation rather than continuous online execution. These results extend continual learning from parameter adaptation to the evolution of agent-level infrastructure, establishing harness evolution as a complementary dimension of continual learning.

Reproducibility Statement

The complete source code has been submitted as supplementary material. Detailed model, data, evaluator, and reporting settings for the stage-wise experiments are provided in Appendix C. Harness state boundaries and candidate-generation procedures are specified in Appendix B, while task-specific anchor criteria are given in Appendix G. Additional settings for the stability analysis, external capability extension, and component ablations are provided in the corresponding appendices.

AI Use Statement

Generative AI tools are used to assist with language editing. All technical content, experimental results, analyses, and claims are reviewed and verified by the authors, who take responsibility for the final content of the paper.

References

Istabrak Abbes, Gopeshh Subbaraj, Matthew Riemer, Nizar Islah, Tsuguchika Tabaru, Hiroaki Kingetsu, Sarath Chandar, and Irina Rish. Revisiting replay and gradient alignment for continual pre-training of large language models. In *Proceedings of the 4th Conference on Lifelong Learning Agents*, pp. 465–486, 2026.

Sami Abuzakuk, Anne-Marie Kermarrec, Rishi Sharma, Rasmus Moorits Veski, and Martijn de Vos. Optimizing Agentic Workflows using Meta-tools, 2026.

Giovanni Bellitto, Federica Proietto Salanitri, Matteo Pennisi, Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Simone Calderara, Simone Palazzo, and Concetto Spampinato. Saliency-driven Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jiayi Chen, Junyi Ye, and Guiling Wang. From Standalone LLMs to Integrated Intelligence: A Survey of Compound AI Systems, 2025.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, and Qing Wang. From Failed Trajectories to Reliable LLM Agents: Diagnosing and Repairing Harness Flaws, 2026a.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. HarnessX: A Composable, Adaptive, and Evolvable Agent Harness Foundry, 2026b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. MemP: Exploring Agent Procedural Memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026. doi: 10.18653/v1/2026.findings-acl.866.

Shangding Gu. From Model Scaling to System Scaling: Scaling the Harness in Agentic AI, 2026.

Chaoyue He, Xin Zhou, Di Wang, Hong Xu, Wei Liu, and Chunyan Miao. Harness Engineering for Language Agents: The Harness Layer as Control, Agency, and Runtime. *Preprints*, 2026. doi: 10.20944/preprints202603.1756.v2.

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?, 2024.

Borui Kang, Lei Wang, Zhiping Wu, Tao Feng, Yawen Li, Yang Gao, and Wenbin Li. Dynamic multi-layer null space projection for vision-language continual learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2077–2086. IEEE, 2025.

Borui Kang, Jinrui Gu, Tao Feng, Qi Fan, Yinghuan Shi, Lei Wang, Wenbin Li, and Yang Gao. Don't forget why you started: Tackling dual forgetting in vision-language continual learning. In *Forty-third International Conference on Machine Learning*, 2026.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Lewandowski, Michał Bortkiewicz, Saurabh Kumar, András György, Dale Schuurmans, Mateusz Ostaszewski, and Marlos C. Machado. Learning Continually by Spectral Regularization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Jianyang Gu, Yingqiang Ge, Weijie Xu, Xi Fang, Xiang Xu, Tianchen Zhao, Youngeun Kim, Tianyang Wang, Jihun Hamm, Smita Krishnaswamy, Jun Huan, and Chandan Reddy. Agent Harness Engineering: A Survey, 2026. Withdrawn TMLR submission.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness Updating Is Not Harness Benefit: Disentangling Evolution Capabilities in Self-Evolving LLM Agents, 2026.

Yang Liu, Toan Nguyen, and Flora D Salim. Cp-moe: Consistency-preserving mixture-of-experts for continual learning. *arXiv preprint arXiv:2605.20247*, 2026a.

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, 2026b.

Ziwei Liu, Borui Kang, Wei Li, Hangjie Yuan, Yanbing Yang, Wenbin Li, Yifan Zhu, Tao Feng, and Jun Luo. Branch, or layer? zeroth-order optimization for continual learning of vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 24026–24034, 2026c.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Aojun Lu, Tao Feng, Hangjie Yuan, Xiaotian Song, and Yanan Sun. Revisiting Neural Networks for Continual Learning: An Architectural Perspective, 2024.

Haodong Lu, Chongyang Zhao, Jason Xue, Lina Yao, Kristen Moore, and Dong Gong. Little by little: Continual learning via incremental mixture of rank-1 associative memory experts. In *Forty-third International Conference on Machine Learning*, 2026.

Qianyu Meng, Yanan Wang, Liyi Chen, Yihang Li, Wei Wu, Wenyuan Jiang, Qimeng Wang, Chengqiang Lu, Yan Gao, Yi Wu, and Yao Hu. Agent Harness for Large Language Model Agents: A Survey. *Preprints*, 2026. doi: 10.20944/preprints202604.0428.v3.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems, 2023.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools, 2023.

Jin Shang, Simone Shao, Tian Tong, Fan Yang, Yetian Chen, Yang Jiao, Jia Liu, and Yan Gao. Divide and Orthogonalize: Efficient Continual Learning with Local Model Space Projection. In *Proceedings of the Forty-First Conference on Uncertainty in Artificial Intelligence*, 2025.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face, 2023.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual Learning of Large Language Models: A Comprehensive Survey, 2025.

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.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. *International Conference on Learning Representations*, 2021.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, Online, 2021. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Edoardo Urettini and Antonio Carta. Online curvature-aware replay: Leveraging second-order information for online continual learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 60590–60609, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, 2024a.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10087–10098. IEEE, 2025a.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024b.

Xinrui Wang, Shao-Yuan Li, Jiaqiang Zhang, and Songcan Chen. Cut out and replay: A simple yet versatile strategy for multi-label online continual learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 63530–63548, 2025b.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary Prompting for Rehearsal-free Continual Learning, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory, 2025c.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments, 2024.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2023.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-Bench: Measuring Harness Effects across Models in Realistic Agent Workflows, 2026.

William Yue, Bo Liu, and Peter Stone. t-dgr: A trajectory-based deep generative replay method for continual learning in decision making. In *Proceedings of the 3rd Conference on Lifelong Learning Agents*, pp. 481–497, 2025.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses That Improve Themselves, 2026a.

Hongsheng Zhang, Zhong Ji, Jingren Liu, Yanwei Pang, and Jungong Han. Multi-stage knowledge integration of vision-language models for continual learning. *IEEE Transactions on Image Processing*, 35:615–628, 2026b. doi: 10.1109/TIP.2026.3652014.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation. In *International Conference on Learning Representations*, 2025.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline Training of Language Model Agents with Functions as Learnable Weights. 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. 60315–60335. PMLR, 2024.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026c.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, Qingnan Ren, Shun Zou, Wenxuan Huang, Lin Chen, Zehui Chen, and Feng Zhao. SkillFlow: Benchmarking Lifelong Skill Discovery and Evolution for Autonomous Agents, 2026d.

Chongyang Zhao, Mingsong Li, Haodong Lu, and Dong Gong. On token's dilemma: Dynamic moe with drift-aware token assignment for continual learning of large vision language models. In *Conference on Computer Vision and Pattern Recognition 2026*, 2026.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. SkillLearnBench: Benchmarking Continual Learning Methods for Agent Skill Generation on Real-World Tasks, 2026.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing Large Language Models with Long-Term Memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:19724–19731, 2024.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in LLM Agents: A Unified Review of Memory, Skills, Protocols and Harness Engineering, 2026.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models Are Human-Level Prompt Engineers, 2023.

A Appendix

The supplementary material provides detailed harness definitions and update boundaries, detailed experimental settings and task-wise results, additional stability–plasticity analyses, external capability details, component ablations, and task-specific anchor criteria.

B Harness Details and Update Boundaries

B.1 Harness and Evaluator Boundaries

Task Interface. The Task Interface I_n constructs $\mathbf{i}_n$ during execution. During candidate generation, the Optimizer may revise its prompts, templates, and parsing or normalization rules. Any such revision enters the deployed harness H_n only when the corresponding candidate is committed.

Raw and Abstract Memory. Raw and Abstract Memory M_n^{raw} and M_n^{abs} supply interaction records and reusable guidance to the Router. The Optimizer may add raw records or revise abstract entries during candidate generation, but these changes become part of H_n only after candidate commitment.

Capability Map. The Capability Map C_n supplies the capabilities available to the Router. The Optimizer may add or revise internal skills, while any resulting change is excluded from the deployed state until the candidate containing it is committed.

Adaptive Router. The Adaptive Router R_n constructs $\mathbf{z}_n$. Its routing prompts, selection criteria, and workflow templates may be revised by the Optimizer, with accepted revisions entering H_n only through candidate commitment.

Anchor Set. The Anchor Set A_n is available only to the Evaluator and is unavailable to both execution and candidate generation. After each adaptation batch, an LLM-based selector chooses a configured number of anchors from the observed examples using their predictions, correctness, format compliance, and task metadata. The selector prioritizes failures, format violations, brittle cases, and representative examples. Invalid or incomplete selections are completed by a deterministic fallback, after which the anchors are deduplicated and retained under a fixed per-task capacity. Historical anchors remain fixed during the optimization of subsequent tasks.

Thus, H_n contains only persistent execution-time contents; $\mathbf{i}_n$, $\mathbf{z}_n$, and $\mathbf{y}_n$ are transient, and A_n remains evaluation-only.

B.2 Optimizer Candidate Generation

Interaction feedback indicates whether an execution is successful but not how the harness should change. In the implementation, memory and skill writes are first evaluated jointly as a single implicit update and are either retained or rolled back. The explicit candidate-generation stage then optimizes a configuration-defined sequence of prompt and context artifacts, including the task-interface structuring prompt and the router's workflow, memory-selection, and context prompts. The task-interface artifact is skipped when no current validation example invokes model-based structuring.

For each applicable artifact, the configured proposal model receives the current artifact, post-execution metrics, current validation statistics, aggregate historical-anchor statistics, and the commit-gate configuration. A single generation call returns up to K alternatives. The returned artifacts are parsed and normalized, after which each alternative is evaluated separately by temporarily replacing only the target artifact. Candidates that fail artifact validation, do not change any comparable behavior, fail to meet the required current-task improvement, exceed the historical-loss budget, or violate the format-compliance threshold are rejected. Among the admissible candidates, the Evaluator prioritizes current-task performance and uses lower anchor loss as the first tie-breaker. If no candidate is admissible, the artifact remains unchanged. Otherwise, the selected artifact is persisted and applied immediately, so each subsequent artifact is generated and evaluated against the already updated harness. Algorithm 1 summarizes this procedure.

Algorithm 1 Sequential Prompt-Artifact Optimization

Require: deployed harness H, post-execution result R_{train}, configured proposal model G, current validation set V, historical anchors A, configured artifact sequence $\mathcal{C}$, candidate budget K
Ensure: updated deployed harness H

$\mathcal{C}' \leftarrow \textsc{ApplicableArtifacts}(\mathcal{C}, V)$
for $c \in \mathcal{C}'$ **in configured order do**
  $R_0 \leftarrow \textsc{Evaluate}(H, V, A)$
  $X_c \leftarrow \textsc{GenerationContext}(H[c], R_{\text{train}}, R_0)$
  $\mathcal{Q}_c \leftarrow \textsc{Normalize}\big(G(c, X_c, K)\big)$ ▷ one call returns at most K alternatives
  for $q \in \mathcal{Q}_c$ **do**
    $H_q \leftarrow \textsc{Replace}(H, c, q)$
    $R_q \leftarrow \textsc{Evaluate}(H_q, V, A)$
  end for
  $\mathcal{Q}_{\text{ok}} \leftarrow \{q \in \mathcal{Q}_c : \textsc{Gate}(q, R_q, R_0) = 1\}$
  if $\mathcal{Q}_{\text{ok}} \neq \varnothing$ **then**
    $q^\star \leftarrow \textsc{SelectBest}(\mathcal{Q}_{\text{ok}}, \{R_q\})$
    $H[c] \leftarrow q^\star$
    $\textsc{Persist}(q^\star)$ ▷ commit before processing the next artifact
  end if
end for
return H

C Experimental Settings and Reporting Notes

C.1 Experimental Settings

The following paragraphs summarize the model, stream, adaptation and evaluator data, and final reporting protocol for the stage-wise experiments retained in this appendix. Unless otherwise stated, counts are per task or category.

Across these experiments, validation cases and historical anchors are restricted to the Evaluator, and final test cases are used only for reporting. The main Stability-HCL and Plasticity-HCL profiles use $B_n = 0$ and $B_n = \infty$, respectively. A main-profile candidate must improve by at least one validation case for discrete metrics or strictly improve the designated continuous score, without introducing an invalid outcome. For ALFWorld and Minecraft, because replaying candidate updates can alter the state of the interactive environment, we replace strict historical backtesting with an LLM-based evaluation that judges candidate acceptance given historical task records. The

independent textual sweep uses 40 proposal opportunities, requires two additional correct predictions among 80 validation cases and at least 90% format compliance, and varies only $B_n \equiv b$ for $b \in \{0, 1, 3, \infty\}$.

ALFWorld main. The stream contains six categories—Pick-and-Place, Look-in-Light, Clean, Heat, Cool, and Two-object—and uses Qwen3.5-9B as the frozen model. Each category provides 10 training episodes with at most 50 interaction steps per episode, and the observed categories are evaluated after every stage. Final reporting uses the 134 official evaluation episodes and includes category macro-average success and average forgetting over the first five categories.

Minecraft main. This stream contains 50 tasks spanning collection, crafting, mining, tool use, placement, smelting, and multi-step dependencies. HCL adapts from sequential environment feedback and uses retained skill tests as historical anchors. We report cumulative task completion and environment actions because completed tasks are not systematically replayed after every update.

Textual main. The task order is MuSiQue $\rightarrow$ ProofWriter $\rightarrow$ GSM8K $\rightarrow$ HotpotQA, with DeepSeek-V4.1-Flash frozen throughout. Thinking is disabled and temperature is set to 0. Each task uses 250 adaptation examples, 50 validation examples, and 500 disjoint test examples. All methods share the same model, task order, data allocation, and evaluation protocol. We report normalized exact-match accuracy, its final task average, and average forgetting.

Multimodal main. The task order is COCO detection $\rightarrow$ COCO captioning $\rightarrow$ RefCOCO grounding $\rightarrow$ VQAv2, with Qwen3.6-27B frozen throughout. Each task uses 250 adaptation examples, 50 validation examples, and 500 disjoint test examples. Final reporting includes the task-specific scores, their unweighted average, and forgetting.

Qwen textual comparison (appendix). The textual experiment follows the same MuSiQue, ProofWriter, GSM8K, and HotpotQA order with Qwen3.6-27B frozen. Each task uses 250 adaptation examples, 50 validation examples, and 500 disjoint test examples. It is reported in Appendix C.3.1 as a cross-backbone comparison and is not mixed with the main textual results.

Textual budget sweep. The independent sweep follows the same textual task order with DeepSeek-V4-Flash. Each task uses 300 adaptation examples, 80 validation examples, 80 anchors for every earlier task, and 600 test examples. The sweep allows 40 proposal opportunities, ten at each task stage.

External-model assistance. The Base-HCL, Opt-HCL, and Online-HCL variants use the same task stream, data allocation, update schedule, optimizer components, and candidate budget. All three variants use the Stability-HCL acceptance policy with $B_n = 0$. Each candidate must yield at least one additional correct validation prediction and must not forget any evaluated historical anchor. The only experimental variable is the set of HCL operations assigned to Qwen3.6-27B.

Capability extension. This experiment uses the same four reasoning tasks and a frozen Qwen3.6-27B model, with an external arithmetic calculator registered in the Capability Map. The evolving selector and Router determine whether the calculator is invoked. We report final per-task accuracy and, for GSM8K, the number of examples routed to the calculator.

Component ablation. The ablation follows the multimodal task order with Qwen3.5-4B frozen. Every task uses 250 adaptation examples, 50 validation examples, and 500 test examples; each variant keeps one persistent harness component fixed. We report final per-task scores, average performance, forgetting, and committed-update counts.

C.2 Evaluation Metrics

For the stage-wise ALFWorld, textual, and multimodal benchmarks, after stage s we evaluate the deployed harness on the current and all previously observed tasks, obtaining $R_{s,j}$ for $j \leq s$. For

Table 5: Textual-reasoning results with frozen Qwen3.6-27B. The best and second-best entries in each column are marked in **bold** and underlined, respectively.

Method	MuSiQue	ProofWriter	GSM8K	HotpotQA	Final Avg. ↑	Avg. Fgt. (%) ↓
Qwen3.6-27B Zero-shot	**57.00**	**83.80**	44.80	62.40	62.00	–
MemP (Fang et al., 2026)	56.40	74.00	54.00	**65.80**	62.55	0.20
MemRL (Zhang et al., 2026c)	55.80	75.20	52.20	65.60	62.20	**0.00**
Plasticity-HCL (Ours)	56.00	79.80	**66.00**	62.80	**66.15**	7.87
Stability-HCL (Ours)	55.80	77.00	51.00	64.20	62.00	1.27

streams whose task scores are reported on a common 0–100 scale, we report

$$\mathrm{Avg}_T = \frac{1}{T}\sum_{j=1}^{T} R_{T,j}, \qquad \mathrm{Fgt}_T = \frac{1}{T-1}\sum_{j=1}^{T-1}\left(\max_{r\in\{j,\ldots,T\}} R_{r,j} - R_{T,j}\right). \tag{11}$$

For the multimodal stream, each task-specific reporting metric is expressed on the 0–100 scale before taking the unweighted final mean and computing forgetting, so forgetting is reported in percent (%). Minecraft is reported separately using cumulative task completion and environment actions, because completed tasks are not systematically replayed after every update. Stability-HCL sets $B_n = 0$ and Plasticity-HCL $B_n = \infty$, with matched current-improvement and validity criteria; B_n constrains anchor loss rather than test-set forgetting.

C.3 Controlled Harness Continual Learning

We next evaluate HCL on task sequences. Within each stream, all HCL profiles share the same foundation model, task order, data allocation, editable artifacts, and candidate generator.

C.3.1 Qwen Textual-Reasoning Experiment

For completeness, we retain the earlier Qwen3.6-27B textual experiment in this appendix. The stream follows the order MuSiQue (Trivedi et al., 2022), ProofWriter (Tafjord et al., 2021), GSM8K (Cobbe et al., 2021), and HotpotQA (Yang et al., 2018). These tasks cover multi-hop question answering, logical deduction, mathematical reasoning, and knowledge-intensive question answering. Qwen3.6-27B remains frozen throughout; each task uses 250 adaptation, 50 validation, and 500 disjoint test examples. The result is presented as a separate cross-backbone comparison and is not pooled with the main textual experiment.

On Qwen3.6-27B, Plasticity-HCL attains the best final average (66.15%) but also substantially higher forgetting (7.87%), whereas Stability-HCL trades away adaptation for lower forgetting (1.27%). This separate result indicates that the realized stability–plasticity outcome depends on the frozen backbone and optimization trajectory, even when the data budget and task order are matched.

C.4 Main-Result Reporting Notes

The complete final task-wise results are reported directly in Tables 1 and 2. Validation cases and historical anchors are never used for final reporting; all reported main scores use the disjoint test sets described above.

D Additional Stability–Plasticity Analysis

For this independent sweep, δ_n requires at least two additional correct validation cases, every candidate must achieve at least 90% output-format compliance and introduce no syntax, tool-use, or environment violations, and only $B_n \equiv b$ is varied. Each task uses 300 adaptation, 80 validation, and 600 test examples, with 80 anchors retained for every earlier task.

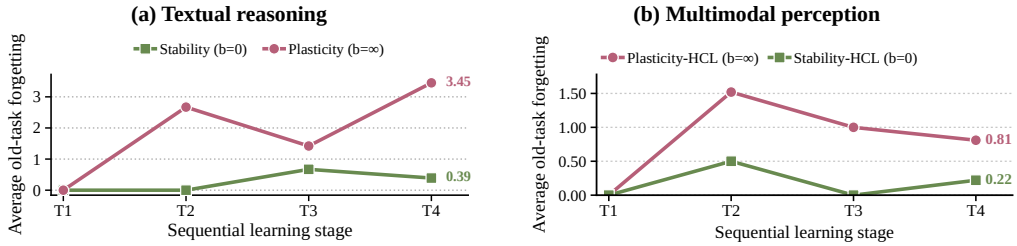


Figure 5: Stage-wise forgetting under different historical-loss tolerances. (a) Textual reasoning comparing Stability-HCL and Plasticity-HCL. Forgetting values are reported in percent. (b) Multimodal perception comparing Stability-HCL and Plasticity-HCL. Forgetting values are reported in percent.

The task-wise sweep results and stage-wise forgetting trajectories are shown in Table 3 and Figure 5 in the main paper. The sweep is kept separate from the DeepSeek-V4.1-Flash textual main experiment because it uses a different configuration and data budget.

Across this sweep, final forgetting increases from 0.39% at $b = 0$ to 3.45% at $b = \infty$. The multimodal trajectory in Figure 5 similarly shows 0.22% rather than 0.81% final forgetting for Stability-HCL versus Plasticity-HCL. Nonzero test forgetting at $b = 0$ reflects the finite coverage of the anchor set: preserving all currently solved anchors does not guarantee unchanged behavior on historical test cases not represented by A_n.

E External Capability Extension

We register an external arithmetic calculator in the Capability Map of a frozen Qwen3.6-27B agent and allow the evolving selector and router to decide whether it should be used for each input. Relative to an earlier Stability-HCL run, the capability-aware run changes final accuracy from 58.60% to 58.60% on MuSiQue, 73.80% to 82.80% on ProofWriter, 53.20% to 90.40% on GSM8K, and 64.40% to 65.80% on HotpotQA. The largest change occurs on GSM8K, where 254 of 500 test examples are routed to the calculator. Because the compared runs also differ in prompts, retries, and memory trajectories, these changes do not isolate the causal contribution of tool execution alone.

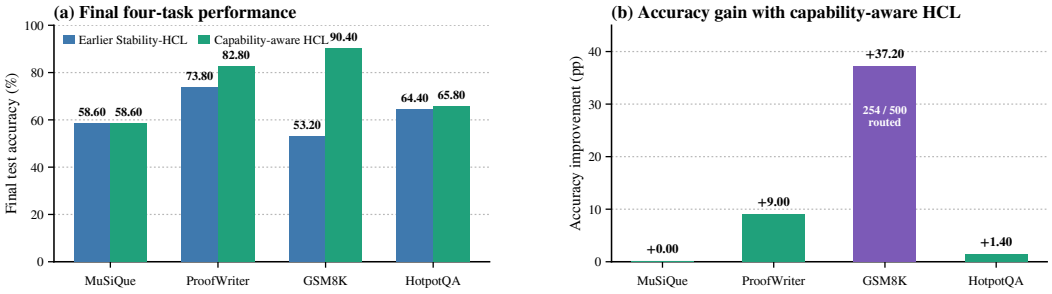


Figure 6: **Capability-aware HCL with an external arithmetic calculator.** (a) Final accuracy compared with the earlier Stability-HCL run. (b) Corresponding accuracy gains; GSM8K shows the largest improvement (+37.20 pp), with 254/500 examples routed to the calculator. The comparison is exploratory because the runs also differ in prompts, retries, and memory trajectories.

F Component Ablation Details

All ablation variants use frozen Qwen3.5-4B and share the task order, data allocation, evaluation criteria, and update schedule. A disabled component remains available during execution but retains

Table 6: Update scope of the component-ablation variants. A ✓ permits persistent updates, while × keeps the corresponding component fixed.

Method	I	M	C	R	Fixed contents
Zero-shot	–	–	–	–	No structured HCL harness or persistent updates.
Full HCL	✓	✓	✓	✓	None.
w/o Interface update	×	✓	✓	✓	Prompts, templates, parsing, and normalization rules.
w/o Memory update	✓	×	✓	✓	Raw and Abstract Memory entries.
w/o Capability update	✓	✓	×	✓	Reusable skills.
w/o Router update	✓	✓	✓	×	Routing prompts, selection criteria, and workflow templates.

Table 7: Full component-ablation results on the controlled multimodal stream. "Committed" counts candidate updates entering the persistent harness.

Method	Detection	Caption	Grounding	VQAv2	Final Avg. ↑	Avg. Fgt. (%) ↓
Zero-shot	35.11	22.98	0.00	81.27	34.84	–
w/o Interface update	53.45	33.56	87.80	74.67	62.37	<u>0.11</u>
w/o Memory update	55.50	28.95	88.00	76.67	62.28	0.83
w/o Capability update	55.11	34.16	86.40	76.80	<u>63.12</u>	**0.06**
w/o Router update	53.59	36.68	87.40	73.40	62.77	0.14
Full HCL	53.07	36.09	87.60	76.87	**63.41**	0.45

its initialized contents throughout the stream. Zero-shot evaluates the frozen model without the structured HCL harness or sequential updates.

F.1 Ablation Configurations

Table 6 summarizes which harness components are allowed to receive persistent updates in each variant.

Because reusable skills may be derived from Abstract Memory, disabling Memory updates also removes this source of newly acquired skills. The corresponding variant therefore measures both direct memory adaptation and its downstream effects on capability construction.

F.2 Full Ablation Results

Table 7 reports the complete task-wise results for the controlled multimodal stream, together with final average performance, average forgetting.

Full HCL achieves the highest final average score of 63.41, compared with 34.84 for the frozen zero-shot model. Restricting any individual component reduces the final average, although the magnitude differs across components. Freezing Interface updates reduces the final average to 62.37, with the main drops appearing on Caption and VQAv2. Freezing Memory updates produces the lowest ablated average, 62.28, and the largest average forgetting, 0.83%, indicating that adaptive memory is particularly important for retaining and reusing experience across the stream. By comparison, disabling Capability updates has a smaller effect on final performance (63.12), consistent with this multimodal stream relying less on long-horizon executable skills than the interactive Minecraft setting. Freezing Router updates yields 62.77 and notably reduces VQAv2 performance to 73.40, suggesting that adaptive execution and selection policies remain useful even when the other harness components can continue to evolve.

G ANCHOR SUCCESS CRITERIA

The fixed task-specific success criterion $q(H, a)$ in Eq. (8) is applied to the same raw input under H_n and $\widetilde{H}_{n+1}$.

Textual reasoning. For MuSiQue and HotpotQA, an anchor is successful when the normalized short answer exactly matches an accepted reference answer. For ProofWriter, the parsed entailment label must match the gold label and the output schema must be valid. For GSM8K, the parsed final number must match the gold value after comma and unit normalization.

Multimodal perception. For COCO detection, the queried category must be correct, the matched box must have IoU ≥ 0.5, and the box schema must be valid. For COCO captioning, the sentence-level CIDEr score must be at least 0.5 on the normalized $[0, 1]$ scale, and the caption schema must be valid. For RefCOCO grounding, the predicted box must be valid and have IoU ≥ 0.5 with the referred-object box. For VQAv2, the standard VQA consensus score must be 1.0 after answer normalization.

Interactive environments. For ALFWorld, the goal predicate must be reached within 50 steps under a valid action sequence. For Minecraft, the retained skill test must reach its predefined inventory or world-state predicate through a valid action sequence.