PRINCIPLES THAT GUIDE, ACTIONS THAT INFORM: AGENT EVOLUTION VIA KNOWLEDGE ABSTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents have demonstrated strong capabilities in interactive environments, yet their ability to continually evolve from experience remains limited. Although fine-tuning provides a pathway for adaptation, its reliance on parameter access and substantial computational resources limits its flexibility and scope of application, particularly for large-scale and closed-source LLMs. External memory provides an alternative by enabling agents to accumulate experience without modifying model parameters. However, existing approaches primarily focus on how experiences are represented and organized, while the acquired knowledge often remains closely coupled with specific tasks and contexts, limiting its generalization for continual agent evolution. A fundamental challenge is therefore how to transform concrete interactions into abstract and reusable knowledge that can guide future decisions and enable agents to generalize beyond individual experiences.

To address this issue, we propose SAGA (**S**elf-evolving **A**gents through Experience-**G**rounded **A**bstraction), a framework for experience-grounded knowledge abstraction and utilization in LLM agents. SAGA progressively transforms interaction trajectories into episodic descriptions, reusable procedures, and principles with explicit applicability conditions, while preserving links to their supporting execution evidence. To make abstract knowledge actionable, SAGA instantiates retrieved principles into task-specific operational guidance and uses principle-derived checks to refine candidate actions through corrective feedback and resampling. These mechanisms form an execution–abstraction feedback loop, in which accumulated knowledge guides subsequent interactions and newly collected experiences update the hierarchical memory. Experiments on ScienceWorld and ALFWorld demonstrate improved task performance, while ablation studies show that contextual instantiation and action regulation are important for realizing the benefits of principle-level knowledge.

1 INTRODUCTION

The rapid advancement of large language models (LLMs) has enabled a new paradigm of autonomous agents, where pretrained models are augmented with external tools, memory modules, and interaction interfaces for complex reasoning and decision-making tasks. Recent LLM-based agents have demonstrated promising capabilities in diverse interactive environments, including embodied systems (Brohan et al., 2023; Wang et al., 2023), text-based environments such as ALFWorld and ScienceWorld (Shridhar et al., 2020; Wang et al., 2022), and emerging computer-use scenarios. However, existing agents largely rely on the capabilities acquired during pretraining and lack mechanisms for continual adaptation after deployment. A fundamental challenge toward more capable agents is therefore enabling self-evolution, where agents can learn from interaction experiences, refine their behaviors, and gradually acquire task-specific or user-specific knowledge beyond the original foundation model.

Recent studies have explored several directions toward self-evolving agents. One direction adapts the model itself through supervised fine-tuning or reinforcement learning, enabling direct modification of agent behaviors but requiring substantial computational resources and access to model parameters (Shao et al., 2024; Schulman et al., 2017). Another direction performs adaptation at

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.

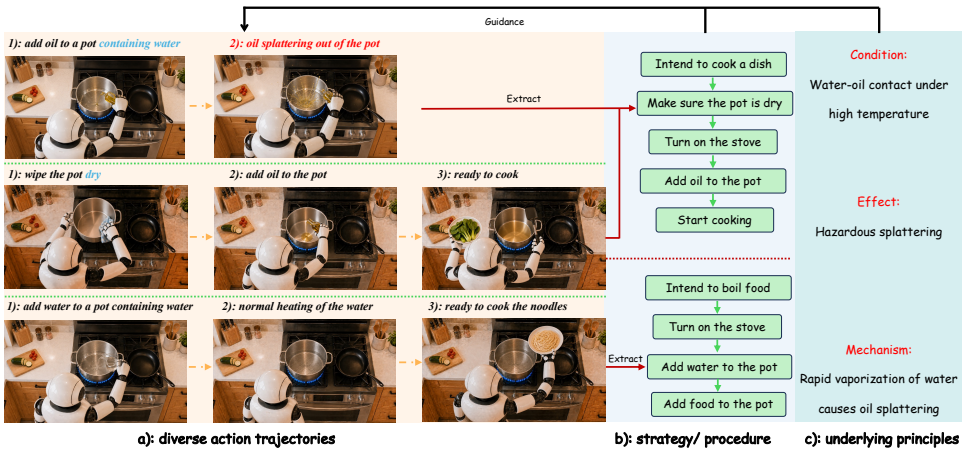

Figure 1: Illustrative Example of Experience-Grounded Knowledge Evolution. Interaction experiences are abstracted into reusable knowledge, which guides future actions.

inference time through reasoning, reflection, and self-improvement mechanisms without updating model parameters (Yao et al., 2022; Madaan et al., 2023; Shinn et al., 2023). Recent works further investigate experience-driven adaptation by extracting reusable skills or strategies from previous interactions (Wang et al., 2023; Zhao et al., 2024; Xue et al., 2026). Among these approaches, external memory provides an effective mechanism for continual adaptation of LLM agents. Existing memory systems study persistent memory management (Packer et al., 2023; Zhong et al., 2024), structured memory representations (Gutiérrez et al., 2024; Rasmussen et al., 2025; Xu et al., 2026), and adaptive memory evolution (Zhang et al., 2026a; Fan et al., 2026; Cao et al., 2026; Zhang et al., 2026b). These methods demonstrate the importance of memory as an interface between foundation models and evolving environments.

Despite recent progress in agent memory systems, existing approaches primarily focus on storing and retrieving information from past interactions. Memory architectures such as MemGPT and MemoryBank maintain persistent interaction histories for long-term context reuse (Packer et al., 2023; Zhong et al., 2024), while retrieval-based methods further improve access to stored experiences through structured memory mechanisms (Gutiérrez et al., 2024). However, these memories are typically represented as episodic records, retrieved contexts, or task-specific procedures, providing limited support for extracting transferable knowledge across tasks and environments. Recent experience-driven approaches, including Reflexion, ExpeL, and Voyager, begin to explore how accumulated experiences can improve future agent behavior through reflection, skill acquisition, and experience reuse (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023). Nevertheless, how to systematically transform interaction experiences into higher-level, experience-grounded knowledge that captures reusable principles and applicability conditions remains a fundamental challenge for lifelong adaptation of LLM-based agents (Zheng et al., 2025; Zhang et al., 2026c).

To address this challenge, we propose SAGA (**S**elf-evolving **A**gents through Experience-**G**rounded **A**bstraction), a framework for experience-driven agent adaptation through evidence-linked knowledge abstraction and utilization. As illustrated in Fig. 1, SAGA establishes an execution–abstraction feedback loop: accumulated knowledge guides subsequent interactions, while newly collected trajectories provide evidence for updating the knowledge representation. The framework organizes experiences into four abstraction levels, from raw trajectories and episodic descriptions to reusable procedures and principles with explicit applicability conditions. Links between adjacent levels preserve the connection between abstract knowledge and its supporting execution evidence. For knowledge utilization, SAGA instantiates retrieved principles into operational guidance conditioned on the current task and environmental context. Principle-derived checks further regulate candidate actions through corrective feedback and bounded resampling. These mechanisms connect knowledge abstraction with both task-level guidance and execution-time action refinement. We evaluate SAGA on ScienceWorld (Wang et al., 2022) and ALFWorld (Shridhar et al., 2020) under memory-based and parameter-based adaptation settings. The results demonstrate improved task performance, while ablation studies identify contextual instantiation and action regulation as complementary mechanisms for effective principle-level knowledge reuse.

Our contributions are summarized as follows. **i)** We develop an evidence-linked knowledge abstraction pipeline that organizes interaction experiences into raw trajectories, episodic knowledge, reusable procedures, and principles with explicit applicability conditions, preserving traceability across abstraction levels. **ii)** We propose SAGA, which integrates task-conditioned principle instantiation and principle-based action regulation within an execution–abstraction feedback loop, connecting experience-derived knowledge with operational guidance and action refinement. **iii)** We evaluate SAGA on ScienceWorld and ALFWorld under both memory-based and parameter-based adaptation settings, and analyze the contributions of knowledge abstraction, contextual instantiation, and action regulation through dual-backbone ablation studies.

2 Related Work

Self-Evolving LLM Agents. LLM agents increasingly incorporate feedback and accumulated experience to improve reasoning and decision making. ReAct (Yao et al., 2022) interleaves reasoning with environmental actions, while Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) use iterative feedback and reflection to improve generated responses and subsequent task execution. Beyond individual attempts, Voyager (Wang et al., 2023) accumulates executable skills for open-ended exploration, and ExpeL (Zhao et al., 2024) combines retrieved experiences with insights extracted from successful and unsuccessful trajectories. StraTA (Xue et al., 2026) jointly optimizes strategy generation and action execution through agentic reinforcement learning. Evolving-RL (Fan et al., 2026), MemRL (Zhang et al., 2026b), and Remember Me, Refine Me (Cao et al., 2026) further explore complementary mechanisms for experience-driven improvement. These studies motivate agent adaptation through both external knowledge and parameter updates. SAGA focuses on connecting experience abstraction with knowledge utilization within a recurring execution–abstraction loop.

Memory for Continual Agent Adaptation. External memory allows agents to retain information across interactions without requiring updates to the underlying model parameters. MemGPT (Packer et al., 2023) introduces memory management inspired by operating systems, while MemoryBank (Zhong et al., 2024) supports long-term memory for personalized interactions. Subsequent studies investigate graph-based retrieval (Gutiérrez et al., 2024), temporal memory architectures (Rasmussen et al., 2025), and agentic memory construction (Xu et al., 2026). Mem0 (Chhikara et al., 2025) explores scalable memory management, while agentic context engineering (Zhang et al., 2026a) treats context as an evolving resource. CoEvo-Mem (Ye et al., 2026) further studies the joint evolution of retrieval policies and memory banks. These approaches provide complementary mechanisms for maintaining, organizing, and updating persistent knowledge. SAGA places particular emphasis on the relationship between abstraction levels and their execution evidence, and on converting retrieved principles into operational guidance and action checks.

Experience Abstraction and Knowledge Utilization. Several studies directly investigate how interaction experience can be transformed into reusable knowledge. Voyager (Wang et al., 2023) represents acquired behaviors as executable skills, and ExpeL (Zhao et al., 2024) extracts cross-task insights from prior trajectories. CLIN (Majumder et al., 2023) maintains a dynamic textual memory of causal abstractions and updates it across trials to support task adaptation and generalization. AutoManual (Chen et al., 2024) constructs instruction manuals through online rule generation and revision, retaining examples and validation records associated with the rules. AutoGuide (Fu et al., 2024) extracts conditional guidelines from offline experiences and selects relevant guidance according to the current interaction context. The experience compression perspective (Zhang et al., 2026c) further relates memories, skills, and rules through different degrees of abstraction. Building on these advances, SAGA integrates knowledge abstraction and execution-time utilization through three connected mechanisms: an evidence-linked hierarchy spanning trajectories, episodes, procedures, and principles; contextual instantiation that converts retrieved principles into task-specific operational guidance; and principle-derived executable checks that refine candidate actions through corrective feedback and bounded resampling. Where AutoManual (Chen et al., 2024) emphasizes rule management and manual construction, and AutoGuide (Fu et al., 2024) emphasizes context-aware guideline selection, SAGA explicitly couples a multi-level abstraction pipeline with post-retrieval instantiation and action regulation. New execution trajectories subsequently provide evidence for updating the hierarchy, closing the loop between knowledge construction and utilization.

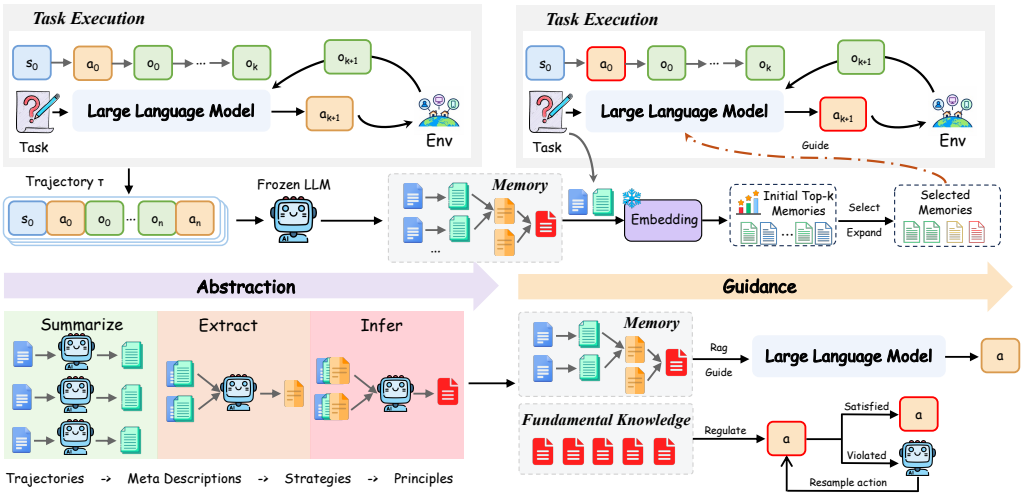

Figure 2: Overview of SAGA, illustrating the close-loop evolving process from experience abstraction to knowledge-guided decision making.

3 SAGA: Experience-Grounded Knowledge Evolution

3.1 Overview

In this work, we propose SAGA (**S**elf-evolving **A**gents through Experience-**G**rounded **A**bstraction), a framework that enables continual evolution of LLM agents through experience-grounded knowledge abstraction. SAGA establishes a closed-loop interaction between task execution and knowledge evolution, where acquired experiences are transformed into reusable knowledge and subsequently leveraged to support future decision making.

As illustrated in Fig. 2, SAGA integrates experience abstraction with knowledge-guided execution in a self-evolving loop. During task execution, the agent collects interaction trajectories from the environment. These trajectories are progressively abstracted into multiple levels of knowledge, including meta descriptions, reusable strategies, and experience-grounded principles, which are organized in a memory library. For subsequent tasks, relevant knowledge is retrieved to provide decision guidance, while extracted principles further serve as high-level behavioral constraints for regulating generated actions. If a candidate action is inconsistent with the acquired principles, SAGA refines the action through resampling. The resulting interactions generate new experiences that further update and refine the knowledge representations, enabling continual agent evolution.

3.2 Hierarchical Knowledge Representation

To enable experience-grounded knowledge evolution, SAGA organizes agent memory into multiple abstraction levels. Following the experience compression perspective, where memories, skills, and rules represent progressively abstract knowledge representations (Zhang et al., 2026c), SAGA transforms concrete interactions into reusable knowledge while preserving the evidence chain from execution trajectories. Specifically, SAGA considers four levels of abstraction: raw experiences, episodic knowledge, procedural knowledge, and experience-grounded principles.

L_0: Raw Experience. Raw experiences represent the original interaction trajectories collected during task execution. A trajectory is represented as

$$\tau_i = \left(s_0, \{(a_t, o_t, r_t)\}_{t=1}^{T}\right),$$

where o_t, a_t, and r_t denote the observation, action, and environmental feedback at step t, respectively. Raw experiences preserve fine-grained execution details and serve as the evidence source for subsequent abstraction. However, their strong dependency on individual task instances limits transferability across diverse tasks and environments.

L_1: Episodic Knowledge. Episodic knowledge provides a compact representation of individual interaction experiences. Each episodic representation preserves task-relevant information, including

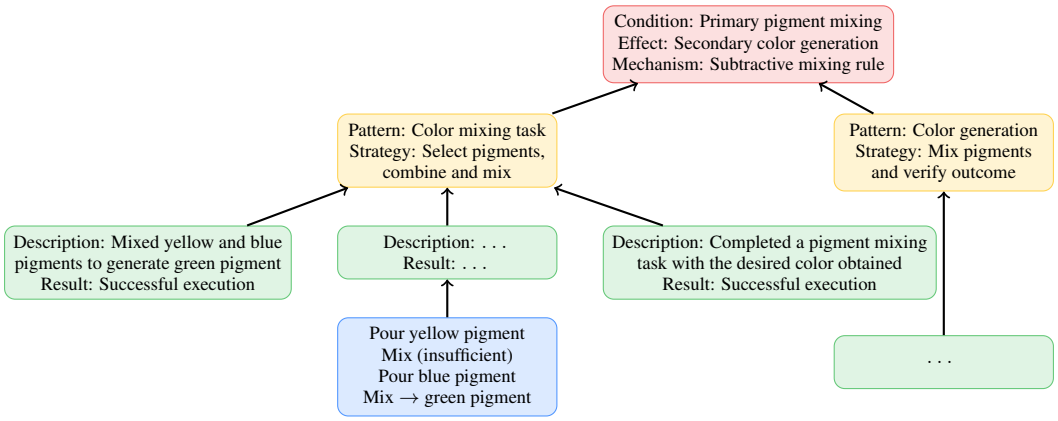

Figure 3: Illustration of hierarchical knowledge abstraction in SAGA. The colors represent different abstraction levels: blue for L_0 raw experiences, green for L_1 episodic knowledge, yellow for L_2 procedural knowledge, and red for L_3 experience-grounded principles.

task objectives, key interactions, outcomes, and encountered obstacles, while removing redundant execution details. Given a raw trajectory τ_i, an LLM-based summarization process generates its corresponding episodic representation:

$$\mathcal{K}_i^1 = \text{Summarize}_{\text{LLM}}(\tau_i).$$

Thus, L_0 and L_1 maintain a one-to-one correspondence, where each episodic representation is grounded in a specific trajectory while providing a compact representation for retrieval and further abstraction.

L_2: Procedural Knowledge. Procedural knowledge captures reusable solution patterns aggregated from multiple episodic experiences with shared task structures. Unlike episodic knowledge that represents individual executions, procedural knowledge describes generalized strategies for a class of related tasks.

Specifically, SAGA identifies related episodic representations and extracts procedural knowledge through an LLM-based extraction process:

$$\mathcal{C}_i^1 = \{\mathcal{K}_j^1\}_{j\in\mathcal{I}_i}, \qquad \mathcal{K}_i^2 = \text{Extract}_{\text{LLM}}\left(\mathcal{C}_i^1, \{\tau_j\}_{j\in\mathcal{I}_i}\right),$$

where $\mathcal{I}_i$ denotes the index set of episodic experiences sharing a common task pattern. The extracted procedure summarizes common behavioral structures across experiences while preserving grounding in the original trajectories. Multiple episodic experiences may contribute to one procedural representation, and different procedures may share overlapping experiences.

L_3: Experience-Grounded Principles. Experience-grounded principles capture fundamental regularities inferred from multiple procedural representations and their supporting experiences. They describe underlying rules, constraints, and relations shared across different procedures.

Specifically, SAGA infers principle knowledge from related procedures through an LLM-based inference process:

$$\mathcal{C}_i^2 = \{\mathcal{K}_j^2\}_{j\in\mathcal{J}_i}, \qquad \mathcal{K}_i^3 = \text{Infer}_{\text{LLM}}\left(\mathcal{C}_i^2, \mathcal{E}_i\right),$$

where $\mathcal{J}_i$ denotes the index set of related procedures, and $\mathcal{E}_i$ denotes their supporting evidence. The resulting principles provide higher-level guidance beyond individual procedures while maintaining traceability to underlying experiences.

Figure 3 illustrates this abstraction process using a pigment mixing example from ScienceWorld, where multiple interaction experiences are progressively transformed into reusable procedures and higher-level principles stored in the hierarchical memory $\mathcal{M}$.

3.3 Knowledge Utilization and Self-Evolving Loop

Given a task instance x_t and the hierarchical memory $\mathcal{M}$, SAGA performs knowledge-guided execution and self-evolution through an iterative retrieval–utilization–abstraction loop, as summarized

Algorithm 1 Knowledge Utilization and Self-Evolving Loop in SAGA

Require: Task instance x_t, hierarchical memory $\mathcal{M}$
Generate task-level plan: $p_t \leftarrow \mathrm{Plan}_{\mathrm{LLM}}(x_t)$
Retrieve hierarchical knowledge: $\mathcal{Z}_t \leftarrow \mathrm{Retrieve}(\mathcal{M}, x_t, p_t)$
Instantiate retrieved principles: $\tilde{\mathcal{Z}}_t^{\mathrm{prin}} \leftarrow \mathrm{Instantiate}(\mathcal{Z}_t^{\mathrm{prin}}, x_t, p_t)$
Initialize trajectory $\tau_t \leftarrow \emptyset$
for each decision step k **do**
  Generate candidate action: $a_k \sim \pi(\cdot|o_k, \tilde{\mathcal{Z}}_t)$
  Evaluate action using principle predicates: $\{\phi_i(a_k)\}_{i=1}^{N_t}$
  if $\exists \phi_i, \phi_i(a_k) = 0$ **then**
    Provide violated principle information for action refinement
    Regenerate action
  end if
  Execute action a_k and collect feedback r_k
  Update trajectory: $\tau_t \leftarrow \tau_t \cup (o_k, a_k, r_k)$
end for
Abstract trajectory following hierarchical abstraction: $\tau_t \rightarrow \mathcal{K}_t$
Update memory: $\mathcal{M} \leftarrow \mathrm{Update}(\mathcal{M}, \mathcal{K}_t)$
return Updated memory $\mathcal{M}$

in Algorithm 1. Different from conventional retrieval-based agents that directly append retrieved information into the context, SAGA explicitly adapts retrieved knowledge to the current task and incorporates it into both decision making and knowledge evolution.

The process begins with task-aware knowledge retrieval. The agent first generates a task-level plan $p_t = \mathrm{Plan}_{\mathrm{LLM}}(x_t)$, which provides additional semantic information about the task objective, intermediate requirements, and potential execution structure. The task description and the generated plan are then used to retrieve relevant knowledge from the hierarchical memory:

$$\mathcal{Z}_t = \mathrm{Retrieve}(\mathcal{M}, x_t, p_t),$$

where $\mathcal{Z}_t$ contains task-relevant knowledge from different abstraction levels introduced in Section 3.2. To enable effective utilization of high-level knowledge, SAGA performs task-conditioned principle instantiation:

$$\tilde{\mathcal{Z}}_t^{\mathrm{prin}} = \mathrm{Instantiate}(\mathcal{Z}_t^{\mathrm{prin}}, x_t, p_t).$$

This step grounds abstract experience-derived principles into task-specific guidance by incorporating the current objective and environmental context. The resulting knowledge context $\tilde{\mathcal{Z}}_t$ is then provided to the decision module for subsequent execution.

During execution, SAGA integrates retrieved knowledge into policy generation and principle-level behavioral regulation. At each decision step k, the policy generates a candidate action $a_k \sim \pi(\cdot|o_k, \tilde{\mathcal{Z}}_t)$ based on the current observation and instantiated knowledge context. To make high-level principles executable, SAGA compiles the instantiated L_3 experience-grounded principles into executable predicates:

$$\mathcal{P}_t = \{\phi_i\}_{i=1}^{N_t}, \qquad \phi_i(a_k) \in \{0, 1\},$$

where each predicate evaluates whether the generated action satisfies the corresponding principle constraint. The candidate action is accepted when all applicable predicates are satisfied. When a violation is detected, SAGA identifies the violated principles and provides their semantic information as feedback for action refinement. Specifically, the violated principles are incorporated into the subsequent decision context: $\tilde{o}_k = (o_k, \{\phi_i | \phi_i(a_k) = 0\})$, which guides the policy to resample an alternative action that better aligns with experience-grounded constraints. This mechanism enables abstract principles to regulate low-level decisions without replacing the underlying generative policy, achieving a balance between knowledge-based guidance and behavioral flexibility.

After task completion, the collected trajectory $\tau_t = \{(o_k, a_k, r_k)\}_{k=1}^{T}$ is incorporated into the knowledge evolution process. Instead of storing the trajectory as an isolated experience, SAGA

Table 1: Main results on ScienceWorld. † denotes prompt-based methods and ∗ denotes training-based methods. Best and second-best performances are highlighted in **bold** and underlined, respectively.

Adaptation Paradigm	Method	Measure	Test-Cond	Find	Chem-Mix	Lifespan	Avg. ↑
Prompt-based	GPT-5.1†	32.1	46.5	41.8	69.6	38.3	43.0
	Claude-4-Sonnet†	45.4	54.7	60.3	45.6	90.0	57.4
	Gemini-2.5-Flash†	40.7	51.3	53.7	33.0	50.0	49.6
	Qwen2.5-7B-Instruct†	2.1	12.3	21.2	9.6	25.0	13.9
	ReAct†	12.2	14.4	33.6	9.6	28.9	20.0
	Ours (DeepSeek-flash)†	25.1	58.1	67.7	24.6	80.0	56.2
	Ours (Qwen3.8-Max)†	**55.9**	**62.2**	**68.0**	**100.0**	**100.0**	**67.0**
Training-based	AgentGym-RL*	24.5	59.0	65.9	0.0	66.7	50.5
	ScalingInter*	34.0	55.4	**88.6**	0.0	73.3	57.0
	PPO*	1.8	57.5	60.9	6.9	97.8	51.4
	GRPO*	31.2	49.8	32.7	14.7	53.3	41.8
	StraTA*	**55.7**	54.4	78.6	16.3	**100.0**	63.5
	Ours (SFT)*	26.8	59.1	64.1	**86.6**	83.3	58.1
	Ours (SFT+GRPO)*	46.2	**63.6**	74.5	69.6	**100.0**	**66.9**

Note: Methods are grouped by adaptation paradigm. For training-based variants, $L0$–$L3$ knowledge is incorporated into the model prompt during training.

applies the hierarchical abstraction mechanism introduced in Section 3.2 to extract reusable knowledge: $\tau_t \rightarrow \mathcal{K}_t$. The extracted knowledge is subsequently integrated into memory:

$$\mathcal{M}_{t+1} = \text{Update}(\mathcal{M}_t, \mathcal{K}_t).$$

Successful executions provide additional evidence for reinforcing effective strategies, while failed interactions reveal ineffective behaviors and support knowledge refinement. Through repeated retrieval, execution, and abstraction, SAGA forms a self-evolving knowledge loop, where accumulated experiences are continuously converted into reusable knowledge to improve future task adaptation.

4 Experiments

4.1 Experimental Setup

We evaluate SAGA on two interactive benchmarks, ScienceWorld (Wang et al., 2022) and ALFWorld (Shridhar et al., 2020), which require agents to perform long-horizon reasoning and decision making through continuous environment interaction. We consider two evaluation paradigms: **prompt-based experience evolution**, where the model parameters remain fixed and agent capabilities are improved through experience-grounded knowledge acquisition and utilization, and **training-based adaptation**, where accumulated experiences are further used to optimize model parameters. Unless otherwise specified, training-based methods (∗) use Qwen2.5-7B-Instruct (Qwen et al., 2025) as the base model. For prompt-based methods (†), we report results with the corresponding backbone models specified in each experiment.

We compare SAGA with representative baselines from three categories. The first category consists of reasoning-based agents, including ReAct (Yao et al., 2022). The second category contains prompt-based experience augmentation approaches, including ExpeL (Zhao et al., 2024), Memento (Zhou et al., 2025), and ReasoningBank (Ouyang et al., 2026). The third category includes training-based adaptation methods, including reinforcement learning approaches such as GRPO (Shao et al., 2024) and SkillRL (Xia et al., 2026), as well as recent self-evolving frameworks such as StraTA (Xue et al., 2026) and Evolving-RL (Fan et al., 2026). For ScienceWorld, we report the final task score averaged over evaluation instances, while for ALFWorld we follow the standard protocol and report task success rates.

4.2 Main Results

ScienceWorld Results. ScienceWorld evaluates an agent's capability to perform long-horizon scientific reasoning and procedural manipulation through interactive environments. We investigate whether experience-grounded knowledge evolution enables agents to acquire and reuse transferable knowledge. Table 1 summarizes the results under both prompt-based and training-based adaptation

Evolution trajectory during GRPO optimization. To further illustrate how accumulated experience contributes to continual agent improvement, we visualize the performance trajectory during GRPO optimization in Fig. 4. The selected checkpoint improves the final evaluation performance from the SFT initialization to 66.9%. Although intermediate updates exhibit fluctuations due to the stochastic nature of reinforcement learning optimization, the overall trend demonstrates continuous refinement of agent capabilities through experience-guided parameter adaptation.

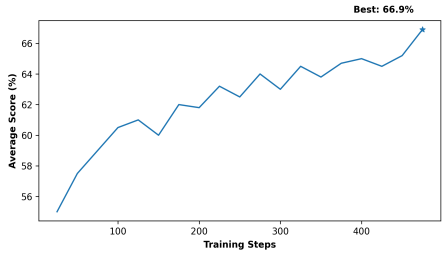

Figure 4: Performance evolution during GRPO optimization in ScienceWorld.

Table 2: Main results on ALFWorld. † denotes prompt-based methods and ∗ denotes training-based methods. Best performances on the overall success rate and unseen tasks (Look/Pick2) are highlighted in **bold**.

Adaptation Paradigm	Method	Pick	Clean	Heat	Cool	Look	Pick2	Avg. ↑
Prompt-based	Base Model†	75.0	57.4	13.0	60.0	18.9	36.5	45.5
	ReasoningBank†	75.0	45.2	21.7	47.6	11.1	35.3	41.1
	ExpeL†	21.0	55.0	52.0	71.0	67.0	6.0	46.3
	Memento†	79.2	64.5	39.1	76.2	0.0	41.2	53.0
	Ours (Qwen3.8-Max, no memory)†	100.0	96.8	87.0	95.2	**100.0**	**100.0**	96.3
	Ours (Qwen3.8-Max, SAGA)†	100.0	100.0	100.0	100.0	**100.0**	94.1	**99.3**
Training-based	GRPO*	96.7	90.3	100.0	100.0	53.3	12.9	79.9
	GRPO (w/ skills)*	95.8	96.1	98.3	98.1	61.1	27.1	83.3
	SkillRL*	93.3	95.2	72.2	80.0	81.2	55.6	81.7
	Evolving-RL (w/o skills)*	94.2	96.8	99.1	100.0	**100.0**	61.2	93.1
	Evolving-RL (w/ skills)*	97.5	97.4	100.0	100.0	95.6	**81.2**	96.0
	Ours (SFT)*	100.0	100.0	87.0	95.2	88.9	58.8	90.3
	Ours (SFT + GRPO)*	95.8	100.0	91.3	95.2	94.4	76.5	93.3
	Ours (SFT + GRPO + External)*	100.0	100.0	95.7	100.0	**100.0**	76.5	**96.3**

Note: Results with "skills" or "SFT + GRPO + External" is additionally allowed to collect skills from the unseen training subset.

paradigms. For prompt-based methods, SAGA consistently improves over conventional prompting and direct trajectory retrieval. Using DeepSeek-flash as the backbone, SAGA improves the average score from 25.0% without memory and 54.8% with trajectory retrieval to 56.2% through hierarchical knowledge evolution, demonstrating the effectiveness of progressively abstracting concrete experiences into reusable knowledge representations. Furthermore, with the stronger Qwen3.8-Max backbone, SAGA achieves an average score of 67.0%, indicating that the proposed experience-grounded abstraction framework generalizes across different backbone models. For training-based adaptation, SAGA further exploits accumulated experiences for parameter optimization. Starting from the Qwen2.5-7B-Instruct backbone (9.6%, get by ourself), supervised fine-tuning improves the average score to 58.1%, and subsequent GRPO optimization further increases the performance to 66.9%, achieving the best overall performance among all compared methods. These results demonstrate that experience-grounded knowledge evolution provides a unified framework for improving LLM agents through both inference-time knowledge utilization and parameter-level adaptation.

ALFWorld Results. ALFWorld evaluates an agent's capability to execute long-horizon embodied instructions in interactive household environments. Table 2 summarizes the performance under both prompt-based and training-based adaptation paradigms. For prompt-based methods, experience-grounded knowledge evolution consistently improves agent performance by transforming interaction trajectories into reusable knowledge representations. With the Qwen2.5-7B-Instruct backbone, memory augmentation improves the success rate from 45.5% to 57.5%, demonstrating the effectiveness of structured experience utilization over direct prompting. Furthermore, for the stronger Qwen3.8-Max backbone, SAGA achieves 99.3% overall success rate, indicating that hierarchical knowledge evolution can effectively complement the capability of stronger foundation models.

For training-based adaptation, SAGA further incorporates accumulated experiences into parameter optimization. Starting from the SFT-based model, SAGA achieves a success rate of 90.3%, while subsequent GRPO optimization further improves the performance to 93.3%. With additional external test-time adaptation, the performance increases to 96.3%, achieving competitive performance with methods that explicitly leverage additional skills. In particular, SAGA maintains strong perfor-

Table 3: Ablation study on knowledge abstraction and utilization on ScienceWorld.

Configuration	Qwen3.8-Max	DeepSeek-v4-Flash
R1: No memory	48.4	25.0
R2: Raw trajectory retrieval (L_0)	63.7	54.8
R3: Episodic + procedural knowledge ($L_1 + L_2$)	64.5	55.0
R4: $L_1 + L_2 + L_3$, direct injection	63.9	54.5
R5: $L_1 + L_2$ + instantiated L_3	66.2	55.8
R6: R5 + principle-based action regulation	**67.0**	**56.2**

mance on unseen tasks, demonstrating the potential of experience-grounded knowledge evolution for improving generalization in long-horizon embodied decision making.

4.3 Ablation Studies

We conduct ablation studies on ScienceWorld to examine three components of SAGA: hierarchical knowledge abstraction, task-conditioned principle instantiation, and principle-based action regulation. Table 3 compares six configurations using Qwen3.8-Max and DeepSeek-v4-Flash, progressively introducing different knowledge representations and utilization mechanisms.

Effect of hierarchical knowledge abstraction. Retrieving raw trajectories (R2) substantially improves performance over memory-free execution (R1), increasing the score from 48.4 to 63.7 with Qwen3.8-Max and from 25.0 to 54.8 with DeepSeek-v4-Flash. These improvements establish direct experience reuse as a strong reference configuration. Replacing raw trajectories with episodic and procedural knowledge (R3) yields further gains of 0.8 and 0.2 points, respectively. However, directly adding principle-level knowledge (R4) decreases the score relative to R3 for both backbones. This pattern indicates that more abstract knowledge does not automatically translate into better execution, motivating mechanisms that explicitly adapt principles to the current task.

Effect of contextual instantiation. The comparison between R4 and R5 examines the contribution of task-conditioned principle instantiation. Converting retrieved principles into operational guidance improves the score from 63.9 to 66.2 with Qwen3.8-Max and from 54.5 to 55.8 with DeepSeek-v4-Flash. Moreover, R5 exceeds the episodic-plus-procedural configuration (R3) by 1.7 and 0.8 points. Thus, principle-level knowledge provides additional value when its applicability and operational implications are grounded in the current task. The consistent ordering across both backbones supports the importance of contextual instantiation for effective utilization of abstract knowledge.

Effect of principle-based action regulation. Adding action regulation to instantiated guidance (R6) further increases the score to 67.0 with Qwen3.8-Max and 56.2 with DeepSeek-v4-Flash, corresponding to gains of 0.8 and 0.4 points over R5. These incremental improvements are consistent with the role of principle-conditioned feedback in refining candidate actions during execution. The complete configuration also outperforms raw trajectory retrieval by 3.3 and 1.4 points, respectively. Together, the comparisons support a complementary role for contextual instantiation and action regulation in converting experience-derived principles into effective decision guidance.

5 Conclusion

In this work, we study continual agent evolution through experience-grounded knowledge abstraction and introduce SAGA, a framework that converts interaction trajectories into progressively more reusable forms of knowledge. By organizing experience into episodic descriptions, reusable procedures, and principles with explicit applicability conditions, SAGA enables agents to move beyond task-specific memory while preserving links between abstract knowledge and its supporting execution evidence. SAGA further closes the loop between knowledge acquisition and action by instantiating retrieved principles into task-specific guidance and using principle-derived checks to regulate subsequent decisions. Experiments on ScienceWorld and ALFWorld show consistent performance improvements, while ablations confirm the importance of both contextual instantiation and action regulation for realizing the benefits of principle-level knowledge. These results suggest that continual improvement in LLM agents depends not only on retaining past experience, but also on systematically abstracting, grounding, and operationalizing that experience for future interaction.

AI USE STATEMENT

In this work, generative AI tools are used to assist with manuscript preparation, including language polishing, improving clarity and readability, and refining the organization and presentation of technical descriptions. These tools are not used to independently conduct experiments, fabricate experimental results, or determine the scientific conclusions of this work. All AI-assisted content is manually reviewed and revised by the authors, with technical statements checked against the implemented methodology, experimental records, and cited literature. Generative models that constitute components of the proposed framework are described separately in the Method and Experiment sections.

ETHICAL CONSIDERATIONS

We will abide by the laws, rules, and regulations of our community, school, work, and country. We will conduct ourselves with integrity, fidelity, and honesty. We will openly take responsibility for our actions and only make agreements that we intend to keep. All data used in this study are intended for research purposes. No personally identifiable information (PII) was collected. The dataset construction and data collection protocol have been reviewed and approved by our organization's internal Ethics Review Board, and all collected data are used solely for research purposes without collecting personally identifiable information. For existing artifacts used in this work, including tools, benchmarks, APIs, and model resources, we follow their stated licenses, access conditions, and terms of use. Any artifacts created in this work, including benchmark data, rubrics, prompts, and evaluation scripts, are intended for research use only. Their use and distribution will be made consistent with the original access conditions of the underlying resources, and derivatives of research-only data will not be used outside research contexts.

REPRODUCIBILITY

To facilitate reproducibility, we provide an anonymized version of our implementation and experimental resources in the supplementary material. The released materials include the core components required to reproduce the reported results. Following the review process, we will publicly release the complete codebase and associated resources to further support research and development in this direction.

REFERENCES

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pp. 287–318. Pmlr, 2023.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, 2026.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631, 2024.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Zhiyuan Fan, Wenwei Jin, Feng Zhang, Bin Li, Yihong Dong, Yao Hu, and Jiawei Li. Evolving-rl: End-to-end optimization of experience-driven self-evolving capability within agents. *arXiv preprint arXiv:2605.10663*, 2026.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents, 2024. URL https://arxiv.org/abs/2403.08978.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization, 2023. URL https://arxiv.org/abs/2310.10134.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Xiangyuan Xue, Yifan Zhou, Zidong Wang, Shengji Tang, Philip Torr, Wanli Ouyang, Lei Bai, and Zhenfei Yin. Strata: Incentivizing agentic reinforcement learning with strategic trajectory abstraction. *arXiv preprint arXiv:2605.06642*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Bowen Ye, Yongchao Xu, Zhijian Li, Xiang Yin, Junkai Ma, and Wenzhao Li. Coevo-mem: Co-evolving retrieval policy and memory bank for llm agents. *arXiv preprint arXiv:2608.01739*, 2026.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026a.

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*, 2026b.

Xing Zhang, Guanghui Wang, Yanwei Cui, Wei Qiu, Ziyuan Li, Bing Zhu, and Peiyang He. Experience compression spectrum: Unifying memory, skills, and rules in llm agents. *arXiv preprint arXiv:2604.15877*, 2026c.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8):1–35, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

A HIERARCHICAL MEMORY CONSTRUCTION

This section formalizes the persistent representation, hierarchical extraction pipeline, and incremental update operators of memory $\mathcal{M}$, complementing the framework-level specification in Algorithm 1.

Schema and persistence. Each level of $\mathcal{M}$ is maintained as an append-only store with an associated embedding index. An L_0 node preserves the raw trajectory τ, including stepwise observations, actions, environment feedback, and the terminal outcome returned by the environment; this authoritative outcome is never replaced by a model-generated summary. An L_1 node encodes the decontextualized goal, key actions, outcome, and obstacle associated with one trajectory, thereby preserving a one-to-one correspondence with its L_0 evidence. An L_2 node represents a transferable behavioral pattern and an ordered, multi-step strategy supported by multiple episodic records rather than by an isolated action. An L_3 node specifies a condition, effect, mechanism, scope, and set of exceptions. Such nodes are interpreted as evidence-grounded explanatory hypotheses, rather than as verified causal laws. Bidirectional parent-child links connect adjacent levels, with each lower-level node assigned to at most two parents to preserve a sparse and auditable evidence graph. Source metadata records the environment, task, variant, split, and round, providing end-to-end traceability from each principle to its supporting trajectories. Retrieval uses Qwen3-0.6B-embedding with cosine similarity. The indexed representation comprises the goal field at L_1, the pattern and strategy fields at L_2, and the scope, condition, effect, and mechanism fields at L_3. Because the index is fully reconstructible from the persistent stores, it does not constitute an independent knowledge source.

Cold start and hierarchical extraction. Memory initialization relies exclusively on memory-free interactions with training tasks and introduces no human-authored seed knowledge. At each subsequent round, the actor interacts with the environment under the previously frozen memory state, collects additional training trajectories, and incrementally updates the inherited store. An LLM transforms each eligible raw trajectory into an L_1 record while preserving distinctions among successful outcomes, task failures, and encountered obstacles. Episodes containing no executed action are excluded, whereas API failures are recorded separately from environment-level failures. For L_2 induction, records are grouped by task and outcome, and each group is consolidated into a compact set of complete procedures with explicit citations to supporting cases. Lexical similarity alone is not treated as evidential support. For L_3 construction, the builder jointly considers related procedures and their supporting evidence to infer a shared condition, effect, mechanism, scope, and set of exceptions. Procedures for which this abstraction is insufficiently supported remain at L_2. Cross-task merging uses semantic similarity with a threshold of 0.75 to generate candidate procedure clusters. A merged principle is admitted only when its evidence spans at least two task types; similarity is used for candidate generation, not as evidence of a shared mechanism.

Incremental update. Algorithm 2 specifies the update operator applied to the current memory and newly collected training trajectories. The selection stage retains at most one trajectory per task variant, ranked lexicographically by terminal score, peak score, and recency. Ties are resolved in favor of the more recent round without removing provenance, and trajectories rejected at this stage do not participate in the current abstraction update. ScienceWorld admits trajectories with a positive peak score. ALFWorld additionally admits executed failures, including zero-score episodes, as negative evidence for actions or strategies that should be avoided. Each newly induced procedure is first matched to existing principles for the same task. Only unmatched procedures are considered for principle lifting, reducing redundant abstraction. Candidate principles are deduplicated, and semantically equivalent candidates are consolidated in proportion to the support supplied by their underlying evidence. Principles that fail validation are recorded in an exclusion registry, preventing semantically equivalent candidates from being reintroduced in subsequent rounds. After each update, the resulting memory is frozen for evaluation. Evaluation trajectories are excluded from all subsequent construction stages. For each round, we record trajectory admissions, replacements, and rejections; L_2 and L_3 counts; principle reuse; cross-task evidence coverage; validation outcomes; and model cost. These structural statistics characterize memory evolution but do not independently establish performance improvement.

The benchmark admission rule in the selection stage retains trajectories with a positive peak score for ScienceWorld, whereas for ALFWorld it additionally retains executed failures, including zero-score episodes, as negative evidence.

Algorithm 2 Incremental Memory Update

Require: Current memory $\mathcal{M} = \{L_0, L_1, L_2, L_3\}$; new training trajectories $\mathcal{T}$; exclusion registry $\mathcal{R}$; merge threshold θ=0.75

Initialize selected set $\mathcal{T}^\star \leftarrow \emptyset$
for each task variant v **do**
  Rank trajectories of v by terminal score, then peak score, then recency
  Add the top trajectory admitted by the benchmark rule to $\mathcal{T}^\star$
end for
Append $\mathcal{T}^\star$ to L_0 and summarize each trajectory into an episodic record in L_1
Group the new L_1 records by task and outcome; induce candidate procedures with supporting-case citations
for each candidate procedure c **do**
  if c matches an existing same-task principle **then**
    Attach c as supporting evidence without lifting a new principle
  else
    Lift c to a candidate principle
  end if
end for
Deduplicate candidates; merge a pair with similarity $\geq\theta$ only when its evidence spans $\geq$2 task types
Remove candidates whose normalized form already appears in $\mathcal{R}$
Validate survivors (ALFWorld: evidence grounding; ScienceWorld: utility ledger)
Commit validated procedures and principles to L_2 and L_3; append rejected candidates to $\mathcal{R}$
Freeze the updated memory $\mathcal{M}'$ and exclude evaluation trajectories from future construction
return Updated memory $\mathcal{M}'$

Table 4: Input-output specification of the LLM-based components in SAGA.

Component	Input	Output
Planning	Task description	One-sentence strategy with a scalar confidence
L_1 summarization	Raw trajectory	Goal, key actions, outcome, obstacle
L_2 induction	Grouped L_1 records with trajectories	Pattern, strategy, supporting-case citations
L_3 lifting	Related L_2 procedures with evidence	Condition, effect, mechanism, scope, exceptions
Cross-task merge	Candidate procedure clusters	Merged principle with multi-task evidence
Instantiation	Principle, task, scene hints	One operational constraint or NOT APPLICABLE
Predicate compilation	Principle	Executable check $\phi_{\mathcal{P}}$ for action regulation

Evidence validation. The two benchmarks employ benchmark-specific quality-control procedures targeting complementary notions of validity. For ALFWorld, an evidence-grounding checker verifies citation syntax, source integrity, realized parent–child links, and consistency between each candidate principle and its supporting trajectories. A cross-task principle must be supported by evidence from at least two task types; textual expansion of its stated scope is insufficient. An API fault prevents the affected stage from being committed and is counted as neither a candidate rejection nor an environment-level task failure. For ScienceWorld, a utility ledger compares paired rollouts with and without knowledge injection on training variants, indexed by task and recalled set. The ledger supports collection screening, leave-one-out diagnosis, and pruning, but applies only to experiments in which utility screening is explicitly enabled; earlier batches are not retrospectively classified as ledger-validated. Evidence grounding evaluates whether cited trajectories support a principle, whereas the utility ledger estimates the empirical effect of deploying that principle. These criteria are therefore reported separately.

Prompt specification. Table 4 summarizes the input-output contract of each LLM-based component. Complete prompt templates, implementation defaults, and entry-point arguments are provided in the anonymized supplementary code.

Table 5: ScienceWorld evaluation configuration (strict alignment with StraTA (Xue et al., 2026)).

Parameter	Value
Evaluation set	strata175: StraTA official test split (200 items) filtered by 14 available levels
Composition	175 unique (task, variation) items, 11 tasks, 5 categories
Category weights	Measure 28, Test-Cond 83, Find 44, Chem-Mix 5, Lifespan 15
Step limit	20
Interaction history	Most recent 2 steps, no length-based episode termination
Admissible-action list	Not provided
Metric	Terminal-step score, item mean $\times 100$
Seed	42

B Knowledge Retrieval and Utilization

Planning and retrieval. Given a task instance x_t, the actor produces a one-sentence plan p_t together with a scalar confidence estimate. The concatenation of x_t and p_t forms the retrieval query; the plan conditions retrieval without superseding task-specific evidence. The default direct-retrieval path matches L_1 records by goal, L_2 procedures by pattern and strategy, and L_3 principles by scope, condition, effect, and mechanism, retaining the top $k{=}3$ items at each level. Plan confidence determines the budget for principle-level knowledge. No principle is injected when confidence is at least 0.8; at most one is injected for confidence in $[0.5, 0.8)$; and at most three are injected otherwise. Unparseable confidence outputs are assigned the conservative maximum budget of three principles. We retain two alternatives for ablation: an evidence path that aggregates scores upward through parent links and a chain path that propagates scores from L_1. Each run records the allocated budget and the numbers of retrieved and injected items. The logs distinguish four outcomes that would otherwise be conflated in aggregate statistics: knowledge deemed unnecessary, not retrieved, filtered, or instantiated as NOT APPLICABLE.

Instantiation and presentation. Each retrieved principle is instantiated once per episode. Conditioned on the principle's mechanism field, the current task, and available scene information, the instantiator produces a single operational constraint; the output NOT APPLICABLE suppresses injection of that principle. The resulting context comprises three labeled blocks: `[Similar case]` supplies episodic evidence and its outcome, `[Procedure]` specifies the complete strategy, and `[Mechanism]` provides the instantiated constraint. The accompanying instruction characterizes retrieved knowledge as applicability-dependent guidance rather than as a prescriptive action sequence. Explicit task instructions retain precedence over memory content. During prompt-based execution, SAGA supplies the actor with retrieved L_1–L_3 knowledge while retaining L_0 as provenance. Raw L_0 trajectories are rendered only in the designated trajectory-retrieval control. The reported training-based variants use the corresponding L_0–L_3 context in their training prompts, as specified in the main tables.

Principle-compiled action regulation. Principles identified as harmful are compiled into executable predicates $\mathcal{P}$. A candidate action a_t is accepted by the regulator only if $\phi_{\mathcal{P}}(a_t){=}1$. A predicate violation initiates bounded, principle-conditioned resampling for at most three attempts. If no sampled candidate satisfies the predicate, the actor executes the original action. This soft intervention preserves the support of the generative policy while supplying principle-level corrective feedback. Successful compilation alone does not establish predicate validity. We therefore audit predicate polarity, empirical trigger frequency, and false-penalty risk separately; principles labeled as beneficial or neutral are never repurposed as negative constraints.

C Evaluation Protocols and Optimization Details

The two benchmarks share the hierarchical memory representation and selected extraction components, while adopting benchmark-specific interaction protocols and evaluation metrics. Tables 5 and 6 define the corresponding evaluation configurations; Table 7 identifies the actor and builder used in each reported setting.

ScienceWorld. Following the reference evaluation harness, we report the terminal-step score rather than the maximum intermediate score. All SAGA rows in Table 1 use the strict configuration in

Table 6: ALFWorld evaluation configuration. Dataset composition and success accounting follow Evolving-RL (Fan et al., 2026); the interaction protocol is reported separately for each model family.

Parameter	Value
Evaluation set	`valid_unseen`, 134 unique games
Task types	Pick 24, Clean 31, Heat 23, Cool 21, Look 18, Pick2 17
Seen / unseen split	Four seen types (99 games); Look and Pick2 unseen (35 games)
Training pool	2,432 games: Pick 790, Clean 650, Heat 459, Cool 533
Step limit	50
Strict 7B protocol	Token-prefix continuation; 16,384-token context
Strong / training protocol	Multi-turn chat; 16,384-token base context
Extended-context rows	32,768-token context, explicitly marked in the reported configuration
Decoding	At most 512 tokens per step; temperature 0, top-p 1, seed 42
Action list	Environment order; exact `help` excluded
Metric	Environment-reported success rate $\times 100$

Table 7: Actor and builder assignments for the reported configurations. The actor denotes the execution policy or the policy being optimized; the builder constructs abstractions and knowledge-conditioned training context.

Configuration	Actor (execution)	Builder (abstraction)
ScienceWorld, prompt-based	DeepSeek-flash	DeepSeek-flash
ScienceWorld, prompt-based	Qwen3.8-max-0902	Qwen3.8-max-0902
ScienceWorld, training-based	Qwen2.5-7B-Instruct	Qwen3.8-max-0902 (offline)
ALFWorld, prompt-based (strict)	Qwen2.5-7B-Instruct (fixed)	Qwen3.8-max-0902 (offline)
ALFWorld, prompt-based (strong)	Qwen3.8-max-0902	Qwen3.8-max-0902
ALFWorld, training-based	Qwen2.5-7B-Instruct	Qwen3.8-max-0902 (offline)

Table 5, matching StraTA in evaluation set, step limit, history window, and admissible-action exposure. Results for comparison methods are reproduced from the values reported by Xue et al. (2026) under their stated protocol. The no-memory Qwen2.5-7B-Instruct score of 13.9 in Table 1 is likewise taken from Xue et al. (2026). Under our strict implementation of Table 5, the same backbone obtains 9.6 without memory; this independently measured score serves as the initialization reference for the training-based experiments. Prompt-based evolution comprises six rounds with DeepSeek-flash (deepseek-v4-flash) and seven rounds with Qwen3.8-Max (qwen3.8-max-0902); the reported DeepSeek-flash result uses the round-5 memory state. At each round, trajectories are collected exclusively from training tasks, the memory is updated and frozen, and the resulting state is subsequently evaluated on strata175.

ALFWorld. All evaluation scenes are drawn from `valid_unseen`. Here, "seen" refers to task types represented in the training pool, rather than to evaluation scenes encountered previously. No evaluation scene contributes to trajectory collection or memory construction. Look and Pick2 remain zero-shot in every non-External configuration: neither task type contributes training trajectories, construction evidence, or test-time adaptation data. The explicitly marked External variant relaxes only the prohibition on test-time adaptation. The fixed-7B strict experiments follow the official solver protocol (reference commit `7ed3d5867b19dbf21a5ddccfea7b77dccfb5346f`; protocol `alf-official-token-v2`). The chat template is applied once, actions are generated by token-prefix completion, outputs ending with `finish_reason=length` are not executed, and actions receive no automatic correction. The Qwen3.8-Max and training-based experiments instead follow the separately specified multi-turn chat protocol. Consequently, comparisons across these protocol families are not interpreted as controlled estimates of model-only effects. At every evolution round, trajectory collection is restricted to the training pool, and the updated memory is frozen before evaluation on all 134 games. In the strict 7B experiments, the no-memory baseline uses online concurrency 32, whereas evolution uses online concurrency 8 and offline-builder concurrency 8. Because serving schedules may affect token-level execution, this difference is reported as a potential implementation-level confound.

Model roles and attribution boundaries. In both training-based settings, Qwen2.5-7B-Instruct is the optimized policy, whereas Qwen3.8-max-0902 acts as the offline builder of hierarchical abstractions and knowledge-conditioned training context. The strict ALFWorld configuration applies the same division of labor while holding the online 7B actor fixed; we refer to this setting as API-assisted memory evolution. Prompt constraints and evidence validation reduce unsupported abstractions but

Table 8: ScienceWorld supervised fine-tuning corpus. Samples are stepwise action targets reconstructed from replayed prompt-based evolution trajectories.

Trajectory source	Samples
Full-score trajectories (stepwise actions)	8,788
Zero-final-score trajectories (prefix to first scoring peak)	4,882
Partial-score trajectories (peak prefix)	9,292
Total	22,962

cannot isolate contributions from the builder's pretrained knowledge. The reported effects therefore characterize the complete actor-builder pipeline rather than a same-model abstraction control.

External test-time adaptation. The SFT + GRPO + External row in Table 2 permits test-time skill acquisition on the two unseen task types, consistent with the evaluation boundary of Evolving-RL (w/ skills). We report this configuration separately because it is not a zero-shot result.

Supervised fine-tuning. The supervised training pipeline explicitly separates trajectory collection, offline abstraction, and policy optimization. For ScienceWorld, trajectories collected during prompt-based evolution are replayed to reconstruct stepwise observations and action targets. Qwen3.8-max-0902 supplies offline abstractions and knowledge context, whereas Qwen2.5-7B-Instruct is optimized as the execution policy. The training corpus comprises three sample categories, summarized in Table 8: stepwise actions from full-score trajectories (8,788 samples), prefixes of zero-final-score trajectories truncated at the first scoring peak (4,882 samples), and peak prefixes from partial-score trajectories (9,292 samples). Because these inclusion criteria depend on observed scores, they do not imply semantic optimality of every retained action. Knowledge-conditioned examples append the retrieved L_0–L_3 context to their inputs, consistent with Table 1. The auxiliary instantiation objective maps a principle, task, and scene to an operational constraint. For ALFWorld, the teacher corpus consists of successful Qwen3.8-Max trajectories from the four seen task types. The corpus is partitioned into training and held-out validation subsets for full-parameter fine-tuning of Qwen2.5-7B-Instruct. The reported SFT variant is knowledge-conditioned; neither Look nor Pick2 contributes supervised trajectories.

GRPO optimization. Table 9 lists the optimization configuration for both training-based arms. Starting from the SFT checkpoint, we optimize a LoRA adapter using group-relative policy optimization (GRPO). Each training item generates eight trajectories, and variable-length step samples are pooled within each group without alignment by step index. The reported ScienceWorld configuration uses a dense reward defined as the per-step score increment, without future-return accumulation. The ALFWorld configuration uses shaped rewards derived from environment success feedback. Trajectory-level terminal-score sharing, rule-based penalties, and principle-predicate penalties are retained as control variants. Step-level advantages are computed from length-normalized log-probabilities of generated tokens. In the dense-reward setting, rewards are normalized by their within-group root-mean-square without mean subtraction, thereby preserving zeros and reward signs. The trajectory-level control instead employs group-centered advantages. Each sampling batch induces a single parameter update, with no PPO-style multi-epoch reuse. The KL penalty relative to the frozen SFT reference has coefficient 0.02, advantages are clipped to magnitude 3, the learning rate is 5×10^{-6}, the maximum sequence length is $16{,}384$ tokens, and evaluation occurs every 25 updates. The ScienceWorld arm uses a 1,890-item training pool, two epochs, and 474 updates with 30,240 sampled episodes. The ALFWorld arms draw only from the 2,432-game pool of four seen task types; Look and Pick2 remain absent from parameter optimization.

Post-training results. On ScienceWorld, GRPO optimization raises the average score to 66.9 on strata175. On ALFWorld, the main table reports each stage under its matched evaluation configuration. Knowledge-conditioned SFT obtains 90.3%, SFT followed by GRPO obtains 93.3%, and the separately marked External variant obtains 96.3%. GRPO uses only the four seen task types. Look and Pick2 remain unseen during parameter optimization; only the External variant permits test-time skill acquisition on these types.

Table 9: Optimization configuration for the training-based variants. GRPO initializes from the SFT checkpoint and optimizes a LoRA adapter; entries marked — are not applicable to the corresponding arm.

Stage / Parameter	ScienceWorld	ALFWorld
SFT scheme	Full-parameter	Full-parameter
Base model	Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct
GRPO adapter	LoRA	LoRA
Group size (trajectories per item)	8	8
Reward	Dense per-step score change	Shaped success feedback
Advantage estimation	Within-group RMS, no mean subtraction	Within-group RMS, no mean subtraction
KL coefficient	0.02	0.02
Advantage clip	3	3
Learning rate	5×10^{-6}	5×10^{-6}
Max sequence length	16,384	16,384
Evaluation interval	Every 25 updates	Every 25 updates
Training pool	1,890 items	2,432 games (four seen types)
Epochs / updates	2 / 474	—
Sampled episodes	30,240	—

Table 10: Prompt-based evolution on ScienceWorld (strata175, strict configuration). All conditions are evaluated on identical items. The DeepSeek-flash row reports the round-5 checkpoint of a six-round evolution batch.

Backbone	Condition	Avg. score	Evolution rounds
DeepSeek-flash	No memory	25.0	—
	Trajectory retrieval (L_0)	54.8	—
	SAGA (L_0–L_3)	56.2	6
Qwen3.8-max-0902	No memory	48.4	—
	Trajectory retrieval (L_0)	63.7	—
	SAGA (L_0–L_3)	67.0	7

D Extended Results and Statistical Analyses

Prompt-based evolution on ScienceWorld. Table 10 supplements the ablation results in Table 3 with the evolution schedule for the no-memory, trajectory-retrieval, and complete-SAGA configurations. With DeepSeek-flash, the reported round-5 checkpoint from a six-round batch obtains 56.2, compared with 25.0 without memory and 54.8 with direct trajectory retrieval. With Qwen3.8-Max, the corresponding values are 48.4, 63.7, and 67.0 after seven evolution rounds. Both backbones exhibit the same rank ordering. Relative to trajectory retrieval, hierarchical memory yields absolute margins of 1.4 points with DeepSeek-flash and 3.3 points with Qwen3.8-Max.

Utilization statistics. Table 11 reports supplementary episode-level paired analyses with Qwen2.5-7B-Instruct, examining the effects of direct knowledge injection and contextual instantiation. Directly injecting the same L_3 knowledge yields a statistically significant negative effect, whereas contextual instantiation yields a statistically significant positive effect; the isolated instantiation contrast quantifies this reversal. Under instantiation, the estimated gain from principle-level knowledge is approximately three times that from procedural knowledge ($+0.0365$ versus $+0.0127$). This analysis uses a harness distinct from the strict configuration in Table 5. Because more than 80% of episodes obtain zero terminal score under the 7B backbone, the paired estimates characterize the effect of knowledge utilization within this evaluation setting. They are reported as a separate supplementary analysis and are not directly comparable to the dual-backbone ablation scores in Table 3.

Generalization to held-out ALFWorld task types. The ALFWorld protocol evaluates generalization to task types excluded from experience collection and parameter optimization. The training pool contains Pick, Clean, Heat, and Cool, whereas Look and Pick2 are held out, comprising 18 and 17 evaluation games, respectively. In every non-External configuration, neither held-out task type contributes trajectories to memory construction, supervised fine-tuning, or GRPO optimization; the memory is frozen before evaluation. The Look and Pick2 columns in Table 2 therefore report performance on held-out task types within the same environment, while all evaluation scenes are drawn from `valid_unseen`. The External variant is reported separately because it permits test-time skill acquisition on these types and is not a zero-shot configuration.

Table 11: Paired utilization analysis on ScienceWorld (Qwen2.5-7B-Instruct, 14 tasks, 3 seeds, n=738 episodes, scores on the 0–1 scale).

Comparison	Paired difference	t statistic
Same L_3 knowledge, direct injection vs. no injection	-0.0155	-3.53
Same L_3 knowledge, instantiated vs. no injection	$+0.0365$	$+5.90$
Instantiation effect on L_3 (isolated)	$+0.0520$	$+7.74$
L_2 knowledge, instantiated vs. no injection	$+0.0127$	—
L_3 vs. L_2 under instantiation	$+0.0238$	$+3.50$

Table 12: Round-wise evolution on ALFWorld (`valid_unseen`, 134 games, success rate in %). No principle from the complete 7B configuration passed validation in R1–R3.

Actor	Arm	R0	R1	R2	R3
Qwen3.8-max-0902	SAGA (L_0–L_3)	96.3	97.8	97.8	99.3
Qwen2.5-7B-Instruct	Trajectory only (L_0/L_1)	45.5	46.3	57.5	46.3
	Complete hierarchy	45.5	48.5	54.5	59.4

Round-wise evolution on ALFWorld. Table 12 reports the evolution process across rounds for the strong backbone and the two fixed-7B configurations. With Qwen3.8-Max, SAGA improves the overall success rate from 96.3% at R0 to 99.3% at R3, corresponding to a net gain of four successful games among the 134 evaluation games. Relative to R1 (97.8%), the gain at R3 corresponds to two additional successful games. At R3, all 99 games from seen task types are successfully solved; the only remaining failure corresponds to an unseen Pick2 instance where the context budget is exhausted. No formatting failure is observed in any round.

For the fixed Qwen2.5-7B-Instruct actor, the memory-augmentation improvement from 45.5% to 57.5% discussed in the main text corresponds to the trajectory-only (L_0/L_1) configuration at R2. Its success rates are 45.5%, 46.3%, 57.5%, and 46.3% at R0–R3, respectively. The complete-hierarchy configuration is a separate experimental arm, with success rates of 45.5%, 48.5%, 54.5%, and 59.4% at R0–R3, respectively. It improves across the reported rounds, although its R2 score remains below the trajectory-only control at the same round. Thus, 57.5% denotes the R2 trajectory-only result, whereas 59.4% denotes the R3 result of the complete-hierarchy pipeline; these results should not be attributed to the same configuration. As stated in the table caption, no L_3 principle passed validation in the complete 7B configuration during R1–R3, so its improvement does not establish a contribution from validated principle-level knowledge.

GRPO trajectory on ScienceWorld. Starting from the SFT initialization, the dense-reward configuration raises the average score to 66.9 on strata175. Figure 4 shows the optimization trajectory over the course of training. The supplementary material retains category-level scores and counts of perfect- and zero-score episodes.

Mechanism of the improvement. Across the seven Qwen3.8-Max rounds, full hierarchical memory yields 11–14 evaluation episodes with regression from a previously attained score, compared with 12–22 under trajectory retrieval. The corresponding estimated rates of self-induced regression are 0.024 and 0.045, and the round-7 perfect-score counts are 41 and 35, respectively. These diagnostics support the hypothesis that the principal observed benefit of knowledge utilization is reduced execution-time score regression, rather than an increased attainable score ceiling. This interpretation is consistent with the constraint-oriented mechanism described in Appendix B.

E Case Studies, Failure Analysis, and Limitations

Knowledge formation and transfer in ScienceWorld. Figure 3 presents the complete abstraction chain for a pigment-mixing task. Raw trajectories in which yellow and blue pigments produce green are summarized as episodic records, consolidated into a procedure for pigment selection, combination, and outcome verification, and lifted into a principle whose condition specifies the mixing of two primary pigments, whose effect is the generation of the corresponding secondary color, and whose mechanism is subtractive color mixing. The functional separation across levels is thereby made explicit: the principle abstracts a regularity that holds beyond any single color, whereas the

Table 13: Per-game episode length (steps at termination) before and after strong-backbone evolution in the strict ALFWorld setting. For the top three games, round 1 terminates at the context budget without success and round 3 solves the task; `pick2/17` is solved in round 1 but exhausts the budget thereafter.

Game	Round 1	Round 3	Outcome
`clean/6`	48	17	solved after evolution
`heat/29`	38	15	solved after evolution
`heat/103`	39	13	solved after evolution
`pick2/17`	37	—	regression (budget exhausted)

procedure and episodes retain the operational specificity and evidential grounding of concrete executions. This organization supports transfer without trajectory replay. For a task whose target color differs from the one observed during construction, such as orange rather than green, the principle is instantiated against the current goal and selects the red and yellow primaries for combination and subsequent verification, rather than reusing the green-producing trajectory retained at L_0 and L_1. The instantiated guidance constrains pigment selection at decision time, and a candidate action that selects a non-primary or off-target pigment is inconsistent with the principle and revised through the resampling step of Algorithm 1; the episodic and procedural records remain available as fallback execution evidence.

Directed behavior after evolution in ALFWorld. Under strong-backbone evolution, the three games that fail in round 1 (`clean/6`, `heat/29`, and `heat/103`) terminate after 48, 38, and 39 steps, respectively, upon exhausting the context budget. At round 3, the same games are solved in 17, 15, and 13 steps, respectively. These reductions are consistent with more directed exploration and cannot be attributed to an expanded interaction budget. The only cross-round regression is `pick2/17`, which is solved within 37 steps in round 1 but exhausts the context budget in subsequent rounds. This residual error occurs in the unseen Pick2 task type, for which no training trajectory is available. Table 13 summarizes these per-game episode lengths.

Validation limitation in the complete-hierarchy arm with a fixed 7B actor. In the strict ALFWorld setting, no principle from the complete 7B configuration passes validation during R1–R3, as reported in Table 12. Despite this limitation, the complete-hierarchy pipeline improves from 45.5% at R0 to 48.5%, 54.5%, and 59.4% at R1, R2, and R3, respectively. At R2, the trajectory-only control obtains 57.5%, exceeding the complete-hierarchy result of 54.5% by 3.0 percentage points; at R3, the complete-hierarchy pipeline obtains 59.4%, compared with 46.3% for the trajectory-only control. These results distinguish task-solving performance from principle-validation outcomes: performance improves in the complete-hierarchy pipeline, but the reported rounds do not demonstrate successful validation or utilization of L_3 principles. Accordingly, we do not attribute the observed gains to validated principles or infer prompt-induced degradation from these aggregate scores alone.

Limitations. The empirical conclusions are subject to several limitations. First, both training-based pipelines use Qwen3.8-max-0902 for abstraction while optimizing Qwen2.5-7B-Instruct as the execution policy. This design does not isolate the contribution of the offline builder's pretrained knowledge; a same-model builder-actor control remains necessary. Second, a substantial subset of residual ScienceWorld failures is execution-bound rather than knowledge-bound. Examples include episodes that locate the target object but fail to submit the final answer, which cannot be corrected through memory content alone. Finally, the extracted principles are explanatory hypotheses supported by execution evidence, not established causal laws. Their generality beyond ScienceWorld and ALFWorld remains unverified.

F Future Work

Personalized experience adaptation. The present formulation treats memory as a task-centric artifact: episodes, procedures, and principles are abstracted from environment outcomes and shared across all interactions with a given benchmark. A natural extension is to condition the hierarchical memory on individual users, so that the same framework accumulates and reuses user-specific preferences, constraints, and interaction habits alongside task knowledge. Concretely, the source

metadata of each node can be augmented with a user identifier, and retrieval can be biased toward evidence originating from, or validated against, the current user. This yields a personalized principle layer whose scope field is qualified not only by task type but also by user context, enabling the actor to instantiate constraints that respect stable individual preferences without re-deriving them at every episode. Realizing this extension raises questions that the current evidence-grounding and utility-screening procedures do not yet address. User preferences are often implicit, occasionally contradictory, and may drift over time, so the validation criteria must distinguish durable preferences from transient instructions and must bound the influence of stale evidence. We regard controlled disentanglement of user-specific from task-general knowledge, together with per-user calibration of the injection budget, as the primary open problems for this direction.

Interactive preference elicitation and lifelong personalization. A second direction couples memory construction with active elicitation. Rather than inferring preferences solely from passively observed trajectories, the actor could issue targeted clarification queries when retrieval returns conflicting or low-confidence user-conditioned principles, and record the resolved preference as a first-class episodic record. Such interactions would supply higher-quality supervision for the abstraction pipeline while keeping the human in the loop for genuinely ambiguous cases. Sustaining personalization over long horizons additionally requires mechanisms for preference revision and forgetting. The exclusion registry introduced for rejected principles offers a starting point, but a personalized deployment must further support explicit user overrides, time-decayed evidence, and auditable provenance so that any behavioral change can be traced to a specific preference update. We leave a systematic study of these lifelong-personalization mechanisms, and their interaction with the training-based pipeline, to future work.