MEMORY AS PROGRAM: SHOULD AGENTS RUN THEIR EXPERIENCE INSTEAD OF REREADING IT?

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory allows LLM agents to carry experience from past interactions into new situations, where earlier successes and failures can inform their decisions. Yet many systems supply retrieved memories as context, leaving their influence on action selection to the model's interpretation. Memory used this way remains advisory, suggesting what to do without explicitly determining action admissibility or candidate values. We introduce **Memory as Program (MemAP)**, a framework that expresses an agent's control policy through executable programs configured by accumulated experience. MemAP exposes memory traces through a programmable symbolic interface, allowing executable programs to compare their decision contexts and use the associated feedback. During execution, these programs retrieve transferable experience, compute return-grounded action values, and enforce admissibility constraints, using novelty to guide exploration when return support is insufficient. Interaction feedback updates MemAP's program state, revising where stored experience applies and how it influences subsequent decisions while leaving program definitions fixed. On WebArena, MemAP achieves 61.9% success with GPT-4o, outperforming the evaluated memory-based baselines in overall success, and reaches 68.7% with GPT-5. Ablations also show that executing the same experience outperforms contextual reuse and that program-state adaptation further improves success. MemAP also better preserves transferable-precedent retrieval as memory grows, achieving twice A-MEM's Hit@5 at the largest tested scale, and reuses fixed source-site memory on held-out real-world websites with higher success and shorter successful trajectories than A-MEM.

1 INTRODUCTION

Large language model (LLM) agents use memory to carry experience across long interaction horizons and draw on it in subsequent decisions. Existing systems organize this experience into consolidated facts (Chhikara et al., 2025), hierarchical working memory (Hu et al., 2025), distilled reasoning patterns (Ouyang et al., 2026), or reusable workflows (Wang et al., 2025b). Although these systems differ in what they retain, experience typically reaches action selection through a similar interface: it is placed in context for the LLM to interpret. These advances enrich what an agent remembers, while the effect of that memory on its next action remains implicit in the model's interpretation. **Memory used this way remains context rather than control**: it can inform a decision, but does not itself specify which actions are admissible or how competing candidates should be valued.

Memory used as context leaves the model to determine how past experience should guide the current decision. Updating memory in response to feedback changes what the model reads, but does not itself specify how its next decision should change. Retrieval-based memory systems support such updates by revising what is stored and recalled (Chhikara et al., 2025; Ouyang et al., 2026; Xu et al., 2025). A complementary line of work gives experience a reusable procedural form through executable tools and skills (Wang et al., 2024; 2025a; Zheng et al., 2025; Shao et al., 2026), with verification and failure attribution supporting their refinement (Li et al., 2026; Xiang et al., 2026). Our focus is not simply whether experience can be executed, but **how accumulated experience parameterizes the decision rule**. A memory program determines where stored evidence applies and uses its feedback to compute action admissibility and value. Its state can be revised as evidence accumulates, changing how previously stored experience contributes to subsequent choices. This makes the use of memory itself programmable, extending adaptation from the content of experience to its role in control.

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.

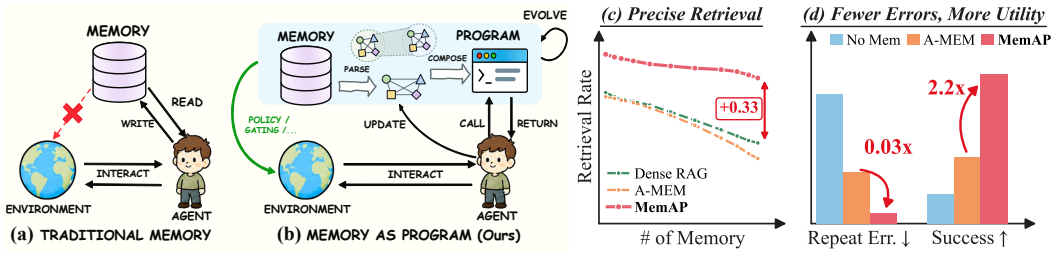

Figure 1: **MemAP provides a programmable interface between memory and policy.** (a) Traditional memory is reread as **context**. (b) MemAP compiles experience into **programs** that compute action values with gates and evolve with feedback. Compared with A-MEM, MemAP (c) maintains more precise retrieval as memory grows and (d) repeats fewer failed actions while improving success.

We take this view further by treating accumulated experience as state that parameterizes a decision program. A programmable symbolic interface would expose memory traces as structured records of contexts, actions, and feedback. These records, transfer relations, and operating parameters would form the persistent program state. Given the current task and agent state, functions over this state could determine which experience applies and compute action admissibility and value. Memory would thereby **parameterize the decision rule**, with its contribution traceable to the supporting records and operations. Revising this state from interaction feedback would allow its decision effects to evolve without changing the function definitions. Realizing this view requires an agent to construct and update the program state from its own experience. This raises the question:



How can an agent turn memory into **an evolving program** whose execution directly shapes the decision rule through admissibility constraints and action values?

We address this gap with **Memory as Program (MemAP)** (Fig. 1b), which turns accumulated experience into executable programs that directly shape the agent's decision rule. MemAP lifts textual interaction traces into a programmable symbolic interface for retrieving applicable records and computing their contributions to action selection. It parameterizes explicit action-value functions with retrieved returns and structural transferability, while admissibility checks exclude actions marked inadmissible in the same symbolic context. These computations **connect memory retrieval with policy control**: the policy exploits highest-valued admissible action when return support is sufficient and otherwise favors actions less covered by prior experience. The constraints and scores remain traceable to supporting records and operations, making memory's influence on each decision explicit.

As experience accumulates, MemAP updates its program state from interaction feedback, revising memory records, transfer relations, and retrieval and reuse thresholds. These updates change more than the available evidence: they can alter where an existing record applies, how its return contributes to action values, and when that support warrants reuse. MemAP thus **adapts the decision role of accumulated experience** through changes to program state, while the executable function definitions remain fixed. The evolution ablation supports the benefit of these updates over executing the program without further state adaptation. These results support four contributions:

❶ We introduce **Memory as Program**, a decision-making framework whose executable functions determine action admissibility and value from accumulated experience.

❷ MemAP **couples structural retrieval with action valuation**: transferability determines which records are retrieved and how their returns contribute to action values, while admissibility constraints and novelty guide selection and exploration.

❸ MemAP **learns how experience is used**, updating records, transfer relations, and decision thresholds from interaction feedback to revise where experience applies and when it warrants reuse, with executable function definitions held fixed.

❹ Experiments show **improved task success on WebArena** (Table 1). MemAP also outperforms A-MEM on held-out real-world websites, with higher success and fewer steps among successful trajectories (Table 2). As memory grows, MemAP better preserves transferable-precedent retrieval, achieving twice A-MEM's Hit@5 at the largest tested memory size (Fig. 1c).

2 Related Work

Agents for Long-Horizon Tasks. Long-horizon tasks require agents to continually adapt decisions over extended interactions as observations and outcomes accumulate. Existing systems address these challenges through different organizations of the decision loop. Single-agent approaches interleave reasoning and action (Yao et al., 2023; Yang et al., 2025), while planner–executor architectures decompose tasks into subgoals and ground them in primitive actions (Zhang et al., 2025b; Erdogan et al., 2025). Outcome-driven training further improves action policies through environmental feedback (Qi et al., 2025). Beyond planning and coordination, structured verification makes execution more inspectable. SkillTracer represents skills as attributed plan graphs to localize failures and support targeted refinement (Li et al., 2026), while NesyProAct checks symbolic predicates against the current state before committing actions (Xiang et al., 2026). Together, these approaches improve how agents organize, execute, and correct extended courses of action. MemAP complements these efforts by studying how experience accumulated across interactions can persistently influence subsequent decisions, providing executable support for action evaluation, admissibility, and exploration.

Memory for LLM Agents. Agent memory preserves experience for reuse across decisions and tasks. Existing approaches organize it as external stores (Chhikara et al., 2025), hierarchical working memory (Hu et al., 2025), procedural guidance (Fang et al., 2026), or distilled reasoning patterns (Ouyang et al., 2026; Liu et al., 2025). Skill-based approaches further compile trajectories into reusable workflows and executable routines (Wang et al., 2025b;a; Zheng et al., 2025). Structured memory captures relationships among experiences: A-MEM maintains interconnected memory notes (Xu et al., 2025), while G-Memory organizes insights, queries, and interaction trajectories through hierarchical graphs (Zhang et al., 2025a). Beyond updating stored content, MemEvolve adapts the memory architecture itself by evolving programmatic implementations of encoding, storage, retrieval, and management (Zhang et al., 2026a). These studies advance the representation, retrieval, reuse, and evolution of agent experience. MemAP focuses on the executable interface between stored experience and decision making. It constructs programs from parsed memory representations, using structural queries, evidence-based scoring, and constraint checking to govern action selection. Execution feedback subsequently updates the experience and revises how it is applied in later tasks.

3 Methodology

We consider an agent that completes a natural-language query q through sequential interaction with an environment using a base policy π_θ. At step t, the agent forms a state s_t from its interaction history, executes an action a_t, and stores the resulting interaction, including its return r_t, in a memory bank $\mathcal{M}_t$. MemAP defines how transferable experience re-enters decisions by exposing it through a programmable symbolic interface rather than rereading it as context. Figure 2 presents an overview of MemAP, from accumulated experience to program-induced policy and program evolution.

3.1 Memory as Program: From Experience to Decision Rules

We introduce MemAP to make accumulated experience explicit in action selection. Given decision context x_t extracted from s_t and q, MemAP maps evidence to a memory-parameterized decision rule through executable functions. We formalize this decision-time interface as *Memory as Program*.

Definition 3.1 Memory as Program

Given accumulated memory $\mathcal{M}_t$, a memory program is a family of executable functions $\mathcal{F}_{\Phi_t}$ configured by an agent-maintained program state Φ_t that governs how $\mathcal{M}_t$ is queried and applied. For a base policy π_θ and decision context x_t, program execution is defined by

$$\underbrace{\Phi_t \longmapsto \mathcal{F}_{\Phi_t}}_{\text{program state configures executable function family}} \quad \Longrightarrow \quad \underbrace{\mathcal{F}_{\Phi_t} : (x_t, \pi_\theta) \longmapsto \pi_{\theta,\Phi_t}(\cdot \mid x_t)}_{\text{program execution induces the decision rule}}. \tag{1}$$

Accumulated experience enters action selection through these executable functions, each with a defined effect on the decision rule. As experience accumulates, execution feedback updates Φ_t, so that the functions applied in subsequent decisions are instantiated from the updated state.

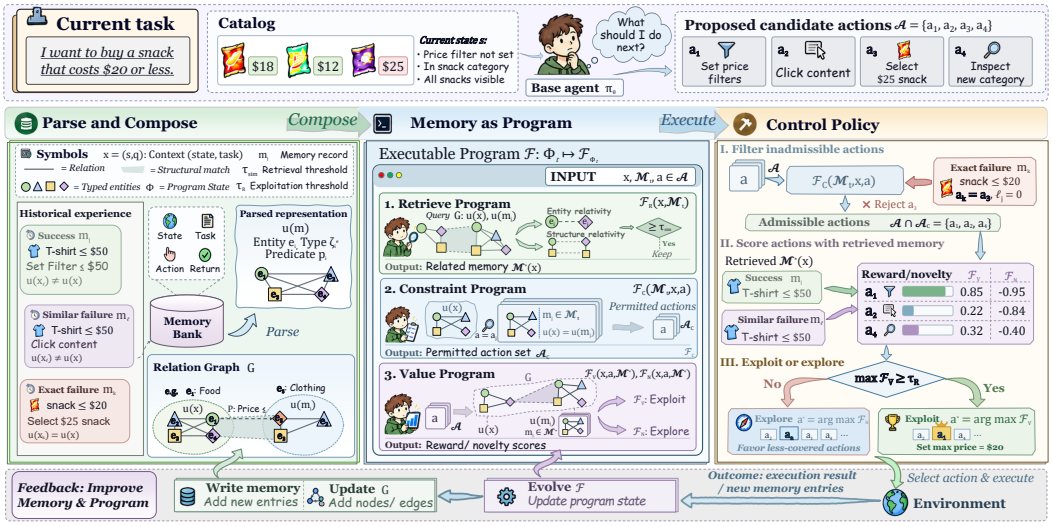

Figure 2: **MemAP turns accumulated experience into an executable decision rule. (1) Compose:** records become typed structures and transfer relations. **(2) Execute:** retrieval, constraint, and value programs filter and score actions; novelty guides selection when evidence is insufficient. **(3) Evolve:** interaction feedback updates memory and program state.

Here, m denotes a memory record and $u(m)$ its symbolic representation in the programmable interface. In context-based systems, retrieved records enter the policy as context, so their effect on action selection depends on in-context interpretation. MemAP instead executes $\mathcal{F}_{\Phi_t}$ over $u(m)$, retrieving structurally transferable records, applying associated constraints to determine admissibility, and using interaction returns to parameterize action values. Because these functions share an interface and their outputs remain linked to the records from which they are computed, memory-induced control is **explicit, composable, and auditable**. As evidence accumulates, interaction feedback updates Φ_t to revise evidence transfer, return-grounded valuation, and active constraints. Memory therefore enters the policy as executable decision logic that is revised through interaction.

3.2 Programmable Memory Interface

We construct a programmable memory interface that maps accumulated interaction traces into structured objects that executable functions can query and compare. At step t, the memory bank contains N_t records, $\mathcal{M}_t = \{m_i = \langle s_i, q_i, a_i, r_i, \ell_i \rangle\}_{i=1}^{N_t}$. Each record associates a decision context (s_i, q_i) and executed action a_i with its feedback return r_i and a feedback-derived admissibility label $\ell_i \in \{0, 1\}$. Each record thus preserves both the conditions under which experience may transfer and the feedback that governs its decision-time effect.

Programmable representation. The parser u maps a decision context (s_i, q_i) to a typed relational representation in $\mathcal{U}$. For a memory record m_i, we write $u(m_i) \equiv u(s_i, q_i)$, where (s_i, q_i) is its stored context; the same parser applies to the current context x_t. Each representation contains entities and relations annotated with types and predicates:

$$u(m) = \Big\langle \{(e_i, \zeta_i^e, p_i)\}_{i=1}^{N_m},\ \{(e_i, e_j, \zeta_{ij}^r, p_{ij})\}_{(i,j)} \Big\rangle, \tag{2}$$

where e_i identifies an entity, ζ_i^e and ζ_{ij}^r are entity and relation types from the shared vocabulary $\mathcal{Z}$, and $p_i, p_{ij} \subseteq \mathcal{P}$ are sets of predicate annotations. The feedback-derived return r_i and admissibility label ℓ_i remain attributes of m_i, separate from its parsed representation. Matching symbolic contexts scope the applicability of admissibility checks.

MemAP maintains a transferability graph G, expressed at the query interface over parsed contexts:

$$\mathsf{G} = (\mathcal{V}, \mathcal{E}), \qquad \mathcal{V} \subseteq \mathcal{U}, \qquad \mathcal{E} : \mathcal{V} \times \mathcal{V} \to [0, 1]. \tag{3}$$

Here $\mathcal{V}$ contains parsed contexts; unseen contexts are added before their first query, with fixed initial weights assigned to new connections. $\mathcal{E}$ assigns directed transferability weights revised through interaction feedback. Specifically, $\mathcal{E}(u(m_j), u(m_i))$ measures transferability from m_j to m_i based on their symbolic structure, giving $\sigma_{\mathsf{G}}(m_i, m_j) = \mathcal{E}(u(m_j), u(m_i))$. These scores govern retrieval and weight stored returns in action evaluation. Together with the retrieval and exploitation thresholds, $\mathcal{M}_t$ and G form the program state Φ_t that configures the executable functions $\mathcal{F}_{\Phi_t}$. Here t indexes decision steps, while k indexes trajectory-level updates in the evolution formulation.

3.3 Program Execution and Memory-Induced Control

MemAP bridges stored experience and action selection by composing retrieval, constraint, value, and novelty functions into an executable decision rule. For the current decision, we define the context as $x \triangleq (s, q)$, where s is the current state and q is the task query. For each stored experience $m_j = \langle s_j, q_j, a_j, r_j, \ell_j \rangle$, its corresponding decision context is $x_j \triangleq (s_j, q_j)$.

Retrieval program. The retrieval program $\mathcal{F}_R$ identifies past experiences transferable to the current decision. Given the current context x and memory bank $\mathcal{M}_t$, it evaluates each record's transferability and returns the retrieved memory set $\mathcal{M}^*(x) = \mathcal{F}_R(x, \mathcal{M}_t)$:

$$\mathcal{F}_R(x, \mathcal{M}_t) = \{m_j \in \mathcal{M}_t \mid \sigma_G(x, m_j) \geq \tau_{\text{sim}}\}. \tag{4}$$

Here τ_{sim} is the minimum transferability score required for a record to enter $\mathcal{M}^*(x)$. The selected records provide the evidence used by the subsequent value programs to evaluate candidate actions.

Constraint program. The constraint program $\mathcal{F}_C$ filters candidate actions proposed by the base policy π_θ, excluding actions marked inadmissible under an identical symbolic context. It takes the memory bank $\mathcal{M}_t$, current context x, and a candidate action a_i as input and returns a binary admissibility indicator. We sample k candidate actions $a_i \sim \pi_\theta(\cdot \mid x)$ to form $\mathcal{A}_k = \{a_i\}_{i=1}^k$, and retain the admissible candidates:

$$\mathcal{F}_C(\mathcal{M}_t, x, a_i) = \prod_{\substack{m_j \in \mathcal{M}_{t,a_i} \\ u(x_j) = u(x)}} \ell_j, \qquad \mathcal{A}_C(\mathcal{M}_t, x) = \{a_i \in \mathcal{A}_k \mid \mathcal{F}_C(\mathcal{M}_t, x, a_i) = 1\}, \tag{5}$$

where $\mathcal{M}_{t,a_i} = \{m_j \in \mathcal{M}_t \mid a_j = a_i\}$ contains the records associated with action a_i, with each admissibility label scoped to the symbolic context $u(x_j)$ in which it was obtained. The product therefore applies only labels whose symbolic scope matches the current context $u(x)$. The output $\mathcal{F}_C(\mathcal{M}_t, x, a_i)$ is 0 if any matching record marks the action inadmissible and 1 otherwise. The empty product is defined as 1, so candidates without matching memories are not excluded. The resulting set $\mathcal{A}_C(\mathcal{M}_t, x) \subseteq \mathcal{A}_k$ contains the admissible candidates. If $\mathcal{A}_C$ is empty, MemAP requests fresh proposals and reapplies the gate. Once a nonempty admissible set is obtained, the control policy below determines whether to exploit or explore.

Value programs. The value program $\mathcal{F}_V$ measures return support for exploitation, whereas the novelty program $\mathcal{F}_N$ measures the absence of transferable support for exploration. Both take the current context x, candidate action a_i, and retrieved memory set $\mathcal{M}^*(x)$ as input, and return the scalar value and novelty scores:

$$\mathcal{F}_V(x, a_i, \mathcal{M}^*) = \frac{1}{|\mathcal{M}^*_{a_i}|} \sum_{m_j \in \mathcal{M}^*_{a_i}} \sigma_G(x, m_j)\, r_j, \quad \mathcal{F}_N(x, a_i, \mathcal{M}^*) = -\frac{1}{|\mathcal{M}^*_{a_i}|} \sum_{m_j \in \mathcal{M}^*_{a_i}} \sigma_G(x, m_j).$$

Here $\mathcal{M}^* = \mathcal{M}^*(x)$, and $\mathcal{M}^*_{a_i} = \{m_j \in \mathcal{M}^* \mid a_j = a_i\}$ contains the retrieved memories associated with action a_i. The value score $\mathcal{F}_V$ averages the similarity-weighted interaction returns, while the novelty score is the negative mean transferability of the available precedents. By convention, both scores are zero when $\mathcal{M}^*_{a_i} = \varnothing$, leaving an uncovered action unsupported for exploitation but available to novelty-guided exploration.

Control policy. The policy selects among admissible candidates using value and novelty scores:

$$a^* = \begin{cases} \arg\max\limits_{a_i \in \mathcal{A}_C(\mathcal{M}_t, x)} \mathcal{F}_V(x, a_i, \mathcal{M}^*(x)), & \max\limits_{a_i \in \mathcal{A}_C(\mathcal{M}_t, x)} \mathcal{F}_V(x, a_i, \mathcal{M}^*(x)) \geq \tau_R \quad \text{(exploitation)}, \\ \arg\max\limits_{a_i \in \mathcal{A}_C(\mathcal{M}_t, x)} \mathcal{F}_N(x, a_i, \mathcal{M}^*(x)), & \text{otherwise} \quad \text{(exploration)}. \end{cases}$$

$\tau_R > 0$ is the minimum return support for exploitation. MemAP selects the admissible action with highest return $\mathcal{F}_V$ when the threshold is met; otherwise, it selects the one with highest novelty $\mathcal{F}_N$.

3.4 Program Evolution from Interaction Feedback

With the evaluation rules in §3.3 fixed, MemAP adapts its program state at update round k:

$$\Phi_k \triangleq (\mathcal{M}_k, \mathsf{G}_k, \tau_{\mathrm{sim},k}, \tau_{R,k}). \tag{6}$$

After each trajectory, the reflection agent updates this state using the interaction trace, retrieved memories, and execution feedback:

$$\Phi_{k+1} = \mathrm{Reflect}_{\mathrm{LLM}}\Big(\Phi_k,\ \big\{(x_t, \mathcal{M}_k^*(x_t), a_t)\big\}_{t=1}^{T},\ R\Big). \tag{7}$$

Here k indexes trajectory-level update rounds, T is the number of decision steps, and R contains observed action outcomes and the task-level result. Using this feedback together with recorded observations and tool responses, the reflection agent assigns or revises record-level returns r_i and admissibility labels ℓ_i. These are agent-assigned assessments, with admissibility evaluated within the corresponding symbolic context using local execution evidence.

During execution, query-time registration expands the graph as $\mathcal{V}_{k+1} = \mathcal{V}_k \cup \{u(x_t)\}_{t=1}^{T}$. After the trajectory, reflection writes or revises memory records and updates the weights of existing graph connections. Reflection also adjusts $\tau_{\mathrm{sim},k}$ and $\tau_{R,k}$, which govern retrieval eligibility and the value support required for exploitation, respectively. The next trajectory runs under Φ_{k+1}, while the executable function definitions remain unchanged.

4 Experiments

4.1 Experimental Setup

Benchmark and Evaluation. We assess the efficacy of MemAP on WebArena (Zhou et al., 2024), a high-fidelity benchmark comprising 812 complex tasks that simulate real-world web interactions. These tasks span five core domains: Shopping, Admin, Reddit, GitLab, and Map, and diverse cross-domain challenges. For our WebArena evaluations, task success is determined strictly by the standard WebArena evaluator, which is widely adopted in prior work such as ASI (Wang et al., 2025a), to ensure a fair comparison. The harness checks the environment's final state (e.g., database records, URL parameters, and DOM elements) to provide a binary success signal. We retain the same evaluator configuration across methods evaluated in our experiments.

Backbone LM and Agent Architecture. We evaluate MemAP with two backbone LLMs: GPT-4o (OpenAI et al., 2024) for standard comparative analysis and GPT-5 (OpenAI, 2025) to evaluate performance under advanced reasoning scales. We utilize the default action set $\mathcal{A}_p$ from WebArena as the foundation for our agentic control, as detailed in Appendix C, and adopt a ReAct-style (Yao et al., 2023) reasoning–acting interface without additional task-specific engineering. For cross-page memory matching, action targets are compared using symbolic descriptors comprising their accessible names and local structural context. Numeric element IDs serve as page-local execution handles and are not part of these memory-matching descriptors.

Baselines. We benchmark MemAP against three paradigms: (i) *Single-Agent Systems*: the ReAct baseline from (Zhou et al., 2024), Branch-and-Browse (He et al., 2026), Plan-MCTS (Zhang et al., 2026b), AgentOccam (Yang et al., 2025), WebOperator (Dihan et al., 2025), and WebNavigator (Zhang et al., 2026c); (ii) *Multi-Agent Systems* (WebPilot (Zhang et al., 2025b) and WALT (Prabhu et al., 2026)); and (iii) *Memory-Augmented Agents*, spanning the three forms memory takes in current agents: trajectory-level memory (AWM (Wang et al., 2025b), CER (Liu et al., 2025), A-MEM (Xu et al., 2025)), skill-based memory, where successful trajectories are compiled into reusable subroutines (SkillWeaver (Zheng et al., 2025), ASI (Wang et al., 2025a)), and neuro-symbolic memory, where experience is stored as predicates checked against the current state (NesyProAct (Xiang et al., 2026)). This last category is the one MemAP belongs to, so it carries the comparison that isolates our contribution: what differs is not whether experience is symbolized, but whether the resulting structures execute as admissibility and value signals at decision time. We compare MemAP with baselines using the same backbone, GPT-4o or GPT-5, to limit differences attributable to model capacity. We evaluate A-MEM and ASI under GPT-4o; the GPT-4o AWM results are reported by Wang et al. (2025a). WebDART and CER report only four and five domains, respectively, so their averages do not cover the full task split.

Methods	LLM Backbone	Domain						Avg.
		Shopping	Admin	Reddit	GitLab	Map	Cross	
Single-Agent Systems								
ReAct baseline (Zhou et al., 2024)	GPT-4o	13.9	10.4	6.6	15.0	15.6	8.3	12.3
Tree Search (Koh et al., 2025)	GPT-4o	27.8	16.5	11.3	13.9	26.6	16.7	19.2
Branch-and-Browse (He et al., 2026)	GPT-4o	34.6	26.4	50.9	36.7	46.8	18.8	35.8
API-Based Agent (Song et al., 2025)	GPT-4o	25.7	41.2	51.9	44.4	45.9	16.7	38.9
Plan-MCTS (Zhang et al., 2026b)	GPT-4o	43.6	40.5	54.1	33.7	24.6	–	39.2
AgentOccam (Yang et al., 2025)	GPT-4o	33.7	46.7	60.4	44.4	40.4	25.0	42.9
WebDART (Yang et al., 2026)	GPT-4o	36.0	41.2	67.9	47.2	–	–	48.1
WebOperator (Dihan et al., 2025)	GPT-4o	49.2	55.0	76.4	52.8	55.2	31.3	54.6
WebNavigator (Zhang et al., 2026c)	GPT-4o	44.4	48.6	73.6	42.2	51.4	50.0	49.9
Multi-Agent Systems								
WebPilot (Zhang et al., 2025b)	GPT-4o	36.9	24.7	65.1	39.4	33.9	–	37.2
WALT (Prabhu et al., 2026)	GPT-5	41.2	56.2	48.5	57.0	58.7	20.8	50.1
Memory-Augmented Agents								
A-MEM† (Xu et al., 2025)	GPT-4o	36.9	19.8	15.1	22.8	50.5	16.7	27.7
ASI† (Wang et al., 2025a)	GPT-4o	31.6	33.2	32.6	28.9	22.5	16.7	29.4
SkillWeaver (Zheng et al., 2025)	GPT-4o	27.2	25.8	50.0	22.2	33.9	–	29.8
AWM (Wang et al., 2025b)	GPT-4o	32.1	29.1	54.7	35.0	42.2	18.8	35.5
CER (Liu et al., 2025)	GPT-4o	32.8	41.2	41.2	37.2	30.4	–	36.7
NesyProAct (Xiang et al., 2026)	GPT-4o	58.1	47.6	**85.8**	56.7	55.0	41.7	58.2
MemAP (Ours)	GPT-4o	63.1	59.3	77.4	57.8	59.6	52.1	61.9
	GPT-5	**69.5**	**68.7**	81.1	**60.0**	**75.2**	**56.3**	**68.7**

Table 1: **Performance comparison on WebArena.** Under the same GPT-4o backbone, memory-based closest baselines (A-MEM, AWM, CER, ASI, SkillWeaver) reach 27.7% – 36.7%, while the two that operate on the candidate set reach 58.2% and 61.9%. MemAP gains most where experience must carry across pages, Admin by 11.7pp and Cross-site by 10.4pp over NesyProAct, where each step can be verified locally. With GPT-4o it also exceeds the GPT-5 multi-agent system WALT. †Evaluated under our GPT-4o evaluation configuration; all other baseline results are taken from the cited work under their original evaluation settings.

4.2 Main Results

We present the benchmark comparison in Table 1, where MemAP achieves the highest overall success rate among the evaluated methods under both GPT-4o and GPT-5.

MemAP scales consistently with stronger backbones. We first evaluate end-to-end task success across the six WebArena domains. As shown in Table 1, MemAP achieves **61.9%** overall success with GPT-4o and **68.7%** with GPT-5. The resulting **6.8pp** improvement is shared by all six domains rather than driven by a single category, with the largest increases on `Map` (**+15.6pp**) and `Admin` (**+9.4pp**). MemAP therefore continues to benefit from improvements in the underlying policy while retaining the same executable memory mechanism.

Executable memory substantially outperforms context and skill reuse. The clearest separation appears among methods that differ in how accumulated experience re-enters the policy. Under GPT-4o and the complete six-domain evaluation, MemAP reaches **61.9%**, compared with **27.7%** for A-MEM, **29.4%** for ASI, and **35.5%** for AWM. These **26.4–34.2pp** margins extend across note-based memory, induced skills, and reusable workflows, rather than arising from comparison with one particular memory representation. Among the two systems in Table 1 that apply symbolic operations to candidate actions, MemAP obtains the higher overall score and leads NesyProAct in five of six domains; NesyProAct remains stronger on `Reddit`. This boundary case is retained rather than attributed to a mechanism not isolated by the present evaluation.

Structural retrieval remains effective as memory grows. We next isolate retrieval from downstream action selection by keeping the transferable precedent in memory while enlarging the bank only with distractors (Figure 1(c) in §1; full protocol and confidence intervals in Appendix A.2 and Figure 3). As the library grows from 59 to 2,325 experiences, MemAP loses 14pp in Hit@5, compared with

29pp for dense retrieval and 35pp for A-MEM. At the largest scale, MemAP is **33pp** above the dense retriever and retains **twice** A-MEM's Hit@5. The increasing separation indicates that the advantage is not a fixed offset at a favorable memory size: MemAP preserves access to transferable precedents as irrelevant experience accumulates around them.

Executable memory prevents repeated failures. Finally, we isolate how memory is used from how it is retrieved by giving every method the same oracle memory at decision points where a failure has already been recorded (Figure 1(d) in §1; full protocol and confidence intervals in Appendix A.3 and Table 5). Across 62 WebArena GitLab decision points, A-MEM, which presents the recorded failures as textual experience, lowers the repeated-error rate from 26.8% to 10.6%, whereas MemAP lowers it to **0.3%**. On the 17 prior-conflict points, where the memory-free agent persistently prefers the failed action, A-MEM still repeats it in 30.6% of selections, compared with **1.2%** for MemAP. The separation is largest where the model's own preference conflicts with the record: MemAP makes a recorded failure act on the decision rather than leaving it as advice that the policy can override.

The experiments locate MemAP's advantage along the complete path from memory to action. Structural retrieval keeps transferable precedents reachable as the memory bank grows; constraint and value programs then make recorded failures binding on action selection rather than advisory. Together with the consistent gains across backbones in Table 1, this turns accumulated experience into a scalable source of decision-time control rather than additional context for the policy to reinterpret.

4.3 Cross-Website Generalization of Memory-Induced Control

Using GPT-4o, we evaluate whether memory acquired in WebArena-Lite Shopping remains useful when the agent moves to a different website, rather than merely to unseen tasks within the source environment. Table 2 reports performance on 39 Amazon and 36 Target tasks, whose layouts and interaction structures were not encountered during source-memory construction. MemAP achieves the highest success rate on both websites, reaching **25.6%** on Amazon and **19.4%** on Target. Relative to A-MEM, these results represent gains of **10.2pp** and **8.3pp**; relative to No memory, the gains are **20.5pp** and **5.5pp**. The direction of transfer is also informative. A-MEM improves over No memory on Amazon but falls **2.8pp** below it on Target, whereas MemAP remains above the no-memory agent on both websites. Access to source-site experience alone therefore does not guarantee positive transfer.

Table 2: **Cross-website generalization to held-out real-world websites** (`WebArena-Lite` shopping → Amazon and Target). With source-site memory held fixed, MemAP leads on both targets, improving success over A-MEM by **66.2%–74.8%** while requiring **8.6%–20.8%** fewer steps on successful trajectories.

Method	**Amazon** (39 tasks)		**Target** (36 tasks)	
	SR (%↑)	**Steps ↓**	**SR (%↑)**	**Steps ↓**
No memory	5.1	15.2	13.9	16.0
A-MEM	15.4	14.0	11.1	15.4
MemAP	**25.6**	**12.8**	**19.4**	**12.2**

The success gains are obtained efficiently with reduced explorations steps. Among successful trajectories, MemAP uses **12.8** steps on Amazon and **12.2** on Target, respectively **1.2** and **3.2** fewer than A-MEM. Higher success and shorter successful trajectories indicate that the transferred memory improves action selection rather than merely extending the search process. Across these two held-out websites, the results support the central role of the programmable interface: prior experience remains useful when its applicability and decision effect can be recomputed for the new interface, rather than supplied as source-site context for the model to reinterpret.

4.4 Ablation Study

We ablate MemAP on WebArena-Lite to examine the contributions of three key modules: structural retrieval, programmatic scoring, and program evolution. We replace structural retrieval with embedding retrieval, remove candidate scoring, or disable feedback-driven evolution in separate variants. In the evolution ablation, memory records continue to accumulate, while transferability weights and both decision thresholds remain fixed at their initial values throughout evaluation. We also introduce *Memory as Context (MemAP-Context)*, which supplies the accumulated experiences and candidate actions as prompt context, with executable gating and scoring disabled. All variants

share the `gpt-4o` backbone, action space, planner, and outer loop. Table 3 reports the results. The ablations distinguish selecting transferable experience from using it to evaluate actions: retaining either retrieval or scoring does not recover full performance. The evolution ablation further shows that these executable operations benefit from updating the evidence and parameters on which they act.

Full MemAP achieves 57.9% success (Table 3). *(i)* **Structural retrieval contributes beyond scoring.** Replacing it with semantic embedding retrieval lowers success to 31.8% (-26.1pp), with downstream scoring unchanged. The same scoring programs thus perform differently depending on which experience supplies their evidence, making retrieval part of the control mechanism rather than merely a lookup step. *(ii)* **Retrieval alone does not recover the benefit of candidate scoring.** Retaining structural retrieval but removing candidate scoring reduces success to 40.9% (-17.0pp). Selecting a transferable precedent and evaluating a current action are distinct operations: scoring uses transferability and recorded returns to turn retrieved experience into candidate-specific values. *(iii)* **Fixed programs benefit from state updates.** Freezing memory records, transfer relations, and decision thresholds while retaining retrieval and scoring lowers success to 33.5% (-24.4pp). The gain supports program state as an effective target of adaptation, through which new experience can revise the evidence used for action selection and the conditions for its reuse without rewriting the executable programs.

Table 3: Controlled ablations (`WebArena-Lite`). Each component contributes to task success; MemAP-Context leaves action selection to in-context interpretation and scores 35.2pp below full MemAP.

Variant	SR (%)	Drop (pp)
MemAP (Full)	**57.9**	—
w/o Programmatic Retrieval	31.8	↓ 26.1
w/o Programmatic Scoring	40.9	↓ 17.0
w/o Programmatic Evolution	33.5	↓ 24.4
Memory as Context	22.7	↓ 35.2

MemAP-Context reaches 22.7% success, 35.2pp below full MemAP (Table 3). Although experience remains available, the model must infer its implications for action selection. Recorded inadmissibility is no longer enforced by a gate, and recorded returns no longer enter an explicit value computation. Beyond changing memory format, this transfers constraint application and candidate valuation back to in-context reasoning. Full MemAP performs these operations explicitly, so applicable constraints affect eligibility and supporting returns enter candidate scores before selection. The comparison therefore concerns how available experience is applied, rather than whether the agent has access to it. We further examine this distinction in a repeated-error probe, where MemAP repeats fewer failed actions than A-MEM when both receive the relevant failure records (Appendix A.3). The ablation supports the task-level benefit of executable memory use, while the probe provides behavioral evidence that MemAP more effectively avoids actions associated with recorded failures.

5 Conclusion

We introduced Memory as Program (MemAP), a framework that gives accumulated experience a defined computational effect on action selection. MemAP uses executable programs to determine which records apply to the current context, how their evidence constrains admissible actions, and how interaction returns contribute to action-value estimates. Experiments show improved task success across two model backbones, retrieval resilience as memory grows, and cross-website transfer with source-site memory held fixed. Controlled ablations show that executing experience through gating and scoring outperforms contextual reuse, while updating program state improves success over fixed-state execution. These findings support the memory–policy interface as a means of turning stored experience into explicit constraints and values that govern action selection.

MemAP adapts its programmable interface through program state. Interaction feedback updates memory records, transfer relations, and retrieval and reuse thresholds, while program definitions remain fixed. The agent can thus revise where experience applies and when it warrants reuse without rewriting the decision procedure. The resulting constraints and scores remain traceable to supporting records and computations. Our evaluation focuses on web tasks, including transfer to held-out shopping websites; extending the symbolic interface to other environments remains an open direction. Memory as Program makes accumulated experience part of the agent's control policy, with an explicit computational role that can be revised through interaction.

ETHICS STATEMENT

The experiments in this work do not involve human subjects or consequential real-world transactions. WebArena is evaluated in sandboxed environments, and the cross-website study is limited to no-login search and navigation tasks on Amazon and Target. No authentication credentials or private personal data are stored in memory. Deployment on real user accounts would require appropriate retention and deletion controls, together with explicit authorization for irreversible actions.

AI USE STATEMENT

In this work, we used generative AI tools for editing assistance, specifically grammar, spelling, and word choice in the manuscript. We did not use generative AI tools for research ideation, experimental design, analysis of results, or the generation of citations. Large language models are also an object of study in this work: GPT-4o and GPT-5 serve as the agent backbones under evaluation, as described in Section 4.1. The authors reviewed all AI-assisted material, verified citations against their original sources and numerical claims against the underlying experimental records, and retained responsibility for all research, design, and interpretive decisions.

REFERENCES

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, 2025. doi: 10.3233/FAIA251160.

Mahir Labib Dihan, Tanzima Hashem, Mohammed Eunus Ali, and Md Rizwan Parvez. Webop-erator: Action-aware tree search for autonomous agents in web environment. *arXiv preprint arXiv:2512.12692*, 2025.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ybA4EcMmUZ.

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. URL https://aclanthology.org/2026.findings-acl.866/.

Shiqi He, Yue Cui, Xinyu Ma, Yaliang Li, Bolin Ding, and Mosharaf Chowdhury. Branch-and-browse: Efficient and controllable web exploration with tree-structured reasoning and action memory. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18407–18418, 2026. doi: 10.18653/v1/2026.acl-long.838.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025. URL https://aclanthology.org/2025.acl-long.1575/.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=QF0N3x2XVm.

Yuyang Li, Yiran Dou, Jie-Jing Shao, Yueming Lyu, Ivor W. Tsang, and Haiyan Yin. Skilltracer: Structural failure attribution and refinement of agentic skills in long-horizon web tasks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2026. doi: 10.1145/3770855.3817981.

Yitao Liu, Chenglei Si, Karthik R. Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), 2025. URL https://aclanthology.org/2025.acl-long.694/.

OpenAI. GPT-5 system card. https://openai.com/index/gpt-5-system-card/, 2025. OpenAI technical report.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, et al. GPT-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25140.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, Krithika Ramakrishnan, Jing Gu, Yanqi Luo, Silvio Savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, and Ran Xu. WALT: Web agents that learn tools. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cgIDqcJcoI.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

Jie-Jing Shao, Haiyan Yin, Yueming Lyu, Xingrui Yu, Lan-Zhe Guo, Ivor W. Tsang, James Kwok, and Yu-Feng Li. Lifting traces to logic: Programmatic skill induction with neuro-symbolic learning for long-horizon agentic tasks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.01293.

Yueqi Song, Frank Xu, Shuyan Zhou, and Graham Neubig. Beyond browsing: API-based web agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11066–11085, 2025. URL https://aclanthology.org/2025.findings-acl.577/.

Zhiruo Wang, Graham Neubig, and Daniel Fried. Trove: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Forty-first International Conference on Machine Learning (ICML 2024)*, 2024. URL https://openreview.net/forum?id=DCNCwaMJjI.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=lsAY6fWsog.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=NTAhi2JEEE.

Keyi Xiang, Tianyi Tang, Jie-Jing Shao, Yueming Lyu, Ivor Tsang, Yew-Soon Ong, and Haiyan Yin. Nesyproact: Proactive neural-symbolic control for web agents. In *ICLR 2026 Workshop on Agents in the Wild: Safety, Security, and Beyond*, 2026. URL https://openreview.net/forum?id=9xteuRjHr7.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jingbo Yang, Bairu Hou, Wei Wei, Shiyu Chang, and Yujia Bao. WebDART: Dynamic decomposition and re-planning for complex web tasks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 11696–11715, 2026. doi: 10.18653/v1/2026.findings-acl.568.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oWdzUpOlkX.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/pdf/2210.03629.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, pp. 14587–14617. Curran Associates, Inc., 2025a. doi: 10.52202/085713-0439. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/136a45cd9b841bf785625709a19c6508-Abstract-Conference.html.

Guibin Zhang, Haotian Ren, Chong Zhan, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-evolution of agent memory systems. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=qpkG0eKx4v.

Weiming Zhang, Jihong Wang, Jiamu Zhou, Qingyao Li, Xinbei Ma, Congmin Zheng, Xingyu Lou, Weiwen Liu, Zhuosheng Zhang, Jun Wang, Yong Yu, and Weinan Zhang. Plan-mcts: Plan exploration for action exploitation in web navigation. *arXiv preprint arXiv:2602.14083*, 2026b.

Xuanwang Zhang, Yuteng Han, Jinnan Qi, Mulong Xie, Zhen Wu, and Xinyu Dai. Webnavigator: Global web navigation via interaction graph retrieval. *arXiv preprint arXiv:2603.20366*, 2026c.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *AAAI*, pp. 23378–23386, 2025b. URL https://doi.org/10.1609/aaai.v39i22.34505.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079. arXiv preprint arXiv:2504.07079.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13854.

A Supplementary Experiment

A.1 Zero-Shot Generalization via Cross-Query Memory Transfer

To evaluate whether memory acquired from earlier executions generalizes to unseen tasks, we conduct a site-level generalization study across five WebArena environments. For each site, we construct 15 independent trials by uniformly sampling a total of 150 tasks. Within each 10-task trial, the first five are executed sequentially to populate the agent's memory substrate, while the remaining five serve as a held-out target set for evaluation.

We isolate the specific impact of this accumulated knowledge by comparing two execution paradigms on the exact same target tasks. Under the *Cold Start* setting, the agent tackles the evaluation tasks *tabula rasa*, without access to any previously acquired memory. In the *Memory Reuse* setting, the agent is granted full access to the symbolic assets aggregated during the first five tasks. By averaging the results across all 15 trials per site (Table 4), we report the Success Rate (SR), the number of execution steps for successful trajectories, and the average relative performance shift (Avg. Δ) from the cold start baseline, providing a statistically robust measure of cross-query memory transfer.

Across the five evaluated sites, memory reuse improves mean relative success by 10.7% and reduces success-only execution steps by 4.8%. These results support cross-query reuse of decision-relevant experience accumulated during the source tasks.

Table 4: **Generalization** of memory reuse across **WebArena** sites (**GPT-4o**). For each site, we sample 10 tasks, use the first 5 to accumulate memory, and evaluate the last 5 under Cold Start and Memory Reuse. We repeat this process 15 times and report the averaged success rate (SR) and success-only execution steps. Avg. Δ denotes the relative change from Cold Start to Memory Reuse averaged across the 5 sites.

Metric	Shop.	Admin	Reddit	GitLab	Map	Avg. Δ
Cold Step(Succ.)	4.4	4.5	5.6	5.0	5.9	–
Reuse Step(Succ.)	4.1	4.2	5.4	4.9	5.6	$\downarrow$ 4.8%
Cold SR (%)	40.0	29.3	61.3	37.3	30.7	–
Reuse SR (%)	42.7	34.7	64.0	41.3	34.7	$\uparrow$ 10.7%

A.2 Retrieval of Borrowable Experience under Memory Growth

An agent that keeps every step it executes accumulates memory without bound, so the operative question is not whether a relevant experience exists in the store but whether it can still be surfaced once thousands of irrelevant ones surround it. We therefore isolate retrieval from execution and measure it directly. We first randomly sample query steps from the collected trajectories, independently of any retrieval method. For each sampled query, two human annotators[1] identify the most *borrowable* prior experience: the one whose procedure transfers to the query, even when its surface entities differ and even when its trajectory failed. Under this criterion, the borrowable experience for *find the nearest McDonald's to the airport* is *find the nearest Starbucks to the school*, not *find the farthest McDonald's to the airport*, which shares every entity but inverts the procedure.

We then adopt a planted-gold protocol: the human-annotated gold is held in the library at every library size, and the library is grown only by adding randomly drawn distractor tasks. Any decline is thus attributable to crowding rather than to the gold being absent. The corpus comprises 2,605 experience steps collected over 184 WebArena shopping tasks; we report Hit@5 over 44 randomly sampled and annotated queries, averaging 20 random distractor draws per library size, with bands giving the 95% bootstrap confidence interval over queries. MemAP is compared against a dense retriever (all-MiniLM-L6-v2) and against A-MEM, whose retrieval path we replay faithfully, including its link expansion.

Figure 3 shows that all three retrievers begin in the same region at 59 experiences (0.98, 0.80, 0.77) and separate as memory accumulates. Growing the library to 2,325 experiences costs MemAP 14 points ($0.98 \rightarrow 0.84$), the dense retriever 29 points ($0.80 \rightarrow 0.51$), and A-MEM 35 points ($0.77 \rightarrow 0.42$); the curve for MemAP is also the only one that flattens, remaining within 6 points of its value at 500 experiences. The gap is therefore a property of scale rather than a constant offset, which matters precisely in the regime that lifelong agent memory targets. We attribute the difference to what each representation preserves. A dense embedding of a trajectory places *nearest* and *farthest* at nearly the same point, so once enough near-duplicates enter the store the gold is no longer separable, whereas MemAP's symbolic decomposition keeps the procedural shape in a distinct slot from the entities that fill it. This reading is supported by an ablation on A-MEM: its LLM-written notes and its memory graph are, as released, invisible to retrieval, since only the raw note text is embedded and link-expanded neighbours are appended after the top-k cut and discarded by it. Repairing both defects raises A-MEM only to 0.43 at full library, still below the plain dense baseline, which indicates that the shortfall is one of representation and not of engineering.

A.3 Repeated-Error Probe

We test whether experience in memory actually steers the agent away from actions that have already failed. From the experience library MemAP accumulated on WebArena GitLab, we collect every decision context that contains a *recorded failure*: a `click` or `hover` action that failed in every record of that context. Keeping the contexts for which the proposer returns at least one valid candidate gives 62 decision points (42 pages, 95 recorded failures). At each point, the agent faces the same

[1] Blind to the retriever that produced each candidate; pooled and shuffled in TREC style.

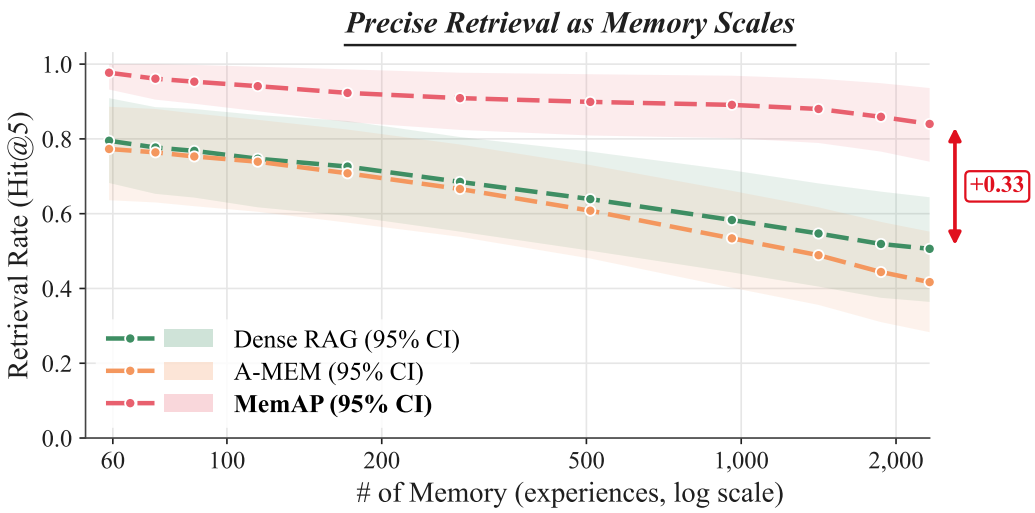

Figure 3: Retrieval of the most borrowable experience as agent memory grows. The annotated gold is planted in the library at every size and only random distractor tasks are added, so the decline measures crowding alone. Curves are Hit@5 over 44 annotated queries, averaged over 20 distractor draws per size; bands are 95% bootstrap confidence intervals over queries. MemAP loses 14 points between 59 and 2,325 experiences, against 29 for the dense retriever and 35 for A-MEM.

Table 5: **Repeated-error rate** on WebArena GitLab (`gpt-4o`, oracle memory). Lower is better. Brackets give 95% bootstrap confidence intervals over decision points.

Method	All points (n=62)	Prior-conflict (n=17)
No memory	0.268 [0.165, 0.381]	0.965 [0.906, 1.000]
A-MEM	0.106 [0.042, 0.184]	0.306 [0.118, 0.506]
MemAP	**0.003** [0.000, 0.010]	**0.012** [0.000, 0.035]

decision context again and chooses from a candidate pool consisting of the proposer's candidates together with the recorded failures.

To isolate how memory is *used* from how it is *retrieved*, every method receives the same oracle memory: all records of that decision context. Thus, differences between methods cannot be attributed to retrieval quality or memory coverage. We compare three methods on the same backbone (`gpt-4o`): *No memory*; *A-MEM*, where the records are rendered as natural language in the prompt and the LLM selects the action; and *MemAP*, where the records are compiled into the value program $\mathcal{F}_V$ and the constraint program $\mathcal{F}_C$, which score and filter the candidates. The *repeated-error rate* (RER) is the fraction of selections that fall on a recorded failure, computed over 5 queries per point. We report it over all points and over the *prior-conflict* points, where the memory-free agent selects the recorded failure in at least 3 of 5 queries. This subset serves as a stress test for cases in which prior experience must override the model's own action preference.

Table 5 shows that providing the same failure experience does not guarantee that it affects subsequent decisions when memory is supplied only as text. Across all decision points, A-MEM reduces RER from 26.8% to 10.6%, but does not eliminate repeated failures. We further isolate prior-conflict points as a stress test in which the memory-free policy exhibits a persistent preference for a previously failed action. Despite receiving oracle memory that explicitly contains this failure, the A-MEM agent still selects the failed action in 30.6% of its decisions. In contrast, MemAP reduces RER to only 0.3% across all points and 1.2% on prior-conflict points.

Since the backbone, candidate pool, page state, and available memory records are held fixed, this difference isolates how retrieved experience enters decision making. With A-MEM, failure records remain advisory evidence whose effect depends on the model's interpretation and can therefore be overridden by its action preference. MemAP instead compiles the same records into deterministic

Table 6: **Stability across repeated runs on WebArena.** We report success rate (SR, %) as mean ± standard deviation over three independent runs.

Method	Shop	Admin	Reddit	GitLab	Map	Cross	Overall
MemAP (GPT-4o)	62.0 ± 2.8	59.2 ± 0.8	76.7 ± 1.1	57.6 ± 0.3	62.4 ± 2.8	51.4 ± 1.2	$\mathbf{61.7 \pm 0.1}$
MemAP (GPT-5)	68.8 ± 3.3	67.9 ± 2.3	81.8 ± 1.1	61.5 ± 1.7	72.5 ± 2.8	55.6 ± 3.2	$\mathbf{68.4 \pm 0.8}$

admissibility and value computations, making their contribution to candidate evaluation explicit and reducing repeated failures by an order of magnitude even when the model's prior preference conflicts with the recorded experience.

A.4 Additional Experiment: Stability Across Repeated Runs

To assess whether the performance gains of MemAP are robust to run-to-run stochasticity, we repeat the full WebArena evaluation three times using both GPT-4o and GPT-5 as the backbone model. Each run evaluates the same 812 tasks under identical experimental settings. Table 6 reports the mean success rate (SR) and standard deviation across the three runs for each domain and the overall benchmark.

Across individual domains, the standard deviation ranges from only 0.3 to 3.3 percentage points, indicating limited variation across repeated executions despite the stochasticity of LLM-based web agents. More importantly, the aggregate results are substantially more stable: the overall standard deviations are only 0.1 and 0.8 percentage points for GPT-4o and GPT-5, respectively. This is expected because the overall metric aggregates performance across all 812 tasks, reducing the effect of task- and domain-specific fluctuations.

The three-run mean results are also highly consistent with the single-run results reported in Table 1. MemAP obtains an average overall SR of 61.7% with GPT-4o and 68.4% with GPT-5, compared with the corresponding single-run results of 61.9% and 68.7%. The differences are therefore only 0.2 and 0.3 percentage points, respectively. Together, these results indicate that the reported performance of MemAP is reproducible across repeated executions and is not attributable to a single favorable run.

B A-MEM and MemAP: Where Memory Adaptation Enters Decision Making

A-MEM already constructs structured notes, establishes links among them, and revises the attributes of existing notes as new memories arrive (Xu et al., 2025). Structure and evolution are therefore shared properties of the two systems, not the mechanism that separates them. The distinguishing mechanism lies in **what is matched, what evidence drives revision, and where the revised memory enters action selection**. A-MEM organizes textual notes around semantic relatedness for retrieval; MemAP matches typed symbolic structure and uses records carrying observed returns to compute admissibility and value.

This difference has a direct policy consequence. A-MEM's retrieved notes condition the model's proposal. MemAP instead applies $\mathcal{F}_C$ to construct $\mathcal{A}_C$ and uses $\mathcal{F}_V$ or $\mathcal{F}_N$ to rank its members before the control rule in Eq. (3.3) selects an action. The program definitions remain fixed, while updates to $(\mathcal{M}, G, \tau_{\text{sim}}, \tau_R)$ change which precedents activate, which actions remain admissible, and how those actions are ranked. Scope and lifecycle operations remain internal to $(\mathcal{M}, G)$; they do not introduce additional decision modules. **A-MEM adapts the organization of experience presented for interpretation; MemAP adapts the return-grounded operator that constrains and values the next decision.**

C Primitive Action Space

To ground the agent's behavior within the simulated web environment, we adopt the standard primitive action space provided by the WebArena benchmark, which interfaces with the browser via the BrowserGym framework. The comprehensive set of these atomic operations is detailed in

Table 7: **Memory adaptation reaches different points in the decision process.** A-MEM changes the organization of experience presented to the model; MemAP changes the admissible action set and the values assigned within it.

Mechanism	A-MEM	MemAP
Matching substrate	Textual-note embeddings seed retrieval; learned links expand the resulting neighborhood	Typed entities, relations, and predicates are compared through G to estimate structural transferability
Update evidence	A new note together with LLM-inferred relations to nearby notes	An observed return r per state–query–action record and a trajectory return R for reflection
Evolved state	Contextual descriptions, tags, and links, changing memory organization and the linked context later exposed	$(\mathcal{M}, G, \tau_{\text{sim}}, \tau_R)$, changing active precedents, constraints, action scores, and the exploitation–exploration boundary
Decision interface	Retrieved notes and linked neighbors condition the LLM; their behavioral effect is mediated by in-context interpretation	$\mathcal{F}_C$ operates over $\mathcal{M}$ to gate admissibility, while $\mathcal{F}_V$ and $\mathcal{F}_N$ operate over $\mathcal{M}^*(x)$ to score admissible actions

Table 8. For structural clarity, this fundamental action space $\mathcal{A}$ is partitioned into four operational dimensions: core element manipulation, page-level navigation, browser tab control, and human-agent communication.

Table 8: Primitive action space $\mathcal{A}$ utilized across all evaluation scenarios.

Action	Functionality
Basic Interactions	
`noop(wait_ms)`	Idle for specified duration.
`click(elem)`	Activate an element.
`hover(elem)`	Move cursor over an element.
`fill(elem, value)`	Input text into an element.
`keyboard_press(key_comb)`	Execute a key combination.
`scroll(x, y)`	Scroll in horizontal or vertical direction.
`select_option(elem, options)`	Choose from available options.
Page Navigation	
`goto(url)`	Navigate to specified URL.
`go_back()`	Return to previous page.
`go_forward()`	Advance to next page.
Tab Management	
`new_tab()`	Create a new browser tab.
`tab_close()`	Close active tab.
`tab_focus(index)`	Switch to specified tab.
User Communication	
`send_msg_to_user(text)`	Deliver message to user.
`report_infeasible(reason)`	Indicate task cannot be completed.

D SYMBOLIC QUERY REPRESENTATION

To bridge the semantic gap between open-ended natural language and deterministic execution, MemAP translates raw user instructions into a formal symbolic representation. As demonstrated in the snapshot below, this parsing step factorizes the query into explicit entities (e_i), semantic types (ζ^e), and symbolic relations (ζ^r). This structured decoupling forms the initial predicate space $\mathcal{P}$ required for precise graph-matching and verification.

```
Raw User Query 1 in Admin:
"What is the top-1 best-selling brand in Quarter 1 2022"

========================================================================

  "symbolic_query": {
    "query_type": "lookup",
    "entities": [
      {
        "entity_id": "E1",
        "entity_type": "product",
        "surface_form": "brand",
        "constraints": [
          {
            "constraint_id": "C1",
            "attribute": "time",
            "operator": "=",
            "value": "Quarter 1 2022"
          }]}
    ],
    "result_structure": {
      "output_mode": "with_output",
      "result_items": [
        {
          "result_id": "R1",
          "result_type": "entity",
          "result_description": "top-1 best-selling brand",
          "linked_entities": [
            "E1"
          ]
        }
      ],
      "output_relation": "single(R1)"
    }
  }
```

```
Raw User Query 2 in Map:
"I will arrive Pittsburgh Airport soon. Provide the name of a Hyatt
    hotel in the vicinity, if available. Then, tell me the the minimal
    driving time to a supermarket from the hotel."

========================================================================

  "symbolic_query": {
    "query_type": "retrieve",
    "entities": [
      {
        "entity_id": "E1",
        "entity_type": "place",
        "surface_form": "Pittsburgh Airport",
        "constraints": []
      },
      {
        "entity_id": "E2",
        "entity_type": "place",
        "surface_form": "Hyatt hotel",
        "constraints": [
          {
            "constraint_id": "C1",
            "attribute": "location",
            "operator": "=",
            "value": "vicinity of Pittsburgh Airport"
          }]},
      {
        "entity_id": "E3",
        "entity_type": "organization",
        "surface_form": "supermarket",
        "constraints": []
      }
    ],
    "result_structure": {
      "output_mode": "with_output",
      "result_items": [
        {
          "result_id": "R1",
          "result_type": "entity",
          "result_description": "name of a Hyatt hotel in the vicinity
              of Pittsburgh Airport",
          "linked_entities": [
            "E1",
            "E2"
          ]},
        {
          "result_id": "R2",
          "result_type": "time",
          "result_description": "minimal driving time from the Hyatt
              hotel to a supermarket",
          "linked_entities": [
            "E2",
            "E3"]}],
      "output_relation": "parallel(R1,R2)"
    }
```

E PROMPT

Prompt for Parser

```
You are a query parser. Convert the user query into the fixed JSON
    schema.
Input: a natural language query. Output: one JSON object ONLY.

JSON schema:
{ "query_type": "",
  "entities": [{
      "entity_id": "","entity_type": "","surface_form": "",
      "constraints": [
        {"constraint_id": "","attribute": "","operator": "","value":
     ""}]}],
  "result_structure": {
    "output_mode": "",
    "result_items": [
      {
        "result_id": "","result_type": "","result_description": "","
    linked_entities": []}],
    "output_relation": ""}
}
    Field Rules:
1. query_type: overall task type of the query
    1.a retrieve: find or select an object. Example: "Tell me the
    closest restaurant to campus."
    1.b lookup: check or read information about an object or
    relation. Example: "What is the price range of wireless earphone
     in the One Stop Market?"
    1.c compare: compare two or more candidates. Example: "Which is
    faster, walking or driving from CMU to the library?"
    1.d act: perform an action. Example: "Open my latest created
    issue with theme editor in the title."
2. entities: list of entities mentioned in the query.
    2.a entity_type: type of the entity, determined by its semantic
    meaning in the query.
        Available types: <<ENTITY_TYPES_DESC>>.
    If none of the above fits, create a new type: use a single
    lowercase word with strong generalizability (e.g. "route", "
    review", "rule"). Avoid "other"
    unless truly no meaningful type can be assigned.
    2.b surface_form: the exact text span in the query that
    corresponds to the entity.
    * Example: query: "Tell me the closest restaurant(s) to
    university center at Carnegie Mellon University" surface_form
    examples: "restaurant(s)", "university center", "Carnegie Mellon
     University"
    * Copy the span from the query as directly as possible. Do not
    normalize or rewrite it unless needed by the schema.
    2.c constraints: list of constraints on the entity, if any. If
    no constraint is needed, use an empty list.
    - attribute: the aspect used to further restrict an entity.
    Example: price: price-related restriction. "< 200 dollars"
    - operator: relation between attribute and value. = : exact
    match. Example: status = closed. != : not equal. Example: status
     != closed. contains: text contains keyword. Example: title
    contains "theme editor". <= : less than or equal. Example: price
     <= 100. >= : greater than or equal. Example: count >= 5. < :
    less than. Example: time < 1 hour. > : greater than. Example:
    distance > 5 miles
```

Prompt for Parser (Cont.)

```
    - value: the value used by the constraint.
    * Examples of full constraints: {"constraint_id": "C1", "
    attribute": "location", "operator": "=", "value": "Pittsburgh"}
3. result_structure: expected structure of the query result.
    3.a output_mode: whether the task expects an explicit returned
    result. - with_output: the task returns one or more results.
    Example: asking for price, distance, count, or entity. -
    without_output: the task mainly performs an action. Example:
    clicking like, opening an issue, submitting a form
    3.b result_type: semantic type of a result item.
        Available types: <<RESULT_TYPES_DESC>>
    If none of the above fits, create a new type: use single
    lowercase word with strong generalizability (e.g. "rating", "
    rank", "percentage"). Avoid "custom/other" unless truly no
    meaningful type can be assigned.
    3.c result_description: short semantic description of the result
     item. Example: "walking duration from Carnegie Mellon
    University to apple store shadyside"
    3.d linked_entities: entities directly involved in this result
    item. Example: if the result is distance from CMU to Starbucks,
    linked_entities can be ["E1", "E2"]
4. output_relation: relation among final result items.
    - single(R1): one final result. Example: "What is the price?" -
    parallel(R1,R2): multiple final results returned side by side.
    Example: "count and total spend"
    - compute(R1,R2): final output is computed from multiple results
    . Example: total duration = walk time + drive time
    - compare(R1,R2): final output compares multiple results.
    Example: compare walking time and driving time
    - list(R1,R2,R3): final output is a list of results. Example:
    top 3 items
    - null: no explicit returned output
    The relation can be recursive. For example, compare(R1, compute(
    R2,R3)) means R1 is compared with the computed result of R2 and
    R3.
Input:<<<USER_QUERY>>>
Output:
```

F CASE STUDY

We present a detailed case study on the task *"What is the top-1 best-selling brand in Quarter 1 2022?"* executed on the shopping administration site. At the decision step analyzed in this case study, the agent has already navigated to the Bestsellers report configuration page (`/admin/reports/sales/bestsellers/`). A representative excerpt of the key interactive elements is shown below:

```
[700] menubar ''
  [156] link '\ue60b SALES'
  [339] link '\ue60a REPORTS'
  [226] link '\ue609 MARKETING'
[714] button 'Show Report'
[741] combobox 'Period' % [options: Day, Month, Year]
[750] combobox 'From' % (date range start)
[760] combobox 'To'   % (date range end)
[771] button '\ue635 Export'
[793] link   'Order Qty' % (sortable column header)
```

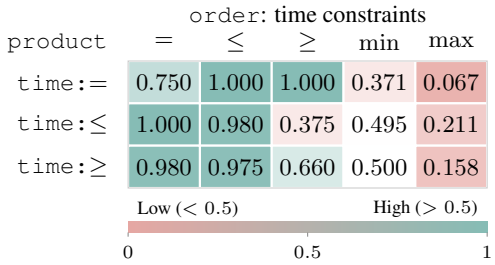

Figure 4: Learned transferability between time constraints on `product` (rows) and `order` (columns), queried from G. Only learned entries are shown.

The agent must select `Year` from the `Period` combobox [741], set the date range to cover Q1 2022, and click `Show Report` [714] to surface the ranked sales data. This single decision step is non-trivial: the action space contains multiple plausible targets (*e.g.*, clicking `Show Report` directly, selecting a period, clicking a column header to sort), each with different expected outcomes. The following subsections decompose this step along three dimensions captured by our framework.

F.1 Structural Retrieval

The current query concerns an entity of type `product`, with a predicate on `time` using the equality operator $=$. Structural retrieval consults the relation graph G for correspondences learned from previous tasks. Figure 4 illustrates the resulting transferability scores for experience involving entities of type `order` under different temporal constraints. The current query corresponds to the `time:=` row: the learned scores are 0.750 for equality and 1.000 for both $\leq$ and $\geq$. These scores encode prior evidence of transferability across entity types and predicates, supporting retrieval beyond exact type matches.

F.2 Proposals and Scoring

As illustrated in Table 9, the system evaluates five candidate actions derived from the current environmental observation. These proposals include: (i) selecting 'Year' in the 'Period' combobox (ID: 741) to set temporal granularity; (ii) clicking the 'Order Qty' link (ID: 793) for sorting; (iii) triggering the 'Export' button (ID: 771); (iv) refreshing the view via 'Show Report' (ID: 714); and (v) interacting with the 'From' date picker (ID: 750).

Table 9: Action proposals and Reward scores.

#	**Action Proposal**	$\mathcal{F}_V$
1	`select_option('741', 'Year')`	0.571
2	`click('793')`	0.485
3	`click('771')`	0.385
4	`click('714')`	0.381
5	`click('750')`	0.379

Proposal 1 achieves the highest score 0.571 which is larger than current $\tau_{R,k}$, demonstrating that the scoring mechanism correctly identifies parameter configuration as a mandatory prerequisite to terminal actions like exporting or refreshing the report.

F.3 SYMBOLIC STATE AND ACTION

The following presents the symbolic state and action representations corresponding to the `select_option('741', 'Year')` operation.

```
Symbolic action:
action = {
    "action_type": "select_option",
    "target_uid": "741",
    "element_name": "Period",
    "action_parameter": "Year",
    "predicate_verifier": ["value='Year'"]
}

Symbolic state:
state = {
    "target_element": {
        "node_id": 741,"role": "combobox","name": "Period","value": "
            Year",
    },
    "local_structure": {
        "parent": {"type": "group", "name": "Filter", "node_id": 731},
        "children": [
            {"type": "option", "name": "Day", "node_id": 742},
            {"type": "option", "name": "Month", "node_id": 743},
            {"type": "option", "name": "Year", "node_id": 744}
        ],
        "neighbors": [
            {"type": "LabelText", "name": "Period", "node_id": 738},
            {"type": "textbox", "name": "From *", "node_id": 750},
        ]}}
```

G Cross-Site Generalization Tasks

G.1 Amazon Tasks

Amazon Cross-Site Generalization Tasks (39 no-login)

Search & Filter (20)

001 Search for an "ergonomic office chair" from the brand 'Herman Miller' that costs between $500 and $1000.

002 Find "27-inch 4k 144hz gaming monitors" that are eligible for Prime shipping.

003 Look for an "air fryer toaster oven combo" with at least a 20-quart capacity and a customer rating of 4 stars or higher.

004 Search for new "Sony WH-1000XM5 headphones" in the color 'Silver'.

005 Find "silicone pet grooming gloves" designed specifically for cats with long hair.

006 Search for 'Apple' "laptops" with at least 16GB of RAM and sort them by price from high to low.

007 Find "robotic vacuums" from the brand 'iRobot' with a self-emptying feature, and sort the results by average customer review.

008 Search for 'Vitamix' "blenders" with at least 1200 watts of power and sort them by price from low to high.

009 Look for "science fiction books" by author 'Andy Weir' in paperback format and sort them by publication date, with newest first.

010 Search for waterproof "men's winter jackets" in size 'Large' and sort by newest arrivals.

011 Find "android tablets" with at least 64GB of storage that cost less than $200.

012 Show me "men's running shoes" from the brand 'Brooks' in size 10.5.

013 Find programmable "coffee makers" under $75 that have a 4-star rating or higher.

014 Look for "QLED 4K televisions" from the brand 'Samsung' with a screen size of '65 inches'.

015 Find waterproof "women's hiking boots" from the brand 'Merrell' in size 8 that are eligible for Prime shipping.

016 Search for "Whey Isolate protein powder", filter by the flavor 'Unflavored', and a customer rating of 4 stars and up.

017 Find "board games" suitable for "2 players" and for ages "12 and up" with an average customer review of at least 4 stars.

018 Look for color-changing "smart home light bulbs" that are compatible with "Amazon Alexa".

019 Find the cheapest "1TB NVMe SSD" with a minimum read speed of 5,000MB/s.

020 From the brand 'Ninja', show me the least expensive "air fryer" with a minimum capacity of 6 quarts

Product Info (10)

021 Find the current price and the screen size in inches of the "Kindle Paperwhite (16 GB)".

022 Go to the product page for the "Instant Pot Duo 7-in-1" and find both its capacity in quarts and its item weight.

023 On the product page for the "Anker PowerCore 10000" power bank, find reviewers who mention the word "travel" and list their names along with the star rating they gave.

024 For the "Sony WH-1000XM5" headphones, summarize the main criticisms mentioned in the 1-star and 2-star reviews.

Amazon Cross-Site Generalization Tasks (39 no-login, Cont.)

025 For the "Breville Barista Express Espresso Machine", find positive customer reviews (4 stars or higher) that specifically talk about the "steam wand".

026 Summarize the positive comments from 5-star reviews for the "Kindle Scribe" that specifically mention the "writing experience".

027 Summarize what customers in 3-star reviews say about the battery life of the "Apple AirPods Pro (2nd Generation)".

028 Find the product page for the book "Dune" by Frank Herbert, list the available formats, and find the price of the Mass Market Paperback edition.

029 Check the product page for the "Logitech MX Master 3S" mouse to see if it is compatible with both macOS and Windows 11.

030 Find the "LEGO Star Wars Millennium Falcon" set (model 75257) and list its item dimensions and the manufacturer's recommended age range.

Site Navigation (9)

031 Navigate to the "Today's Deals" section and then filter to see only deals in the "Electronics" category.

032 Go to the "Best Sellers in Books" page and then navigate to the "Science Fiction & Fantasy" sub-category.

033 Show me the new releases in the "Electronics" category from the "Last 30 days".

034 Go to the Amazon "Gift Cards" page and then find the "eGift Cards" section.

035 Navigate to the customer service section and find help related to "A delivery, order or return".

036 Find Amazon's return policy page and determine the return window for a new television.

037 Find the help page on tracking packages and identify what to do if the tracking information is not updating.

038 Navigate to the "Coupons" section of the website and filter to see coupons for "Grocery & Gourmet" products.

039 Find "Amazon Basics" products in the "Home & Kitchen" category and sort the results by newest arrivals.

G.2 TARGET TASKS

Target Cross-Site Generalization Tasks (36 no-login)

Search & Filter (20)

001 Search for an "ergonomic office chair" from the brand 'Threshold' that costs between $100 and $250.

002 Find "27-inch 4K computer monitors" that are available for same-day delivery.

003 Look for an "air fryer toaster oven combo" with at least a 20-quart capacity and a guest rating of 4 stars or higher.

004 Search for new "Beats Studio Pro headphones" in the color 'Navy'.

005 Find "kitchen towels" from the brand 'Hearth & Hand with Magnolia' that are 100% cotton.

006 Search for "laptops" with at least 16GB of RAM from the brand 'HP' and sort them by price from high to low.

007 Find "robotic vacuums" from the brand 'iRobot' with a self-emptying feature, and sort the results by "Best sellers".

008 Search for "single-serve coffee makers" from the brand 'Keurig' and sort them by price from low to high.

009 Look for "LEGO Creator 3-in-1" sets and sort them by "Newest".

010 Search for "men's winter jackets" that are waterproof and in size 'Large', then sort by guest rating.

011 Find "patio conversation sets" that include a fire pit and cost less than $750.

012 Show me "women's athletic shorts" from the brand 'All in Motion' in a size 'Medium'.

013 Find "electric toothbrushes" from the brand 'Philips Sonicare' with a guest rating of 4 stars or higher.

014 Look for "OLED 4K smart TVs" from the brand 'Sony' with a screen size of '65 inches', and filter for "Order Pickup".

015 Find "men's hiking boots" from the brand 'Merrell' in size 10 that are available for "Same Day Delivery".

016 Search for "blenders", filter for the 'Ninja' brand, models with Auto-iQ, and a guest rating of 5 stars.

017 Find "nursery gliders" from the brand 'DaVinci' that are currently on sale and available in the color 'Gray'.

018 Look for "Nintendo Switch" games that are rated "Everyone 10+" and have the "Action" genre selected.

019 Find the cheapest "8-cube organizer shelf" from the brand 'Brightroom' available in 'White'.

020 Show me the least expensive "carry-on luggage" that has a hardside exterior, spinner wheels, and is from the brand 'Made By Design'.

Product Info (9)

021 Find the current price and overall dimensions of the "Threshold designed with Studio McGee Herriman Wooden Console Table".

022 Go to the product page for the "Keurig K-Mini Single-Serve Coffee Maker" and find out if it has an auto shut-off feature and its water tank capacity from the item details.

023 On the product page for "Good & Gather Organic Blue Corn Tortilla Chips", check the nutrition details to see the amount of sodium and dietary fiber per serving.

Target Cross-Site Generalization Tasks (36 no-login, Cont.)

024 For the "Beats Studio Pro" headphones, go to the Q&A section and find out the reported battery life and if they come with a carrying case.

025 Find customer reviews for the "Dyson V8 Origin Cordless Stick Vacuum" that mention its performance on both "pet hair" and "hardwood floors".

026 What do customers in the 1- and 2-star reviews say about the fit and fabric quality of the "A New Day Women's High-Rise Slim Fit Ankle Jeans"?

027 Find the product page for the book "Fourth Wing" by Rebecca Yarros. What is the listed genre and the total number of pages in the item details?

028 Check the product page for the "Apple iPad 10.9-inch (10th Generation)" to see how many colors it is available in and if it is compatible with the 1st generation Apple Pencil.

029 Find the "Fisher-Price Laugh & Learn Smart Stages Piggy Bank" toy and identify the manufacturer's suggested age range and the number of batteries required from the product details.

Site Navigation (7)

030 Navigate to the "Weekly Ad" section and find a deal on "Good & Gather" brand ground beef.

031 Go to the "Clearance" section of the website and filter for "Home Deals" that are discounted by 50% or more.

032 Go to the "Gift Cards" page and find the section for "Thank You" themed e-gift cards.

033 Find the store locator page and check the guest service hours and Starbucks hours for the Target store in Somerville, MA.

034 Find information on Target's return policy for electronics that have been opened and used.

035 From the homepage, navigate to the "Toys" category and filter for "Action Figures & Playsets" from the brand "Marvel".

036 Find the "Top Deals" in the "Home" category and sort them by "Discount: High-low". .