

From Trajectories to Workflows: Online Procedural Memory Abstraction for Language Agents

Swathi Shree Narashiman^{a*}, Ishan Jindal^b, Mahesh Chandran^b

^aStanford University, ^bFujitsu Research of India
narashim@stanford.edu, {ishan.jindal, mahesh.chandran}@fujitsu.com

Abstract

Procedural memory for language agents typically accumulates as trajectories, scripts, or task-specific skills, so memory grows with the number of interactions rather than the number of distinct procedures learned. We propose a workflow-centric procedural memory framework that continually consolidates repeated experience into a bounded set of natural-language workflows. These workflows are merged online under outcome-conditioned update policies. On ALFWorld and ScienceWorld, our memory is competitive with trajectory-level procedural memory on task success while using $3.5\times$ fewer memory tokens on average ($1.1\text{--}9.1\times$ across settings), showing that representation besides construction, retrieval and updating is also a key design axis for scalable procedural memory.

1 Introduction

1.1 Memory-Augmented Language Agents

Recent advances in LLM agents have enabled increasingly capable systems for long-horizon planning, web navigation, software engineering, and embodied decision making (Zhou et al., 2024a,b; Yang et al., 2024; Wang et al., 2024a; Ouyang et al., 2026; Senrayan et al., 2026). Since these tasks often exceed the context window of the underlying language model, agents increasingly rely on persistent external memory to accumulate knowledge across interactions, commonly distinguished into *episodic*, *semantic*, and *procedural* memory (Park et al., 2023; Zhong et al., 2023; Packer et al., 2024; Wang et al., 2023; Fang et al., 2026a). Among these, procedural memory has emerged as a key component for long-horizon reasoning, capturing reusable execution strategies that can be extracted, retrieved, and refined across interactions (Fang

et al., 2026b; Mi et al., 2026; Cai et al., 2026; Wang et al., 2024b).

1.2 The Problem: Procedural Memory Only Grows

Despite recent progress, existing procedural memory systems fundamentally remain accumulative. Trajectories, scripts, skills, or induced workflows are continually extracted and added to memory over time (Fang et al., 2026b; Mi et al., 2026; Cai et al., 2026; Wang et al., 2024b). While recent methods increasingly learn higher-level procedural abstractions, these abstractions remain additional memory entries rather than replacements for the underlying memory representation. For example, Agent Workflow Memory also induces and integrates reusable workflows from experience (Wang et al., 2024b). Our focus instead is on continual consolidation: we maintain a persistent workflow for each task family and update it with subsequent trajectories, explicitly studying whether replacing experience-level memory with evolving procedural abstractions can bound memory growth over time.

As an agent repeatedly solves similar tasks, successful executions often differ only in incidental details such as object locations, environment states, or interaction order while sharing nearly identical procedural structure. Existing systems do not explicitly consolidate these recurring procedures into a shared memory representation. Consequently, memory size continues to scale with accumulated experiences rather than the number of distinct procedures the agent has learned, forcing retrieval over increasingly redundant experiences.

1.3 Our Perspective: Consolidation as Representation

Human procedural memory does not accumulate this way. Studies of human memory suggest that repeated experience is progressively consolidated into schemas that preserve shared structure while

^{*}Work conducted during an internship at Fujitsu Research of India.

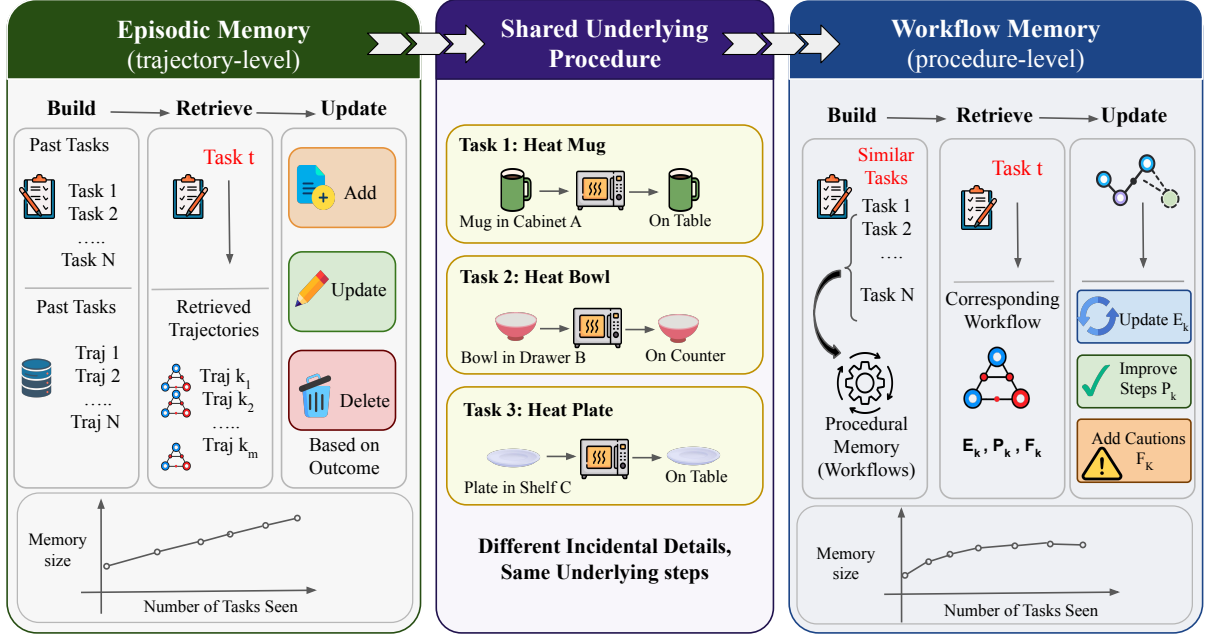


Figure 1: **Overview of workflow-centric procedural memory.** (Left) Trajectory-level methods retain individual experiences, so memory grows with interactions. (Middle) Tasks with different objects or states often share the same procedure. (Right) Our method consolidates such experiences into reusable workflows refined online. Each workflow contains environment knowledge (E_k), a high-level procedure (P_k), and failure guidance (F_k).

discarding incidental detail (Bartlett, 1932; Carbon and Albrecht, 2012; Tse et al., 2007; van Kesteren et al., 2012). New instances overwrite and refine existing structure rather than sitting alongside it indefinitely. We treat this as a design principle, not an analogy: instead of storing trajectories and abstracting over them separately, the abstraction *is* the procedural memory. We instantiate this as a workflow-centric framework in which each new trajectory merges into an existing workflow. This shifts memory from storing experiences to storing reusable procedural abstractions, reducing memory growth from the number of observed trajectories to the number of distinct procedures.

Figure 1 illustrates this shift from trajectory-centric procedural memory to workflow-centric procedural memory.

Our contributions are threefold:

- We propose a workflow-centric representation for procedural memory that consolidates repeated interactions into reusable workflows with bounded memory growth.
- We develop an online framework for constructing, updating, and retrieving workflow memories, and study the effects of workflow organization and outcome-conditioned updates.

- We show on ALFWorld and ScienceWorld that workflow memory is competitive with trajectory-level procedural memory while using substantially fewer memory tokens.

2 Reusable Workflows

We represent procedural memory as a set of evolving natural-language workflows. Each workflow summarizes reusable knowledge accumulated from multiple interactions and is provided to the agent as additional guidance at inference time. The underlying language model is not fine-tuned; adaptation occurs entirely through updates to the external workflow memory.

2.1 Workflow Representation

A workflow is a structured textual abstraction of procedural knowledge. It separates reusable environmental regularities, execution strategies, and failure guidance from the incidental details of individual trajectories. We write a workflow as

$$W_k = (E_k, P_k, F_k), \quad (1)$$

where E_k contains reusable task and environment knowledge, P_k describes the high-level procedure, and F_k contains cautions or failure patterns relevant to future execution. Workflow schema

Table 1: **Main Performance Results.** Results are Mean $\pm$ Std Dev. Within each backbone, **bold** denotes the best and underline the second-best result among No Memory, MemP, and TaskWF (Ours); Memory Tokens excludes No Memory, as zero usage isn’t comparable. **TaskWF** (ours) is our workflow-centric procedural memory under Vanilla and Validation update policies; **MemP** (Fang et al., 2026b) is a trajectory-level baseline; **No Memory** is a memory-free baseline. TaskWF is competitive with MemP’s success rate/score on ALFWorld and ScienceWorld-Easy for both backbones, using 1.1–9.1 $\times$ fewer memory tokens overall (3.5 $\times$ on average).

Backbone	Memory Method	Update Policy	ALFWorld		ScienceWorld (Easy)		ScienceWorld (Difficult)	
			Success Rate $\uparrow$	Memory Tokens $\downarrow$	Score $\uparrow$	Memory Tokens $\downarrow$	Score $\uparrow$	Memory Tokens $\downarrow$
Qwen2.5-32B Instruct	No Memory		0.645 $\pm$ 0.015	0 $\pm$ 0	57.9 $\pm$ 0.73	0 $\pm$ 0	34.0 $\pm$ 0.30	0 $\pm$ 0
	MemP	Vanilla	0.795 $\pm$ 0.017	42,634 $\pm$ 161	50.9 $\pm$ 1.62	33,016 $\pm$ 322	33.6 $\pm$ 0.77	33,721 $\pm$ 196
		Validation	0.783 $\pm$ 0.024	39,197 $\pm$ 299	53.6 $\pm$ 0.96	19,889 $\pm$ 220	31.7 $\pm$ 0.98	16,965 $\pm$ 254
	TaskWF (Ours)	Vanilla	0.800 $\pm$ 0.021	7,430 $\pm$ 1673	54.5 $\pm$ 1.09	12,894 $\pm$ 71	31.1 $\pm$ 1.44	12,528 $\pm$ 252
		Validation	0.788 $\pm$ 0.012	8,435 $\pm$ 1047	57.2 $\pm$ 0.68	11,901 $\pm$ 154	33.0 $\pm$ 0.18	11,785 $\pm$ 56
GPT-5.1-mini	No Memory		0.719 $\pm$ 0.009	0 $\pm$ 0	46.0 $\pm$ 0.36	0 $\pm$ 0	23.5 $\pm$ 0.57	0 $\pm$ 0
	MemP	Vanilla	0.814 $\pm$ 0.015	55,016 $\pm$ 269	52.0 $\pm$ 1.70	45,651 $\pm$ 507	29.5 $\pm$ 1.22	45,381 $\pm$ 294
		Validation	0.805 $\pm$ 0.015	50,139 $\pm$ 374	53.5 $\pm$ 1.21	31,160 $\pm$ 299	29.3 $\pm$ 3.08	27,310 $\pm$ 111
	TaskWF (Ours)	Vanilla	0.800 $\pm$ 0.023	6,055 $\pm$ 258	47.4 $\pm$ 1.95	25,492 $\pm$ 393	20.4 $\pm$ 2.78	26,034 $\pm$ 222
		Validation	0.819 $\pm$ 0.039	5,841 $\pm$ 81	48.8 $\pm$ 3.06	23,659 $\pm$ 291	19.9 $\pm$ 2.13	25,544 $\pm$ 176

and prompting details are provided in Appendix C. Appendix C. Appendix E illustrates this evolution from an early to a later-stage workflow for one ALFWorld task family, explaining how successful and failed trajectories modify the workflow.

2.2 Online Workflow Construction

Workflow memory is constructed incrementally as the agent interacts with the environment. Given an interaction (q_i, τ_i, r_i) : query, trajectory and reward, the workflow builder first produces a task-level summary,

$$s_i = \mathcal{B}(q_i, \tau_i, r_i), \quad (2)$$

Further, the summary is merged into an existing workflow,

$$W_k^{(i)} = \mathcal{U}\left(W_k^{(i-1)}, s_i, r_i\right), \quad (3)$$

where W_k denotes the workflow associated with memory unit k . In this work, k corresponds to a task family.

2.3 Online Workflow Update Policies

The update rule in Eq. 3 is shared across policies; what differs is which interactions (q_i, τ_i, r_i) are used to update the workflow.

The outcome signal r_i guides workflow construction. Successful trajectories contribute reusable procedural knowledge, whereas failed trajectories are summarized as cautions and failure guidance.

Following MemP (Fang et al., 2026b), we study two outcome-conditioned update policies:

Vanilla. Every trajectory updates the workflow, regardless of task outcome.

Validation. Only successful trajectories are used for workflow updates.

These policies isolate the effect of outcome supervision while keeping the workflow representation and retrieval mechanism unchanged.

2.4 Workflow Organization and Retrieval

In this work, memory consists of one workflow per task family. Given an instruction q_i , a task classifier retrieves the corresponding workflow $\mathcal{R}(q_i, \mathcal{M})$, which is provided as additional context during execution. The agent then generates actions according to

$$a_i^t \sim \pi_\theta\left(a \mid q_i, o_i^{\leq t}, \mathcal{R}(q_i, \mathcal{M})\right), \quad (4)$$

where the model parameters θ remain fixed. The retrieved workflow serves as reusable procedural guidance rather than an exact action sequence, allowing the agent to adapt the learned procedure to the current environment state.

3 Experiments

3.1 Setup

We evaluate on **ALFWorld** (Shridhar et al., 2021) and **ScienceWorld** (Wang et al., 2022), two text-

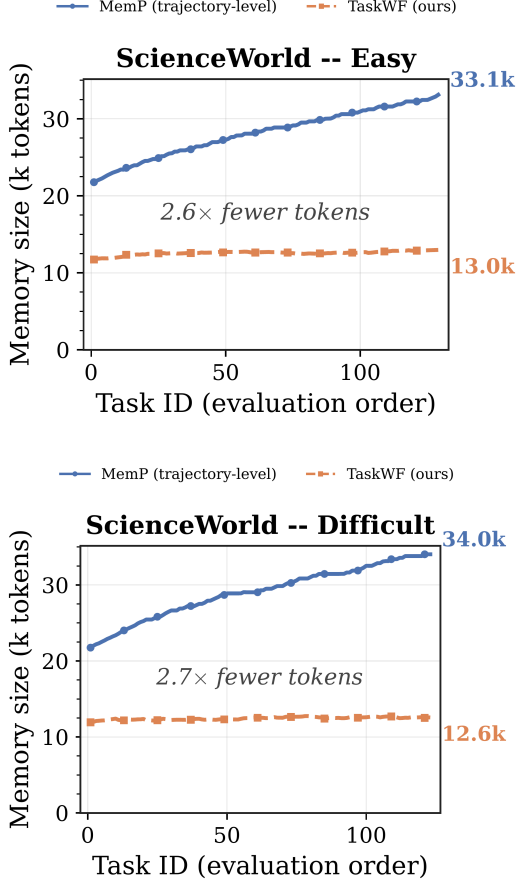


Figure 2: **Memory growth during ScienceWorld evaluation using Qwen2.5-32B-Instruct.** Top: ScienceWorld-Easy. Bottom: ScienceWorld-Difficult. TaskWF consolidates repeated experiences into workflows, substantially reducing memory growth compared with trajectory-level procedural memory (MemP).

based interactive environments organized into task families that share procedural structure but vary in incidental details (object placement, layout, substances/instruments involved), and both provide a per-episode outcome signal required to compare our update policies (§2.3). We restrict evaluation to these two environments as they combine sufficient within-family variation and a usable reward signal.

We use **Qwen2.5-32B-Instruct** and **GPT-5.1-mini** via the OpenAI API as the fixed backbone language models for both task execution and workflow construction/update, without fine-tuning. We compare against a no-memory baseline and a trajectory-accumulation baseline representative of prior experience-centric procedural memory (Fang et al., 2026b). We report task success on held-out instances and memory size as a function of interactions seen to evaluate whether workflow consolidation yields bounded memory growth relative to

trajectory accumulation. Memory size is measured by tokenizing the persistent external memory using the corresponding backbone language model. Full task families, hyperparameters, and experimental protocols are provided in Appendix A.

3.2 Results

Table 1 and Figure 2 evaluate whether workflow-centric procedural memory achieves the central objective of this work: reducing memory growth through workflow consolidation while preserving downstream task performance.

Workflow consolidation substantially reduces memory growth. Our primary hypothesis is that procedural memory should scale with the number of reusable procedures rather than the number of observed trajectories. Figure 2 confirms this behavior. Whereas trajectory-level memory (MemP) grows steadily as new interactions are accumulated, TaskWF remains nearly constant throughout evaluation because repeated experiences are merged into existing workflows instead of stored independently. This translates to approximately 2.6–2.7× fewer memory tokens on ScienceWorld, while Table 1 shows reductions of up to 9× across all benchmarks. Appendix E illustrates this qualitatively with an early- vs. late-stage workflow example.

Compact workflow memory preserves task performance. Despite using substantially less memory, TaskWF remains competitive with trajectory-level procedural memory across both environments. On ALFWorld, TaskWF performs on par with MemP for both Qwen2.5-32B-Instruct and GPT-5.1-mini while requiring roughly 5–9× fewer memory tokens. On ScienceWorld-Easy, TaskWF remains competitive under both backbones while using approximately 2–3× less memory. These results indicate that consolidating repeated experiences removes substantial redundancy without eliminating the procedural information needed for successful task execution.

3.3 Update Policy and Interaction Efficiency

Outcome-conditioned updates matter more for challenging environments. The effect of the update policy depends on task difficulty. On ALFWorld, Vanilla and Validation updates perform similarly, whereas Validation consistently improves the overall ScienceWorld score across both backbones and both memory methods (Table 2). The largest gain occurs for Qwen2.5-32B-Instruct with

Table 2: **ScienceWorld aggregate performance and interaction efficiency.** ScienceWorld Overall is the average score across the Easy and Difficult partitions. Steps denote the average number of interaction steps. Validation improves the overall ScienceWorld score for both memory methods and both backbones. TaskWF incurs comparable or fewer interaction steps than trajectory-level memory despite substantially lower memory usage.

Backbone	Method	Update	ScienceWorld Overall $\uparrow$	ALFWorld Steps	ScienceWorld Steps
Qwen2.5-32B Instruct	MemP	Vanilla	42.28	14.38 ± 0.21	24.36 ± 1.67
		Validation	42.67	14.70 ± 0.46	23.16 ± 0.58
	TaskWF	Vanilla	42.83	14.11 ± 0.55	23.46 ± 0.41
		Validation	45.09	14.21 ± 0.29	22.30 ± 1.07
GPT-5.1-mini	MemP	Vanilla	40.73	13.24 ± 0.21	10.14 ± 0.33
		Validation	41.40	13.40 ± 0.73	10.80 ± 0.24
	TaskWF	Vanilla	33.94	13.72 ± 0.60	8.09 ± 1.05
		Validation	34.34	13.32 ± 1.00	7.72 ± 0.42

TaskWF, where the overall ScienceWorld score improves from 42.83 to 45.09. This suggests that filtering unsuccessful trajectories becomes more useful as procedural complexity increases.

The benefit of outcome-conditioned updates, however, does not fully resolve performance degradation in the most difficult setting. On ScienceWorld-Difficult with GPT-5.1-mini, both TaskWF variants underperform No Memory (20.4 and 19.9 vs. 23.5), whereas MemP remains above the No Memory baseline (29.5 and 29.3; Table 1). SwiftSage (Lin et al., 2023) reports similar degradation on harder ScienceWorld variations, suggesting that part of the performance drop is intrinsic to the environment. Difficulty alone, however, cannot explain why TaskWF falls below No Memory while MemP does not. Moreover, the memory-token budgets of TaskWF and MemP on this split do not provide an explanation for the gap. We therefore leave identifying its source to future work.

Workflow consolidation does not increase interaction length. The memory reduction achieved by TaskWF does not come at the cost of longer agent trajectories. On ALFWorld, its average interaction length is comparable to MemP for both backbones. On ScienceWorld, TaskWF is similarly comparable for Qwen2.5-32B-Instruct and requires fewer interaction steps for GPT-5.1-mini: 7.72–8.09 steps compared with 10.14–10.80 for MemP (Table 2). Thus, despite storing substantially fewer memory tokens, workflow consolidation does not make the agent require longer interactions to execute the retrieved procedural guidance.

4 Conclusions

We introduced workflow-centric procedural memory, which consolidates repeated agent experience into reusable workflows instead of accumulating trajectories. Across ALFWorld and ScienceWorld, workflow memory achieves comparable task performance while requiring substantially fewer memory tokens, demonstrating that representation is also an important design axis for scalable memory.

Limitations

Our evaluation is limited to environments with well-defined task families, allowing a separate workflow to be maintained for each family. Extending workflow-centric memory to more open-ended agents will require automatically discovering task boundaries and determining when new experiences should update an existing workflow or initialize a new one. Additionally, because workflows are represented and updated in natural language, consolidation quality depends on the underlying language model and may degrade over prolonged updates.

References

- Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. 2026. [jina-embeddings-v5-text: Task-targeted embedding distillation](#). *Preprint*, arXiv:2602.15547.
- Frederic Charles Bartlett. 1932. Remembering: A study in experimental and social psychology.
- Yishuo Cai, Xingyu Guo, Xuancheng Huang, Jinhua Du, Can Huang, Wenxuan Huang, Wenhao Ma, Yuyang Hu, Aohan Zeng, Jie Tang, and Xu Sun. 2026. [From player to master: Enhancing test-time learning of llm agents via reinforcement learning over memory](#). *Preprint*, arXiv:2606.08656.

- Claus-Christian Carbon and Sabine Albrecht. 2012. [Bartlett’s schema theory: The unreplicated “portrait d’homme” series from 1932](#). *Quarterly Journal of Experimental Psychology*, 65:2258–2270.
- Jizhan Fang, Xinle Deng, Haoming Xu, Ziyang Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. 2026a. [Lightmem: Lightweight and efficient memory-augmented generation](#). *Preprint*, arXiv:2510.18866.
- Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2026b. [Memp: Exploring agent procedural memory](#). *Preprint*, arXiv:2508.06433.
- Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. [Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks](#). *Preprint*, arXiv:2305.17390.
- Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. 2026. [Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents](#). *Preprint*, arXiv:2602.01869.
- 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. 2026. [Reasoningbank: Scaling agent self-evolving with reasoning memory](#). *Preprint*, arXiv:2509.25140.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. [Memgpt: Towards llms as operating systems](#). *Preprint*, arXiv:2310.08560.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simulacra of human behavior](#). *Preprint*, arXiv:2304.03442.
- Ganesh Senrayan, Moyuru Yamada, Ishan Jindal, and Kiran Purohit. 2026. [Exploratory and assimilating reflection: Reflective recall cycle for long-term memory](#). *Preprint*, arXiv:2607.17879.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. [Alfworld: Aligning text and embodied environments for interactive learning](#). *Preprint*, arXiv:2010.03768.
- Dorothy Tse, Rosamund Langston, Masaki Kakeyama, Ingrid Bethus, Patrick Spooner, Emma Wood, Menno Witter, and Richard Morris. 2007. [Schemas and memory consolidation](#). *Science*, 316:76–82.
- Marlieke van Kesteren, Dirk Ruiter, Guillén Fernández, and Richard Henson. 2012. [How schema and novelty augment memory formation](#). *Trends in neurosciences*, 35:211–9.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. [Voyager: An open-ended embodied agent with large language models](#). *Transactions on Machine Learning Research*.
- Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. [Scienceworld: Is your agent smarter than a 5th grader?](#) *Preprint*, arXiv:2203.07540.
- Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. [Augmenting language models with long-term memory](#). *Preprint*, arXiv:2306.07174.
- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. [Agent workflow memory](#). *Preprint*, arXiv:2409.07429.
- John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. [Swe-agent: Agent-computer interfaces enable automated software engineering](#). *Preprint*, arXiv:2405.15793.
- Wanjuan Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. [Memorybank: Enhancing large language models with long-term memory](#). *Preprint*, arXiv:2305.10250.
- Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024a. [Language agent tree search unifies reasoning, acting, and planning in language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160. PMLR.
- 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. 2024b. [Webarena: A realistic web environment for building autonomous agents](#). *Preprint*, arXiv:2307.13854.

A Experimental Details

A.1 Environments

We evaluate on ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022) because both satisfy two properties required by our study. First, they contain recurring task families whose instances share an underlying procedural structure while varying in incidental details (e.g., object placement, environment state, or task wording),

making them suitable for evaluating workflow consolidation. Second, both provide an explicit outcome signal required for the update policies introduced in Section 2.3. ALFWorld provides a binary success signal, whereas ScienceWorld reports a normalized task score between 0 and 100.

ScienceWorld consists of 30 task types organized into 10 scientific topics: (i) Matter (Changes of State), (ii) Measurement, (iii) Electricity, (iv) Classification, (v) Life Cycle, (vi) Mixtures, (vii) Lifespan, (viii) Life Stages, (ix) Inclined Plane & Friction, and (x) Mendelian Genetics.

Following SwiftSage (Lin et al., 2023), some ScienceWorld episodes terminate after an unrecoverable invalid action and receive a negative environment score. In these cases, we report the last non-negative score obtained before termination.

A.2 Implementation Details

Within each experiment, the same backbone language model (Qwen2.5-32B-Instruct or GPT-5.1-mini) is used for workflow construction, workflow updates, and task execution without fine-tuning.

Workflow retrieval uses jina-embeddings-v5-text-small (Akram et al., 2026). During execution, the task instruction is mapped to its corresponding task family and the associated workflow is retrieved.

ALFWorld does not provide task-family annotations. We therefore classify each instruction into one of six task families using the same backbone language model: (i) Pick & Place, (ii) Pick Clean & Place, (iii) Pick Cool & Place, (iv) Pick Heat & Place, (v) Pick Two Objects & Place, and (vi) Look At Object In Light.

ScienceWorld provides task-family annotations, which are used directly for workflow construction and retrieval.

A.3 Workflow Initialization

Workflow memories are initialized through an offline cold-start stage before online updates begin. Cold-start trajectories are used only to initialize workflow memory and are never included in evaluation.

For ALFWorld, the initial workflow memory is constructed from 300 trajectories distributed across the six task families.

For ScienceWorld, the initial workflow memory is constructed from 144 trajectories, restricting initialization to at most 10 trajectories per task family.

A.4 Evaluation Protocol

After cold-start initialization, workflow memories are updated online throughout evaluation.

ALFWorld is evaluated on the development test split containing 140 task instances.

For ScienceWorld, we additionally construct Easy and Difficult evaluation partitions to distinguish between tasks that are semantically similar and those requiring greater procedural generalization. We embed both the cold-start and evaluation task instructions using jina-embeddings-v5-text-small, and use FAISS to measure the semantic distance between each evaluation task and its nearest cold-start example. Tasks with FAISS distance below 0.5 are assigned to the Easy partition, while those above 0.5 form the Difficult partition. The resulting evaluation consists of 129 Easy tasks and 123 Difficult tasks, restricting evaluation to at most 10 instances per task family.

To ensure a fair comparison across methods, the evaluation order is kept identical across all experiments so that every memory method observes the same sequence of interactions. The results reported in Table 1 are the mean and standard deviation over 3 repeated runs under this protocol.

B Task Execution Prompts

This appendix provides the prompts used for agent action generation in each environment. Workflow construction and update prompts ($\mathcal{B}$, $\mathcal{U}$, Eq. 2–3) are provided in Appendix C.

B.1 ALFWorld

The following system prompt governs agent action generation in ALFWorld across all memory conditions (No Memory, MemP, TaskWF); retrieved memory content, when present, is inserted as additional context alongside this prompt.

```
Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish.
For each turn, you will be given the observation of the last turn. You should first think about the current condition and plan for your future actions, and then output your action in this turn. Your output must strictly follow this format:
Thought: your thoughts.
Action: your next action
The available actions are:
1. go to recep
2. take obj from recep
3. move obj to recep
4. open recep
5. close recep
6. use obj
7. clean obj with recep
8. heat obj with recep
9. cool obj with recep
where obj and recep correspond to objects and receptacles.
After each turn, the environment will give immediate feedback. If the environment outputs "Nothing happened", the previous action is invalid and you should try another option.
Your response should use the following format:
Thought: <your thoughts>
Action: <your next action>
```

B.2 ScienceWorld

The following system prompt governs agent action generation in ScienceWorld across all memory conditions.

```
You are an agent acting in a ScienceWorld text environment to solve the current task. At the beginning of an episode, you will receive the task name, task description, current observation, inventory, goal progress, and a sample of valid actions available in the current state.
On each turn, use the current observation, inventory, goal progress, previous action, and valid action sample to decide the next move.
Important rules:
- Choose one action only from the valid actions shown for the current state.
- Return exactly one action string and nothing else.
- Do not output thoughts, explanations, prefixes, bullet points, or multiple candidate actions.
- If the last action was invalid or had no useful effect, choose a different valid action instead of repeating it.
- If the state is unclear, inspect or navigate purposefully. Do not spam "look around" repeatedly when the observation is unchanged.
- Do not invent an action that is not in the valid action list, even if it sounds reasonable from the task description.
```

Note the two environments impose different output contracts: ALFWorld requires an explicit Thought/Action pair, while ScienceWorld requires a single bare action string with no reasoning trace. This distinction is relevant because $\mathcal{B}$ (Appendix C) must parse each format differently when constructing task-level summaries from raw trajectories

C Workflow Construction and Update Prompts

This appendix provides the prompts implementing $\mathcal{B}$ (Eq. 2) and $\mathcal{U}$ (Eq. 3) for each environment. $\mathcal{B}$ produces a task-level summary s_i from a single trajectory; $\mathcal{U}$ merges s_i into the existing workflow $W_k^{(i-1)}$. Both prompts are structured to populate the three workflow components introduced in Eq. 1: object/location and state-change knowledge populate E_k , the numbered procedure populates P_k , and the Cautions section populates F_k .

C.1 ALFWorld

$\mathcal{B}$: task-level summary. Given a raw trajectory, task category, query, and outcome, the following prompt produces a structured task-level summary.

```
You are analyzing raw ALFWorld trajectories and must combine knowledge from them only.
Task category: <category>
Task query: <raw query>
Trajectory outcome: <outcome>

Produce a task-level summary that captures reusable procedural knowledge.

Output plain text only, with exactly these sections in this order:

Common Object Search Knowledge:
- Start with a single compact header line:
  object | common locations ranked decreasingly
- Then list object knowledge under that header using compact lines like:
  cup | countertop, cabinet, drawer
  knife | countertop, drawer, diningtable
- Rank locations from most common / most plausible to less common.
- Try to cover as many object categories from the trajectory as possible.
- Keep the section compact and avoid repeating the same object twice unless the new line adds
  genuinely new location evidence.

State-Change Knowledge:
- If the trajectory involves heating, cooling, or cleaning, identify what appliance/receptacle
  seems usable for that operation.
- Use lines like:
  heat | microwave, stoveburner
  cool | fridge
  clean | sinkbasin

Placement Knowledge:
- Summarize placement knowledge compactly under this single header.
- Prefer one grouped line for destinations that need opening and one grouped line for destinations
  that do not, for example:
  open first | drawer, cabinet, fridge, microwave
  no open needed | countertop, diningtable, sinkbasin, desk
- If the trajectory gives strong object-to-destination evidence, you may add compact lines like:
  mug | coffeemachine
  plate | cabinet

Steps:
- Then provide numbered steps like 1. 2. 3.
- Keep the steps to at most <step limit>.
- The steps should summarize the common procedure for this task family.

Cautions:
- Under Cautions, provide short dash bullets.
- Include no-op, invalid-action, repeated-search, or repeated-placement failure patterns if they
  appear.
- Keep cautions to at most <caution limit>.

Important:
- Use only evidence from the raw trajectory below.
- Do not mention trajectory numbers.
- Do not mention object ids or room ids.
```

- Generalize names like drawer 1 to drawer, countertop 2 to countertop, etc.
- Keep object and placement sections compact rather than repetitive.
- If the trajectory failed, preserve useful partial knowledge and failure patterns, but do not rewrite it as a successful plan unless the successful step was demonstrated.

Raw trajectory:
<trajectory text>

$\mathcal{U}$: online merge. Given the existing workflow for a task category and a new task-level summary, the following prompt merges them into a single updated workflow. If no workflow yet exists for the category, the new summary is used directly to initialize W_k .

You are given an existing task-family workflow summary and a new task-level summary derived from a raw ALFWorld trajectory.

Task category: <category>

Combine them into one updated workflow text.

Output plain text only, with exactly these sections in this order:

Combined online workflow for <category>

Common Object Search Knowledge:

- Aggregate object-wise search knowledge across the existing workflow and new task summary.
- Start with a single compact header line:
object | common locations ranked decreasingly
- Then list compact lines like:
cup | countertop, cabinet, drawer
knife | countertop, drawer, diningtable
- Do not repeat prefixes like "Object:" on every line.
- Rank locations from most common / most reliable to less common.
- Try to cover as many object categories as possible.
- Keep the section compact and merge duplicate object knowledge instead of repeating the same object on many lines.

State-Change Knowledge:

- Aggregate which appliances or receptacles can be used for heating, cooling, or cleaning when those operations are relevant.
- Use lines like:
heat | microwave, stoveburner
cool | fridge
clean | sinkbasin

Placement Knowledge:

- Aggregate destination receptacles and whether they need to be opened first.
- Prefer grouped lines like:
open first | drawer, cabinet, fridge, microwave
no open needed | countertop, diningtable, sinkbasin, desk
- If supported, you may add compact object-to-destination lines like:
mug | coffeemachine
plate | cabinet

Steps:

- Then provide numbered steps like 1. 2. 3.
- Keep the steps to at most <step limit>.
- Combine the common procedure across the existing workflow and new task summary.

Cautions:

- Under Cautions, provide short dash bullets.
- Include failure/no-op/repeated-search/repeated-placement cautions when supported by either source.
- Keep cautions to at most <caution limit>.

Important:

- Preserve the object-location knowledge, state-change appliance knowledge, and placement/open-state knowledge from both sources.

- Treat failed-trajectory cautions as negative guidance, not as required steps.
- Keep object and placement sections compact rather than repetitive.
- Do not mention implementation details.

Existing workflow:
<existing workflow>

New task-level summary:
<new task summary>

C.2 ScienceWorld

B: task-level summary. Given a task name, query, trajectory, and reward, the following prompt produces a structured task-level summary. Relative to ALFWorld, the ScienceWorld prompt separates object roles from tool/device knowledge and adds explicit decision-rule, precondition, and verification-check sections, reflecting the greater procedural and diagnostic complexity of ScienceWorld tasks.

You are summarizing a ScienceWorld trajectory into reusable task-name-specific workflow guidance.
Task name: <task name>
Outcome: <successful / failed>
Query: <query>

Produce plain text only, with exactly these sections in this order:

Subgoal Sequence:

- List the stable ordered subgoals for this task name.
- Use compact numbered lines like:
 1. focus on instrument
 2. focus on target substance
 3. choose correct answer box
- Keep only subgoals that generalize across variations.

Object Roles:

- Aggregate reusable object-role knowledge rather than one-off object names.
- Use compact lines like:
 - instrument | thermometer
 - target object | unknown substance
 - answer container | colored answer box
 - mixing container | bowl, jug
 - power source | battery, solar panel
- If location knowledge is important, append it compactly:
 - instrument | thermometer | kitchen

Tool / Device / Test Knowledge:

- Summarize which tools, devices, or tests help complete which subgoal.
- Use compact lines like:
 - measure | thermometer
 - heat | stove, furnace
 - cool | freezer, fridge
 - conductivity test | battery, connectors, bulb
 - grow | soil, water, pot
- If not relevant, keep this section short.

Decision Rules:

- Summarize the rule that maps evidence or state to the final action.
- Include threshold, classification, ordering, or placement rules like:
 - above threshold -> yellow box
 - below threshold -> orange box
 - nonconductive -> blue box
 - make secondary color first -> then add remaining primary
- If this task has no strong final decision rule, say so briefly.

Preconditions:

- State what must already be true before an action or subgoal is useful.
- Use compact lines like:
 - before answer box focus | thermometer and substance already focused

before activate | components connected
before place object | class/result confirmed

Verification Checks:

- State how the agent can tell a subgoal is completed.
- Use compact lines like:
 - correct focus -> score increases or goal progress advances
 - correct state change -> observation mentions new state
 - correct setup -> device powers on
 - correct classification -> final placement subgoal becomes relevant

Steps:

- Provide 3 to 7 concise numbered steps for the common procedure.
- Use action-oriented steps such as inspect, navigate, pick up, pour, connect, activate, measure, compare, or place.
- If the trajectory failed, keep only steps that are still supported by the evidence.

Cautions:

- Provide short dash bullets for failure patterns, invalid actions, wrong target choices, loops, no-op actions, premature focusing, wrong ordering, or missing preconditions.
- If the trajectory failed, treat the failure as negative guidance. Explain what not to do in a reusable way.
- Prefer reusable anti-patterns such as:
 - do not focus on the container when the task requires the substance
 - do not choose the answer box before the measurement or classification is established
 - do not repeat the same zero-reward action

Important:

- Ground everything in the trajectory below.
- Keep it concise, specific to this task name, and useful across variations of the same task.
- Prefer reusable object roles and decision rules over exact one-off variation details unless the exact object type is essential.
- Failed trajectories should contribute more to Cautions, Preconditions, and Verification Checks than to Steps.
- Do not mention variation ids.
- Do not mention this instruction block.

Trajectory:

<trajectory>

U: online merge. Given the existing workflow for a task name and a new trajectory-derived summary, the following prompt merges them into a single improved workflow.

You are given an existing ScienceWorld task-name workflow and a new trajectory-derived summary for the same task name.
Task name: <task name>

Merge them into one improved workflow.

Output plain text only, with exactly these sections in this order:

Subgoal Sequence:

Object Roles:

Tool / Device / Test Knowledge:

Decision Rules:

Preconditions:

Verification Checks:

Steps:

Cautions:

Important:

- Preserve useful older knowledge unless the new summary clearly contradicts it.
- Merge reusable subgoal knowledge, object-role knowledge, tool/test knowledge, decision rules, preconditions, and verification checks rather than repeating them.
- Prefer concise, non-redundant guidance.
- Keep Steps to at most 7 numbered lines.
- Keep Cautions short and high-signal.

- Failed-trajectory evidence should mostly strengthen Cautions, Preconditions, and Verification Checks, not add bad steps.
- Remove one-off variation details when they do not generalize across the same task name.
- Keep the workflow operational: what subgoal comes first, what object roles matter, what tools or tests matter, what final rule matters, what preconditions matter, how to verify progress, and what mistakes to avoid.

Existing workflow:
<existing workflow>

New summary:
<new summary>

D MemP Baseline Prompts

For comparison with *B/U* (Appendix C), this appendix provides the prompt used to construct the MemP (Fang et al., 2026b) trajectory-level baseline. Given a query and trajectory, MemP directly produces a single free-form paragraph workflow, without our schema decomposition into E_k , P_k , and F_k . Each trajectory produces its own workflow entry, which is added to the MemP memory pool rather than merged into an existing entry; this is the source of the trajectory-level memory growth reported in §3.2 (Figure 2).

D.1 ALFWorld

You are provided with an ALFWorld household task query and a trajectory taken to solve the query. The trajectory consists of multiple steps of thought, action and observation. Your task is to generate a workflow based on critical steps to help solve similar ALFWorld queries in the future.

A critical step is one that has a significant impact on fulfilling the query, such as locating the target object, opening a needed receptacle, taking the correct object, applying the required state change (clean, heat, cool, inspect under light), and placing the object in the correct destination. Notice: Write the workflow as a natural, coherent paragraph (not as a bullet list or numbered steps). Use clear, concise language to describe what actions should be taken and in what general order.

—EXAMPLE WORKFLOW—

To solve this query, begin by searching the most likely receptacles or surfaces where the target object may be found, such as cabinets, drawers, countertops, tables, or appliances mentioned in the environment. If a receptacle is closed, open it before searching inside. Once the correct object is located, take it and complete any required state change, such as cleaning it in the sinkbasin, heating it in the microwave, cooling it in the fridge, or making sure the relevant light source is on before inspection. Then go to the target destination, open it first if needed, and place or inspect the object in the required way. If the object is not found in the first location, continue searching other likely locations instead of repeating ineffective actions.

—EXAMPLE END—

Query:

<query>

Trajectory:

<trajectory>

Output the workflow without any explanation or context: Be concise. Do not explain.

D.2 ScienceWorld

You are provided with a ScienceWorld task query and a trajectory taken to solve the query. The trajectory consists of multiple steps of observation, action, reward, and score updates. Your task is to generate a reusable workflow that helps solve similar ScienceWorld tasks in the future.

A critical step is one that helps identify the relevant objects, gather measurements or evidence, change the state of objects correctly, navigate to the right location efficiently, and verify that the task goal is satisfied.

Notice: Write the workflow as a natural, coherent paragraph (not as a bullet list or numbered steps). Use clear, concise language to describe what actions should be taken and in what general order.

—EXAMPLE WORKFLOW 1—

To solve this type of ScienceWorld task, first inspect the environment and identify the objects that are directly relevant to the task description. Gather any needed tools or containers before

attempting state changes or measurements. Use observations, inventory, and goal progress to verify that each intermediate step is helping the task rather than repeating unproductive actions. When the task requires a physical change such as heating, cooling, mixing, measuring, or placing an object somewhere, complete that change carefully and then re-check the environment to confirm the effect. If progress stalls, revisit the task description and search nearby rooms or containers for missing required objects.

—EXAMPLE WORKFLOW 2—

To solve this kind of ScienceWorld comparison or classification task, begin by locating the objects or organisms named in the task and examining them one by one to gather distinguishing evidence. Move between rooms only when the task or observations suggest the next relevant location, and use inventory or containers only if they are needed to carry, combine, or test objects. Keep checking the goal progress after each meaningful action so you know whether the task expects more evidence, a state change, or a final ordering or identification. If an action fails, switch to a nearby inspection, navigation, or manipulation action that is consistent with the current observation instead of repeating the same ineffective command.

—EXAMPLE END—

Query:

<query>

Trajectory:

<trajectory>

Output the workflow without any explanation or context: Be concise. Do not explain.

E Illustrative Workflow Evolution: Pick Clean & Place

To make the abstraction in Eq. 1 concrete beyond aggregate token counts, we present a hand-constructed illustrative example showing how a workflow for the ALFWorld *Pick Clean & Place* task family might evolve under our schema (§C.1), grounded in representative queries including (i) “*pick a clean plate and place on the countertop*”, (ii) “*pick some clean apples and place it in the fridge*,” and (iii) “*pick a clean mug and place it on the diningtable*.” This example is constructed by hand to match the exact output format specified in Appendix C.1 for expository purposes only; it is not a sampled system output.

Early-stage workflow (after cold-start initialization, 2 trajectories). With only the plate and apple trajectories observed, object knowledge is sparse and state-change knowledge has not yet been established, since neither cold-start trajectory required heating or cooling.

Illustrative Workflow Output

Combined online workflow for Pick Clean & Place

Common Object Search Knowledge:

object | common locations ranked decreasingly

plate | cabinet

apple | countertop

State-Change Knowledge:

clean | sinkbasin

Placement Knowledge:

no open needed | countertop

open first | fridge

Steps:

1. Locate the target object on a nearby surface or receptacle.
2. Take the object.
3. Go to the sinkbasin and clean the object.
4. Go to the destination receptacle, opening it first if needed.
5. Place the object at the destination.

Cautions:

- Do not place the object before cleaning it; the task requires the clean state to register before placement is accepted.

Update from a failed trajectory. Consider a subsequent query: “*pick a clean mug and place it on the diningtable*.” The agent searches for the mug by opening cabinets sequentially. After exhausting the available cabinets without finding the mug, however, it begins revisiting previously searched locations, repeating the same search cycle until reaching the maximum step limit.

Illustrative Trajectory Output

Illustrative failed trajectory:

Agent: The mug was not found in cabinet 1. I will check cabinet 2.
Environment: cabinet 2 contains plate 1, bowl 1, and cup 1.

Agent: The mug was not found in cabinet 2. I will check cabinet 3.
Environment: cabinet 3 contains plate 2 and bowl 2.

Agent: The mug was not found in cabinet 3. I will check cabinet 4.
Environment: cabinet 4 contains cup 2 and plate 3.

...

Agent: The mug was not found in cabinet 10. I will check cabinet 1.
Environment: cabinet 1 contains plate 1 and bowl 3.

Agent: The mug was not found in cabinet 1. I will check cabinet 2.
Environment: cabinet 2 contains plate 1, bowl 1, and cup 1.

Agent: The mug was not found in cabinet 2. I will check cabinet 3.
Environment: cabinet 3 contains plate 2 and bowl 2.

...

Until maximum number of steps reached

Rather than retaining the failed trajectory verbatim, the workflow generalizes the failure into a reusable search constraint and updates its **Cautions** section:

Illustrative Workflow Output

New Caution:

- Do not repeatedly revisit locations that have already been searched without finding the target object; prioritize unexplored likely locations unless the environment state has changed.

Update from a successful trajectory. On another *Pick Clean & Place* instance requiring a clean mug to be placed on the diningtable, the agent successfully finds the mug in the coffeemachine, cleans it at the sinkbasin, and places it on the diningtable. This successful trajectory contributes reusable object-location knowledge rather than a new caution.

Illustrative Trajectory Output

Illustrative successful trajectory:

Agent: I will search the coffeemachine for the mug.
Environment: coffeemachine contains mug 1.

Agent: I found mug 1. I will take it.
Environment: You pick up mug 1.

Agent: The mug must be clean before placement. I will clean it at the sinkbasin.
Environment: mug 1 is now clean.

Agent: I will place the clean mug on the diningtable.
Environment: Task completed successfully.

The successful trajectory is consolidated into the workflow's **Common Object Search Knowledge**, **State change knowledge** and **Placement knowledge**.

Illustrative Workflow Output

Updated Knowledge:

object | common locations ranked decreasingly
mug | coffeemachine

State-Change Knowledge:

clean | sinkbasin

Placement Knowledge:

no open needed | diningtable

Later-stage workflow (after 15+ merged trajectories across plates, apples, mugs, lettuce, and cups). Object coverage broadens substantially, and placement knowledge becomes more specific. State-change knowledge additionally records heat- and cool-relevant receptacles the agent incidentally passed while searching for objects in prior trajectories, even though heating/cooling is not itself required by this task family.

Illustrative Workflow Output

Combined online workflow for Pick Clean & Place

Common Object Search Knowledge:
object | common locations ranked decreasingly
plate | countertop, cabinet, diningtable
apple | fridge, countertop, garbagecan
mug | coffeemachine, countertop, cabinet
lettuce | fridge, countertop, sinkbasin
cup | cabinet, countertop, drawer

State-Change Knowledge:
heat | microwave, oven
cool | fridge, refrigerator
clean | sinkbasin

Placement Knowledge:
open first | fridge, cabinet, microwave
no open needed | countertop, diningtable, sinkbasin
apple | fridge
plate | countertop
mug | coffeemachine

Steps:

1. Search likely locations for the target object, prioritizing the highest-ranked location first.
2. Take the object once located.
3. Go to the sinkbasin and use “clean X with sinkbasin” to clean it.
4. Confirm the clean state before proceeding.
5. Go to the destination receptacle; open it first if it belongs to the “open first” group.
6. Place the object at the destination.

Cautions:

- Do not place the object at the destination before cleaning it; unclean placement does not satisfy the goal and wastes a turn.
- Repeated “clean X with sinkbasin” actions on an already-clean object return a no-op; move on to placement instead.
- Placement actions silently fail (“Nothing happened”) when attempted on an unopened receptacle; always open first if the destination belongs to the “open first” group.
- Do not confuse cleaning with cooling: apples and lettuce are frequently found in the fridge, but their presence there is unrelated to the clean-state requirement.

Comparing the two snapshots, the later-stage workflow covers $5\times$ more objects, adds a fully-populated State-Change Knowledge section, refines Placement Knowledge with object-specific destination lines, and accumulates cautions grounded in observed failure patterns (e.g., placement-before-cleaning, redundant cleaning, unopened receptacles) while remaining a single bounded workflow entry rather than a growing list of per-trajectory summaries, consistent with the memory-growth behavior reported in Figure 2.