AGENTIC LOCAL ENVIRONMENT ALIGNMENT VIA DECISION-ORIENTED KNOWLEDGE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents often operate in environments whose useful regularities are absent from task instructions: where objects are likely to be found, and which action procedures lead to success. These regularities can be learned from interaction, but trajectories also contain transient states and incidental actions that should not be carried across instances. We call regularities shared across related instances of a target setting local environment knowledge and organize them according to the decisions they support. Search decisions use probabilistic object-location relations, while execution decisions use procedural rules. We instantiate this representation in Structured Knowledge Learning (SKL), which keeps the LLM backbone frozen and learns both components from interaction: relations are estimated by combining commonsense priors with observation statistics across training instances, while rules are induced by contrasting trajectories with different returns. Both knowledge components are supplied to a frozen LLM agent at inference time. Under a shared evaluation protocol on ALFWorld and ScienceWorld, SKL substantially improves reward over prompting- and reflection-based baselines with GPT-4o-mini and Gemini-2.0-flash. On ALFWorld, for example, SKL improves reward from 62.6 to 81.3 with GPT-4o-mini and from 61.2 to 88.0 with Gemini-2.0-flash. Ablations show complementary gains from relations and rules, while budget and cost analyses show that these gains persist with fewer interactions and lower test-time token consumption. The learned knowledge also transfers across backbones: on ALFWorld, knowledge acquired with either GPT-4o-mini or Gemini-2.0-flash improves Gemini-2.5-flash, and GPT-4o-mini with SKL achieves higher reward than Gemini-2.5-flash without SKL.

1 INTRODUCTION

Large language model (LLM) agents increasingly act in interactive environments rather than only respond to static inputs (Wang et al., 2024b;a; Zhou et al., 2024; Koh et al., 2024; de Chezelles et al., 2025; Liu et al., 2025). Their performance therefore depends on knowledge that a task instruction may not provide. Some missing information can be retrieved from documents or the web (Singh et al., 2025; Li et al., 2025). In household environments and simulated laboratories, however, agents must also learn regularities of the environment in which they act: where objects tend to appear, which operations are available, and how those operations must be ordered. Such regularities are revealed through interaction with the target environment.

We call regularities that recur across related task instances *local environment knowledge*. They differ from transient information about the current instance and need not hold outside the target environment or task distribution. A trajectory can provide evidence for this knowledge, but it also records incidental actions and states specific to one attempt. Reusing trajectories effectively thus requires deciding which regularities to extract, how to represent them, and when an agent should use them. Existing methods preserve experience as memories, workflows, or skills (Shinn et al., 2023; Zhao et al., 2024; Fu et al., 2024; Chen et al., 2024; Wang et al., 2024a; 2025; Fang et al., 2026; Agrawal et al., 2026; Zhang et al., 2026a; Zhou et al., 2025), or incorporate it through updates to a policy, world model, or critic (Qiao et al., 2024; Zhu et al., 2025; Xiang et al., 2024; Tan et al., 2026; Zhang et al., 2026b). We study an external knowledge interface that makes the extracted regularities available to a frozen agent at the decisions they inform.

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.

Our starting point is that different decisions require different evidence. Consider an agent asked to cool an apple. It must first locate the apple and then perform the actions needed to cool it. To guide search, the agent needs estimates of where an apple is likely to appear in the target environment. These estimates can draw on object-location observations across task instances. To guide execution, the agent needs rules about valid actions and their order. Such rules can be inferred by comparing attempts at the same task that achieve different returns. Treating both needs as undifferentiated trajectory memories obscures the distinct evidence available for learning them. We therefore represent local environment knowledge in two components: probabilistic object-location relations for search and procedural rules for execution.

We instantiate this decision-oriented representation as **S**tructured **K**nowledge **L**earning (SKL), which keeps the LLM backbone frozen and learns both components from interaction experience, each from the evidence suited to its decision (Figure 1). For search, SKL estimates probabilistic object-location relations by Bayesian fusion of the agent's commonsense priors with observation statistics across training instances. For execution, it induces procedural rules through reflective rule evolution, in which an LLM diagnoses where a lower- and a higher-return trajectory on the same task diverge and proposes rule edits. A revised rule set is kept only if it improves aggregate return over the training set. At test time, both are provided to the agent as additional context. Since the learned knowledge is external to the model, we also study whether it transfers across backbones.

We evaluate SKL on ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022) with GPT-4o-mini (OpenAI, 2024) and Gemini-2.0-flash (Google, 2025). Adding the learned knowledge raises overall average reward on ALFWorld from 41.9 to 81.3 with GPT-4o-mini and from 45.0 to 88.0 with Gemini-2.0-flash, and on ScienceWorld from 33.2 to 53.6 and from 29.1 to 66.8. Ablations show that the two knowledge components provide complementary benefits. The gains persist under smaller interaction budgets, and SKL uses fewer test-time tokens than the controlled baselines. The learned knowledge also transfers across backbones: knowledge acquired with GPT-4o-mini or Gemini-2.0-flash improves Gemini-2.5-flash on ALFWorld from 74.6 to 85.3 and 86.8, respectively. GPT-4o-mini with SKL reaches 81.3, compared with 74.6 for Gemini-2.5-flash without learned knowledge. In summary, our contributions are threefold:

- We formulate local environment knowledge as cross-instance regularities and organize its representation around two agent decisions: where to search and how to execute a task.
- We introduce SKL, which learns probabilistic object-location relations from observation statistics and commonsense priors, learns procedural rules from same-task trajectory contrasts with aggregate-return validation, and supplies both to a frozen LLM agent.
- We show on ALFWorld and ScienceWorld that these components improve performance under controlled comparisons, provide complementary benefits, and can be reused across the evaluated agent backbones.

2 Related Work

Externalizing interaction experience. Prior work has made agent experience reusable in several forms. Reflexion and ExpeL distill feedback or insights from earlier attempts (Shinn et al., 2023; Zhao et al., 2024); AutoGuide derives context-aware guidelines from contrasting trajectories, while AutoManual constructs instruction manuals (Fu et al., 2024; Chen et al., 2024). Other methods maintain executable skills, workflows, or procedural memories (Wang et al., 2024a; 2025; Fang et al., 2026; Yang et al., 2025; Forouzandeh et al., 2026). WALL-E instead learns symbolic rules to align a world model used for planning (Zhou et al., 2025). SKL shares the goal of reusing interaction experience, but separates knowledge according to the decision it supports. It estimates probabilistic object-location relations from observations pooled across training instances to guide search, while separately learning procedural rules from trajectory contrasts to guide execution. The relations combine empirical evidence with commonsense priors and are instantiated for the locations available in each test environment. Both components directly condition a frozen agent.

Object-location beliefs. Belief-based world models have exposed object-location probabilities to LLM agents on ALFWorld and ScienceWorld (Kumar et al., 2026). Their beliefs are initialized from environment placement constraints and updated as observations eliminate possible locations within an episode. SKL addresses a different learning problem: estimating reusable object-location

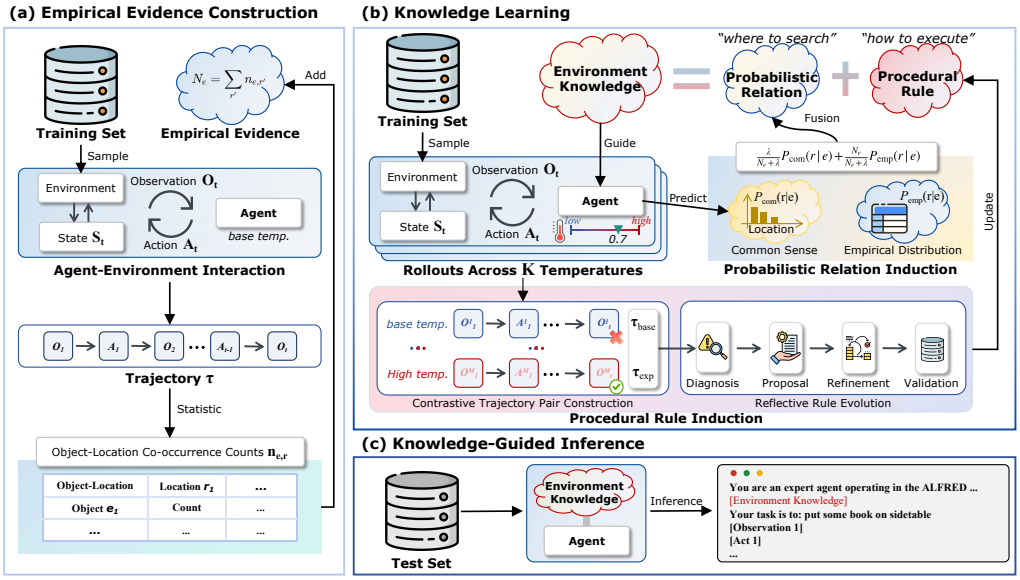


Figure 1: **Overview of SKL. (a) Empirical evidence construction.** The agent interacts with the training environments and records object-location co-occurrence counts. **(b) Knowledge learning.** SKL fuses commonsense priors with the counts into a probabilistic relation set $\mathcal{K}_{\text{rel}}$ (Section 3.3), and induces a procedural rule set $\mathcal{K}_{\text{rule}}$ from contrastive trajectory pairs through reflective rule evolution (Section 3.4). **(c) Knowledge-guided inference.** The rule block and the relations instantiated for the current task are injected as prompt context to guide planning in test environments.

tendencies from interactions across training instances, then combining these relations with independently learned execution rules. Its relation component provides search guidance without requiring the agent to query or simulate a world model.

Reflective Knowledge and Context Optimization. ExpeL (Zhao et al., 2024) extracts insights from same-task success–failure comparisons and groups of successful trajectories, maintaining them through edits and importance votes. ACE (Zhang et al., 2026a) organizes experience into a playbook that evolves through reflection, incremental updates, and curation, while GEPA (Agrawal et al., 2026) combines trace-based reflection with performance-based prompt selection. Our reflective rule evolution proposes procedural edits by contrasting a base trajectory with a higher-return exploratory trajectory from the same task and knowledge state. The revised rule block is accepted only if it improves the frozen agent's aggregate return across the complete training set. This performance-based acceptance distinguishes our approach from ExpeL's importance-based insight maintenance and ACE's incremental playbook curation. Within SKL, this rule optimization supports execution, alongside probabilistic search relations learned separately from cross-instance observation statistics.

Learning from Interaction via Parameter Updates. A complementary line internalizes interaction experience through policy, world-model, or critic optimization: ETO (Song et al., 2024), WKM (Qiao et al., 2024), and KnowAgent (Zhu et al., 2025) optimize agent behavior using trajectories or structured planning knowledge, Retrospex (Xiang et al., 2024) uses an offline reinforcement learning critic, and HCAPO (Tan et al., 2026) and Q-Evolve (Zhang et al., 2026b) derive learning signals for policy improvement. In contrast, SKL keeps the backbone frozen and optimizes an explicit environment-knowledge context that can be inspected and reused across model backbones.

3 Method

3.1 Problem Formulation

A dynamic interactive task can be modeled as a Partially Observable Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, R, H)$ with latent state, action, and observation spaces $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$, transition and observation functions $\mathcal{T}$ and Ω, reward function R, and maximum horizon H. Let $\mathcal{D}$ be the task

distribution, where each task instance $g \sim \mathcal{D}$ is associated with a natural-language instruction u_g. We refer to this POMDP together with its task distribution as the *local environment*, the particular task setting in which the agent is deployed, with its characteristic object placements, admissible operations, and procedural conventions. In our experiments, each ALFWorld task type and each ScienceWorld topic is one local environment (Section 4.1). Regularities that recur across instances $g \sim \mathcal{D}$ constitute local environment knowledge. They hold within the local environment but not necessarily beyond it and may depart from the commonsense expectations of a pretrained LLM. At step t, the agent observes $o_t \in \mathcal{O}$ rather than the latent state $s_t \in \mathcal{S}$, and selects an action based on its interaction history $h_t = (u_g, o_1, a_1, \ldots, o_t)$. In our scenario, the agent π is backed by an LLM with parameters θ and an external context $\mathcal{K}$, and samples the next action as $a_t \sim \pi_{\theta,\mathcal{K}}(\cdot \mid h_t)$.

Aligning the agent with its local environment then means making its decisions reflect these regularities. Rather than updating θ, we align the frozen agent through the context $\mathcal{K}$. In this work, $\mathcal{K}$ is instantiated as structured knowledge in the form of explicit, human-readable entities, weighted relations, and natural-language rules. Formally, let τ denote the trajectory generated by executing $\pi_{\theta,\mathcal{K}}$ on task g and $R(\tau)$ its cumulative environment return (defined in Section 3.4). We aim to obtain $\mathcal{K}^\star = \arg\max_{\mathcal{K}} \mathbb{E}_{g\sim\mathcal{D},\,\tau\sim\pi_{\theta,\mathcal{K}}}[R(\tau)]$. $\mathcal{K}$ captures what recurs across instances, not the state of the current instance, which the agent continues to read from its observations.

3.2 Method Overview

SKL learns $\mathcal{K}$ from self-generated interactions with the local environment while the backbone remains frozen throughout learning and inference. Following the distinction drawn in the introduction, $\mathcal{K} = \{\mathcal{K}_{\text{rel}}, \mathcal{K}_{\text{rule}}\}$ has one component per decision and one learning signal per component. $\mathcal{K}_{\text{rel}}$ is a probabilistic relation set for search decisions, estimated from cross-instance observation statistics, and $\mathcal{K}_{\text{rule}}$ is a procedural rule set for execution decisions, refined from behavioral differences between trajectories with different returns. As shown in Fig. 1, SKL proceeds in three phases. In empirical evidence construction (a), the agent performs an initial pass over the training set and records object-location co-occurrence counts from its observations. In knowledge learning (b), $\mathcal{K}_{\text{rel}}$ is obtained by fusing these counts with the agent's own commonsense priors, and $\mathcal{K}_{\text{rule}}$ is induced by reflectively updating rules over contrastive trajectory pairs collected in a second pass. In knowledge-guided inference (c), the learned knowledge is injected into the prompt of the frozen agent to guide planning on the test set. Fig. 1 follows this pipeline, whereas the next two sections are organized by component, since each component spans the first two phases. Section 3.3 covers the co-occurrence counting in (a) and the prior fusion in (b), Section 3.4 covers the rule induction in (b), and Section 3.5 describes phase (c).

3.3 Probabilistic Relations for Search

SKL first constructs empirical evidence from interaction with the training set $\mathcal{D}_{\text{train}}$. For each instance $g \in \mathcal{D}_{\text{train}}$, the frozen LLM agent executes a rollout and obtains a trajectory $\tau = (o_1, a_1, o_2, a_2, \ldots, a_{T_\tau-1}, o_{T_\tau})$, where T_τ is the number of observations in τ, o_t is the observation at step t, and a_t is the action taken after o_t.

Since the observations used in this work follow fixed textual templates, SKL uses a deterministic parser to extract object-location tuples. The parser matches predefined textual patterns, strips instance indices, and preserves the entity and location names used in the observations. Let $\phi(o_t)$ denote the set of tuples (e, r) extracted from observation o_t, where e is an entity and r is a location. The object-location co-occurrence count is then defined at the trajectory level as

$$n_{e,r} = \sum_{\tau \in \mathcal{T}_{\text{train}}} \mathbb{I}\big[\exists\, t \le T_\tau :\ (e, r) \in \phi(o_t)\big],$$

where $\mathcal{T}_{\text{train}}$ is the set of collected trajectories. We further define $N_e = \sum_{r' \in \mathcal{R}} n_{e,r'}$, where $\mathcal{R}$ is the set of candidate locations.

Each trajectory contributes at most one count per pair, so $n_{e,r}$ records in how many training instances e was observed at r, and revisiting a location within an episode does not inflate it. These counts summarize observed co-occurrence under the collecting policy rather than exhaustive environment occupancy: an unvisited or unopened location yields no evidence, not negative evidence, which motivates the prior-weighted smoothing below.

Given these counts, the goal is to learn where entities are likely to appear. Pre-trained LLMs encode rich commonsense knowledge about where objects are typically located. However, directly relying on such priors can be unreliable, because the target environment may exhibit object-location regularities that differ from common patterns. In contrast, empirical evidence is domain-grounded and corrective, yet sparse due to limited exploration coverage. SKL therefore integrates the two sources through a Bayesian fusion mechanism that balances generality and specificity.

For each entity e and candidate location set $\mathcal{R}$, SKL first elicits a commonsense prior from the frozen LLM, which assigns confidence scores $s(e, r) \in [0, 100]$ to a sparse subset of likely locations (zero elsewhere), normalized into a prior $P_{\text{com}}(r \mid e) = \frac{s(e,r)}{\sum_{r' \in \mathcal{R}} s(e,r')}$. When empirical evidence is available, we define the empirical distribution as $P_{\text{emp}}(r \mid e) = \frac{n_{e,r}}{N_e}$. SKL then fuses commonsense priors and empirical evidence through a smoothed estimate:

$$\mathcal{K}_{\text{rel}}(e, r) = \frac{\lambda}{N_e + \lambda} P_{\text{com}}(r \mid e) + \frac{N_e}{N_e + \lambda} P_{\text{emp}}(r \mid e),$$

where $\lambda > 0$ is a concentration hyperparameter controlling the influence of the commonsense prior. If no empirical evidence is available for e ($N_e = 0$), the relation reduces to P_{com}; if the LLM assigns no positive score to any candidate location, it reduces to P_{emp}; if neither source is available, no relation context is provided for e. This relation can be interpreted as a weighted relational fact $\text{likely_loc}(e, r) \mapsto \mathcal{K}_{\text{rel}}(e, r)$; for example, a learned relation may assign high confidence to $\text{likely_loc}(\texttt{apple}, \texttt{fridge})$ if apples are frequently observed in fridges in the training set. As interaction evidence grows, the empirical distribution receives larger weight and corrects the initial commonsense prior.

3.4 Procedural Rules for Execution

Successful interaction also requires procedural knowledge about how to execute. SKL therefore learns a natural-language rule block $\mathcal{K}_{\text{rule}}$ that encodes reusable action constraints, task procedures, and failure-avoidance strategies, through an iterative cycle of contrastive trajectory pair construction, reflective rule evolution, and performance-based validation (Algorithm 1).

Contrastive Trajectory Pair Construction. Effective reflection requires pinpointing where the agent made mistakes, i.e., credit assignment (Sutton & Barto, 1998; Shinn et al., 2023); a single failure analyzed in isolation lacks a positive reference, making it difficult to distinguish strategic flaws from stochastic execution errors in long-horizon tasks. SKL therefore constructs contrastive trajectory pairs that expose decision points associated with performance differences. For each training instance $x \in \mathcal{D}_{\text{train}}$, the agent first generates a base trajectory τ_{base} using the current $\mathcal{K}_{\text{rel}}$ and $\mathcal{K}_{\text{rule}}$ at temperature γ_{base}. It then generates $K-1$ exploratory trajectories $\{\tau_i\}_{i=1}^{K-1}$ at temperatures $\Gamma_{\text{exp}} = \{\gamma_i\}_{i=1}^{K-1}$. Let $R(\tau) = \sum_{t=1}^{T_\tau - 1} r_t$ denote the cumulative environment return of trajectory τ, where r_t is the reward received after action a_t. Among the exploratory trajectories, SKL selects τ_{exp} lexicographically: highest return first, then shorter length when returns are tied, then lower temperature when both are tied. A contrastive pair $(\tau_{\text{base}}, \tau_{\text{exp}})$ is retained only when $R(\tau_{\text{exp}}) > R(\tau_{\text{base}})$, so that the exploratory trajectory serves as a higher-return reference for diagnosis.

Reflective Rule Evolution. Given a contrastive pair $(\tau_{\text{base}}, \tau_{\text{exp}})$, SKL invokes three LLM-based modules to construct a candidate update to $\mathcal{K}_{\text{rule}}$. The *Diagnosis* module compares the two trajectories, identifies the critical divergence points between them, and produces a structured critique C that specifies the failure mode, the desired behavior, and a generalizable lesson. The *Proposal* module converts this critique into a structured set of rule-edit operations, denoted by E, specifying ADD, REMOVE, or STRENGTHEN modifications to the current rule block. The *Refinement* module applies these edits to $\mathcal{K}_{\text{rule}}$ and rewrites the complete procedural rule block, producing a candidate $\mathcal{K}'_{\text{rule}}$. During refinement, redundant or conflicting instructions are consolidated to maintain a coherent rule block. If any module produces an output that violates the predefined schema, the candidate update is discarded before performance evaluation. For instance, in an ALFWorld Cool task, a base trajectory that exhausts its budget opening cabinets, paired with an exploratory trajectory that finds the apple on a countertop, yields a critique of the ineffective cabinet search, an ADD edit, and a rule to check visible countertop objects before exhaustive cabinet search (Appendix G.1). Unlike insights extracted from independently collected successes and failures (Zhao et al., 2024) or curated from accumulated context (Zhang et al., 2026a), both trajectories in a pair share the task instance and the knowledge state, so their divergence isolates a decision difference.

Algorithm 1 Procedural Rule Induction

Require: Training set $\mathcal{D}_{\text{train}}$, relation set $\mathcal{K}_{\text{rel}}$, number of rollouts K, base temperature γ_{base}, and exploration temperatures Γ_{exp}
Ensure: Procedural rule block $\mathcal{K}_{\text{rule}}$

$\mathcal{K}_{\text{rule}} \leftarrow \varnothing$; $S_{\text{best}} \leftarrow V(\mathcal{K}_{\text{rule}})$
for all $x \in \mathcal{D}_{\text{train}}$ **do**
  $\tau_{\text{base}} \leftarrow \textsc{Rollout}(x, \mathcal{K}_{\text{rel}}, \mathcal{K}_{\text{rule}}, \gamma_{\text{base}})$; $\tau_i \leftarrow \textsc{Rollout}(x, \mathcal{K}_{\text{rel}}, \mathcal{K}_{\text{rule}}, \gamma_i)$ for $i = 1, \ldots, K-1$
  $\tau_{\text{exp}} \leftarrow \textsc{Select}(\{(\tau_i, \gamma_i)\}_{i=1}^{K-1})$ ▷ highest return, then shortest, then lowest temperature
  if $R(\tau_{\text{exp}}) \leq R(\tau_{\text{base}})$ **then continue**
  end if
  $C \leftarrow \textsc{Diagnose}(x, \tau_{\text{base}}, \tau_{\text{exp}})$; $E \leftarrow \textsc{Propose}(\mathcal{K}_{\text{rule}}, C)$; $\mathcal{K}'_{\text{rule}} \leftarrow \textsc{Refine}(\mathcal{K}_{\text{rule}}, C, E)$
  $S' \leftarrow V(\mathcal{K}'_{\text{rule}})$
  if $S' > S_{\text{best}}$ **then** $\mathcal{K}_{\text{rule}} \leftarrow \mathcal{K}'_{\text{rule}}$; $S_{\text{best}} \leftarrow S'$
  end if
end for
return $\mathcal{K}_{\text{rule}}$

Performance-Based Validation. Since each candidate update to $\mathcal{K}_{\text{rule}}$ is induced from a single contrastive trajectory pair, it may improve performance on the current instance without generalizing to other training instances, and may even degrade their performance. `SKL` therefore evaluates each candidate rule block based on its aggregate performance over the entire training set. For a procedural rule block $\mathcal{K}_{\text{rule}}$, let $\tau_x(\mathcal{K}_{\text{rule}})$ denote the trajectory generated by one rollout on training instance x using $\mathcal{K}_{\text{rel}}$ and $\mathcal{K}_{\text{rule}}$ at temperature γ_{base}. We define its performance score as

$$V(\mathcal{K}_{\text{rule}}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{x \in \mathcal{D}_{\text{train}}} R\big(\tau_x(\mathcal{K}_{\text{rule}})\big).$$

The induction process is initialized with $\mathcal{K}_{\text{rule}} = \varnothing$, and the score of the currently accepted rule block is cached as $S_{\text{best}} = V(\mathcal{K}_{\text{rule}})$. For each candidate $\mathcal{K}'_{\text{rule}}$, `SKL` computes $S' = V(\mathcal{K}'_{\text{rule}})$. The candidate is accepted only when $S' > S_{\text{best}}$, in which case `SKL` sets $\mathcal{K}_{\text{rule}} \leftarrow \mathcal{K}'_{\text{rule}}$ and $S_{\text{best}} \leftarrow S'$; otherwise, the current rule block and its cached score are retained. By repeating this cycle over the training instances, `SKL` converts instance-level contrastive evidence into an iteratively refined procedural rule block. Local trajectory comparisons propose targeted rule edits, while aggregate training-set performance filters out candidates that fail to improve the agent's overall behavior. Because acceptance uses the training score, V is a selection criterion rather than a held-out validation estimate; generalization is assessed separately on the test partitions.

3.5 Knowledge-Guided Inference

Before held-out evaluation, the object-location counts and the rule block $\mathcal{K}_{\text{rule}}$ are frozen; no test-time observation or reward updates them. The rule block is injected verbatim as a natural-language list, and the relation component is instantiated per test instance. In ALFWorld, the target entity e is extracted from the task instruction, and the candidate location set $\mathcal{R}$ is read from the initial observation of the current environment, which lists the receptacles present and is visible to every agent, so `SKL` receives no information beyond what the baselines observe. The prior $P_{\text{com}}(\cdot \mid e)$ is elicited over this $\mathcal{R}$, fused with the frozen counts as above, renormalized, and rendered as a ranked list with rounded percentages, e.g., `apple: fridge (39%), countertop (32%), diningtable (13%), ...`; entries supported only by training counts are kept, and one whose location type is absent from the current environment is simply inapplicable. In ScienceWorld, topic-level relation entries are supplied with their stored weights (Appendix G.2). Both components enter the prompt alongside the instruction, current observation, and interaction history; the agent still has to locate the object from its observations and issue admissible actions (exact serialization in Appendix D; learned knowledge in Appendix F).

Table 1: **Performance on ALFWorld**. Each result is the mean ± standard deviation of average reward over five independent evaluations of the full test set. **Bold** and underlining denote the best and runner-up results, respectively.

Method	Task type						All
	Clean	Cool	Examine	Heat	Pick	Pick2	
GPT-4o-mini							
Base	27.24 ± 0.77	17.83 ± 1.82	59.35 ± 7.43	30.77 ± 3.14	72.20 ± 4.08	43.41 ± 5.00	41.90 ± 2.15
ReAct	75.86 ± 2.73	36.09 ± 3.95	47.10 ± 8.41	50.26 ± 6.93	81.36 ± 3.17	**70.24** ± 3.18	62.63 ± 2.90
Reflexion	36.90 ± 3.13	32.17 ± 2.83	87.10 ± 4.56	43.59 ± 1.81	92.20 ± 2.57	63.41 ± 6.22	58.61 ± 1.05
SKL (ours)	**88.62** ± 4.15	**71.30** ± 3.89	**92.90** ± 2.70	**67.69** ± 6.44	**96.27** ± 1.42	64.88 ± 6.36	**81.31** ± 2.33
Gemini-2.0-flash							
Base	38.28 ± 3.08	18.26 ± 3.30	50.97 ± 6.99	17.44 ± 4.93	83.05 ± 3.60	51.22 ± 4.56	44.96 ± 1.49
ReAct	34.14 ± 4.78	37.39 ± 3.57	44.52 ± 3.53	23.59 ± 3.34	28.81 ± 2.68	63.90 ± 2.04	37.66 ± 2.21
Reflexion	61.72 ± 2.83	32.61 ± 6.15	64.52 ± 5.10	37.95 ± 7.12	91.53 ± 3.60	68.29 ± 4.56	61.17 ± 2.52
SKL (ours)	**91.72** ± 3.74	**88.70** ± 3.57	**78.71** ± 2.89	**73.85** ± 5.56	**98.64** ± 0.76	**87.32** ± 2.67	**88.03** ± 1.22

Table 2: **Performance on ScienceWorld.** Each result is the mean ± standard deviation of average reward over five independent evaluations of the full test set. Notation follows Table 1.

Method	Topic						All
	Life Span	Classif.	Life Stage	Chemistry	Measure.	Matter	
GPT-4o-mini							
Base	58.49 ± 3.40	60.83 ± 4.16	10.80 ± 2.66	9.50 ± 1.02	8.33 ± 2.67	7.73 ± 3.66	33.20 ± 0.83
ReAct	63.67 ± 3.26	53.24 ± 1.94	28.18 ± 5.20	**27.37** ± 0.54	11.15 ± 2.91	16.47 ± 2.86	37.42 ± 0.47
Reflexion	**70.70** ± 2.93	65.35 ± 2.19	11.56 ± 1.69	8.51 ± 1.23	10.27 ± 3.63	15.72 ± 2.15	38.91 ± 0.69
SKL (ours)	68.44 ± 3.95	**85.23** ± 1.90	**38.57** ± 4.82	24.14 ± 3.03	**24.16** ± 4.19	**37.30** ± 2.71	**53.61** ± 1.20
Gemini-2.0-flash							
Base	70.95 ± 2.62	26.18 ± 0.81	8.23 ± 4.33	16.31 ± 2.45	27.05 ± 2.84	5.82 ± 0.81	29.10 ± 0.57
ReAct	56.66 ± 1.83	51.53 ± 1.49	38.51 ± 3.19	46.11 ± 4.84	26.70 ± 5.38	20.90 ± 3.92	41.28 ± 1.72
Reflexion	**80.82** ± 2.09	47.65 ± 1.35	8.19 ± 6.08	30.79 ± 4.20	33.41 ± 3.26	38.21 ± 6.61	46.61 ± 1.44
SKL (ours)	76.00 ± 3.46	**89.76** ± 1.46	**41.69** ± 1.36	**48.14** ± 5.90	**48.34** ± 4.59	**56.42** ± 2.01	**66.83** ± 0.43

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Datasets and Metrics. We evaluate SKL on ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022). For ALFWorld, we use 100 instances from the official training split for each of the 6 task types and evaluate on the complete official seen and unseen test splits (600 training and 274 test instances). For ScienceWorld, we follow the ETO (Song et al., 2024) task split and evaluate 17 tasks across 6 topics; after excluding Tasks 3, 5 and 6.3 due to known benchmark inconsistencies, the split contains 1,064 training and 291 test variants. We learn a separate $\mathcal{K}$ for each ALFWorld task type and each ScienceWorld topic from the corresponding training split, and evaluate it on the corresponding test split. We report average reward on both benchmarks. ALFWorld rewards are 0 or 100 per episode, so the average equals the task success rate; ScienceWorld provides a dense completion score in $[0, 100]$, so its average reflects partial progress and is not a full-task success rate.

Models, Baselines, and Implementation. We evaluate SKL with GPT-4o-mini (OpenAI, 2024) and Gemini-2.0-flash (Google, 2025), and compare it under the same implementation and evaluation protocol with the Base agent, ReAct (Yao et al., 2023), and Reflexion (Shinn et al., 2023). The base agent uses interleaved thought-action prompting with a memory buffer of recent observations and actions, and is evaluated at temperature 0. We set $\lambda = 10$ and sample $K = 8$ rollouts per training instance, one base trajectory ($\gamma_{\text{base}} = 0$) and seven stochastic trajectories with $\Gamma_{\text{exp}} = \{0.1, \ldots, 0.7\}$.

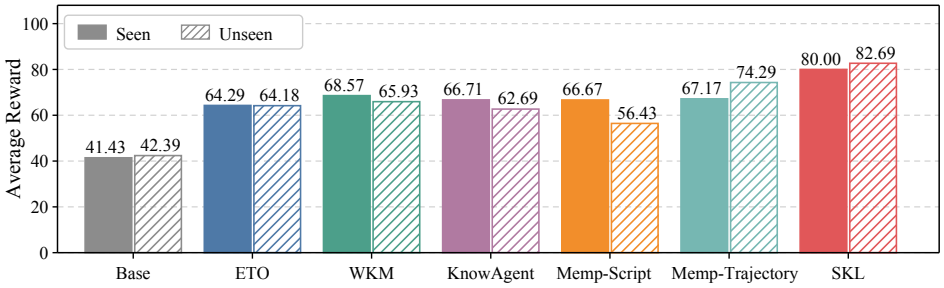
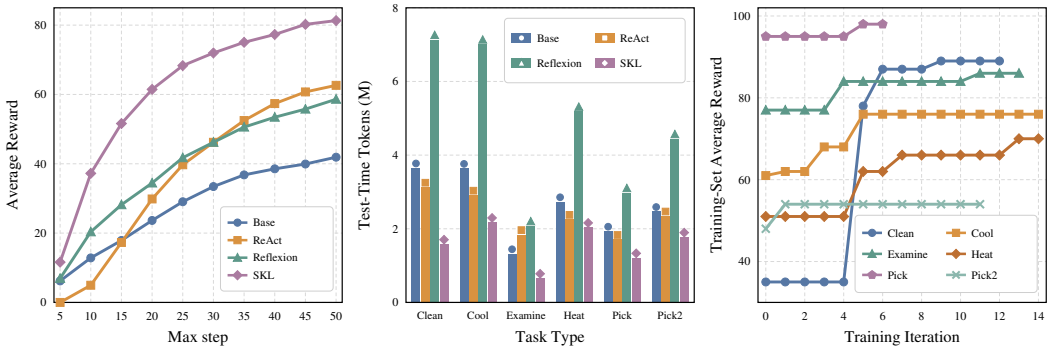


Figure 2: **Comparison with training-based and memory-based baselines on ALFWorld.** We report the average reward for Base, ETO, WKM, KnowAgent, Memp variants, and SKL (Ours) on Seen and Unseen tasks.



(a) Performance under Interaction Budgets (b) Test-Time Token Consumption (c) Knowledge-Learning Curves

Figure 3: **Efficiency analysis on ALFWorld with GPT-4o-mini.** (a) Average reward under different test-time environment-step budgets. (b) Test-time LLM-token usage per task type in one evaluation run. (c) Training-set average reward after each accepted rule update on 100 training tasks.

Every reported result is the mean and standard deviation over five evaluations of the full test partition with one learned $\mathcal{K}$; these repeats characterize execution variability, not variation across independent knowledge-learning runs. Appendix B gives the complete protocol.

4.2 Main Results

Comparison with Prompting- and Reflection-based Baselines. As shown in Tables 1 and 2, SKL achieves the best overall average reward on both benchmarks with both backbones. Over the runner-up, it improves overall reward on ALFWorld by 18.68 points with GPT-4o-mini and 26.86 points with Gemini-2.0-flash (62.63 $\rightarrow$ 81.31, 61.17 $\rightarrow$ 88.03), and on ScienceWorld by 14.70 and 20.22 points (38.91 $\rightarrow$ 53.61, 46.61 $\rightarrow$ 66.83).

Contextual Comparison with Training- and Memory-based Methods. Figure 2 places our GPT-4o-mini results next to published ALFWorld results for the training-based methods ETO (Song et al., 2024), WKM (Qiao et al., 2024), and KnowAgent (Zhu et al., 2025), which fine-tune Llama-3-8B, and for the memory-based method Memp (Fang et al., 2026), which uses GPT-4o. Because these systems use their original backbones, learning budgets, and implementations, the figure provides literature context rather than a controlled comparison, and the margins cannot be attributed to the method alone. Two observations are nevertheless informative. First, with the same GPT-4o-mini backbone, Base obtains 41.43 on Seen and 42.39 on Unseen, whereas SKL obtains 80.00 and 82.69, above the published values of all four systems; the margin over Memp is obtained with a smaller backbone than Memp's GPT-4o. Second, SKL shows no degradation from Seen to Unseen, whereas KnowAgent and Memp-Script decline, consistent with the learned knowledge capturing regularities that hold in unseen room layouts.

4.3 ANALYSIS

Are the two components complementary? Table 3 evaluates the contributions of $\mathcal{K}_{\text{rel}}$ and $\mathcal{K}_{\text{rule}}$. Each single-component variant removes the other component in both learning and inference. The two variants improve over Base by 13.94 and 20.14 points, and combining both reaches 81.31, which suggests complementary guidance, with $\mathcal{K}_{\text{rel}}$ prioritizing likely locations and $\mathcal{K}_{\text{rule}}$ guiding valid action sequences. The relation-only sensitivity study (Appendix C.2) isolates the fusion: empirical counts alone ($\lambda = 0$) reach 52.92 versus 55.84 when fused at $\lambda = 10$, and larger prior weights hurt, so interaction evidence carries most of the gain.

Does the knowledge transfer to a stronger backbone? We test whether knowledge learned by a weaker backbone (by Base performance) improves a stronger frozen agent. We transfer the complete $\mathcal{K}$ learned by GPT-4o-mini or Gemini-2.0-flash to Gemini-2.5-flash (Comanici et al., 2025). As shown in Table 4, the transferred knowledge improves the overall reward of Gemini-2.5-flash from 74.60 to 85.26 and 86.79, with gains on all six task types. Notably, GPT-4o-mini with SKL (81.31, Table 1) surpasses Gemini-2.5-flash without it (74.60). Appendix C.1 reports a corroborating evaluation with the open-weight Qwen3-8B.

Table 3: **Component ablation.** Average reward on ALFWorld with GPT-4o-mini.

Variant	$\mathcal{K}_{\text{rel}}$	$\mathcal{K}_{\text{rule}}$	Reward	Gain
Base	✗	✗	41.90 ± 2.15	–
$\mathcal{K}_{\text{rel}}$-only	✓	✗	55.84 ± 1.73	+13.94
$\mathcal{K}_{\text{rule}}$-only	✗	✓	62.04 ± 1.54	+20.14
Full SKL	✓	✓	**81.31** ± 2.33	**+39.41**

Table 4: **Weak-to-strong transfer.** Average reward of Gemini-2.5-flash on ALFWorld without learned knowledge (Base) or with the $\mathcal{K}$ learned by a weaker backbone.

Task type	Base	$\mathcal{K}$ learned with	
		GPT-4o-mini	Gemini-2.0-flash
Clean	71.72 ± 1.97	80.34 ± 3.36	**86.21** ± 2.73
Cool	56.96 ± 4.71	83.48 ± 2.48	**84.78** ± 1.54
Examine	66.45 ± 5.40	**86.45** ± 4.78	76.13 ± 4.33
Heat	52.82 ± 5.32	62.56 ± 2.92	**70.77** ± 6.44
Pick	97.97 ± 0.76	**99.66** ± 0.76	**99.66** ± 0.76
Pick2	91.71 ± 3.27	94.15 ± 1.34	**94.63** ± 2.04
Overall	74.60 ± 1.38	85.26 ± 0.99	**86.79** ± 1.25

Does return-grounded contrast matter? The Failure-only variant keeps $\mathcal{K}_{\text{rel}}$ and the same update and validation procedure but removes exploratory rollouts and learns from the failed base trajectory alone. On ALFWorld-Cool with GPT-4o-mini, it reaches 64.35 ± 2.22 versus 71.30 ± 3.89 for full SKL, supporting return-grounded contrast, although this does not isolate the higher-return reference from the extra rollouts (Appendix C).

What does the knowledge save, and what does it cost? At test time, SKL achieves the highest reward across all tested interaction-step budgets in Figure 3(a), exceeding 60 with only 20 steps, and uses fewer test-time LLM tokens than the compared agents across all six task types (Figure 3(b)). Figure 3(c) plots training-set reward after each accepted rule update on the 100 training instances. It stabilizes within about 14 updates and, since acceptance uses this score, describes optimization dynamics rather than sample efficiency. Offline learning consumes approximately 90M LLM tokens per task type, mostly in performance-based validation (Appendix C.6), a one-time cost amortized over the test instances that reuse the knowledge.

5 CONCLUSION

We studied how an LLM agent can be aligned with its local environment without updating its parameters. SKL organizes local environment knowledge by the decisions it supports, learning probabilistic relations for search and procedural rules for execution from distinct interaction signals. On ALFWorld and ScienceWorld, this knowledge substantially improves average reward with both backbones. On ALFWorld, the two components are complementary, and the gains hold under smaller interaction budgets. Because the knowledge is external to the model, it can be handed to a different backbone, and in our experiments a weaker model equipped with learned knowledge surpassed a stronger model without it. These results suggest that a substantial part of what separates agents in such environments is environment-specific knowledge rather than backbone capability, and that this part can be learned once and reused across models. Our evidence is limited to two text environments with templated observations and to reuse within a task family. Extending this view to richer environments and other decisions, such as tool selection or verification, remains future work.

AI USE STATEMENT

Large language models are components of the studied agents and knowledge-learning procedure, as described in Section 3 and Appendix E. Outside these experimental uses, we used large language models solely for language polishing to improve the grammar, clarity, and readability of the manuscript. They were not used for research ideation, methodological or experimental design, method implementation, or interpretation of results. The authors take full responsibility for the final manuscript, including all scientific claims and experimental results.

ETHICS STATEMENT

This work investigates knowledge learning for LLM agents using the ALFWorld and ScienceWorld simulation benchmarks. All experiments are conducted within these benchmark environments. Real-world deployment is outside the scope of this study.

REPRODUCIBILITY STATEMENT

Section 3 describes our method, with pseudocode for procedural rule induction provided in Algorithm 1. Section 4.1 and Appendix B document the benchmarks, data splits, baseline protocols, interaction budgets, model and decoding settings, repeated evaluation procedure, and computational environment. Appendix D details probabilistic relation construction, Appendix E provides functional prompt templates, and Appendix F presents representative learned knowledge. These descriptions specify how knowledge is constructed from training interactions and fixed before held-out evaluation. We will release the complete codebase upon acceptance.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Generating instruction manuals by LLM agents via interactive environmental learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Pwl9n4zlf5.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Saman Forouzandeh, Wei Peng, Parham Moradi, Xinghuo Yu, and Mahdi Jalili. Learning hierarchical procedural memory for llm agents through bayesian selection and contrastive refinement. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '26, pp. 1820–1828, Richland, SC, 2026. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400723179. doi: 10.65109/FKYO8341. URL https://doi.org/10.65109/FKYO8341.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mRIQz8Zd6O.

Google. Gemini 2.0 Flash model card. https://modelcards.withgoogle.com/assets/documents/gemini-2-flash.pdf, 2025. Accessed 2026-01-27.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, A. J. Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Shubham Kumar, Harshit Kumar, Narendra Ahuja, and Saurabh Jha. Towards a belief-based world model for llm agents, 2026. URL https://arxiv.org/abs/2609.00455.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.276. URL https://aclanthology.org/2025.emnlp-main.276/.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems, 2025. URL https://arxiv.org/abs/2504.01990.

Meta. Introducing meta llama 3: The most capable openly available llm to date. Meta AI Blog, April 2024. URL https://ai.meta.com/blog/meta-llama-3/. Accessed 2026-01-29.

OpenAI. GPT-4o mini model. https://platform.openai.com/docs/models/gpt-4o-mini, 2024. Accessed 2026-01-27.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=j6kJSS9O6I.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. Agentic retrieval-augmented generation: A survey on agentic RAG, 2025. URL https://arxiv.org/abs/2501.09136.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction*. MIT press Cambridge, 1998.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon LLM agents, 2026. URL https://arxiv.org/abs/2603.08754.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

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

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

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

Yufei Xiang, Yiqun Shen, Yeqin Zhang, and Cam-Tu Nguyen. Retrospex: Language agent meets offline reinforcement learning critic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4650–4666, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.268. URL https://aclanthology.org/2024.emnlp-main.268/.

Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. Coarse-to-fine grounded memory for LLM agent planning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13029–13056, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.659. URL https://aclanthology.org/2025.emnlp-main.659/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=eC4ygDs02R.

Yudi Zhang, Meng Fang, Zhenfang Chen, and Mykola Pechenizkiy. Self-evolving LLM agents with in-distribution optimization, 2026b. URL https://arxiv.org/abs/2606.07367.

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

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://openreview.net/forum?id=oKn9c6ytLx.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-e: World alignment by neurosymbolic learning improves world model-based LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=DorAT49sxj.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. KnowAgent: Knowledge-augmented planning for LLM-based agents. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3709–3732, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.205. URL https://aclanthology.org/2025.findings-naacl.205/.

A Scope and Organization

This appendix provides implementation, protocol, and analysis details; notation follows the main paper. In particular, SKL learns external structured environment knowledge $\mathcal{K} = \{\mathcal{K}_{\text{rel}}, \mathcal{K}_{\text{rule}}\}$ while keeping the backbone language model frozen. Section B specifies the benchmarks, information boundary, baseline protocols, and reproducibility settings. Section C reports additional backbone, sensitivity, mechanism, learning-curve, and cost results. Sections D, E, and F provide relation-construction details, functional prompt templates, and representative learned knowledge, respectively. Section G closes with the ALFWorld and ScienceWorld case studies.

B Experimental Setup and Reproducibility

B.1 Benchmark Details

The two benchmarks use different organizational units. We describe them using the hierarchy *benchmark suite* → *knowledge-sharing group* → *task or instance* → *episode*. In ALFWorld, a knowledge-sharing group is one of six household task types, and each instance specifies a concrete goal in an environment. In ScienceWorld, a knowledge-sharing group is one of six scientific topics; each topic contains multiple named task templates, and each template is instantiated through parameterized variations. This distinction determines the granularity at which SKL learns and reuses knowledge.

B.1.1 ALFWorld

ALFWorld (Shridhar et al., 2021) provides a textual interactive environment aligned with the household goals in ALFRED. At the beginning of an episode, the agent receives a natural-language goal. At each subsequent step, it observes a textual description of its current surroundings and issues a textual command to navigate, inspect, take, place, or otherwise manipulate objects. An instance pairs a concrete goal specification with an initial household configuration; an episode is one attempt to complete that instance within the interaction budget.

The six task types differ in their goal conditions and required action sequences. *Pick & Place* (Pick) requires placing one target object in a designated receptacle. *Examine in Light* (Examine) requires inspecting a target object under a light source. *Clean & Place*, *Heat & Place*, and *Cool & Place* require applying the named state-changing operation before placing the object in its target receptacle. *Pick Two & Place* (Pick2) requires placing two instances of the target object in a receptacle. These task types share the household action vocabulary but require different object-location regularities and procedural constraints.

We sample 100 instances from the official training split for each task type and evaluate on the complete official Seen and Unseen test partitions. Each episode is capped at 50 environment interactions and terminates earlier when the task is completed. The same 50-step budget is used for Base, ReAct, Reflexion, and SKL on every ALFWorld instance. Seen contains held-out goal instances in scene layouts represented in the training environments, whereas Unseen contains goal instances in held-out scene layouts. Both partitions retain the same six task types; thus, Unseen tests transfer to new environments within known task schemas rather than to a seventh, previously unobserved task type. ALFWorld returns binary episode success, represented in our results as 0 or 100. Table 5 gives the exact counts.

B.1.2 ScienceWorld

ScienceWorld (Wang et al., 2022) is a text-based simulated environment for grounded, multi-step scientific reasoning. Its world contains interconnected rooms, containers, ordinary objects, materials, and scientific devices. The agent must navigate through this world and perform operations such as locating, moving, observing, measuring, heating, cooling, and mixing objects. Rather than answering a static science question, the agent must change or inspect the simulated world until the goal conditions are satisfied. The benchmark assigns partial credit as required and intermediate goal conditions are completed, producing a dense degree-of-completion score that we report on a $[0, 100]$ scale.

Table 5: ALFWorld training and held-out evaluation partitions. The test set contains 274 instances in total.

Category	Train	Seen	Unseen	Total
Pick	100	35	24	159
Examine	100	13	18	131
Clean	100	27	31	158
Heat	100	16	23	139
Cool	100	25	21	146
Pick2	100	24	17	141
All	600	140	134	874

Table 6: ScienceWorld split after filtering inconsistent tasks. Seen and Unseen together form the 291-variation evaluation set reported in the main paper. Max Steps gives the task-specific interaction budget adopted from the ETO evaluation protocol.

Topic	Task	Train	Seen	Unseen	Total	Max Steps
Matter	melt	14	7	9	30	80
Matter	change-the-state-of-matter-of	14	7	9	30	80
Matter	freeze	14	7	9	30	80
Matter	boil	14	6	9	29	100
Measurement	use-thermometer	120	10	10	140	30
Measurement	measure-melting-point-known-substance	120	10	10	140	120
Classification	find-animal	120	10	10	140	15
Classification	find-non-living-thing	120	10	10	140	15
Classification	find-plant	120	10	10	140	15
Classification	find-living-thing	120	10	10	140	15
Chemistry	chemistry-mix	16	8	8	32	60
Chemistry	chemistry-mix-paint-secondary-color	18	9	9	36	15
Biology (Life Span)	lifespan-shortest-lived	83	10	10	103	10
Biology (Life Span)	lifespan-longest-lived	83	10	10	103	10
Biology (Life Span)	lifespan-longest-lived-then-shortest-lived	78	10	10	98	12
Biology (Life Stages)	identify-life-stages-1	6	3	5	14	30
Biology (Life Stages)	identify-life-stages-2	4	2	4	10	30
All		**1,064**	**139**	**152**	**1,355**	–

The original benchmark organizes its elementary-school science curriculum into 30 named task templates and thousands of parameterized variations. A named task specifies the scientific objective and its scoring conditions; a variation instantiates that template with particular target entities and an initial world configuration. An episode is one interaction attempt on one such variation. Consequently, two variations may require the same underlying scientific procedure while differing in the objects that must be found, tested, ordered, or transformed.

Our filtered evaluation contains 17 named tasks in six topics. *Matter* covers melting, freezing, boiling, and other changes of state. *Measurement* requires using a thermometer or determining the melting point of a known substance. *Classification* asks the agent to identify living things, nonliving things, plants, or animals. *Chemistry* requires mixing substances or producing a secondary paint color. *Biology (Life Span)* compares organisms by lifespan, whereas *Biology (Life Stages)* requires identifying or ordering stages in an organism's lifecycle.

Following ETO (Song et al., 2024), we retain its training, Seen, and Unseen index lists and exclude three tasks whose simulator behavior or annotations make evaluation unreliable: Task 3 has a documented anode–cathode reversal issue, Task 5 applies inconsistent pollination requirements to otherwise identical plant instances, and Task 6.3 has a task description inconsistent with its ground-truth annotations. After filtering, the training list contains 1,064 variations. The ETO development/

Seen list contributes 139 in-distribution evaluation variations, and its test/Unseen list contributes 152 out-of-distribution variations. We combine the latter two only when reporting the 291-variation ScienceWorld test aggregate in the main paper. Both evaluation partitions remain within the same selected 17 named tasks and six topics; they do not constitute previously unevaluated topics or task templates. Table 6 gives the per-task breakdown and the corresponding maximum interaction budgets.

For ScienceWorld, we follow the task-specific episode budgets used by the ETO evaluation protocol rather than imposing one uniform horizon. Across the 17 evaluated tasks, these budgets range from 10 to 120 environment interactions. An episode terminates when the task is completed or when its task-specific budget is exhausted. For a given task, the same budget is applied to Base, ReAct, Reflexion, and SKL.

B.2 Knowledge Granularity

The implementation learns six ALFWorld knowledge objects, one per task type, and six ScienceWorld knowledge objects, one per topic, for twelve knowledge objects in total. For ScienceWorld, all selected named tasks assigned to the same topic contribute training trajectories to the same topic-level $\mathcal{K}$; there is not a separate object for each of the 17 named tasks. Each object is learned only from its corresponding training list, frozen before evaluation, and supplied only to episodes from its associated task type or topic.

This grouping determines the supported generalization claim. ALFWorld tests reuse across instances of a known task type in the official Seen and Unseen environment partitions. ScienceWorld tests reuse of topic-level knowledge across the Seen and Unseen variations of the selected named tasks. Because every evaluated ScienceWorld variation belongs to one of the same 17 named tasks and six topics used to organize training, these experiments do not isolate zero-shot transfer to an entirely new named task or topic, nor do they assume one universal knowledge object across unrelated groups.

B.3 Information Boundary and Evaluation

All empirical counts, exploratory trajectories, return comparisons, rule edits, and candidate-rule acceptance evaluations are obtained from entries in the training list. The Seen and Unseen evaluation lists are not separately queried during knowledge construction. Before evaluation, the group-specific statistics and the accepted procedural rule block are frozen.

During evaluation, the relevant probabilistic relations are instantiated for the current target entity and the accessible candidate-location vocabulary provided by the environment interface. This operation adapts which relation entries are rendered for the current environment; it does not update the training counts or use evaluation rewards as learning signals. The procedural rule block is likewise used only as fixed prompt context. Evaluation observations are consumed by the executing agent in the ordinary interactive loop, and returns are used only to score the episode.

ALFWorld episode rewards are represented as 0 or 100. ScienceWorld provides a dense degree-of-completion score in $[0, 100]$. The common metric is average reward. Every controlled result in the main tables is the mean and standard deviation over five independent evaluations of the full test set.

B.4 Baselines and Evaluation Protocols

B.4.1 Controlled Comparisons

The controlled comparison comprises Base, ReAct, Reflexion, and SKL. All four use the same backbone, task instances, benchmark partition, maximum interaction-step budget, environment-facing action interface, and evaluation metric. Model parameters remain frozen. The methods differ in the additional context supplied to the agent and in whether reusable environment knowledge is constructed before held-out evaluation. We introduce the three baselines below.

- **Base.** The Base agent uses an interleaved reasoning–action interface. At each decision step, the prompt provides the current observation together with a bounded buffer of recent observations and actions, and instructs the model to produce a brief reasoning trace before issuing one environment action. Base receives no demonstration, verbal reflection,

or learned structured environment knowledge. The environment-specific Base prompts are given in Section E.1.

- **ReAct** (Yao et al., 2023) integrates Chain-of-Thought-style reasoning with interactive planning through a Thought–Action–Observation loop. We closely follow the original ReAct prompting setup and use one in-context demonstration to condition the LLM on the required reasoning and action pattern. ReAct otherwise uses the same backbone, benchmark instances, interaction budget, and scoring protocol as Base, and it does not update model parameters or receive the structured knowledge learned by SKL.
- **Reflexion** (Shinn et al., 2023) improves planning with verbal feedback: after an unsuccessful trajectory, the model is prompted to diagnose the failure and formulate a revised plan for the next attempt. We implement Reflexion on top of the Base agent and provide one in-context demonstration to steer the reflection process. We allow a single reflection iteration and report the resulting performance; thus, a failed first attempt can be followed by at most one reflection-conditioned re-planning attempt. The backbone parameters remain unchanged.

SKL uses the same Base execution interface, but augments its prompt with the probabilistic relations and procedural rules learned from the corresponding training partition. This knowledge is fixed before held-out evaluation; no evaluation reward is used to update it. Consequently, comparisons among Base, ReAct, Reflexion, and SKL permit direct comparison while holding the backbone, data partition, environment interface, and episode budget fixed. Sections D and F provide construction details and representative knowledge artifacts.

B.4.2 Contextual Cross-Paper Comparisons

The training- and memory-based systems in the main paper (Figure 2) provide broader literature-level context rather than a controlled head-to-head comparison. We summarize the role of each external baseline below.

- **Memp** (Fang et al., 2026) equips an LLM agent with procedural memory by distilling past trajectories into reusable task-solving knowledge and retrieving relevant memories for subsequent planning. We include two representative variants reported in the original paper. **Memp-Script** abstracts successful trajectories into high-level procedural instructions, whereas **Memp-Trajectory** retains complete execution traces as concrete examples. These variants therefore represent two memory granularities: abstract procedural guidance and retrieved trajectory demonstrations. We use the published ALFWorld results for both variants with GPT-4o (Hurst et al., 2024) as the reported backbone; we do not rerun Memp with our GPT-4o-mini implementation.
- **ETO** (Song et al., 2024) is a training-based approach that gathers exploration failures, constructs contrastive trajectory preference pairs, and updates the agent policy with contrastive preference learning such as direct preference optimization. It therefore internalizes trial-and-error experience through parameter updates, in contrast to the frozen-backbone, prompt-level knowledge interface used by SKL.
- **WKM** (Qiao et al., 2024) trains a parametric world knowledge model using knowledge synthesized from expert and sampled trajectories. It supplies prior task knowledge for global planning and dynamic state knowledge for local planning. Its environment knowledge is encoded in a trained model rather than in an explicit pair of probabilistic relations and procedural rules.
- **KnowAgent** (Zhu et al., 2025) introduces explicit action knowledge and a self-learning mechanism to constrain and improve action planning. The published system acquires this capability through its knowledgeable self-learning procedure, whereas SKL keeps the backbone fixed and supplies learned environment knowledge as inspectable prompt context.

For ETO, WKM, and KnowAgent, we follow the comparison protocol used by WKM and report their published ALFWorld results with Llama 3-8B (Meta, 2024) as the backbone. Differences in backbone, method-specific training or memory construction, and implementation mean that these values cannot isolate a method effect relative to our GPT-4o-mini runs. In addition, discrepancies in

Table 7: Implementation and evaluation settings used by SKL.

Item	Setting
Main-paper backbones	GPT-4o-mini and Gemini-2.0-flash.
Additional backbone	Qwen3-8B, evaluated through an API in the additional ALFWorld-Cool experiment reported in Section C.1.
Transfer target	Gemini-2.5-flash; transferred knowledge is used without target-side knowledge learning or parameter updates.
Model parameters	Frozen throughout knowledge learning and held-out evaluation.
Evaluation decoding	Base-agent temperature $\gamma_{\text{base}} = 0$.
Contrastive rollouts	$K = 8$: one base rollout at temperature 0 and seven exploratory rollouts at $\Gamma_{\text{exp}} = \{0.1, 0.2, \ldots, 0.7\}$.
Relation fusion	Concentration hyperparameter $\lambda = 10$.
Repeated evaluation	Main benchmark, component, and reported transfer results use five independent evaluations of the complete held-out test partition and are reported as mean $\pm$ standard deviation. Explicitly identified additional single-run or uncertainty-unavailable results are exceptions.
Knowledge granularity	One $\mathcal{K}$ per ALFWorld task category and one $\mathcal{K}$ per ScienceWorld topic.
Metrics	Average reward; ALFWorld uses binary episode outcomes represented as $\{0, 100\}$ and ScienceWorld uses dense completion scores in $[0, 100]$.
Acceptance signal	Aggregate training-set return at γ_{base}; no separate validation split and no test-set feedback are used for rule acceptance.

the ScienceWorld data splits prevent a reliable comparison with the published values. We therefore restrict this cross-paper comparison to ALFWorld and use it only to contextualize the controlled results.

B.5 Reproducibility Summary

All benchmark simulation, experiment orchestration, result recording, and evaluation code ran on a local server with two Intel Xeon Gold 6430 processors, providing 64 physical cores and 128 hardware threads in total. The server had 314 GiB RAM and 2.0 GiB swap and ran Ubuntu 22.04.5 LTS with Linux kernel 6.8.0-124-generic. Inference for GPT-4o-mini, Gemini-2.0/2.5-flash, and Qwen3-8B was accessed through APIs; consequently, the local pipeline used no GPU acceleration. These specifications describe the local experimental pipeline rather than the model providers' serving hardware. Table 7 summarizes the remaining model, decoding, sampling, evaluation, and knowledge-granularity settings.

B.6 Computational Environment

The local benchmark simulators and experiment orchestration ran in the versioned software environment summarized in Table 8. User-specific installation paths are omitted.

C Additional Experimental Results

C.1 Additional Qwen3-8B Evaluation

To evaluate SKL with a recent open-weight backbone under a controlled protocol, we additionally tested Base, ReAct, Reflexion, and SKL with API-served Qwen3-8B on the 46-instance ALFWorld-Cool test subset, comprising 25 Seen and 21 Unseen instances. We used the same prompts, maximum interaction-step budget, data split, and average-reward metric as in the main evaluation. Decoding temperature was 0, and all other generation parameters retained the official defaults exposed by the API.

Table 8: Software environment used by the benchmark and experiment code. User-specific installation paths are omitted.

Layer	Versioned dependencies
Runtime	Python 3.9.21 in a Conda environment.
Core experiment stack	`openai==1.84.0`, `numpy==1.26.4`, `scipy==1.13.1`, `PyYAML==6.0.2`, `tiktoken==0.9.0`, and `scikit-learn==1.6.1`.
ALFWorld	`alfworld==0.4.2`, `textworld==1.6.2`, `fast_downward_textworld==20.6.3`, `jericho==3.3.0`, `ai2thor==2.1.0`, `torch==2.7.0`, `torchvision==0.22.0`, and `opencv-python==4.11.0.86`.
ScienceWorld	`scienceworld==1.2.3`, `py4j==0.10.9.9`, and OpenJDK 11.0.31.

Table 9: Controlled evaluation with API-served Qwen3-8B on all 46 ALFWorld-Cool test instances. Values are from one complete run.

Method	Avg. reward
Base	45.65
ReAct	32.61
Reflexion	78.26
SKL	**89.13**

In this run, SKL obtains the highest reward among the four controlled methods. Because the result is based on one complete run of a single task subset, we treat it as corroborating evidence for compatibility with this open-weight 8B backbone, rather than evidence of run-level stability or uniform superiority over methods evaluated under other protocols.

Table 10: Cross-backbone reuse on ALFWorld-Cool. Knowledge learned with Qwen3-8B is supplied to Gemini-2.5-flash without target-side knowledge learning. Results are mean $\pm$ standard deviation over five runs.

Target setting	Avg. reward	Gain
Target Base	56.96 ± 4.71	–
+ Qwen3-8B knowledge	$\mathbf{84.35 \pm 3.22}$	$+27.39$

The transferred knowledge improves Gemini-2.5-flash by 27.39 points in this setting. This result supports reuse of the same task-specific structured knowledge across these two API-served backbones on ALFWorld-Cool; it does not establish transfer across arbitrary models, benchmarks, or task definitions.

C.2 Relation-Prior Sensitivity

The relation-only sensitivity study varies λ on ALFWorld with GPT-4o-mini while disabling procedural rules. Base obtains 41.90 average reward. With empirical relations but no commonsense weight ($\lambda = 0$), reward rises to 52.92. The best evaluated setting is $\lambda = 10$, where the relation-only variant obtains 55.84 ± 1.73. Larger values reduce performance, consistent with over-weighting a generic prior relative to environment-specific evidence. The full SKL score of 81.31 ± 2.33 remains substantially higher, which is consistent with the two knowledge components serving complementary search and execution roles.

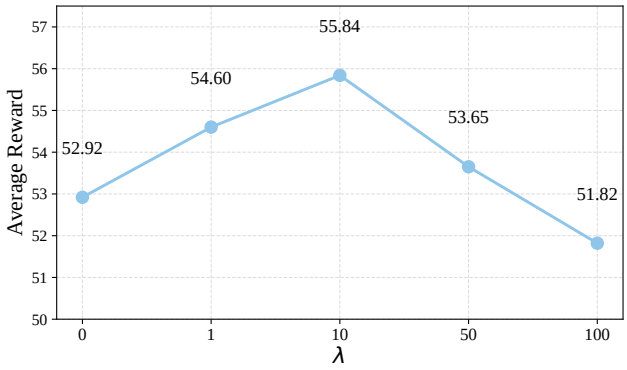


Figure 4: Relation-prior sensitivity on ALFWorld with GPT-4o-mini after disabling procedural rules. The best evaluated relation-only setting is $\lambda = 10$, which obtains 55.84 ± 1.73 average reward, compared with 41.90 for Base.

Table 11: Sensitivity to the rollout budget on 46 held-out ALFWorld-Cool instances with GPT-4o-mini. $K = 8$ is the default configuration in the main paper. Run-level uncertainty was not available for the added $K = 2$ and $K = 4$ configurations.

K	**Rollout temperatures**	**Avg. reward**
2	$\{0.0, 0.7\}$	27.39
4	$\{0.0, 0.2, 0.5, 0.7\}$	41.74
8	$\{0.0, 0.1, \ldots, 0.7\}$	**71.30**

C.3 Sensitivity to the Number of Sampled Trajectories

We additionally vary the rollout budget on the 46-instance ALFWorld-Cool test subset with GPT-4o-mini. Table 11 reports the three evaluated configurations. Each configuration contains the temperature-zero base rollout and the listed exploratory temperatures.

Across the evaluated configurations, reducing the rollout budget from $K = 8$ to $K = 4$ or $K = 2$ substantially lowers the observed reward. This pattern is consistent with a larger rollout set providing more opportunities to obtain an informative higher-return reference for contrastive rule refinement. However, K and the temperature set vary together in this experiment, and trajectory diversity was not measured separately; the results therefore characterize the complete sampling configurations rather than isolating a causal effect of K alone.

C.4 Contrastive Construction

The Failure-only comparison in the main paper retains $\mathcal{K}_{\text{rel}}$ and the same downstream edit and validation procedure, but removes exploratory rollouts and learns from the failed base trajectory alone. It obtains 64.35 ± 2.22 on ALFWorld-Cool with GPT-4o-mini, compared with 71.30 ± 3.89 for full SKL. This ablation tests the exploration-and-comparison mechanism as a whole. It does not isolate the effect of the higher-return reference independently of the extra rollouts used to obtain that reference.

C.5 Learning-Curve Interpretation

The knowledge-learning curves in the main paper plot training-set average reward after successive rule updates. Each curve uses the 100 training instances for its ALFWorld task category. The curves rise rapidly and largely stabilize within 14 updates. They describe optimization dynamics on the training set, not held-out sample complexity; Seen and Unseen generalization is evaluated separately on the official test partitions.

C.6 STAGE-WISE TOKEN ACCOUNTING

For the recorded ALFWorld-Cool knowledge-learning run with GPT-4o-mini, the complete offline process—empirical-evidence construction, eight-rollout contrastive trajectory construction, rule refinement, and repeated training-set validation—consumes approximately 89M input tokens and 1M output tokens. This is the approximately 90M-token per-task offline cost reported in the main paper. The cost is incurred once for that task-specific knowledge object and is not a claim that knowledge learning is cost-free. A separate test-time accounting over the 46 held-out ALFWorld-Cool instances in one GPT-4o-mini evaluation gives the values in Table 12.

Table 12: Aggregate LLM-token use over 46 held-out ALFWorld-Cool instances with GPT-4o-mini in one evaluation. This task-scoped accounting does not establish lower total cost or benchmark-wide amortization.

Method	**Test-Time Tokens**
Base	3.6M
ReAct	2.9M
Reflexion	7.0M
SKL	**2.2M**

These measurements separate one-time offline learning from downstream inference. Any amortization argument applies only across held-out instances that reuse the same task-specific knowledge object. They do not establish that SKL has lower total token consumption than all baselines after including offline learning.

D PROBABILISTIC RELATION CONSTRUCTION

D.1 OBSERVATION-ONLY TUPLE EXTRACTION

Let $\phi(o_t)$ be the set of entity–location tuples extracted from the textual observation o_t. The extractor operates on the same structured text visible to the agent. It first applies deterministic normalization, including case and whitespace normalization and removal of simulator instance indices, and then matches the fixed object–location templates used by the benchmark observations. Matched object and receptacle descriptions are canonicalized to the corresponding entity and location types. The extractor neither queries a language model nor accesses a latent simulator state.

For example, the representative ALFWorld case study contains the following verbatim observation after the agent reaches `countertop 2`:

Raw ALFWorld Observation from the Case Study

```
You arrive at countertop 2. On the countertop 2,
you see a apple 1, a bowl 1, a bread 1,
a cellphone 3, a ladle 3, a peppershaker 1,
a pot 2, a potato 1, a soapbottle 2, and a spoon 2.
```

After removing simulator instance indices and canonicalizing the receptacle name, the extracted set $\phi(o_t)$ comprises the following ten pairs:

```
(apple, countertop), (bowl, countertop)
(bread, countertop), (cellphone, countertop)
(ladle, countertop), (pot, countertop)
(peppershaker, countertop), (potato, countertop)
(soapbottle, countertop), (spoon, countertop)
```

The simulator-generated grammar (e.g., `a apple 1`) is preserved in the quoted observation but does not affect normalization. Because $\phi(o_t)$ is a set, repeated mentions of the same pair within one observation contribute once. Counts are further deduplicated at the trajectory level: the same pair observed at several time steps of one trajectory, for example when the agent revisits a location,

contributes once. Across the collected training trajectories $\mathcal{T}_{\text{train}}$, the count is

$$n_{e,r} = \sum_{\tau \in \mathcal{T}_{\text{train}}} \mathbb{I}\big[\exists\, t \le T_\tau :\ (e,r) \in \phi(o_t)\big], \qquad N_e = \sum_{r' \in \mathcal{R}} n_{e,r'},$$

where $\mathcal{R}$ is the candidate-location vocabulary. Thus $n_{e,r}$ is the number of training instances in which e was observed at r, which limits the influence of the collecting policy's revisitation behavior; unvisited or unopened locations still provide no evidence rather than negative evidence.

D.2 Prior Elicitation and Fusion

Given an entity e and candidate set $\mathcal{R}$, the frozen language model selects the three most likely locations from $\mathcal{R}$ and assigns each a verbalized confidence score $s(e,r) \in [0, 100]$. Unselected locations receive score zero. The normalized commonsense prior is

$$P_{\text{com}}(r \mid e) = \frac{s(e,r)}{\sum_{r' \in \mathcal{R}} s(e,r')}.$$

When $N_e > 0$, the empirical distribution is $P_{\text{emp}}(r \mid e) = n_{e,r}/N_e$, and the fused relation is

$$\mathcal{K}_{\text{rel}}(e,r) = \frac{\lambda}{N_e + \lambda} P_{\text{com}}(r \mid e) + \frac{N_e}{N_e + \lambda} P_{\text{emp}}(r \mid e).$$

If an entity has no empirical observation ($N_e = 0$), `SKL` falls back to $P_{\text{com}}(r \mid e)$; if the language model assigns no positive score to any candidate location, the relation reduces to $P_{\text{emp}}(r \mid e)$; if neither source is available, no relation context is rendered for e. As evidence accumulates, the empirical term receives more weight relative to generic commonsense priors. These normalized weights summarize observed co-occurrence; incomplete and policy-dependent coverage prevents interpreting them as calibrated environment occupancy probabilities.

D.3 Test-Time Instantiation

The relation component is not stored as one fixed textual paragraph for an entire task category. Instead, the task-specific counts are retained and instantiated at test time. In the ALFWorld implementation, the target entity is extracted from the task instruction with one language-model call, and the candidate location vocabulary $\mathcal{R}$ is read from the initial observation of the current environment, which enumerates the receptacles present and is the same observation every baseline receives. The commonsense prior is elicited over this vocabulary, fused with the frozen counts as above, renormalized over all locations with positive weight, and rendered in descending order as rounded percentages:

```
For the object "[OBJ]", it most likely appears at:
[Loc1] (∼XX%), [Loc2] (∼XX%), ...
```

Locations that carry training counts but do not appear in the current vocabulary are not filtered from this list; such entries are simply inapplicable in the current environment. The statistics are not updated from held-out observations or rewards.

E Functional Prompt Templates

The following templates reproduce the functional prompt content used in the source appendix. Runtime wrappers and transport-specific message formatting are omitted. Placeholders in braces are populated at runtime.

E.1 ALFWorld Base Agent

System Prompt for ALFWorld Base Agent

```
You are an expert agent operating in the ALFRED Embodied Environment.
```

```
{environment_knowledge}

Your task is to: {task_description}

Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and
corresponding actions: {action_history}

You are now at step {current_step}. Your current observation is:
{current_observation}

Your admissible actions are: [{admissible_actions}]

First reason step by step inside <think> </think>.
Then choose one admissible action and place it inside <action>
</action>.
```

E.2 SCIENCEWORLD BASE AGENT

The ScienceWorld implementation selects between two user-prompt templates according to whether the interaction history is empty. The shared code contains the placeholder {`variable`}, but this field is the empty string for the Base agent; it is populated only when optional learned knowledge is supplied to `SKL`. At runtime, the task description is read from ScienceWorld's `taskDesc` field, the current observation from its `look` field, and the admissible actions from the environment's valid action–object combinations. The admissible actions are rendered as one `- action` entry per line. The resulting Base prompts are as follows.

User Prompt for ScienceWorld Base Agent (First Step)

```
You are an expert agent operating in the Text-Embodied Environment.
{task_description}
{variable}

Your current observation is:
{current_observation}

Your admissible actions of the current situation are:
{admissible_actions}

Now it is your turn to take an action.
First reason step-by-step inside <think> </think> tags.
Then choose one action and write it inside <action> </action> tags.
```

User Prompt for ScienceWorld Base Agent (With History)

```
You are an expert agent operating in the Text-Embodied Environment.
{task_description}
{variable}

Before this step, you have already taken {step_count} step(s). Below
are the most recent {history_length} observations and actions:
{action_history}

You are now at step {current_step}. Your current observation is:
{current_observation}

Your admissible actions of the current situation are:
{admissible_actions}
```

```
Now it is your turn to take an action.
First reason step-by-step inside <think> </think> tags.
Then choose one action and write it inside <action> </action> tags.
```

E.3 COMMONSENSE LOCATION PRIOR

System Prompt for Commonsense Location Prior

```
You are a location prior predictor for an embodied environment agent.

Given the target object and location vocabulary, predict the TOP-3
most likely locations and assign a confidence score from 0 to 100 to
each.

Rules:
- Use ONLY commonsense priors.
- Choose ONLY from the provided vocabulary.
- Order locations by decreasing score.
- Output: Location: Score, separated by commas.

Example:
Target object: lettuce
Location vocabulary: cabinet, drawer, fridge, sinkbasin, diningtable
Output:
fridge: 95, diningtable: 30, sinkbasin: 10

Target object: {target_obj}
Location vocabulary: {vocab_str}
Output:
```

E.4 DIAGNOSIS

System Prompt for Diagnosis Module

```
Role:
You are the feedback model in an optimization system for interactive
environments.

Inputs:
- task
- agent trajectory
- automatically selected higher-return reference trajectory

Goal:
Critique the AGENT_TRAJECTORY only. Identify the earliest divergence
from the higher-return reference, explain the failure mode, and
state a general desired behavior.

Output exactly:
RESPONSE_GRADIENT:
- Divergence points: ...
- Failure modes: ...
- Desired behavior: ...
- Generalization: ...
```

E.5 PROPOSAL

System Prompt for Proposal Module

```
Role:
You are the gradient model in an optimization system for interactive
environments.

Input:
- episode context
- RESPONSE_GRADIENT

Goal:
Translate the critique into reusable edits to the VARIABLE that
improve planning, state tracking, and recovery across tasks.

Do not rewrite the VARIABLE. Return only:
VARIABLE_GRADIENT:
- Add: ...
- Remove: ...
- Strengthen: ...
```

E.6 REFINEMENT

System Prompt for Refinement Module

```
Role:
You are the optimizer in an optimization system for interactive
environments.

Inputs:
- OBJECTIVE
- current VARIABLE
- agent trajectory
- RESPONSE_GRADIENT
- one or more VARIABLE_GRADIENTs

Goal:
Produce an improved VARIABLE. Incorporate the critiques as stable,
general instructions; keep the result concise and operational;
preserve helpful existing structure.

Output only:
<IMPROVED>
...
</IMPROVED>
```

F REPRESENTATIVE LEARNED KNOWLEDGE

F.1 RELATION ARTIFACT

The probabilistic relation component is instantiated using the template in Section D; the object name, available locations, and probabilities vary by task instance. It is therefore more accurately viewed as task-specific statistics plus an instance-level rendering function than as one fixed sentence per ALFWorld category.

F.2 PROCEDURAL RULE ARTIFACTS

The blocks below are recorded outputs learned by GPT-4o-mini for the six ALFWorld task categories. They are shown to make the learned representation inspectable. Some original outputs use the terms "expert" or "expert-like." In this context, those terms refer to the automatically selected higher-return reference trajectory used by the Diagnosis module; they do not indicate a human-written or human-annotated demonstration.

ALFWorld — Pick

The agent policy system should prioritize moving directly towards locations of required objects based on observations. It should minimize unnecessary exploration of irrelevant locations and strengthen the decision-making process to include efficient pathfinding towards known object locations after examining relevant areas. The policy should also incorporate long-horizon planning with subgoals and prefer expert-like decisions when available.

ALFWorld — Examine

The agent policy system should prioritize actions that directly relate to the task goal, especially when specific tools or items, such as the desk lamp, are available. After examining relevant locations, the agent should immediately plan to use these tools to fulfill the task requirements. A mechanism should evaluate action outcomes and adapt the trajectory based on success or failure, prioritizing new areas or objects after repeated failures. The policy should avoid actions that do not contribute to the objective and should not repeat actions already found unproductive, such as opening the same drawer multiple times without new information. It should maintain long-horizon subgoals and follow higher-return reference decisions when available.

ALFWorld — Clean

The agent policy should first assess the environment for relevant objects, prioritizing locations where the target object is likely to be found. It should retrieve the required item, proceed directly to the sink basin, clean the item immediately after retrieval, and then move it to the specified final location. It should minimize exploration of irrelevant locations and evaluate candidate actions by their contribution to the task goal, including whether the item has been cleaned before final placement.

ALFWorld — Heat

After the task begins, go directly to the likely location of the target object and retrieve it. Proceed to the microwave to heat it, then move the heated object directly to the requested final receptacle. Avoid examining unrelated cabinets or receptacles unless they are required by the task, and prioritize actions that directly advance heating and placement.

ALFWorld — Cool

Prioritize visible objects on countertops before searching through cabinets. After retrieving the target object, move directly to the fridge, open it, and cool the object in one continuous sequence of relevant actions. Minimize unnecessary actions, transition between subgoals as soon as their preconditions are satisfied, and prefer efficient action sequences observed in the higher-return reference.

ALFWorld — Pick2

Move each acquired item to the designated location immediately after pickup, particularly when this advances the primary objective. Place the first item before searching for the second, prioritize task-relevant receptacles, and eliminate exploration that does not contribute to completion. The policy should track completed subgoals to avoid loops and preserve progress across the two required placements.

F.3 SCIENCEWORLD TOPIC-LEVEL PROCEDURAL RULES

The blocks below are the six topic-level procedural-rule artifacts learned by GPT-4o-mini for ScienceWorld, with exactly one knowledge object shared within each evaluated topic. We reproduce the stored text in full, with only typographic normalization for the paper. These are learned outputs rather than manually generalized templates: concrete objects and locations therefore reflect

the training evidence from which each artifact was induced. During held-out evaluation, the same fixed topic-level artifact is supplied across the different tasks and variations assigned to that topic. The terms "expert" and "expert-like" have the same meaning as above—the automatically selected higher-return reference trajectory, not a human-authored demonstration.

ScienceWorld — Matter

The agent policy should prioritize focusing on the orange juice immediately after opening the fridge, as this is the primary task object. The agent should confirm the state of the orange juice and then use the lighter on it to apply heat, facilitating the melting process. The decision-making process must emphasize efficient pathways to achieve the objective of melting the orange juice, minimizing distractions and unnecessary actions. The agent should continuously evaluate the situation and select actions that directly contribute to the task goal of changing the state of the orange juice.

ScienceWorld — Measurement

The agent policy system should immediately focus on the thermometer upon arriving in the kitchen to measure the temperature of sodium chloride. After measuring, the agent should teleport directly to the living room and focus on the unknown substance B without unnecessary movements. The policy must include a clear directive to prioritize critical tools or objects, ensuring that the agent engages in long-horizon planning with well-defined subgoals. The decision-making process should favor locations containing task-related objects, ensuring efficient planning and execution. When an expert trajectory is available, the agent should prefer expert-like decisions to maximize episode score.

ScienceWorld — Classification

Prioritize immediate objectives upon observing relevant items. When an item related to the task is seen, such as the apple tree in flower pot 2, focus on it immediately. Emphasize identifying and interacting with the most relevant objects that align with the task requirements, particularly when expert trajectories are available. Follow efficient action sequences that directly lead to task completion, especially when specific items need to be collected and moved to designated locations, like the green box in the living room. Avoid unnecessary actions that do not contribute to the task goal.

ScienceWorld — Chemistry

The agent policy system should prioritize clear command recognition for pouring and mixing actions, ensuring no ambiguity exists in action commands for pouring paint. The policy must emphasize a clear sequence of actions: first, teleport directly to the art studio to access the paint, then pour yellow paint into the bowl, followed by pouring red paint into the bowl, and finally mixing the contents of the bowl to create orange paint. This sequence should closely follow the expert trajectory to efficiently achieve the goal of creating orange paint while minimizing unnecessary actions.

ScienceWorld — Biology (Life Span)

The agent policy system should prioritize actions that align with expert trajectories, especially when they lead to identifying the animal with the longest lifespan. The policy must ensure that the agent immediately examines items likely associated with longer lifespans, such as the parrot egg, as soon as they are observed. Additionally, the decision-making process should be strengthened to ensure the agent consistently evaluates and selects multiple targets based on their relevance to the task objectives, particularly in scenarios with multiple options, rather than fixating on any single item.

ScienceWorld — Biology (Life Stages)

The agent policy should prioritize long-term observation and information gathering. After examining a subject of interest, such as the baby beaver, the agent should wait and observe its behavior and surroundings for a sufficient duration to gather insights into its life stage. The agent should minimize unnecessary teleportation and focus on understanding the task requirements by aligning actions with expert decision-making processes. This includes being patient and attentive to the context of the current task, ensuring that actions contribute to maximizing the episode score.

Table 13: Repeated outcomes on the same held-out ALFWorld-Cool instance with GPT-4o-mini. An episode length of 50 indicates that the interaction budget was exhausted without completing the task.

Method	Successes	Episode lengths
Base	0/5	50, 50, 50, 50, 50
ReAct	0/5	50, 50, 50, 50, 50
Reflexion	0/5	50, 50, 50, 50, 50
SKL	**5/5**	20, 20, 12, 7, 7

G Case Studies

G.1 ALFWorld: Cooling an Apple

We examine one held-out Seen instance from the ALFWorld Cool category with GPT-4o-mini. The goal is `put a cool apple in garbagecan.` Base, ReAct, Reflexion, and SKL receive the same task instance and 50-interaction budget. We repeat the evaluation five times for each method. As summarized in Table 13, the three baselines do not complete this instance in any run, whereas SKL completes it in all five runs.

Knowledge supplied to SKL. This episode does not receive an instance-specific demonstration, trajectory, or rule. It uses the Cool-category knowledge object learned from the training partition and frozen before held-out evaluation. The relation context ranks the rounded location probabilities for `apple` as follows: `fridge` (39%), `countertop` (32%), `diningtable` (13%), `sinkbasin` (6%), `microwave` (4%), `cabinet` (3%), `garbagecan` (1%), and `sidetable` (1%). The task-level procedural rule prioritizes visible countertop objects before exhaustive cabinet search and, after retrieving the target, directs the agent to transition to the fridge for cooling. These inputs narrow the search and specify the relevant operation, but they do not reveal the apple's actual location in this test instance.

Case analysis. The repeated outcomes in Table 13 correspond to distinct behaviors visible in the representative traces. **Base** commits early to the wrong object and then spends most of its remaining budget on sequential cabinet search, without inspecting the visible countertop that contains the requested apple. **ReAct** finds and cools the correct apple, but its placement attempts—`put`, `drop`, `open`, and `throw`—receive `Nothing happens.`; it never adopts ALFWorld's valid `move ... to ...` convention. **Reflexion** exhausts the budget through cabinet-and-drawer search and revisits already inspected locations. In contrast, **SKL** checks the relevant countertop, retrieves and cools the apple, and uses the valid final placement command.

Interpretive scope. This contrast is consistent with the intended complementary roles of $\mathcal{K}_{\text{rel}}$ and $\mathcal{K}_{\text{rule}}$, but it does not isolate their individual causal effects because both are supplied together. The instance was selected post hoc for a clear contrast rather than sampled randomly. It is therefore an illustrative failure analysis, not an estimate of how often each failure mode occurs; the aggregate results and component ablations provide the corresponding quantitative evidence.

Detailed representative trajectories. The following boxes expose the recorded `obs–act` evidence rather than only an action summary. The source traces use two native serializations: Base, Reflexion, and SKL pair the recorded `obs` with the `act` selected from it, whereas ReAct records an `act` followed by the resulting `obs`. For presentation, every environment output is labeled `obs` and every agent decision is labeled `act`; this normalization does not change the native temporal order or the original step indices. Long inventories and reasoning strings are shortened only where an explicit omission marker appears; all task-relevant objects, environment feedback, and environment-facing actions in the selected windows are retained. Line breaks and punctuation are normalized for typesetting. The baseline windows were selected post hoc to expose the documented divergence points.

Trace legend: **obs** denotes environment evidence; **act** denotes an agent decision; **red italic obs** marks a rejected or ineffective command; and **green bold obs** marks a completed subgoal. Typography and explicit labels duplicate every color cue.

Base—failed run (Steps 1–7 and 23–26 of 50)

[Step 1]
obs: You are in the middle of a room. Looking quickly around you, you see cabinet 9 ... cabinet 1, countertop 2, countertop 1, [irrelevant inventory omitted], fridge 1, garbagecan 1, microwave 1, and sinkbasin 1.
act: **inventory**

[Step 2]
obs: You are not carrying anything.
act: **go to fridge 1**

[Step 3]
obs: You arrive at fridge 1. The fridge 1 is closed.
act: **open fridge 1**

[Step 4]
obs: You open the fridge 1. The fridge 1 is open. In it, you see a egg 2, a plate 2, a potato 2, and a tomato 1.
act: **take tomato 1 from fridge 1**

[Step 5]
obs: You pick up the tomato 1 from the fridge 1.
act: **go to cabinet 1**

[Step 6]
obs: You arrive at cabinet 1. The cabinet 1 is closed.
act: **open cabinet 1**

[Step 7]
obs: You open the cabinet 1. The cabinet 1 is open. In it, you see a soapbottle 1.
act: **examine tomato 1**
[Steps 8--22 omitted: sequential cabinet search without finding the apple.]

[Step 23]
obs: You open the cabinet 9. The cabinet 9 is open. In it, you see a kettle 2.
act: **go to fridge 1**

[Step 24]
obs: You arrive at fridge 1. The fridge 1 is open. In it, you see a egg 2, a plate 2, and a potato 2.
act: **go to garbagecan 1**

[Step 25]
obs: You arrive at garbagecan 1. On the garbagecan 1, you see nothing.
act: **move tomato 1 to garbagecan 1**

[Step 26]
obs: You move the tomato 1 to the garbagecan 1.
act: **examine garbagecan 1**

ReAct—failed run (Steps 7–13 and 19–29 of 50)

[Step 7]
act: **> go to countertop 2**
obs: You arrive at countertop 2. On the countertop 2, you see a apple 1, a bowl 1, a bread 1, [irrelevant inventory omitted], and a spoon 2.

[Step 8]
act: **> take apple 1 from countertop 2**
obs: You pick up the apple 1 from the countertop 2.

[Step 9]
act: **> think: Now I have the apple (1). Next, I need to ensure it is cool before putting it in the garbage can.**
[reasoning remainder omitted]
obs: OK.

[Step 10]
act: **> think: Since I found the apple on the countertop, it is likely not cool. I need to cool it in the fridge before putting it in the garbage can.**
obs: OK.

[Step 11]
act: **> go to fridge 1**
obs: You arrive at fridge 1. The fridge 1 is open. In it, you see a egg 2, a plate 2, a potato 2, and a tomato 1.

[Step 12]
act: **> think: The fridge is open, but I need to cool the apple (1) in it. I will place the apple inside the fridge.**

obs: OK.

[Step 13]
act: > put apple 1 in fridge 1
obs: *Nothing happens.*
[Steps 14--18 omitted: the agent closes and reopens the fridge, then plans an alternative cooling command.]

[Step 19]
act: > cool apple 1 with fridge 1
obs: You cool the apple 1 using the fridge 1.

[Step 20]
act: > think: Now that I have cooled the apple (1), I need to move it to the garbage can.
obs: OK.

[Step 21]
act: > go to garbagecan 1
obs: You arrive at garbagecan 1. On the garbagecan 1, you see nothing.

[Step 22]
act: > think: Now that I am at the garbage can, I can put the cool apple in it.
obs: OK.

[Step 23]
act: > put apple 1 in garbagecan 1
obs: *Nothing happens.*

[Step 24]
act: > think: It seems I cannot directly put the apple in the garbage can. I will try to drop the apple instead.
obs: OK.

[Step 25]
act: > drop apple 1
obs: *Nothing happens.*

[Step 26]
act: > think: I should check whether another command can dispose of the apple.
[reasoning remainder omitted]
obs: OK.

[Step 27]
act: > open garbagecan 1
obs: *Nothing happens.*

[Step 28]
act: > think: I will try to throw the apple away instead.
obs: OK.

[Step 29]
act: > throw apple 1
obs: *Nothing happens.*

Reflexion—failed run (Steps 1–8 / 38–40 / 48–50 of 50)

[Step 1]
obs: You are in the middle of a room. Looking quickly around you, you see cabinets 1--9, countertops 1--2, [irrelevant inventory omitted], fridge 1, and garbagecan 1.
act: go to fridge 1

[Step 2]
obs: You arrive at fridge 1. The fridge 1 is closed.
act: open fridge 1

[Step 3]
obs: You open the fridge 1. The fridge 1 is open. In it, you see a egg 2, a plate 2, a potato 2, and a tomato 1.
act: go to countertop 1

[Step 4]
obs: You arrive at countertop 1. On the countertop 1, you see a bread 2, a egg 3, a fork 1, a knife 1, a lettuce 1, a pan 2, a pan 1, a plate 1, a pot 3, a spoon 1, a window 3, and a window 1.
act: go to cabinet 1

[Step 5]
obs: You arrive at cabinet 1. The cabinet 1 is closed.
act: open cabinet 1

[Step 6]
obs: You open the cabinet 1. The cabinet 1 is open. In it, you see a soapbottle 1.
act: open cabinet 2

[Step 7]
obs: You open the cabinet 2. The cabinet 2 is open. In it, you see a cup 1.
act: go to cabinet 3

```
[Step 8]
obs:  You arrive at cabinet 3.  The cabinet 3 is closed.
act:  open cabinet 3
[Steps 9--37 omitted:  continued cabinet/drawer search.]
[Step 38]
obs:  You open the drawer 11.  The drawer 11 is open.  In it, you see nothing.
act:  go to fridge 1
[Step 39]
obs:  You arrive at fridge 1.  The fridge 1 is open.  In it, you see a egg 2, a plate 2, a
potato 2, and a tomato 1.
act:  go to countertop 1
[Step 40]
obs:  You arrive at countertop 1.  On the countertop 1, you see a bread 2, a egg 3, a fork
1, a knife 1, a lettuce 1, a pan 2, a pan 1, a plate 1, a pot 3, a spoon 1, a window 3,
and a window 1.
act:  go to cabinet 2
[Steps 41--47 omitted:  revisiting already inspected cabinets.]
[Step 48]
obs:  You arrive at cabinet 9.  The cabinet 9 is open.  In it, you see a kettle 2.
act:  go to countertop 1
[Step 49]
obs:  You arrive at countertop 1.  On the countertop 1, you see a bread 2, a egg 3, a fork
1, a knife 1, a lettuce 1, a pan 2, a pan 1, a plate 1, a pot 3, a spoon 1, a window 3,
and a window 1.
act:  go to fridge 1
[Step 50]
obs:  You arrive at fridge 1.  The fridge 1 is open.  In it, you see a egg 2, a plate 2, a
potato 2, and a tomato 1.
act:  go to cabinet 1
```

SKL—successful run (complete 7-step trajectory)

```
[Step 1]
obs:  You are in the middle of a room.  Looking quickly around you, you see cabinets 1--9,
countertop 2, countertop 1, [irrelevant inventory omitted], fridge 1, and garbagecan 1.
act:  go to countertop 2
[Step 2]
obs: You arrive at countertop 2.  On it, you see a apple 1, a bowl 1, a bread 1,
[irrelevant inventory omitted], a potato 1, and a spoon 2.
act:  take apple 1 from countertop 2
[Step 3]
obs: You pick up the apple 1 from the countertop 2.
act:  go to fridge 1
[Step 4]
obs:  You arrive at fridge 1.  The fridge 1 is closed.
act:  open fridge 1
[Step 5]
obs:  You open the fridge 1.  The fridge 1 is open.  In it, you see a egg 2, a plate 2, a
potato 2, and a tomato 1.
act:  cool apple 1 with fridge 1
[Step 6]
obs: You cool the apple 1 using the fridge 1.
act:  go to garbagecan 1
[Step 7]
obs:  You arrive at garbagecan 1.  On the garbagecan 1, you see nothing.
act:  move apple 1 to garbagecan 1
```

G.2 SCIENCEWORLD: MELTING ICE CREAM

We next examine the held-out ScienceWorld instance `seen_Matter_melt_14` with GPT-4o-mini. The instruction is: "Your task is to melt ice cream. First, focus on the substance. Then, take actions that will cause it to change its state of matter." The episode budget is 80 interactions, and a score of 100 denotes complete success. Each method is evaluated five times on the same instance. Table 14 reports every run-level score and episode length. Base, ReAct, and Reflexion never reach 100 in these runs, whereas `SKL` reaches 100 in all five. Several failed baseline episodes terminate before step 80, so an episode length below the budget should not be interpreted as success.

Table 14: Five-run outcomes on the same held-out ScienceWorld Matter instance with GPT-4o-mini. The success threshold is 100; the maximum episode length is 80 interactions.

Method	Successes	Scores	Episode lengths
Base	0/5	5, 0, 5, 3, 5	80, 54, 80, 36, 80
ReAct	0/5	3, 7, 7, 5, 8	80, 59, 80, 80, 80
Reflexion	0/5	0, 0, 5, 0, 3	2, 5, 80, 42, 80
SKL	**5/5**	100, 100, 100, 100, 100	6, 6, 7, 6, 6

Table 15: Selected Matter-topic relation entries available to SKL in the ScienceWorld case study. Values are stored relation weights, not normalized probabilities.

Entity	Recorded locations and weights
freezer	kitchen (52)
lighter	kitchen (54)
glass cup	kitchen (50); bathroom (8); freezer (1); table (1)
fridge	kitchen (45)
stove	kitchen (53)
oven	kitchen (50)
orange juice	kitchen (6)
solid orange juice	kitchen (2)

Knowledge supplied to SKL. The stored Matter procedural rule reproduced above is not an instance-specific solution: it mentions *orange juice* in a *fridge*, whereas the held-out task concerns *ice cream* in a *freezer*. Its reusable content is the subprocedure of focusing on the target substance and applying heat with a lighter while avoiding irrelevant actions. The relation component provides complementary layout evidence. Table 15 lists the entries relevant to interpreting this episode. The displayed numbers are stored, unnormalized relation weights; they are neither calibrated probabilities nor percentages. Notably, the displayed relation subset contains no ice cream entry. It instead associates both freezer and lighter strongly with the kitchen, while the current observation must still identify the instance-specific target.

Case analysis. The trajectories make the repeated contrast in Table 14 inspectable. **Base** locates both the ice cream and the lighter, but fails to turn the disambiguated object reference into an accepted command. **ReAct** identifies the target and the intended heating subgoal, yet repeatedly reformulates the same invalid pickup command instead of resolving the environment's ambiguity feedback. **Reflexion** reaches the kitchen but attempts to focus on a glass cup before observing and grounding the task-relevant ice cream, after which the representative run terminates. In contrast, **SKL** locates the freezer, focuses on the ice cream itself, and applies the available lighter. Because the stored rule refers to orange juice in a fridge, this success reflects instance-grounded reuse of a procedure rather than literal replay.

Interpretive scope. The case is consistent with the relation entries supporting location choice and the topic-level rule supplying a transferable operation order. It does not isolate either component's causal contribution, and five repeated runs on this instance do not estimate benchmark-level performance. The instance was selected post hoc using the strict contrastive criterion of zero complete successes for all three baselines and five for SKL, rather than sampled randomly.

Detailed representative trajectories. The boxes below preserve source step indices and native temporal order. For presentation, every environment output is labeled obs and every agent decision is labeled act. Base, Reflexion, and SKL retain obs-then-act order; ReAct retains act-then-resulting-obs order. Inventory lists and long reasoning actions are shortened only where marked. The Base and ReAct windows retain the environment messages needed to audit command grounding; the Reflexion and SKL boxes contain their complete representative runs. Line breaks and punctuation are normalized for typesetting.

Base—failed run (Steps 3–8 of 80; final score 5)

```
[Step 3]
obs:  This room is called the kitchen.  You see a freezer (closed), a fridge (closed), a
lighter, [irrelevant inventory omitted], and a table with an empty glass cup.
act:  open freezer
[Step 4]
obs:  The freezer is now open.
act:  look around
[Step 5]
obs: This room is called the kitchen.  The freezer is open.  In the freezer is a glass cup
(containing ice cream).  A lighter is also in the kitchen.  [irrelevant inventory omitted]
act:  use lighter on glass cup
[Step 6]
obs: Ambiguous request:  Please enter the number for the action you intended (or blank to
cancel):  0:  use lighter (in kitchen) on glass cup (in table, in kitchen); 1:  use
lighter (in kitchen) on glass cup (in freezer, in kitchen).
act:  use lighter (in kitchen) on glass cup (in freezer, in kitchen)
[Step 7]
obs: Unknown action.  Type 'help' for a list of actions, and 'objects' for a list of
possible object referents.
act:  use lighter (in kitchen) on glass cup (in freezer, in kitchen)
[Step 8]
obs: No known action matches that input.
act:  pick up glass cup (containing ice cream)
```

ReAct—failed run (Steps 5–18 of 80; final score 3)

```
[Step 5]
act:  > open freezer
obs:  The freezer is now open.
[Step 6]
act:  > look around in freezer
obs: No known action matches that input.
[Step 7]
act:  > look inside freezer
obs: No known action matches that input.
[Step 8]
act:  > look around
obs: This room is called the kitchen.  The freezer is open.  In it is a glass cup
(containing ice cream).  A lighter is in the kitchen.  [irrelevant inventory omitted]
[Step 9]
act:  > think:  I have found the ice cream in the glass cup inside the freezer.  I need to
take the glass cup out and apply heat.
[reasoning remainder omitted]
obs:  OK.
[Step 10]
act:  > pick up glass cup
obs: Ambiguous request:  Please enter the number for the action you intended:  0:  pick up
glass cup (in table, in kitchen); 1:  pick up glass cup (in freezer, in kitchen).
[Step 11]
act:  > pick up glass cup (in freezer, in kitchen)
obs: Unknown action.  Type 'help' for a list of actions, and 'objects' for possible
referents.
[Step 12]
act:  > pick up glass cup (in freezer)
obs: No known action matches that input.
[Step 13]
act:  > pick up glass cup
obs: Ambiguous request:  0:  pick up glass cup (in table, in kitchen); 1:  pick up glass
cup (in freezer, in kitchen).
[Step 14]
act:  > pick up glass cup (in freezer, in kitchen)
obs: Unknown action.  Type 'help' for a list of actions, and 'objects' for possible
referents.
[Step 15]
act:  > pick up glass cup (in freezer, in kitchen)
obs: No known action matches that input.
[Step 16]
act:  > pick up glass cup (in freezer)
obs: No known action matches that input.
```

```
[Step 17]
act:  > think:  I am having trouble picking up the glass cup.  I will try to move it to
the table first.
obs:  OK.
[Step 18]
act:  > move glass cup (in freezer) to table
obs: No known action matches that input.
```

Reflexion—failed run (complete 2-step trajectory; final score 0)

```
[Step 1]
obs:  This room is called the art studio.  In it, you see the agent; air; a large cupboard
(closed); a table with a jug (containing nothing) and wood cups containing blue, red, and
yellow paint; and a door to the hallway (open).
act:  teleport to kitchen
[Step 2]
obs:  You teleport to the kitchen.
act:  focus on glass cup
[The representative run terminates after Step 2.]
```

SKL—successful run (complete 6-step trajectory; final score 100)

```
[Step 1]
obs:  This room is called the art studio.  In it, you see the agent; air; a large cupboard
(closed); a table with a jug (containing nothing) and wood cups containing blue, red, and
yellow paint; and a door to the hallway (open).
act:  teleport to kitchen
[Step 2]
obs:  You teleport to the kitchen.
act:  look around
[Step 3]
obs:  This room is called the kitchen.  You see a freezer (closed), a fridge (closed), a
lighter, [irrelevant inventory omitted], and a table with an empty glass cup.
act:  open freezer
[Step 4]
obs:  The freezer is now open.
act:  look around
[Step 5]
obs: This room is called the kitchen.  The freezer is open.  In the freezer is a glass cup
(containing ice cream).  A lighter is also in the kitchen.  [irrelevant inventory omitted]
act:  focus on ice cream
[Step 6]
obs: You focus on the ice cream.
act:  use lighter on ice cream
```