CLIN: A Continually Learning Language Agent for Rapid Task Adaptation and Generalization

Anonymous Authors[1]

Abstract

Language agents have shown some ability to interact with an external environment, e.g., a virtual world such as ScienceWorld, to perform complex tasks, e.g., growing a plant, without the startup costs of reinforcement learning. However, despite their zero-shot capabilities, these agents to date do not continually improve over time, beyond performance refinement on a specific task. Here we present CLIN, the first language-based agent to achieve this, so that it continually improves over multiple trials, including when both the environment and task are varied, and without requiring parameter updates. Our approach is to use a persistent, dynamic, textual memory, centered on *causal abstractions* (rather than general "helpful hints"), that is regularly updated after each trial so that the agent gradually learns useful knowledge. CLIN is able to continually improve on repeated trials on the same task and environment, outperforming state-of-the-art reflective language agents like Reflexion by 23 points in ScienceWorld and 1.4 points in ALFWorld benchmarks. CLIN can also transfer its learning to new environments and tasks, enhancing performance by 21 points in ScienceWorld and 11 points in ALFWorld. This suggests a new architecture for agents built on frozen models that can still continually and rapidly improve over time.

1. Introduction

Large language models (LLMs) have been increasingly used to interact with external environments (e.g., simulated worlds) as goal-driven agents (Reed et al., 2022). However, it has been challenging for these language agents to efficiently learn from trial-and-error as traditional reinforcement learning methods require extensive training samples and expensive model fine-tuning (Chen et al., 2021; Ammanabrolu et al., 2020). More recently, new techniques have appeared in which an agent reflects on its own past experience solving a task in a particular environment, and generates language-based insights to help it retry the task, e.g., Reflexion (Shinn et al., 2023). Such methods have the advantage of not requiring parameter updates (particularly with the frozen large language models). However, the style of such insights plays a crucial role in performance, and not all insights improve generalization performance. For example, a specific insight such as "I should go to desk 1 and find the lamp" (Shinn et al., 2023) may have limited value (or even hurt) for a different environment or task.

Our goal is a system that will continually improve over time, both while attempting the same task in the same environment, and across different tasks and environments. Our approach builds on prior work on reflection in two ways: First, we conjecture that a specific *style* of insight will be useful, namely one that captures **causal abstractions** about agent's actions, e.g., "opening doors may be necessary for movement between rooms". Causal abstractions can potentially help the agent decide which action to take in the future, and can be viewed as a kind of action model learning (Arora et al., 2018), but placed in the modern context of language models. Second, we maintain these abstractions in a **continually evolving, dynamic memory**, which is regularly updated as the agent gains experience, allowing useful causal knowledge to persist (and unhelpful knowledge to be dropped) over time and between tasks and environments, as illustrated in Figure 1.

We operationalize and evaluate this approach in a memory-augmented language agent called CLIN (**c**ontinual **l**earning from **in**teractions)[1]. CLIN is an agent that operates in a virtual, text-based environment (e.g., ScienceWorld (Wang et al., 2022), ALFWorld (Shridhar et al., 2021)) in which an agent is tasked with goals, e.g., boiling a liquid or growing a plant. We find that CLIN is able to rapidly learn about the environment and its action vocabulary and continually improve on repeated trials on the same task and environment, outperforming state-of-the-art (SOTA) reflective language

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]We will release code upon publication.

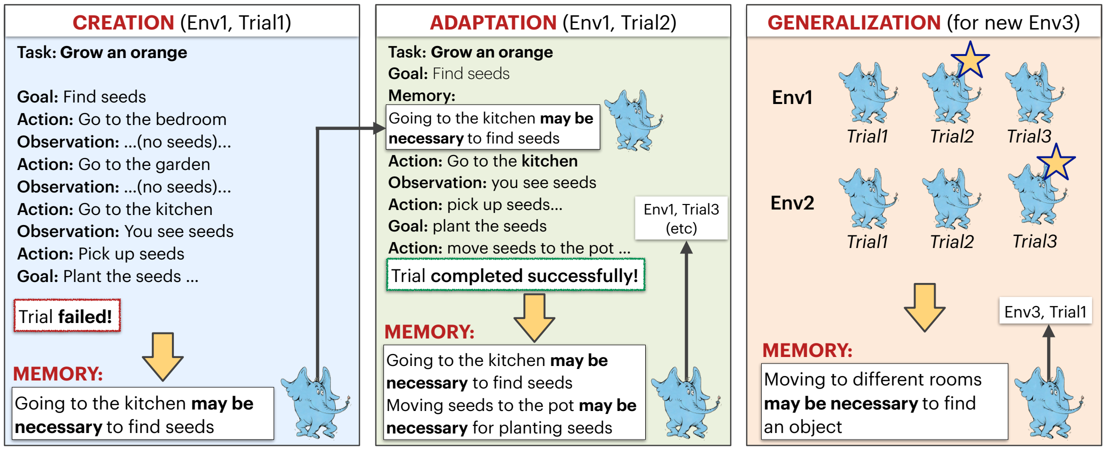

Figure 1. CLIN creates (Trial1) or adapts (Trial2+) a **memory of causal abstractions** to help in future trials by reflecting on the last trial and current memory. It does this using a suitably prompted LLM to generate the updated memory (Section 3.4 **Adaptation**). Here, reflecting on Trial1, CLIN notes in memory that going to the kitchen helped with finding seeds, enabling it to find the seeds faster in Trial2. From there, it also learns that moving the seeds to the pot helped plant the seeds. To further generalize across episodes (sequences of trials, right figure) for use in new environments, CLIN generates a summary ("meta-memory") of the best (starred) memories from each prior episode, here generating the generalization that moving to different rooms helps finding objects (Section 3.4 **Generalization**).

agents like Reflexion by 23 points in ScienceWorld. CLIN can also transfer its learning to new environments (or tasks), through continual memory updates and achieving 21 (20 for new tasks) points performance boost. Similarly, in ALFWorld, CLIN enhances its base performance by 11 points in unseen tasks/environments. Our contributions:

- We describe and evaluate CLIN, an architecture for a novel nonparametric learning paradigm. We show using a dynamic, evolving memory over time, CLIN learns faster than the short-term "reflect, use, then discard" approach used in Reflexion and other memory-based agents and generalizes better to new tasks and new environments, achieving state-of-the-art.
- We show that memory of causal abstractions (or "action models") is effective at helping the agents learn over an extended period and for varying tasks and environments—first to apply in the modern context of language-based agents.
- Overall, this work suggests that a dynamic memory, centered around causal knowledge, is a promising way forward for agents built on frozen models to continually improve over time.

2. Related Work

There is a long literature of work on agents that can navigate complex environments. A common approach is to use reinforcement learning (RL), e.g., DRRN (He et al., 2015), KG-A2C (Ammanabrolu & Hausknecht, 2020), CALM (Yao et al., 2020), where agents learn a task over repeated trials. However, while effective, such agents typically require a large number of trials to learn and have trouble adapting to unexpected changes in the test environment. More recently, (Adaptive-Agent-Team et al., 2023) demonstrated AdA, an agent that could rapidly adapt to open-ended novel 3D problems, using meta-reinforcement learning, essentially being able to change its policy on the fly. However, AdA required vast amounts of pretraining, and this skill was still limited to the style of environments and problems seen in pretraining.

Recently, LLMs have provided a new tool for building goal-directed agents (Huang et al., 2022). Given a linguistic description of the world state, a task, and a history, the LLM can be prompted to suggest next actions to take to achieve a goal, exploiting their wealth of semantic knowledge about the world and requiring little training, e.g., SayCan (Ahn et al., 2022), ReAct (Yao et al., 2022), and more recently SwiftSage (Lin et al., 2023), which combines a supervised agent and a deliberative agent together. However, while performing reasonably with little training data, such agents are unable to learn and adapt from experience.

Two recent systems have demonstrated how a frozen-model-based agent could improve at a task. Voyager (Wang et al., 2023) operates in the world of Minecraft, growing a (code-based) skill library from rich feedback of its failures. Reflexion (Shinn et al., 2023) improves at a task by *reflecting* on a failed attempt at that task and devise a new plan that accounted for that mistake, used in the subsequent prompt to retry the task. While Reflexion did not have a long-term memory, and its reflections were task- and environment-specific, e.g., "In the next trial, I will go to desk 1 and find the lamp.", we take inspiration from it to build an agent, CLIN, which continually maintains and adapts a long-term, persistent memory of reflections, useful across different

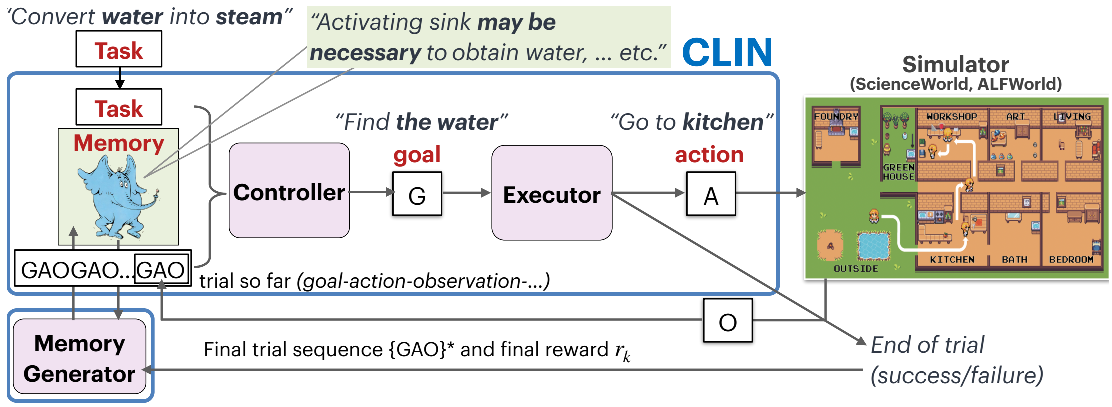

Figure 2. The architecture of CLIN. A **controller** takes the current task, retrievals from memory, and the trial so far, to generate the next goal to achieve. The **executor** then converts this to a valid action to perform towards that goal. The simulator then performs the action and returns an observation of that action's effect. Memory is updated at the end of each trial by the **memory generator** (Section 3.4).

trials, tasks, and environments.

More generally, others have found that a memory of useful learnings can be used to improve frozen LLM behavior, e.g., in QA (Dalvi et al., 2022; Tandon et al., 2022; Madaan et al., 2023), or for modeling social behavior (Park et al., 2023). We apply this finding to goal-directed agents.

Finally, we note that the *content* of experiential memory is also important. Specifically, CLIN learns a memory of *causal abstractions*, which can be seen as learning a linguistic form of action model, describing the causal effects of actions. While there has been work in the planning community of learning action models in a fully formalized context (Arora et al., 2018; Aineto et al., 2018), CLIN loosely applies this idea in the linguistic world of LLM agents.

3. Approach

3.1. Acting in the World

We follow the normal formalization for an agent performing actions in a partially observable environment, but add a memory S as an additional input for decision making. The memory contains learned task/environment knowledge to help the agent make better decisions in the next trial, and is updated at the end of each trial (described shortly). At each time step t, given a task m (e.g., "grow an orange"), memory S, and the history of actions so far, the agent decides on its next goal g and action a in pursuit of that goal. In response, the environment returns the result of executing a in the form of an observation o and a reward r. This repeats until an end state is reached (such as completing, failing, or timing out). Thus at each step t, the history so far is $\mathcal{T}_{\leq t} = \{g_i, a_i, o_i\}^*_{i \leq t}$, and the agent's decision-making task at each time step t can be described as:

$$m + e + \mathcal{T}_{\leq t} + S \rightarrow g_{t+1} \rightarrow a_{t+1} \quad (1)$$

We describe the implementation of this shortly. The full sequence of steps $\mathcal{T}_{end}$ to an end-state is called a *trial*. If the *same* task m is attempted in the *same* environment K times (resetting the environment each time), we call the collection of the K trials an *episode*.

3.2. Continual Learning

The agent's goal is to maximize its reward r (e.g., completing the task) on new trials. Learning occurs by updating the dynamic memory S between (not during) trials, using a memory update function that takes memories from old trials as input, and generates a new memory for the next trial. Note that the memory does not grow monotonically; it may drop previous memory items and add new ones. Also note it is not perfect; some memory items may be erroneous, and ideally be dropped or modified in subsequent iterations. Learning is continuous in the sense that each new memory is generated from an ever-growing collection of previous memories. We define two classes of learning:

1. **Within-episode learning ("adaptation")** - same task m and environment e. After each trial k, a new memory S_{k+1} is generated from the most recent trial history $\mathcal{T}_k$ and final reward r_k, and memories from prior trials:

$$\mathcal{T}_k + r_k + \{S_{\leq k}\} \rightarrow S_{k+1} \quad (2)$$

   S_{k+1} is then used to retry the same task m, next trial.
2. **Cross-episode learning ("generalization")** - new task m_{new} or environment e_{new}. Given a *new* task or environment, an initial starting memory is generated using memories from *other* episodes. Specifically, we select the "best" (defined later) memory from each prior episode as inputs, and generate a new memory for use in this unexplored task/environment as output:

$$m_{\text{new}} + e_{\text{new}} + \{S_{\text{best}}, r_k\}_{\text{prior\_episodes}} \rightarrow S_{\text{new}} \quad (3)$$

Because this new memory generalizes prior memories, we also refer to it as a "meta-memory". Note that some generalizations in S_{new} may be overly specific or wrong. However, we see both a net initial benefit in using S_{new}, and further task improvement in subsequent trials as S_{new} is refined (adaptation), described later in Section 4.

3.3. CLIN: Agent Architecture

Our implementation, called CLIN, comprises three components for acting: the **memory**, a **controller**, and an **executor**. Learning then occurs using a fourth module, a **memory generator**, to generate an updated memory after each trial. These are illustrated in Figure 2.

Memory. CLIN's memory ($\mathcal{S}$) is a persistent, dynamic collection of NL sentences expressing CLIN's current understanding of actions and their effects. Specifically, each sentence expresses a *causal abstraction* between actions, e.g., "opening the fridge is `necessary` to access apple juice", as well as negative learnings, e.g., "moving to another room `does not contribute` to freezing mercury.". Such statements are learned from past experiences (described shortly). Their role is to help CLIN make better action choices (Eqn 1). Causal abstractions constitute CLIN's current understanding of the way the world behaves, and can be viewed as a modern version of *action models* used in formal planning, describing the effects of actions in the world (Arora et al., 2018).

Controller. At each time step in a trial, the controller generates the next goal to pursue in service of the overall task m. In CLIN the controller is a frozen LLM, whose prompt includes the current **task** m, e.g., "convert water into steam", selected statements from the **current memory** $\mathcal{S}$ (a list of sentences), and the **trial so far** (the sequence of goal-action-observation triples. It is prompted to output the next **goal** g_{t+1} to pursue, e.g., "find water". Note we only use selected statements from memory (rather than the whole memory), in order to avoid irrelevant knowledge distracting the controller. Selection is itself done with a separate query to the LLM, prompting it to list relevant memory items, with the full memory S and the task included in that prompt.

Executor. The role of the executor is to convert the generated goal g_{t+1} into a valid **action** a_{t+1} that can be executed in the environment in pursuit of that goal. In other words, it serves to map goals into the specific action space of the environment. Again a (frozen) LLM is used, whose prompt includes the goal g_{t+1} (from the controller, above), the trial so far, and all the possible actions that can be performed in the current state (provided by the environment, as is standard practice in current generative agent research (Ahn et al., 2022; Yao et al., 2022; Lin et al., 2023; Park et al., 2023)). The list of possible actions is expressed as possible action templates and available objects that can instantiate them,

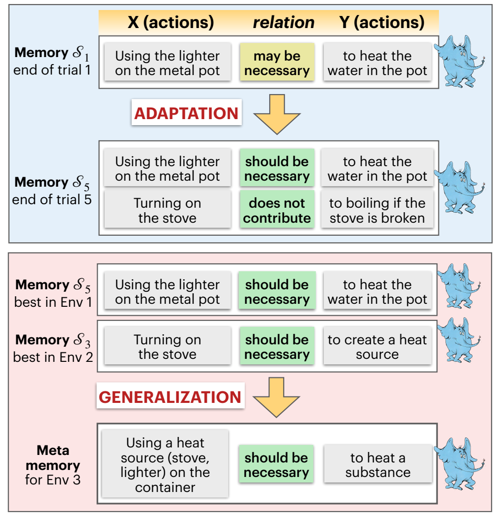

Figure 3. Examples of Memory Update in CLIN.

rather than a combinatorially large enumeration of possible actions. The model is then prompted to generate a candidate action to perform (see prompt in Figure 6). Finally, CLIN checks this candidate action is one of the valid actions. If it is not, it finds the most similar valid action using the pre-trained embeddings from the sentence-transformer model (Reimers & Gurevych, 2019). If the top-ranked valid action has a similarity score greater than a threshold (here, 0.9, chosen as a hyperparameter), the action is selected. Otherwise, we perform iterative refinement (Madaan et al., 2023) by suffixing the context with feedback that the generated candidate action is not executable. This allows the executor to retry the generation up to a max number of tries (here, 5).

Finally, upon executing the action a_{t+1}, CLIN receives a partial next state, as an **observation**, from the environment and the reward $(r) \in [0, 1]$, provided by the environment. A snapshot of a full trial is given in lines 4-10 in Algorithm 1.

Note that CLIN does not make use of any gold data to identify goals and memories. Rather, we expect CLIN to perform a balanced act of exploration-exploitation by interacting, learning, and adapting to unseen tasks or environment configurations—a key difference from few-shot generative agents in previous work (Ahn et al., 2022; Yao et al., 2022; Lin et al., 2023; Park et al., 2023).

3.4. Continual Learning in CLIN

At the end of each trial (completion or failure), CLIN uses a **memory generator** to create or update its memory. The memory generator is a (frozen) LLM prompted to reflect on the current trial and memory, and generate a new memory of insights in the form of (English sentences expressing) useful **causal abstractions**.

To make the LLM to generate causal abstractions, we use

Algorithm 1 Continual Learning with CLIN

procedure ADAPTATION(Task: m, Env: e, Memory: $\mathcal{S}$):
Initialize Memory: $\mathcal{S}_0$
for $k \in 1, \cdots, K$ **do**:
  Intialize Trial $\mathcal{T}$, t
  while $t <$ max. steps or task not complete **do**:
    $g_t = \texttt{Controller}\ (m, e, \mathcal{T}_{<t}, \mathcal{S}_{k-1})$
    $a_t = \texttt{Executor}\ (g_t$, admissible actions$)$
    $r_t, o_t = \texttt{Simulator}\ (\mathcal{T}_{<t}, a_t)$
    $\mathcal{T}_{<t+1} = \mathcal{T}_{<t} + (g_t, a_t, o_t, r_t)$
  Final reward $r_{\mathrm{k}} = r_t$
  $\mathcal{S}_k = \texttt{memory-generator}\ (\{\mathcal{S}_{<k}\}, \mathcal{T}_k, r_k)$

procedure GENERALIZATION(Task: m, Env: e, past m'/e')
$\{\mathcal{S}_{\text{best}}, r_k\} = \texttt{best-memories}$ (past m'/e')
$\mathcal{S}_{\text{meta}} = \texttt{meta-memory}\ (\{\mathcal{S}_{\text{best}}, r_k\}, m, e)$
ADAPTATION(m, e, $\mathcal{S}_{\text{meta}}$)

special instructions in the prompt that ask the LLM to generate insights in a particular templated syntax (see prompt in Figure 7). To capture actions enabling desired changes and helpful state transitions, we use the template "`X is NECESSARY to Y`", and to capture contrastive examples of unsuitable actions and state transitions, we employ "`X DOES NOT CONTRIBUTE to Y`"(Figure 3), where `X`, `Y` are related to actions. These abstractions are functionally analogous to hindsight experience replay (Andrychowicz et al., 2017), obtained from CLIN's past self-explorations. In addition, to allow the LLM to express uncertainty, we encourage it to use modifiers: "`X may ...`" to denote moderate to high uncertainty, and "`X should ...`" to indicate low uncertainty (See Figure 3).

As described earlier, there are two kinds of memory update needed: (a) re-generating the memory when retrying the same task in the same environment ("adaptation") (b) re-generating the memory for a *new* task / environment ("generalization"), as we now describe.

Within-episode learning ("adaptation"). To update the memory after each trial within an episode (Eqn 2), the memory generator is prompted with the most recent trial (a sequence of (g_t, a_t, o_t) tuples and the final reward r_k[2]), and the memories from the three most recent trials $\{\mathcal{S}_{k-2}, \mathcal{S}_{k-1}, \mathcal{S}_k\}$. It is then prompted to generate an updated memory $\mathcal{S}_{k+1}$, namely a new list of semi-structured causal abstractions in the forms described above, for use in the next trial. Although we do not specify a maximum size for the memory, we observe that size of the generated memory (i.e., the number of causal abstractions generated) is far less than the number of actions executed in the trial, indicating the memory-generator additionally performs a saliency-based pruning to keep only important insights based on the success of the trial (final reward r_k for trial $\mathcal{T}_k$).

Cross-episode learning ("generalization"). Given a new task m_{new} or environment e_{new}, the memory generator is prompted to generate a suitable memory generalizing from the best trials in previous episodes on different tasks/environments, suitable for this new situation (Eqn 3). Following the prioritized level replay scheme (Jiang et al., 2021), we choose the most successful trial per episode (based on the reward r_k) and retrieve memories abstracted from those trials with a fixed archive of size 10, a hyperparameter. If the *environment* is new, the prompt instructs the LLM to generate a memory helpful "to solve the same task in a new environment configuration", given the new task description. The prompt is designed to encourage the LLM to generate generic causal insights about the task, not tied to specific environmental details (Figure 8). Similarly, if the task is new, the prompt is modified accordingly (Figure 9).

4. Results and Analysis

Experimental Setup. Test-time adaptation and generalization via continual learning require a variety of complex tasks and environment configurations to allow an agent to explore, learn latent causal insights from interactions, and exploit them in the future. We evaluate CLIN's performance in two benchmarks: **ScienceWorld** (Wang et al., 2022) and **ALFWorld** (Shridhar et al., 2021). Both benchmarks consists of a text-based interactive environment requiring complex interactive reasoning processes to solve a plethora of tasks. ScienceWorld focuses on science-theory-based tasks[3] spanning several diverse classes (e.g., thermodynamics, genetics, friction, etc.). ALFWorld has 6 categories of household tasks: Pick, Clean, Heat, Cool, Look, and Pick-two-items.

For ScienceWorld, we evaluate on 18 tasks (two task instances from 9 task classes) in several environment configurations from the test split resulting in a total of 164 task-environment combinations. For ALFWorld, we have a total of 134 task-environment combinations (test split). We evaluate based on the final score provided by the simulator; for ScienceWorld, the score ranges from $0 \rightarrow 100$, and for ALFWorld score is binary, success/failure. Now, we define our setups for zero-shot adaptation (ADAPT) and generalization (GEN-ENV and GEN-TASK).

ADAPT: This setup focuses on CLIN's ability to adapt to a task by attempting it for several trials in the same environment configuration. Most importantly, CLIN initializes with an empty memory at the beginning of the first trial and generates memory at the end of each trial. While the environment gets reset at the trial boundary, CLIN's mem-

[2]The reward is converted to NL feedback for a LLM using 7 simple rules, e.g., "if score >= 0 and score < 20 then feedback = *"The agent performed poorly and made some progress but not enough to solve the task."*

[3]ScienceWorld tasks are grouped into Short (S), e.g., *pick & place* and Long (L), e.g., *grow plant*, based on the # of gold actions.

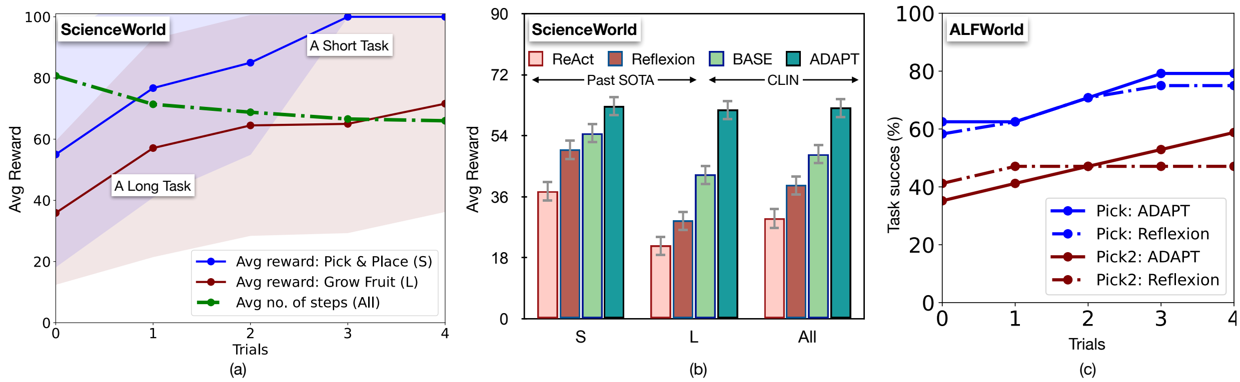

Figure 4. **Rapid task adaptation with CLIN. (a)** Example tasks with CLIN's adaptation. For CLIN, Trial-0 is BASE, Trial-4 is ADAPT. Comparison of CLIN with Reflexion (Shinn et al., 2023) in **(b)** ScienceWorld and **(c)** ALFWorld. (More in Appendix C).

ory continues to be updated, capturing informative causal abstractions pertaining to both successful and failed actions. Here, we compare with Reflexion (Shinn et al., 2023), a SOTA, however, CLIN differs from Reflexion by how the memory is abstracted.

GEN-ENV: In this setup, we focus on CLIN's ability to transfer its learning from past experiences to solve tasks in an unseen environment. For a task m, we run CLIN for 10 different (train) environment settings (with varying objects and starting locations) and then create meta-memories from its exploration to solve the same task in an unseen (test) environment. Here, we compare CLIN with RL methods DRRN (He et al., 2015), KG-A2C (Ammanabrolu & Hausknecht, 2020), and CALM (Yao et al., 2020) trained on all (large) training variations with simulator reward and Generative Language agents, SayCan (Ahn et al., 2022), ReAct (Yao et al., 2022), and Reflexion (Shinn et al., 2023), prompted with few-shot demonstrations.

GEN-TASK: In this setup, we focus on CLIN's ability to transfer its learning from past experiences to solve a new task in the same environment. For an environment e, we run CLIN for to solve a task m and then condense its learning to solve a novel task m' in the environment e. We took all test examples where we have a different task defined in the same environment configuration. (Adaptive-Agent-Team et al., 2023) suggests that transferring learning from a random task can be very hard; hence we couple tasks that are related (revolve around overlapping task-critical objects/locations such water, kitchen), such as *boil* and *freeze* to measure transfer learning from one to the other. This is a novel setup where we do not have any off-the-shelf baselines. However, here, we compare against CLIN-BASE, a strong baseline.

GEN-ADAPT (G+A): If CLIN, in GEN-ENV or GEN-TASK setting, does not successfully complete the new task, it can continue learning and retrying that task. We refer to this setup as GEN-ADAPT. CLIN can use any instruction-tuned LLM (Chung et al., 2022) as part of the controller, executor, and memory generator. In this paper, we use `gpt-4`, the same as our generative agent baselines.

4.1. CLIN Exhibits Rapid Task Adaptation

Figure 4a demonstrates two example trends in ScienceWorld where CLIN learns from its own prior attempts (ADAPT) and gets better at solving a given task. Apart from length, the difficulty level of a task also depends on the environment configuration (hence, variance across environment configurations for each task). CLIN quickly adapts to a short task, *Pick & Place*, solving it in its 4th attempt, whereas a longer task, *Grow Fruit* is not solved after 5 (max) tries. Furthermore, CLIN becomes more efficient in later trials by solving the tasks with a lower number of (average) steps.

Next, we compare CLIN with Reflexion, the reflective SOTA agent, in Figure 4b. CLIN already starts off with a stronger base performance (see discussion in 4.3), however, CLIN's relative improvement in ADAPT is significantly stronger than Reflexion's gain from its base agent ReAct. CLIN's relative improvement is higher for longer tasks. This can be attributed to CLIN's persistent memory, which gets refined over past trials. CLIN accumulates both useful (for the task) and harmful (for the task) causal learnings, whereas Reflexion only learns from its mistakes, lacking comprehensive learning.

We found similar trends on the ALFWorld benchmark. Figure 4c shows how CLIN can improve its performance across trials for task types: Pick and Pick2. During adaptation, CLIN (1) learns facts specific to the environment: e.g., "Searching on sofa should be necessary to find the second keychain.", and (2) hypothesizes for sub-goals it couldn't achieve "Checking other locations like drawers and shelves may be necessary to finding the second CD.". After 5 trials, CLIN achieves highest performance on 5 out of 6 task types (Table 2). CLIN-ADAPT outperforms ReAct by 8.9 points and Reflexion by 1.4 points when averaged over all tasks.

		RL Methods			Generative Language Agents			CLIN (ours)		
Task	**Type**	**DRRN**	**KGA2C**	**CALM**	**SayCan**	**ReAct**	**Reflexion**	**Base**	**Gen-Env**	**G+A**
Temp	S	6.6	6.0	1.0	**26.4**	7.2	5.9	25.2	15.7	13.8
Temp	S	5.5	11.0	1.0	8.0	6.1	28.6	53.2	49.7	**58.2**
Pick&Place	S	15.0	18.0	10.0	22.9	26.7	64.9	92.5	59.2	**100.0**
Pick&Place	S	21.7	16.0	10.0	20.9	53.3	16.4	55.0	**100.0**	**100.0**
Chemistry	S	15.8	17.0	3.0	47.8	51.0	**70.4**	44.5	42.2	51.7
Chemistry	S	26.7	19.0	6.0	39.3	58.9	70.7	56.7	85.6	**93.3**
Lifespan	S	50.0	43.0	6.0	80.0	60.0	**100.0**	85.0	65.0	**100.0**
Lifespan	S	50.0	32.0	10.0	67.5	67.5	84.4	70.0	75.0	**90.0**
Biology	S	8.0	10.0	0.0	16.0	8.0	8.0	10.0	32.0	**32.0**
Boil	L	3.5	0.0	0.0	**33.1**	3.5	4.2	7.0	4.4	16.3
Freeze	L	0.0	4.0	0.0	3.9	7.8	7.8	**10.0**	8.9	**10.0**
GrowPlant	L	8.0	6.0	2.0	9.9	9.1	7.3	10.2	10.9	**11.2**
GrowFruit	L	14.3	11.0	4.0	13.9	18.6	13.0	35.9	70.8	**94.5**
Biology	L	21.0	5.0	4.0	20.9	27.7	2.6	70.0	42.8	**85.6**
Force	L	10.0	4.0	0.0	21.9	40.5	50.6	53.5	70.0	**100.0**
Friction	L	10.0	4.0	3.0	32.3	44.0	**100.0**	56.5	70.0	94.0
Genetics	L	16.8	11.0	2.0	67.5	25.7	50.9	77.4	84.5	**100.0**
Genetics	L	17.0	11.0	2.0	59.5	16.8	23.7	62.3	61.4	**100.0**
	S	22.1	19.1	5.2	36.5	37.6	49.9	54.7	58.3	**71.0**
	L	11.2	6.2	1.9	29.2	21.5	28.9	42.5	47.1	**68.0**
	All	16.7	12.7	3.6	32.9	29.6	39.4	48.6	52.7	**69.5**

Table 1. Comparing CLIN with baselines for **generalization across unseen environments** in ScienceWorld

Method	**Pick**	**Clean**	**Heat**	**Cool**	**Look**	**Pick2**	**All**
ReAct	58.3	71.0	87.0	81.0	94.4	41.2	72.4
Reflexion	75.0	74.2	**91.3**	90.5	**100.0**	47.1	79.9
CLIN							
Adapt	79.2	74.2	87.0	90.5	**100.0**	58.8	81.3
Gen-Env+A	**83.3**	**77.4**	87.0	**95.2**	**100.0**	58.8	**83.6**
Gen-Task+A	79.2	74.2	**91.3**	90.5	**100.0**	**64.7**	82.8

Table 2. **Adaptation** and **generalization** results in ALFWorld

4.2. CLIN Outperforms SOTA, Generalizing to Novel Environments and Tasks

New ScienceWorld environments. Table 1 compares CLIN with baselines that learn from training environmental variants for a task to improve its performance in a novel environment [4]. Language agents (including CLIN) that use NL feedback from the ScienceWorld (e.g., "Door to the kitchen is closed") perform significantly better compared to RL methods that purely rely on (sparse) numeric rewards from the environment to learn a policy. We observe a positive generalization effect in Gen-Env (average 4 point gain) compared to Base where CLIN tries to solve the tasks zero-shot. With a strong Base performance, CLIN beats all baselines in generalization performance. Furthermore, in G+A, CLIN shows a substantial 16 additional improvement, beating the SOTA reflective agent by 23 points. Figure 5(a) additionally shows trend of improvement compared to when CLIN does not start with a meta-memory. Meta-memory helps CLIN with a stronger start than Base (52.7 vs. 48.6), with a continued gain in scores till the end of Trial-4 (G+A: 69.5 vs. Adapt: 62.2). The stronger start for CLIN with meta-memory also results in fewer steps to solve a task. Unlike imitation learning-based agents, TDT (Wang et al., 2022) and SwiftSage (Lin et al., 2023), CLIN (and most baselines) do not use any gold trajectories. Learning only from self-generated trajectories, CLIN outperforms TDT on all 18 tasks and SwiftSage on 8/18 (mostly long) tasks.

[4] Baseline numbers are derived from Table 1 in (Lin et al., 2023)

New ScienceWorld tasks. Mirroring trends from Gen-Env, CLIN demonstrates strong transfer learning to new tasks (Figure 5(b)) with 13-point improvement over its Base performance, being better at 38.8% of datapoints. The improvement attributes to critical learning about the environment ("apple juice is in the fridge", required for both boiling and freezing it), leading to improvement in previously low-performing tasks in both Adapt and Gen-Env setups. This transfer learning in Gen-Task and G+A helps CLIN to solve the tasks with fewer steps[5] and achieve higher rewards.

New ALFWorld environments/tasks. Table 2 shows that CLIN can generalize its learnings across environments (Gen-Env) and across tasks (Gen-Task) to improve its success rate further. In the Gen-Env setting, CLIN had access to memories from other tasks of same type. This helped CLIN improve its success rate by 11.2% for Gen-Env and by 10.4% for Gen-Task from its base performance.

4.3. Discussion

A qualitative example. Figure 3 depicts how memory items get refined during task adaptation and for generaliza-

[5] # steps in Figure 5(a),(b) are normalized between 0-1, 1 being maximum #steps allowed for a task.

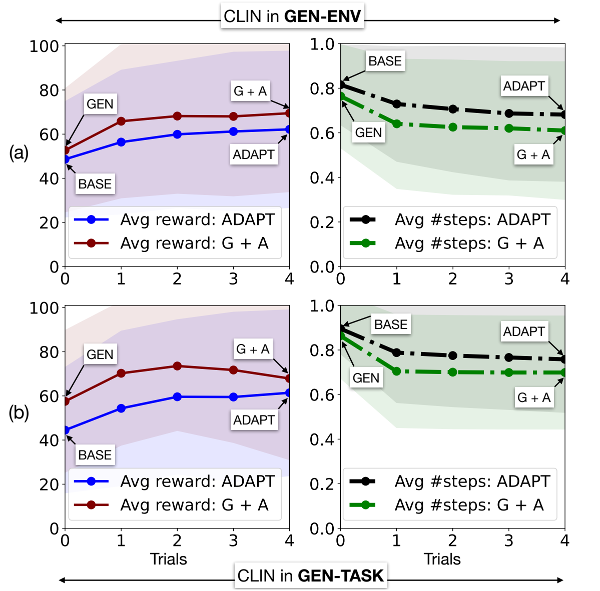

Figure 5. Reward and #steps trends for CLIN in **(a)** GEN-ENV and **(b)** GEN-TASK for ScienceWorld.

tion for a task *boil*. `Env2` has a working stove, whereas in `Env1`, the stove is broken, but a lighter is available as an alternative. CLIN acquires that knowledge through adaptation, which later can be applied to cross-episode learning via the generalized meta-memory. Appendix B contains example memories for adaptation and generalization.

Importance of memory structure. CLIN extracts causal abstractions structured around 'necessary' and 'does not contribute' relations. To ablate, we modified our memory generator to generate free-form advice for future trials, which, however, ended up generating generic insights without any causal abstractions (Figure 13). In ScieneWorld, the average reward drops by 6 points (in 10% cases than CLIN), and in ALFWorld, the success rate drops by 1.4 points when using the unstructured memory, indicating the usefulness of causal abstractions, as shown in Table 3.

Ablation Setup	**Δavg score (↓)**	**%ep. drop. (↑)**
ScienceWorld	-6.2	10.0
ALFWorld	–	1.4

Table 3. Ablation for CLIN's causal memory

Memory correctness. While the final performance with memory is indicative of their effectiveness, we performed additional human evaluation of generated memory insights for correctness. For generalization setups, we randomly 10 task-environment combinations to evaluate the correctness of memories used in them, notably the meta-memory used for trial 0 (GEN) and memory adapted for the best trial (GEN-ADAPT). Two annotators rated the insights (cohen's $\kappa = 0.78$) for correctness with reference to gold trajectories. Table 4 shows that some meta-memories may not be applicable initially; however, with adaptation, in later trials, the correctness of the memory insights significantly improves, leading to a direct increase in task performance.

	ScienceWorld				ALFWorld	
Insights	GEN-ENV	G+A	GEN-TASK	G+A	GEN	G+A
Total	100	105	98	107	94	92
Correct	72.0%	**91.4%**	73.9%	**91.1%**	72.3%	**90.1%**

Table 4. Memory correctness for CLIN

Limitation: Lack of exploration. CLIN's learnings are dependent on its own past experience. An insight related to an unobserved location or unexplored action can never be generated. Hence, exploration becomes important when task-critical location or action is unknown to CLIN from past trials. For example, to create orange paint, the agent must find red and yellow paint from the art studio. However, the art studio is not visible when CLIN starts from the 'outside.' Without that knowledge, CLIN tries alternative methods failingly to create orange paints from other irrelevant objects (e.g., an orange) and remains unsuccessful. If an insight related to the art studio appears from past exploration, CLIN is able to successfully complete the task.

Limitation: Poor memory retrieval. For a task of boiling gallium, CLIN is supposed to use oven/blast furnace and not a stove. In the meta-memory for boiling tasks, there are two insights regarding the act of boiling: "Activating stove should be necessary to boil a substance" and "Using an alternative heat source (e.g., oven or fire pit) may be necessary if the initial heat source is insufficient." However, CLIN repeatedly retrieves the former and hence failing at the task despite performing other actions (e.g., finding gallium) correctly. This problem intensifies at the initial trial during generalization due to the presence of insights with varied initial conditions for them to be applied. This can be circumvented by improved memory representation, which we leave as a future work.

5. Conclusion

Our goal is a system that can continually improve over time, both while rapidly adapting to a task by multiple retries and efficiently generalizing to novel tasks and environments. We propose CLIN, an architecture for language agents that constructs a persistent, dynamic memory of causal abstractions, refines it over time and uses it effectively to improve its performance on future tasks, achieving state-of-the-art performance. Our work systematically evaluates a novel nonparametric learning paradigm, promising never-ending learning abilities to frozen language agents.

6. Impact Statement

This work aims to develop a novel learning paradigm for frozen models using a memory-based framework requiring parameter updates. Since our agent operates in a closed environment, we do not foresee any negative consequences of our system. We hope to contribute to the growing literature on language agents by formally exploring their capabilities in continual learning setups. We will release our code upon publication for reproducibility.

References

Adaptive-Agent-Team, Bauer, J., Baumli, K., Baveja, S., Behbahani, F. M. P., Bhoopchand, A., Bradley-Schmieg, N., Chang, M., Clay, N., Collister, A., Dasagi, V., Gonzalez, L., Gregor, K., Hughes, E., Kashem, S., Loks-Thompson, M., Openshaw, H., Parker-Holder, J., Pathak, S., Nieves, N. P., Rakicevic, N., Rocktäschel, T., Schroecker, Y., Sygnowski, J., Tuyls, K., York, S., Zacherl, A., and Zhang, L. M. Human-timescale adaptation in an open-ended task space. In International Conference on Machine Learning, 2023. URL https://api.semanticscholar.org/CorpusID:255998274.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R. C., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D. M., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., and Yan, M. Do as i can, not as i say: Grounding language in robotic affordances. In Conference on Robot Learning, 2022. URL https://api.semanticscholar.org/CorpusID:247939706.

Aineto, D., Jiménez, S., and Onaindía, E. Learning strips action models with classical planning. In International Conference on Automated Planning and Scheduling, 2018. URL https://api.semanticscholar.org/CorpusID:49405691.

Ammanabrolu, P. and Hausknecht, M. J. Graph constrained reinforcement learning for natural language action spaces. In ICLR, 2020.

Ammanabrolu, P., Urbanek, J., Li, M., Szlam, A., Rocktaschel, T., and Weston, J. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In North American Chapter of the Association for Computational Linguistics, 2020. URL https://api.semanticscholar.org/CorpusID:222125301.

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. ArXiv, abs/1707.01495, 2017. URL https://api.semanticscholar.org/CorpusID:3532908.

Arora, A., Fiorino, H., Pellier, D., Métivier, M., and Pesty, S. A review of learning planning action models. The Knowledge Engineering Review, 33, 2018. URL https://api.semanticscholar.org/CorpusID:56483203.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In Neural Information Processing Systems, 2021. URL https://api.semanticscholar.org/CorpusID:235294299.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Valter, D., Narang, S., Mishra, G., Yu, A. W., Zhao, V., Huang, Y., Dai, A. M., Yu, H., Petrov, S., hsin Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models. ArXiv, abs/2210.11416, 2022. URL https://api.semanticscholar.org/CorpusID:253018554.

Dalvi, B., Tafjord, O., and Clark, P. Towards teachable reasoning systems: Using a dynamic memory of user feedback for continual system improvement. In EMNLP, 2022.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. Deep reinforcement learning with a natural language action space. arXiv: Artificial Intelligence, 2015. URL https://api.semanticscholar.org/CorpusID:15986631.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In International Conference on Machine Learning, pp. 9118–9147. PMLR, 2022.

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J. N., Grefenstette, E., and Rocktaschel, T. Replay-guided adversarial environment design. In Neural Information Processing Systems, 2021. URL https://api.semanticscholar.org/CorpusID:238408352.

Lin, B. Y., Fu, Y., Yang, K., Ammanabrolu, P., Brahman, F., Huang, S., Bhagavatula, C., Choi, Y., and Ren, X.

Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. ArXiv, abs/2305.17390, 2023. URL https://api.semanticscholar.org/CorpusID:258960143.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Welleck, S., Majumder, B. P., Gupta, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. ArXiv, abs/2303.17651, 2023. URL https://api.semanticscholar.org/CorpusID:257900871.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. arXiv preprint arXiv:2304.03442, 2023.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A. D., Heess, N. M. O., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. Trans. Mach. Learn. Res., 2022, 2022. URL https://api.semanticscholar.org/CorpusID:248722148.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Shinn, N., Labash, B., and Gopinath, A. Reflexion: an autonomous agent with dynamic memory and self-reflection. arXiv preprint arXiv:2303.11366, 2023.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. URL https://arxiv.org/abs/2010.03768.

Tandon, N., Madaan, A., Clark, P., and Yang, Y. Memory-assisted prompt editing to improve GPT-3 after deployment. In ACL Workshop on Commonsense Representation and Reasoning (CSRR'22), 2022. (also arxiv:2201.06009).

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L. J., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. ArXiv, abs/2305.16291, 2023. URL https://api.semanticscholar.org/CorpusID:258887849.

Wang, R., Jansen, P. A., Côté, M.-A., and Ammanabrolu, P. Scienceworld: Is your agent smarter than a 5th grader? In Conference on Empirical Methods in Natural Language Processing, 2022. URL https://api.semanticscholar.org/CorpusID:247451124.

Yao, S., Rao, R., Hausknecht, M. J., and Narasimhan, K. Keep calm and explore: Language models for action generation in text-based games. ArXiv, abs/2010.02903, 2020. URL https://api.semanticscholar.org/CorpusID:222142129.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. ArXiv, abs/2210.03629, 2022. URL https://api.semanticscholar.org/CorpusID:252762395.

A. CLIN prompts

Figures 6 to 9 are the complete prompts for next-action generation (controller + executor), memory-generator during ADAPT, GEN-ENV, and GEN-TASK.

B. Example Memories

Example generated memory for ADAPT, GEN-ENV, and GEN-TASKsetups in Figures 10 to 12.

C. More results

Full results for CLIN outperforming Reflexion is in Table 5. For the ScienceWorld benchmark, we exclude electricity tasks since they deviate from standard electrical conventions, prohibiting us from fairly using LLM agents. We choose the first 10 test variants for each 18 tasks selected. The full list of 18 tasks from the benchmark, with the number of test variants used in parentheses:

grow-plant (10), identify-life-stages-1 (5), grow-fruit (10), measure-melting-point-known-substance (10), mendelian-genetics-unknown-plant (10), chemistry-mix-paint-secondary-color (9), freeze (9), lifespan-longest-lived (10), inclined-plane-determine-angle (10), boil (9), use-thermometer (10), chemistry-mix (8), lifespan-shortest-lived (10), find-plant (10), find-living-thing (10), identify-life-stages-2 (4), mendelian-genetics-known-plant (10), inclined-plane-friction-named-surfaces (10).

Short tasks have oracle lengths less than 37 steps (median), and Long tasks have oracle lengths more than equal to 37 steps.

The map to the short names used for tasks in the paper:

Temp: use-thermometer, measure-melting-point-known-substance; Pick&Place: find-plant, find-living-thing; Chemistry: chemistry-mix, chemistry-mix-paint-secondary-color; Lifespan: lifespan-longest-lived, lifespan-shortest-lived; Biology: identify-life-stages-1, identify-life-stages-2, Boil; Freeze; Grow Plant, Grow Fruit; Force: inclined-plane-determine-angle; Friction: inclined-plane-friction-named-surfaces; Genetics: mendelian-genetics-known-plant, mendelian-genetics-unknown-plant.

Superior BASE performance. Figure 4 depicts a superior BASE performance for CLIN than the final performance of both ReAct and Reflexion despite using the same underlying LLM (here, `gpt-4`). We find if we ablate for the controller module in CLIN, responsible for generating a goal before outputting the next action, CLIN's BASE performance drops in 44% cases. With an 18-point drop in average reward, the Abl-Contoller-BASE version of CLIN becomes equivalent to ReAct, the base agent for Reflexion, demonstrating the importance of the controller.

		Generative L. Agents		**CLIN (ours)**	
Task	Type	ReAct	Reflexion	BASE	ADAPT
Temp	S	7.2	5.9	25.2	14.3
Temp	S	6.1	28.6	53.2	51.8
Pick&Place	S	26.7	64.9	92.5	100.0
Pick&Place	S	53.3	16.4	55.0	100.0
Chemistry	S	51.0	70.4	44.5	44.4
Chemistry	S	58.9	70.7	56.7	56.7
Lifespan	S	60.0	100.0	85.0	100.0
Lifespan	S	67.5	84.4	70.0	90.0
Biology	S	8.0	8.0	10.0	8.0
Boil	L	3.5	4.2	7.0	15.2
Freeze	L	7.8	7.8	10.0	10.0
GrowPlant	L	9.1	7.3	10.2	11.1
GrowFruit	L	18.6	13.0	35.9	71.6
Biology	L	27.7	2.6	70.0	81.0
Force	L	40.5	50.6	53.5	100.0
Friction	L	44.0	100.0	56.5	72.5
Genetics	L	25.7	50.9	77.4	100.0
Genetics	L	16.8	23.7	62.3	92.6
	S	37.6	49.9	54.7	**62.8**
	L	21.5	28.9	42.5	**61.6**
	All	29.6	39.4	48.6	**62.2**

Table 5. Comparing CLIN with baselines for **adaptation** in ScienceWorld

Type	**#trials to success (↓)**	**%ep. improv.**
S	3.3	29.2
L	3.2	37.2
All	3.3	33.2

Table 6. CLIN's ADAPT improvements in ScienceWorld

Type	**GEN-TASK**		**G + A**	
	Δavg score	**%ep. improv.**	**Δavg score**	**%ep. improv.**
S	14.6	40.0	4.9	5.7
L	10.3	36.7	9.2	15.6
All	13.0	38.8	6.5	9.3

Table 7. CLIN's GEN-TASK improvements in ScienceWorld

```
[System]: You are an AI agent helping execute a science experiment in a simulated
environment with limited number of objects and actions available at each step.

[User]:
Possible objects ( value an OBJ can take ):
{objects_str}

Your next action should be in one of the following formats:
Possible actions:
{actions_str}

If I say \"Ambiguous request\", your action might mean multiple things. In that case,
respond with the number corresponding to the action you want to take.

What action would you like to do next?

First, scan the (unordered) list of learnings, if provided. Decide if any of the
learnings are applicable given the last observation to make progress in this task. Then
only use selected learnings, if any, to construct a rationale for picking the next
action. If no Learning is selected, construct the rationale based on the last
observation. Format your response as follows:

Write 'I used learning id(s):' as a comma separated list; the list can be empty if no
learnings selected. Then, write $$$ followed by the rationale. Finally, write ###
followed by the single next action you would like to take.

If you think you have completed the task, please write TASK_COMPLETE as the next action.

If the task requires you to 'focus' on something (OBJ), please write FOCUS ON <OBJ> as
the next action. FOCUS is a extremely critical action that can be only used the number of
times 'focus' is mentioned in the task description. Using it more than that or
inappropiately (such as on a wrong object) will terminate the session and the task will
be rendered as incomplete.

If you performed an action that requires waiting to see the effect, please write 'wait'
as the next action.
```

Figure 6. Prompt for the Controller and the Executor

```
[System]: You are an expert assistant.

[User]:
You are given CURRENT TRACE, a sequence of actions that an agent made in a world to
accomplish a task.

Task is detailed at the beginning.
For each action, there is a rationale why the agent made that action.
There is an observation that provide details about the new state of the world after each
action was executed.
The CURRENT TRACE is accompanied by an EVALUATION REPORT indicating the success of the
attempt to the task.

You can also be provided with PREVIOUS LEARNINGS which are learnings from the previous
attempts by the agent for the same task in the same environment/world. TASK indicates the
task description. EPISODE indicates the number of previous attempts of the task.

Generate a summary of learning, as a numbered list, that will help the agent to
successfully accomplish the SAME task AGAIN, in the SAME world.

Each numbered item in the summary can ONLY be of the form:
X MAY BE NECCESSARY to Y.
X SHOULD BE NECCESSARY to Y.
X MAY BE CONTRIBUTE to Y.
X DOES NOT CONTRIBUTE to Y.

{CURRENT TRACE}
Action: ...
Observation: ...
...
EVALUATION REPORT:
REWARD_FINAL: 100. This means: The agent has performed exceptionally well and
successfully solved the task.

Summary of learning as a numbered list:
```

Figure 7. Prompt for CLIN's memory generator during ADAPT

```
[System]: You are an expert assistant.

[User]: You are given a collection of learning lists, that are derived from actions made
by an agent and subsequent observations from a world to accomplish a TYPE of TASKs. All
of these TASKs belong to a same TYPE (such as 'boiling') but they are executed in
different ENVIRONMENT configurations. A different ENVIRONMENT configuration means there
are presence of a different set of objects (lighter instead of a stove) that are critical
for solving the TASK, presence of a different set of distractor objects that are not
useful for the TASK, a different floor plan, etc.

For each learning list, the TASK description is provided at the beginning as TASK:

Each learning list indicates a list of learnings from the agent's best attempt to solve
the TASK.

Each learning list is associated with an EVALUATION REPORT indicated how sucessful the
respective attempt was for solving the task.

Consider all learning lists and combine them in to a summary of learnings, as a numbered
list, that will help the agent to successfully accomplish a NEW TASK related to the
previous TASKs (such as 'boliing') in an ENVIRONMENT configuration that it has not seen
before. The NEW TASK description will be provided.

Each numbered item in the summary can ONLY be of the form:
X MAY BE NECCESSARY to Y.
X SHOULD BE NECCESSARY to Y.
X MAY NOT CONTRIBUTE to Y.
X DOES NOT CONTRIBUTE to Y.

{PREVIOUS LEARNINGS}
TASK: ...
LEARNINGS:...
EVALUATION REPORT:
REWARD_FINAL: 100. This means: The agent has performed exceptionally well and
successfully solved the task.
...

NEW TASK: ...
Summary of learning as a numbered list:
```

Figure 8. Prompt for CLIN's memory generator during GEN-ENV

```
[System]: You are an expert assistant.

[User]: You may be given a list of learnings, that are derived from actions made by an
agent and subsequent observations from a world to accomplish a TASK in an ENVIRONMENT
CONFIGURATION.

For the learning list, the TASK description is provided at the beginning as TASK:

The learnings are from the agent's best attempt to solve the TASK.

The learning list is associated with an EVALUATION REPORT indicated how sucessful the
attempt was for solving the task.

Now, generate a summary of learnings from the existing ones if provided, such that they
will be useful to the NEW TASK in the SAME ENVIRONMENT CONFIGURATION. The NEW TASK may
require different actions which are not captured in the given learnings but given
learnings can be used to infer about the ENVIRONMENT CONFIGURATION. The NEW TASK
description will be given. If PREVIOUS LEARNINGS says 'No learnings available', improvise
learnings for the NEW TASK.

Each numbered item in the summary can ONLY be of the form:
X MAY BE NECCESSARY to Y.
X SHOULD BE NECCESSARY to Y.
X MAY NOT CONTRIBUTE to Y.
X DOES NOT CONTRIBUTE to Y.

{PREVIOUS LEARNINGS}
TASK: ...
LEARNINGS:...
EVALUATION REPORT:
REWARD_FINAL: 100. This means: The agent has performed exceptionally well and
successfully solved the task.
...

NEW TASK: ...
Summary of learning as a numbered list:
```

Figure 9. Prompt for CLIN's memory generator during GEN-TASK

```
Task: Your task is to determine whether round seed shape is a dominant or recessive trait
in the pea plant. If the trait is dominant, focus on the blue box. If the trait is
recessive, focus on the orange box.

Memory (ADAPT), after Trial5:
1. Looking around MAY BE NECESSARY to gather information about the environment.
2. Moving to the greenhouse SHOULD BE NECESSARY to find the pea seeds.
3. Picking the jar SHOULD BE NECESSARY to access the pea seeds.
4. Moving the round green pea seed to flower pot 2 SHOULD BE NECESSARY to grow the pea
plants.
5. Moving the wrinkly green pea seed to flower pot 3 SHOULD BE NECESSARY to grow the pea
plants.
6. Waiting SHOULD BE NECESSARY for the pea plants to grow and show their traits.
7. Looking at the adult pea plant in flower pots 2 and 3 MAY BE NECESSARY to observe the
traits of the offspring.
8. Focusing on the blue box SHOULD BE NECESSARY to indicate the trait of the seed shape.
9. Continuously looking at the same environment DOES NOT CONTRIBUTE to the task of
determining the dominant or recessive trait in the pea plant.
10. Moving to the workshop DOES NOT CONTRIBUTE to determining the dominant or recessive
trait in the pea plant.
11. Picking up the battery or the wires in the workshop DOES NOT CONTRIBUTE to finding
the pea seeds.
12. Waiting for longer periods MAY BE NECESSARY to allow the pea plants to fully grow and
show their traits.
```

Figure 10. Memory generated after Trial5 in ADAPT for a Genetics task in ScienceWorld.

```
Task: Your task is to determine which of the two inclined planes (aluminum, platinum) has
the most friction. After completing your experiment, focus on the inclined plane with the
most friction.

Meta-memory (GEN-ENV):
1. Moving to the hallway SHOULD BE NECESSARY to reach the workshop.
2. Moving to the workshop SHOULD BE NECESSARY to find the block.
3. Picking up the block SHOULD BE NECESSARY to move it to the inclined planes.
4. Placing the block on the first inclined plane (either aluminum or platinum) SHOULD BE
NECESSARY to measure the friction.
5. Activating the stopwatch SHOULD BE NECESSARY to time the experiment.
6. Waiting for a certain period MAY CONTRIBUTE to observing the friction effect.
7. Deactivating the stopwatch SHOULD BE NECESSARY to stop timing the experiment.
8. Moving the block to the second inclined plane (either aluminum or platinum) SHOULD BE
NECESSARY to compare the friction.
9. Activating the stopwatch again SHOULD BE NECESSARY to time the second part of the
experiment.
10. Waiting for a certain period again MAY BE NECESSARY to observe the friction effect.
11. Deactivating the stopwatch again SHOULD BE NECESSARY to stop timing the experiment.
12. Focusing on the inclined plane with the most friction SHOULD BE NECESSARY to conclude
the experiment.
13. Repeating the experiment multiple times MAY BE NECESSARY for more accurate results.
14. Looking around in the initial room multiple times DOES NOT CONTRIBUTE to the task.
15. Moving the block back and forth between the two inclined planes DOES NOT CONTRIBUTE
to the task.
```

Figure 11. Meta-memory used in GEN-ENV for a Friction task in ScienceWorld.

```
Task: Your task is to freeze mercury. First, focus on the substance. Then, take actions
that will cause it to change its state of matter.

Meta-memory (GEN-TASK):
1. Looking around MAY BE NECESSARY to identify the available resources and the layout of
the environment.
2. Moving to different rooms SHOULD BE NECESSARY to find the tools and materials needed
to change the state of the substance.
3. Picking up items like glass cups or metal pots SHOULD BE NECESSARY to contain the
substance for changing its state.
4. Focusing on the substance SHOULD BE NECESSARY to understand its properties and how to
interact with it.
5. Picking up the thermometer SHOULD BE NECESSARY to monitor the temperature of the
substance.
6. Using the thermometer on the substance SHOULD BE NECESSARY to monitor the progress of
the task.
7. Puring the substance into the container SHOULD BE NECESSARY to prepare it for cooling.
8. Moving the container to a cooling device SHOULD BE NECESSARY to cool the substance.
9. Waiting for a period of time after cooling the substance SHOULD BE NECESSARY to allow
the substance to change state.
10. Repeatedly checking the temperature of the substance SHOULD BE NECESSARY to monitor
the progress of the task.
11. Activating the stove DOES NOT CONTRIBUTE to the task as it does not progress the
task.
12. Picking up unrelated items like a lighter DOES NOT CONTRIBUTE to the task as it does
not progress the task.
13. Moving to unrelated rooms like the workshop DOES NOT CONTRIBUTE to the task as it
does not progress the task.
14. Teleporting to the kitchen MAY BE NECESSARY for the task as it speeds up the process
of moving between rooms.
15. Using the thermometer multiple times on the substance after it reaches freezing point
DOES NOT CONTRIBUTE to the task as it does not progress the task.
```

Figure 12. Meta-memory used in GEN-TASK for a Freeze task in ScienceWorld.

```
Task: Your task is to make a red paper.

Ablated memory (GEN-TASK):
1. Avoid repeating actions: The agent performed the "look around" action twice at the
beginning of the task. This is unnecessary and wastes time. Once you have a clear
understanding of your surroundings, move on to the next step.
2. Be mindful of the order of actions: The agent moved to the workshop and read the
instructions after looking around twice. It's better to first read the instructions to
understand the task requirements.
3. Use the correct containers for mixing: The agent used a cup for mixing the
ingredients, which was successful. Continue using suitable containers like a cup or a
bowl for mixing.
4. Be clear with your actions: The agent made an ambiguous request when trying to mix the
cup containing red paint. Be specific with your actions to avoid confusion.
5. Avoid unnecessary movements: The agent teleported to the art studio after picking up
the paper. Plan your actions carefully to avoid unnecessary movements.
6. Use the correct method to combine ingredients: The agent successfully mixed the
ingredients in the cup. Continue using suitable methods to combine the ingredients, such
as using a blender or a food processor.
7. Keep track of your actions: The agent moved the paper to the cup and then back to the
inventory. This is unnecessary and wastes time. Keep track of your actions to avoid
repeating them.
8. Always refer back to the instructions: The agent seemed to forget the instructions to
make the red paper. Always refer back to the instructions to ensure you are following the
correct steps.
```

Figure 13. Meta-memory generated for ablation in GEN-ENV for a Chemistry task in ScienceWorld.