# NESYFS: A NEURO-SYMBOLIC FAST-SLOW THINKING FRAMEWORK FOR LLM AGENT UNDER PARTIAL OBSERVABILITY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recently Large Language Models (LLMs) have been increasingly deployed as autonomous agents in applications such as self-reflection, retrieval-augmented generation, and scientific discovery. In these settings, agents must act based on limited observations rather than full environmental states, leading to partial observability. This introduces several key challenges: belief state inference, task objective misalignment, and planning under uncertainty. Prior approaches typically condition actions on full or summarized action-observation histories whose redundant and irrelevant information can mislead the decision making of LLM agent. Inspired by human cognition, we propose a novel **Ne**uro-**Sy**mbolic **F**ast-**S**low thinking (NeSyFS) framework for LLM agent, addressing the challenges introduced by partial observability in a unified approach. We use a knowledge graph (KG) to represent the belief state, providing triplets as context for every module of NeSyFS. The fast-thinking module performs reactive action, while slow-thinking conducts a new uncertainty-aware planning by following the high-level structure of twisted sequential Monte Carlo (TSMC) algorithm. To mitigate the misalignment of task objective, a reflection module is used to reflect fast-thinking actions, and also switches to the slow-thinking module whenever reactive actions repeatedly fail. Experiments on three representative benchmarks, i.e. ALFWorld, Webshop, and ScienceWorld, demonstrate significant advantages over previous methods.

## 1 INTRODUCTION

Large Language Models (LLMs) have achieved notable success in agentic applications by extending beyond static text generation to support autonomous decision-making, enabling capabilities such as tool use (Schick et al., 2023), retrieval-augmented reasoning (Gao et al., 2023), multi-step planning (Rawat et al., 2025), and self-reflection (Renze & Guven, 2024) in complex environments. These advances have allowed LLM agents to perform tasks in domains ranging from embodied interaction to software automation (Jin et al., 2024) and scientific workflows (Yamada et al., 2025). However, a fundamental challenge arises from partial observability: unlike supervised settings with complete input context, agentic environments typically reveal only local and immediate observations, while the true underlying state remains latent (Shridhar et al., 2020; Yao et al., 2022a). As a result, LLM agents must infer a belief over the hidden state from incomplete and potentially noisy interaction histories (Xi et al., 2024; Ma et al., 2024), which are often long and difficult to compress effectively within limited context windows. In addition, since the agent must act without access to the true underlying state, increased amount of uncertainty is introduced, and planning under uncertainty becomes a challenge for the LLM agent. Moreover, the uncertainty and missing state information from partial observability can make the LLM agent produce actions misaligned with task objective, degrading the performance of task completion.

In this work, we propose a novel neuro-symbolic fast–slow thinking framework (NeSyFS) for LLM agents that addresses the challenges of partial observability in a unified manner, enabling latent state inference, mitigating task-objective misalignment, and supporting planning under uncertainty. Unlike prior full-history-based or summarization-based approaches that rely on unstructured memory, we employ a knowledge graph (KG) to represent the latent state of the environment, thereby encoding interaction histories into a structured and symbolic memory. Specifically, each incoming observation



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.

is transformed into triplets that capture relationships and attributes of entities and events. Newly acquired facts are incorporated into the KG, while outdated triplets in KG are pruned, ensuring a consistent and dynamically updated representation of the latent state. The KG-retrieved triplets are used as context in every module of NeSyFS.

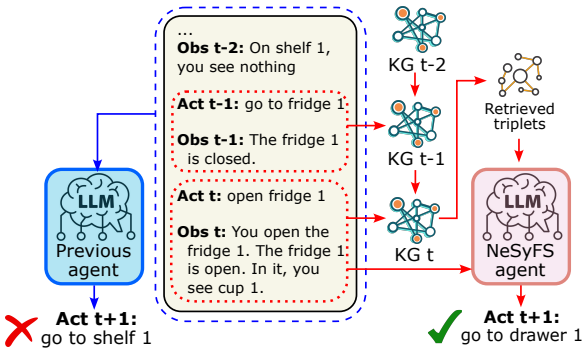



Figure 1: Comparison of previous and NeSyFS agent. Previous agent uses full or summarized interaction history as context for decision making, while our agent updates a KG-based memory dynamically and uses KG-retrieved triplets as context. The KG $\mathcal{G}_{t-1}$ is updated with action $a_t$ and observation $o_t$, i.e. $\mathcal{G}_t = \text{Update}(\mathcal{G}_{t-1}, o_t, a_t)$. The triplets related with solving task $\eta$ are retrieved from KG, i.e. $\tau_t = \mathcal{R}(\mathcal{G}_t, \eta)$, which can provide structured context to help LLM make correct decision.

Furthermore, inspired by human cognition (Kahneman, 2011), the proposed framework decomposes decision-making into fast and slow thinking processes. The fast thinking module generates reactive actions by retrieving relevant triplets from the KG to construct a symbolic approximation of the latent state. In contrast, the slow thinking module performs deliberative planning under uncertainty. Slow thinking is used only when fast-thinking decision making has too much uncertainty or is unreliable. To this end, we propose a new uncertainty-aware planning algorithm which follows the high-level structure of twisted sequential Monte Carlo (TSMC) (Del Moral et al., 2006). Specifically, it maintains a set of particles and uses resampling to guide them toward the target distribution modeling task-completing trajectories. LLM is prompted to predict next observation and existence of task progress in the state transition. The sampling mechanism of this method can make the particles robust to mistakes in LLM-based evaluations of task progress, and reach a good balance between exploration and exploitation. As far as we know, we are the first to develop a TSMC-style planning method for LLM agent.

Although fast thinking emphasizes locally plausible actions, it does not explicitly guarantee alignment with global task objectives. To address this limitation, we incorporate a reflection module that evaluates the consistency of fast-thinking actions with the task objective, conditioned on relevant triplets retrieved from the KG. To control computational overhead, slow thinking is invoked only when fast-thinking actions repeatedly fail in this reflective evaluation. With KG-provided triplets as context, stepwise reflection is enabled and is more efficient and robust than conventional trajectory-wise reflection.

Our contributions are summarized as follows.

- We propose a neuro-symbolic fast-slow thinking framework that addresses challenges introduced by partial observability in a unifying manner.
- Based on the context retrieved from the KG, a neuro-symbolic TSMC-style planning algorithm is introduced within the slow-thinking module to address the uncertainty under partial observability.
- We design a KG-augmented reflection module that aligns fast-thinking actions with task objectives. Using KG-retrieved triplets, the module enables stepwise reflection that is more efficient and robust than conventional trajectory-level reflection.
- Experiments on three benchmarks demonstrate that using KG-retrieved triplets as context improves the LLM performance on decision-making, reflection, and world modeling.

## 2 Preliminary Background

### 2.1 Partial Observability

We consider agentic tasks in which an autonomous LLM responds to a user query through iterative interaction with an external environment $\mathcal{E}$. Each episode begins with a user query and unfolds over a finite horizon $H$.

Such agentic tasks can be naturally formulated as a Partially Observable Markov Decision Process (POMDP) (He et al., 2024; Zhang et al., 2025), represented by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, O, R)$. Here, $\mathcal{S}$ denotes the latent state space of the environment, $\mathcal{A}$ the action space, and $\mathcal{O}$ the observation space consisting of textual environmental response. The environment dynamics are governed by the transition function $\mathcal{T}(s' \mid s, a)$ and the observation model $O(o \mid s, a)$. The reward function $R(s)$ is defined only on terminal states and indicates task success.

At time step $t$, the agent executes an action $a_t$, after which the environment transitions to a latent state $s_t \sim \mathcal{T}(\cdot \mid s_{t-1}, a_t)$ and returns an observation $o_t \sim O(\cdot \mid s_t, a_t)$. During inference, neither the latent state $s_t$ nor the reward function is directly observable to the agent. This results in partial observability, which introduces several fundamental challenges, including latent state inference, task objective misalignment, and planning under uncertainty. In particular, inferring $s_t$ from the interaction history is difficult because the history often contains substantial redundant and noisy information. Moreover, not knowing $s_t$ creates a lot of uncertainty and further complicates the prediction of future situations in planning. In this work, we propose a novel framework to unifiedly resolve these challenges.

### 2.2 Knowledge Graph

Knowledge graphs (KGs) (Ehrlinger & Wöß, 2016) represent information as graph-structured data, where nodes correspond to entities and edges encode relationships among them. KGs can be constructed from diverse sources, including structured databases, unstructured text, and other heterogeneous data modalities. Similar to traditional databases, graph databases support structured querying mechanisms, such as the Cypher query language (Francis et al., 2018).

The relationship between a pair of nodes in a KG is commonly referred to as a triplet of subject, predicate, and object. In this work, we consider the following triplet forms:

- (entity 1, relationship, entity 2)
- (entity 1, attribute, boolean/value)
- (entity 1, verb in the past tense, entity 2)

which describe relationships, attributes and historic events of entities. A KG is denoted as $\mathcal{G} = (V, E)$ where $V$ is a set of semantic vertices corresponding to entities in the environment, and $E$ is a set of semantic edges representing triplets in the above forms. Our framework has a retrieval-augmented generation (RAG) component which retrieves the relevant triplets from a KG and feeds them to LLM as context approximating the underlying state of the POMDP environment.

### 2.3 Twisted Sequential Monte Carlo

Twisted Sequential Monte Carlo (TSMC) (Doucet et al., 2001; Del Moral et al., 2006; Chopin et al., 2020; Zhao et al., 2024b; Feng et al., 2025b) is a probabilistic inference framework that incrementally guides particles toward high-probability regions of a target distribution $\sigma$ through intermediate twisted distributions $\psi_t$. At each step, particles are propagated using a proposal distribution and resampled according to importance weights, allowing computational effort to focus on promising trajectories while reducing variance. In this work, to address uncertainty under partial observability, the slow-thinking module follows a TSMC-like structure, where LLMs are used to propose future states and observations and to score particles.

Sequential Monte Carlo (SMC) and its variants have been applied into planning and reinforcement learning by many previous work (Piché et al., 2018; Macfarlane et al., 2024; Abdulsamad et al., 2026; Zhao et al., 2024b). Compared with other planning methods, such as Monte Carlo Tree Search (MCTS) (Chen et al., 2024), SMC-style methods can better address the uncertainty of states and reduce computational complexity, since particles representing less-likely states will be discarded and not branched in large probability. However, in tree-search-based methods, every state will be branched and its child nodes will be evaluated. In this work, we develop a TSMC-style LLM-based planning method for slow-thinking module, where the state is approximated by related triplets retrieved from memory KG. As far as we know, this work is the first of doing so.

# 3 METHODOLOGY

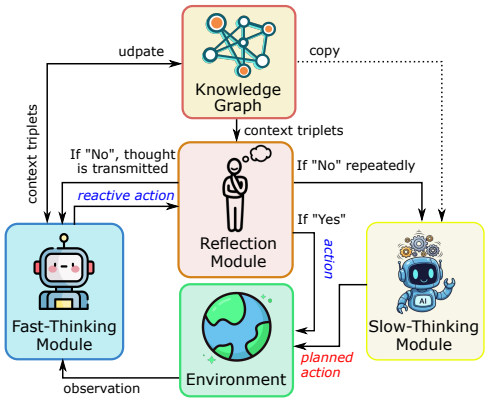



Figure 2: Diagram of NeSyFS framework. Knowledge graph (KG), representing the latent state of environment, is updated by fast-thinking module with new observation and retrieved by all the modules. Every reactive action is reflected. If "No", fast-thinking is called again with thought from reflection module. If "No" repeatedly for $K$ number of times, the slow-thinking module is initiated to search optimal action via planning, where KG is copied to initialize the simulation of future latent states and observations.

In this section, we introduce the NeSyFS framework, which maintains a memory KG as a proxy for the latent state of the environment. The diagram of the proposed framework is shown in Figure 2. Using triplets retrieved from the memory KG as a context, the fast-thinking module generates reactive actions, while the slow-thinking module performs uncertainty-aware deliberative planning. A reflection module evaluates the fast-thinking actions with respect to the task objective, where the KG-retrieved triplets facilitate reflection in the stepwise granularity. We will first introduce the fast-thinking module, including the memory KG. Then, reflection module will be presented with details. Finally, the slow-thinking module, including the proposed TSMC-style planning algorithm, is introduced.

## 3.1 FAST THINKING MODULE

We use a KG $\mathcal{G} = (V, E)$ as working memory to represent the latent state of the environment (POMDP $\mathcal{M}$). Instead of using the entire interaction history or its summary as unstructured context, this memory organizes environmental knowledge into structured triplets which describe attributes and relations of entities and past events, in the forms in Section 2.2.

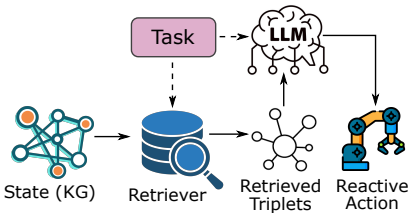



Figure 3: Diagram of decision making in fast-thinking module.

At each time step $t$, given the current observation $o_t$ and task description $\eta$, the fast-thinking module first retrieves task-relevant triplets $\tau_t$ from memory KG $\mathcal{G}_t$ by calling the retrieval method $\mathcal{R}(\mathcal{G}_t, \eta)$. The details of $\mathcal{R}$ are introduced in Appendix D.1.2. Then, $\tau_t$ are then provided as contextual input to the LLM which is prompted to produce the next action for accomplishing task $\eta$ with observation $o_t$. During this process, chain-of-thought (CoT) reasoning (Wei et al., 2022) is employed, making the fast-thinking action selection process analogous to the ReAct method (Yao et al., 2022b). This can be formally expressed as $a_t = LLM(P^{\text{fast}}; \mathcal{R}(\mathcal{G}_t), o_t, \eta)$, where $P^{\text{fast}}$ is the reactive decision making prompt template, $\mathcal{G}_t$ is the KG representing current latent state. Diagram of decision making in the fast-thinking module is shown in Figure 3.

Whenever $a_t$ is applied into the environment, the agent receives a new observation $o_t$. The KG $\mathcal{G}_t$ will be updated with $o_t$ and $a_t$ to produce $\mathcal{G}_{t+1}$ to represent next state. This process is denoted as $\mathcal{G}_{t+1} = \text{Update}(\mathcal{G}_t, o_t, a_t)$, where new triplets from $o_t$ and $a_t$ will be added to $\mathcal{G}_t$, and outdated triplets in $\mathcal{G}_t$ will be removed. The details of update process are presented in Appendix D.1.1.

## 3.2 REFLECTION MODULE

Due to incomplete information about the underlying environment state, partial observability can introduce misalignment between the agent's actions and the task objective, which has emerged as a major challenge for LLM agents (Fang et al., 2025; Kim et al., 2025; Chung et al., 2025). To mitigate this issue, we introduce a prompt-based reflection module that detects misalignment between the actions produced by the fast-thinking component and the task objective. Unlike self-reflection methods (Shinn et al., 2023), in which the agent introspects and explains its own behavior, the proposed reflection module adopts a third-person perspective to verify whether a candidate action aligns with the task objective. Our innovation here is to use the KG-retrieved triplets as

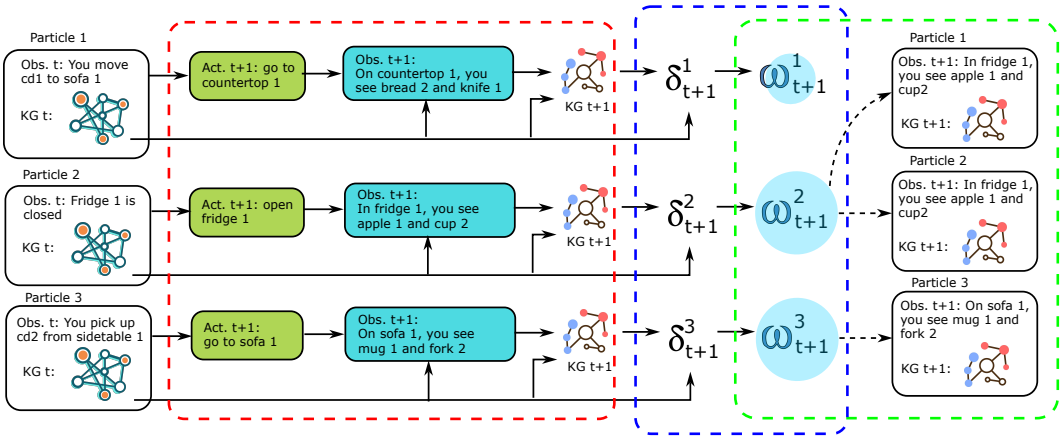



Figure 4: Diagram of the proposed TSMC-style uncertainty-aware planning algorithm, where three particles are updated in the planning step $t$, consisting of propagation, weight update and resampling processes. The red box shows the propagation process, where the prediction of KG at time $t+1$ is essentially updating KG at $t$ with $a_{t+1}$ and $o_{t+1}$. The blue box is the weight update process, where the blue circle represents the value of an updated weight. The green box represents the process of resampling. Every particle stores all the historic observations, actions, states (KG), and weights.

context to facilitate reflection in the stepwise granularity, since context triplets provide sufficient state information for reflection.

The reflection can be expressed as $Y^{\text{ref}} = LLM(P^{\text{ref}}; a_t, o_t, \mathcal{R}(\mathcal{G}_t), \eta)$, where $P^{\text{ref}}$ is the prompt for refection, $a_t$ is the fast-thinking action, $\mathcal{R}$ is the operation of KG retrieval introduced in previous section, $\eta$ is the task description, and output $Y^{\text{ref}} \in \{\text{Yes}, \text{No}\}$ which indicates whether $a_t$ aligns with $\eta$ or not. To mitigate the randomness of LLM output, we propose to adopt a self-consistency method (Wang et al., 2022b) which draws $n$ samples of $Y^{\text{ref}}$ from LLM and sets the final output $\bar{Y}^{\text{ref}}$ as "Yes" only when $80\%$ of $Y^{\text{ref}}$ samples are "Yes". We use $n = 5$ in experiments.

As shown in Figure 2, if the final reflection answer $\bar{Y}^{\text{ref}}$ is "Yes", then $a_t$ is applied to the environment. Otherwise, the thought of reflection module which produces "No" is sent back to the fast-thinking module and a new reactive action is produced by calling the fast-thinking module again. If $\bar{Y}^{\text{ref}}$ is repeatedly not "Yes" for $K$ number of times, say $K = 3$, the slow thinking module is activated.

### 3.3 Slow Thinking Module

As shown in Figure 2, when the fast-thinking is unreliable, the agent transitions to the slow-thinking module, which performs deliberative planning by explicitly searching over candidate action sequences in a step-by-step manner. To handle the uncertainty arising from partial observability, we propose a novel neuro-symbolic uncertainty-aware planning algorithm which follows the high-level framework of twisted sequential Monte Carlo (TSMC) algorithm introduced in Section 2.3.

The proposed planning algorithm is illustrated in Figure 4. Specifically, the algorithm maintains a set of weighted particles that represent possible future observations, latent states (i.e., KGs), and actions over a planning horizon $H$. To identify an optimal plan, these particles are guided toward high-probability regions of a target distribution $\sigma$, which corresponds to trajectories that successfully accomplish the task. At each planning step, every particle is propagated by predicting the next action, observation, and latent state (KG). The particle weights are then updated according to the likelihood of the newly predicted state achieving task progress. Finally, a resampling procedure based on the updated weights preserves particles that make meaningful task progress while filtering out less promising particles with a large probability.

To address the difficulty LLMs face in evaluating task progress (Setlur et al., 2025; Zheng et al., 2025), particle weights are updated using binary LLM feedback indicating whether a given state transition makes task progress or not. Moreover, the concise state representation provided by KG can improve the correctness of LLM judgment. In addition, the resampling mechanism can make our

approach more robust to LLM's mistakes, as it still allows promising particles to survive even when they are temporarily misjudged by LLM.

Compared with previous tree-search planning methods, such as MCTS (Chen et al., 2024), the proposed TSMC-style method is better at addressing state uncertainty and more computationally efficient. Previous methods are difficult at predicting next state or observation under partial observability, while the proposed method leverages memory KG to accurately predict next KG by updating current KG with new action and observation. In addition, in tree-search methods, every leaf node is expanded by predicting multiple next states as child nodes. However, in the proposed method, particles with poor task progress will be discarded and not be expanded with a large probability, hence reducing the computation complexity.

**Definition.** Denote $N$ as number of particles. Denote particle $i$ at the planning step $\tau$ as $\boldsymbol{p}_\tau^i = (\hat{\boldsymbol{o}}_{1:\tau}^i, \hat{\boldsymbol{\mathcal{G}}}_{1:\tau}^i, \hat{\boldsymbol{a}}_{1:\tau}^i, \omega_\tau^i)$, where $\boldsymbol{o}_{1:\tau}^i = (\hat{o}_1^i, \ldots, \hat{o}_\tau^i)$ is the sequence of observations predicted by LLM until time step $\tau$, and same for $\hat{\boldsymbol{\mathcal{G}}}_{1:\tau}^i$ and $\hat{\boldsymbol{a}}_{1:\tau}^i$.

**Initialization.** Denote $t_0$ as the time step when slow thinking is activated. For each particle, set the first observation, latent state and action as copies of the actual observation, state (KG), and action at time step $t_0$, i.e. $\hat{o}_1^i = o_{t_0}, \hat{\mathcal{G}}_1^i = \mathcal{G}_{t_0}, \hat{a}_1^i = a_{t_0}$ for $i = 1, \ldots, N$, so that every particle starts from the same origin. Set the weight of each particle to be 1, i.e. $\omega_1^i = 1$.

**Propagation.** As shown in the red box of Figure 4, in each planning step $\tau$, every particle $\boldsymbol{p}_\tau^i$ is propagated forward to generate a new particle $\boldsymbol{p}_{\tau+1}^i$ by predicting the next action $\hat{a}_{\tau+1}^i$, observation $\hat{o}_{\tau+1}^i$, and latent state $\hat{\mathcal{G}}_{\tau+1}^i$ one-by-one.

First, the next action is produced by an LLM-based policy, described as $\hat{a}_{\tau+1}^i = LLM(P^{\text{fast}}; \mathcal{R}(\hat{\mathcal{G}}_\tau^i), \hat{o}_\tau^i, \eta)$, where $P^{\text{fast}}$ is the prompt template used in the fast-thinking module, and $\eta$ is the task description.

Subsequently, for each particle $i$, we need to predict the next observation $\hat{o}_{\tau+1}^i$ caused by action $\hat{a}_{\tau+1}^i$. Specifically, task-relevant context triplets are retrieved from the current KG $\hat{\mathcal{G}}_\tau^i$ and used as state information, which is the same process as the retrieval $\mathcal{R}(\hat{\mathcal{G}}_\tau^i, \eta)$ described in the fast-thinking section. Then with these triplets and $\hat{a}_{\tau+1}^i$ as input, LLM is prompted to predict the next observation $\hat{o}_{\tau+1}^i$. This can be formally written as $\hat{o}_{\tau+1} = LLM(P^{\text{obs}}; \mathcal{R}(\hat{\mathcal{G}}_\tau^i, \eta), \hat{a}_{\tau+1}^i)$. In this prediction, the LLM is asked to infer plausible future observations using its commonsense knowledge and reasoning capabilities.

Then, $\hat{o}_{\tau+1}^i$ is parsed into triplets which are incorporated into $\hat{\mathcal{G}}_\tau^i$ to produce the next latent state (KG) $\hat{\mathcal{G}}_{\tau+1}^i$, following the process of updating KG introduced in the fast-thinking section, i.e. $\hat{\mathcal{G}}_{\tau+1}^i = \text{Update}(\hat{\mathcal{G}}_\tau^i, \hat{o}_{\tau+1}^i, \hat{a}_{\tau+1})$. The newly acquired triplets in updating process are denoted as $\tau_{\text{new}}$.

**Weight Update.** As shown in the blue box of Figure 4, each particle $i$ is assigned a new weight $\omega_{\tau+1}^i$ which is updated as $\omega_{\tau+1}^i \leftarrow \exp(\lambda \delta_{\tau+1}^i)$. Specifically, $\delta_{\tau+1}^i$ is the task progress detector and is the LLM evaluation on whether the transition $(\hat{\mathcal{G}}_\tau^i, \hat{a}_\tau^i, \hat{\mathcal{G}}_{\tau+1}^i)$ makes progress toward task completion or not. Formally, we can write $\delta_{\tau+1}^i = LLM(P^{\text{prog}}; \hat{a}_\tau^i, \mathcal{R}(\hat{\mathcal{G}}_\tau^i), \tau_{\text{new}}, \eta)$, where $P^{\text{prog}}$ is the prompt template in the task progress evaluation, $\delta_{\tau+1}^i \in \{0, 1\}$, and $\tau_{\text{new}}$ is obtained in the propagation process. In contrast to previous LLM-based evaluation methods (Zheng et al., 2025), in our approach, the binary response of LLM and concise representation provided by context triplets can reduce hallucination and make LLM focus on task-related information in the input.

**Why likelihood ratio of $p_0$ and $q$ in the weight update can be dropped?** In previous applications of TSMC (Del Moral et al., 2006; Zhao et al., 2024b), at time $t$, the weight of particle $k$ is updated as

$$\omega_\tau^k \leftarrow \frac{p_0(s_\tau^k | s_{1:\tau-1}^k)}{q(s_\tau^k | s_{1:\tau-1}^k)} \frac{\psi_t(s_{1:\tau}^k)}{\psi_{\tau-1}(s_{1:\tau-1}^k)}, \text{when } \tau < T; \quad \omega_\tau^k \leftarrow \frac{p_0(s_\tau^k | s_{1:\tau-1}^k)}{q(s_\tau^k | s_{1:\tau-1}^k)} \frac{\phi(s_{1:\tau}^k)}{\psi_{\tau-1}(s_{1:\tau-1}^k)}, \text{when } \tau = T \quad (1)$$

However, in our work, we use the pre-trained LLM, such as GPT-5, as the proposal model, so that the base model is directly used as the proposal model, which is the case of "base model as proposal" (Zhao et al., 2024b). According to (Zhao et al., 2024b), we have $p_0(s_\tau^k | s_{1:\tau-1}^k) = q(s_\tau^k | s_{1:\tau-1}^k)$.

Therefore, the likelihood ratio in equation 1 can be dropped, and the new update rule is written as

$$\omega_\tau^k \leftarrow \frac{\psi_\tau(s_{1:\tau}^k)}{\psi_{\tau-1}(s_{1:\tau-1}^k)}, \text{when } \tau < T; \;\; \omega_\tau^k \leftarrow \frac{\phi(s_{1:\tau}^k)}{\psi_{\tau-1}(s_{1:\tau-1}^k)}, \text{when } \tau = T \tag{2}$$

Since the ratio $\psi_\tau / \psi_{\tau-1}$ or $\phi / \psi_{\tau-1}$ is essentially $\exp(\lambda \delta_{\tau+1}^i)$, our weight update rule is justified. The detailed theoretical justification for the weight update is in Appendix D.2.

**Resampling.** The resampling step is to probabilistically keep particles with good task progress and discard particles without poor progress. At planning step $\tau$, the likelihood of a particle $i$ being kept is proportional to its weight $\omega_\tau^i$.

Repeating the operations of propagation, weight update and resampling over the planning horizon $H$, the remaining particles are close to successful trajectories with a large probability. This can be guaranteed by the theoretical properties of TSMC (Del Moral et al., 2006). After planning step $H$, uniformly sample a particle $j \sim \text{Unif}([1, \ldots, N])$, and apply the first action in the particle $j$ (i.e. $\hat{a}_1^j$) to the actual environment, as shown in Figure 2.

# 4 Experiments

## 4.1 Experimental Settings

**Benchmarks.** We evaluate NeSyFS on three text-based benchmarks: ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022a), and ScienceWorld (Wang et al., 2022a). ALFWorld evaluates household tasks, WebShop (Yao et al., 2022a) tests multi-step decision-making in online shopping, and ScienceWorld (Wang et al., 2022a) assesses procedural and scientific reasoning. ALFWorld uses binary task success, while WebShop and ScienceWorld provide dense rewards, allowing evaluation by success rate and average reward across tasks. Additional details are provided in Appendix E.1.

Table 1: Performance comparison of NeSyFS with ReAct, Reflexion, RAFA and SwiftSage across ALFWorld, WebShop and ScienceWorld. SR and AR denote success ratio and average reward, respectively.

| Model | Method | ALFWorld | WebShop | | ScienceWorld | | Average |
|---|---|---|---|---|---|---|---|
| | | SR | AR | SR | AR | SR | |
| GPT-5-mini | ReAct | 71.2 | 42.5 | 32.7 | 65.1 | 28.3 | 44.1 |
| | Reflexion | 75.6 | 49.5 | 35.7 | 69.7 | 31.5 | 47.6 |
| | ABBEL | 72.1 | 43.2 | 33.3 | 65.9 | 29.5 | 44.9 |
| | RAFA | 73.5 | 40.1 | 27.3 | 70.2 | 34.5 | 45.1 |
| | SwiftSage | 76.7 | 52.3 | 38.2 | 72.1 | 35.7 | 51.0 |
| | AriGraph | 78.1 | 53.3 | 41.3 | 73.9 | 39.1 | 52.8 |
| | **NeSyFS** | **91.1** | **61.2** | **51.3** | **82.2** | **61.2** | **63.5** |
| GPT-5 | ReAct | 78.2 | 49.6 | 38.7 | 72.2 | 35.5 | 50.8 |
| | Reflexion | 80.3 | 52.3 | 40.2 | 75.7 | 38.9 | 53.1 |
| | ABBEL | 77.6 | 51.2 | 39.8 | 73.9 | 36.0 | 51.1 |
| | RAFA | 83.2 | 48.1 | 35.3 | 76.3 | 42.7 | 53.7 |
| | SwiftSage | 81.9 | 56.3 | 41.2 | 75.3 | 40.2 | 54.4 |
| | AriGraph | 82.1 | 59.5 | 47.1 | 73.9 | 45.6 | 58.3 |
| | **NeSyFS** | **93.6** | **69.4** | **63.3** | **86.1** | **69.4** | **75.3** |
| Llama-3.3-70B | ReAct | 79.6 | 45.6 | 37.1 | 69.2 | 33.1 | 49.9 |
| | Reflexion | 81.2 | 47.3 | 38.9 | 73.1 | 36.9 | 52.3 |
| | ABBEL | 78.6 | 45.1 | 36.9 | 70.1 | 32.5 | 49.3 |
| | RAFA | 82.3 | 47.2 | 37.1 | 75.1 | 39.2 | 52.9 |
| | SwiftSage | 80.6 | 51.2 | 39.9 | 76.1 | 41.3 | 53.9 |
| | **NeSyFS** | **89.6** | **68.1** | **62.5** | **85.2** | **67.1** | **73.0** |

In the following, we first conduct experiments to compare the overall framework against some previous representative methods. Then, the effect of reflection module is investigated empirically, independently of the slow-thinking module. The ablation study of the slow-thinking module is conducted in Appendix E.3. Finally, the computation and token cost are summarized in Appendix E.4 and E.5. Every result reported here is an average of three random seeds.

### 4.2 Overall Evaluation

**Baseline.** Some representative decision making methods of LLM agent are selected as baselines. Details of each baseline are introduced in Appendix E.2.

- ReAct (Yao et al., 2022b): This method reasons about next action by using CoT Wei et al. (2022), laying the foundation of LLM decision making.
- Reflexion (Shinn et al., 2023): This method reflects every failed trajectory and helps LLMs improve after each failed attempt.
- ABBEL (Lidayan et al., 2025): It uses LLM to summarize observation and full interaction history to infer the belief state.
- RAFA (Liu et al., 2023): At each time step, the agent first conducts a short-horizon planning through tree search, executes the first action of the plan, and then replans at next step.
- AriGraph (Anokhin et al., 2025): A KG-augmented LLM agentic framework which is similar as the fast-thinking module alone of NeSyFS.
- SwiftSage (Lin et al., 2023): In this fast-slow thinking framework, the Swift module represents fast and intuitive thinking, and the Sage module plans and grounds subgoals to emulate deliberate thought processes.

**Results.** Table 1 presents the performance comparison between NeSyFS and several representative baselines, including ReAct, Reflexion, ABBEL, RAFA, and SwiftSage, across ALFWorld, WebShop, and ScienceWorld. The results show that NeSyFS consistently and significantly outperforms all baseline methods. In particular, NeSyFS achieves nearly a 100% improvement over ReAct and Reflexion, which serve as the foundation of many state-of-the-art LLM agents, highlighting the effectiveness of the proposed KG-based memory, reflection, and slow-thinking modules.

Furthermore, the substantial improvement over summarization-based method, such as ABBEL, suggests that KG-based contextual representations provide more effective state representations for decision making than summarization-based belief states, since the summarization of history may lose critical information for belief updates. RAFA performs short-horizon planning at each time step through tree search, a mechanism conceptually related to our slow-thinking module. The superior performance of NeSyFS over RAFA therefore underscores the importance of integrating KG-based memory and reflection mechanisms in addition to planning. SwiftSage represents a typical fast–slow thinking framework, and the advantage of NeSyFS further demonstrates the benefits of KG-based memory in enhancing the synergy between fast and slow reasoning processes. The advantage over AriGraph shows the effectiveness of the proposed reflection and slow-thinking modules.

### 4.3 KG-provided Context and Reflection

In this section, we first evaluate the correctness of the KG-augmented reflection module, and then demonstrate the advantage and effect of KG-retrieved context triplets in the overall task-completion evaluations.

The reflection module of NeSyFS leverages KG-retrieved triplets as contextual information. In the first part of experiment, to evaluate the effect of KG-provided context in reflection, we compare the correctness of our reflection module against baselines that use the entire interaction history and LLM-summarized belief state as context, which are short as "history" and "belief", respectively. The proposed reflection method is short as "KG". All the methods are evaluated on a set of randomly-collected trajectories in which the correctness of each action is manually annotated.

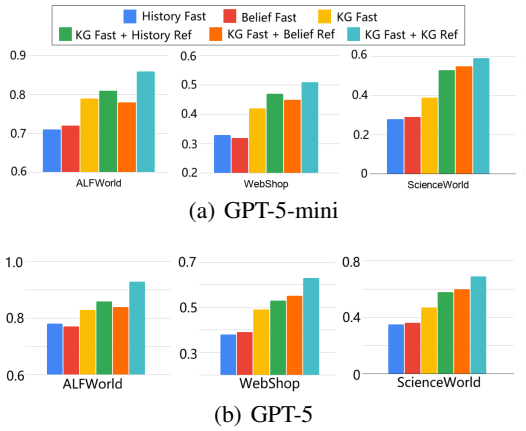
We adopt two evaluation metrics. The first is Total Detection Errors (TDE), defined as the sum of false positives (misaligned actions that are not detected) and false negatives (correct actions incorrectly classified as misaligned). TDE measures the overall number of reflection errors. The second metric is Effective Reliability (ER), defined as $\frac{TP-FP}{TP+FP}$, where $TP$ and $FP$ denote the numbers of true positives and false positives, respectively. Hence, ER evaluates the reliability of actions that are approved by the reflection module.

The performance comparison is shown in Table 2. We can see that the reflection method using KG-provided context significantly outperforms that of using interaction history or LLM-summarized belief

Table 2: Comparison of reflection with different representations of context.

| Model | Method | ALFWorld | | WebShop | | ScienceWorld | |
|---|---|---|---|---|---|---|---|
| | | TDE | ER | TDE | ER | TDE | ER |
| GPT-5 | History | 57 | 0.55 | 87 | 0.51 | 95 | 0.53 |
| | Belief | 52 | 0.59 | 89 | 0.50 | 89 | 0.55 |
| | **KG** | **42** | **0.71** | **61** | **0.66** | **77** | **0.65** |
| GPT-5-mini | History | 82 | 0.45 | 113 | 0.43 | 135 | 0.42 |
| | Belief | 77 | 0.41 | 110 | 0.42 | 125 | 0.45 |
| | **KG** | **58** | **0.65** | **72** | **0.58** | **91** | **0.59** |

state as context. This is because KG-retrieved triplets can provide concise state representation, making the LLM focus on decision-making-related information in the prompt when reflecting. Notably, the ER of "KG" method is significantly higher than baselines, meaning that KG-provided context could prevent more wrong actions from being applied into the environment. "History" performs worse since the noisy and redundant information in the interaction history can distract LLM. The problem of "Belief" is that the LLM-based summarization can lose important information.




(a) GPT-5-mini

(b) GPT-5

Figure 5: Performance comparison of different reflection methods.

In the second part of the experiment, we empirically investigate the effect of KG-provided context within the fast-thinking and reflection modules by comparing different methods of forming the context to approximate the underlying environment state. Both GPT-5 and GPT-5-mini are used as the underlying LLMs in these evaluations. The evaluation metric, corresponding to y-axis in the plots, is the task success rate.

The methods “History Fast”, “Belief Fast”, and “KG Fast” use the full interaction history, an LLM-summarized belief state, and KG-retrieved triplets, respectively, as context for action selection in the fast-thinking module, without employing any reflection module or slow-thinking module. In contrast, “History Ref”, “Belief Ref”, and “KG Ref” incorporate the reflection module while using the same three forms of context representations. Specifically, in the fast-thinking and reflection modules, the KG is retrieved and updated using the same procedures as those employed in NeSyFS, as described in Section 3.1. Besides, the belief state is updated using the same method as ABBEL introduced in Appendix E.2. No slow-thinking module is used in any of these evaluations here. Notably, the “Fast” method are conceptually equivalent to previous KG-augmented LLM agents (Agarwal et al., 2025; Anokhin et al., 2025).

As illustrated in Figure 5, incorporating the reflection module consistently improves performance across all benchmarks. Moreover, the use of KG-provided context leads to additional performance gains, demonstrating both the importance of reflection in mitigating misalignment with task objectives and the effectiveness of the symbolic, structured representations provided by the KG.

## 5 Conclusion

In this work, we propose a neuro-symbolic framework which addresses the challenges introduced by partial observability in a unified manner. Under partial observability, the agent may have difficulties on latent state inference, task objective misalignment, and planning under uncertainty. In NeSyFS, the latent state of the environment is represented and maintained as a memory KG, and triplets retrieved from KG are used as context in every module of NeSyFS. In addition, a KG-augmented reflection module is proposed to address the misalignment of the task objective. Besides, a neuro-symbolic TSMC-style planning algorithm is proposed to tackle the uncertainty in observation prediction and task progress evaluation. We empirically validate the proposed framework and its modules in three popular benchmarks.

## 6 AI ASSISTANCE

In this work, we used generative AI tools for no tasks with required disclosure. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for draft parts of a research paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Hany Abdulsamad, Sahel Mohammad Iqbal, and Simo Sarkka. Sequential monte carlo for policy optimization in continuous pomdps. *Advances in Neural Information Processing Systems*, 38: 41241–41269, 2026.

Krish Agarwal, Yuqian Jiang, Jiaheng Hu, Bo Liu, and Peter Stone. L3m+ p: Lifelong planning with large language models. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3344–3351. IEEE, 2025.

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: learning knowledge graph world models with episodic memory for llm agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 12–20, 2025.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, et al. Iterresearch: Rethinking long-horizon agents with interaction scaling. *arXiv preprint arXiv:2511.07327*, 2025a.

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. Atlas: Agent tuning via learning critical steps. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25334–25349, 2025b.

Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for llm planning? it depends on the discriminator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13659–13678, 2024.

Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

Andy Chung, Yichi Zhang, Kaixiang Lin, Aditya Rawal, Qiaozi Gao, and Joyce Chai. Evaluating long-context reasoning in llm-based webagents. In *NeurIPS Workshop on Bridging Language, Agent, and World Models for Reasoning and Planning*, 2025.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2, 2016.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. Preemptive detection and correction of misaligned actions in llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 222–244, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025a.

Shengyu Feng, Xiang Kong, Shuang Ma, Aonan Zhang, Dong Yin, Chong Wang, Ruoming Pang, and Yiming Yang. Step-by-step reasoning for math problems via twisted sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*, pp. 1433–1445, 2018.

Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pp. 56–60, 2017.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Jianliang He, Siyu Chen, Fengzhuo Zhang, and Zhuoran Yang. From words to actions: Unveiling the theoretical underpinnings of llm-driven autonomous systems. In *International Conference on Machine Learning*, pp. 17807–17841. PMLR, 2024.

Haolin Jin, Linghan Huang, Haipeng Cai, Jun Yan, Bo Li, and Huaming Chen. From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479*, 2024.

Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*, 2025.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. Reflact: World-grounded decision making in llm agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33421–33453, 2025.

Yiming Lei, Jiawei Xu, Chia Xin Liang, Ziqian Bi, Xiaoming Li, Danyang Zhang, Junhao Song, and Zhenyu Yu. Large language model agents: A comprehensive survey on architectures, capabilities, and applications. 2025.

Hui Li, Xuejun Liao, et al. Multi-task reinforcement learning in partially observable stochastic environments. *Journal of Machine Learning Research*, 10(5), 2009.

Aly Lidayan, Jakob Brandt Bjorner, Satvik Golechha, and Alane Suhr. Abbel: Llm agents acting through belief bottlenecks expressed in language. In *NeurIPS Workshop on Bridging Language, Agent, and World Models for Reasoning and Planning*, 2025.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*, 36:23813–23825, 2023.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. *arXiv preprint arXiv:2309.17382*, 2023.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Scaling llm multi-turn rl with end-to-end summarization-based context management. *arXiv preprint arXiv:2510.06727*, 2025.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Matthew Macfarlane, Edan Toledo, Donal John Byrne, Paul Duckworth, and Alexandre Laterre. Spo: Sequential monte carlo policy optimisation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, et al. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*, 2026.

Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2018.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Mrinal Rawat, Ambuje Gupta, Rushil Goomer, Alessandro Di Bari, Neha Gupta, and Roberto Pieraccini. Pre-act: Multi-step planning and reasoning improves acting in llm agents. *arXiv preprint arXiv:2505.09970*, 2025.

Matthew Renze and Erhan Guven. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*, 2024.

Andres Rodriguez, Ronald Parr, and Daphne Koller. Reinforcement learning using approximate belief states. *Advances in Neural Information Processing Systems*, 12, 1999.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40, 2005.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

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

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.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, 2024.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Z Xi, Y Ding, W Chen, B Hong, H Guo, J Wang, D Yang, C Liao, X Guo, W He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

Xuanchen Xiang and Simon Foo. Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1—fundamentals and applications in games, robotics and natural language processing. *Machine Learning and Knowledge Extraction*, 3(3):554–581, 2021.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

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*, 2022b.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *Transactions on Machine Learning Research*, 2025.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Ming Zhu, Juntao Tan, Thai Hoang, Zuxin Liu, Liangwei Yang, et al. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv preprint arXiv:2402.15506*, 2024.

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

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential monte carlo. In *International Conference on Machine Learning*, pp. 60704–60748. PMLR, 2024b.

Congming Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, et al. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*, 2025.