OntoGuard: Enforcing Action Admissibility for LLM Agents in Complex Interactive Environments

Pengxiang Liu Tao Ren* Wei Xiong Tingrui Yang
Junjie Wang Jun Hu*
State Key Laboratory of Intelligent Game, Institute of Software Chinese Academy of Sciences,
University of Chinese Academy of Sciences, Beijing, China
{liupengxiang2024, rentao22, xiongwei2023}@iscas.ac.cn
{yangtingrui2023, junjie, hujun}@iscas.ac.cn
*Corresponding authors

Abstract

Large Language Models (LLMs) have shown impressive reasoning capabilities in agents for complex interactive environments. However, these agents often suffer from hallucinations and lack grounding, leading to unreliable actions that conflict with real-world constraints. Existing approaches mitigate some issues through implicit imitation or sparse reinforcement learning but rely on fitting data distributions without explicitly understanding environmental constraints, often generating actions that are behaviorally distorted or environmentally impermissible. To address this, we introduce OntoGuard, an ontological framework designed to guard LLM agents by enforcing environmental and behavioral admissibility. These constraints are constructed by extracting knowledge from oracle demonstrations, supplemented with world knowledge inherent in LLMs and general knowledge bases. During inference, OntoGuard functions as an active interceptor, using a graph-based constraint-checking mechanism to reject invalid actions and prompt self-correction before acting. Experiments on both ScienceWorld and VirtualHome demonstrate OntoGuard's advantage over state-of-the-art methods, validating its ability to enforce physical and behavioral constraints while preventing invalid actions.

1 Introduction

Agents operating in complex interactive environments represent a significant leap towards general-purpose AI (Wang et al., 2022, 2023; Song et al., 2022; Mu et al., 2023). Unlike static NLP tasks, these agents must perceive dynamic state changes, distinguish intricate entity properties, and reason about long-horizon causal dependencies to achieve goals (Barsalou, 2008; Yao et al., 2023). The ability to navigate such environments is critical for automating sophisticated workflows, ranging from digital simulations to real-world decision-making processes, where adherence to the underlying logic of the world is paramount.

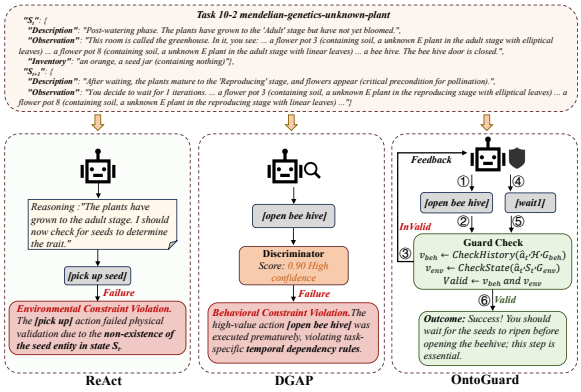

Figure 1: Comparison of execution flows on the "Mendelian Genetics" task. ReAct hallucinates that plants in the "Adult" stage are ready for harvest, triggering an environmental constraint violation due to the non-existence of seeds. DGAP, biased by implicit statistics, commits a behavioral constraint violation due to opening the bee hive prematurely before the flowering stage. In contrast, OntoGuard intercepts these invalid actions by verifying them against explicit logic, obeying the critical precondition (ripening).

However, the core challenge in deploying these agents lies in the strict adherence to the invisible rules and physical laws governing complex environments. Existing approaches largely fall into two categories: **implicit** *imitation learning* from demonstrations and *reinforcement learning* with sparse feedback (Lin et al., 2023; Chen et al., 2021; Luo et al., 2024). While these methods aim to capture high-level behavioral patterns, they share a critical limitation: relying on fitting the data distribution **without explicitly comprehending** the underlying environmental constraints. Consequently, lacking this explicit grounding, such agents often generate actions that are behaviorally distorted or physically impermissible (Li et al., 2022; Lambert et al., 2025).

In fact, environment constraints are often com-

Findings of the Association for Computational Linguistics: ACL 2026, pages 20937–20952

plex and difficult to model. As shown in Figure 1 with the "Mendelian Genetics" task from ScienceWorld (Wang et al., 2022), successful interaction requires strict adherence to constraints: plants must mature to the flowering stage before pollinators are introduced, a detail not easily captured from demonstration data. As a result, ReAct (Yao et al., 2022) misidentifies the "Adult" (vegetative) stage as ready for harvest, attempting an invalid `[pick up seed]` action that violates environmental constraints. DGAP (Qian et al., 2025), biased by the statistical preference of the `[open bee hive]` action, executes it prematurely before the critical precondition (flower presence is met), violating behavioral admissibility. In contrast, a more effective approach is to intercept such invalid actions by verifying them against explicit logic. This raises the technical challenge: *How can we build reliable admissibility rules and localize relevant constraints during interaction with complex environments?*

To address this, we propose **OntoGuard**, an Ontological framework that Guards LLM agents by enforcing **Environmental** and **Behavioral** Admissibility. We build reliable admissibility rules by extracting knowledge from oracle demonstrations, supplemented with the generalized world knowledge inherent in LLMs. During inference, OntoGuard employs a graph-based verification mechanism that acts as an active interceptor to dynamically intercept generated actions, block those that violate established rules, and provide the agent with interpretable feedback for self-correction. OntoGuard achieves state-of-the-art performance, significantly outperforming baselines by ensuring strict adherence to environmental dynamics.

Our contributions are summarized as follows:

- We propose a novel framework for LLM agents in complex interactive environments, shifting the paradigm from implicit learning to explicit verification, thereby preventing invalid actions and ensuring strict adherence to environmental constraints.
- We develop an admissibility-rule construction mechanism that combines the precision of domain-specific oracle logic with the breadth of LLM commonsense knowledge.
- We achieve state-of-the-art performance on both ScienceWorld and VirtualHome benchmarks, demonstrating that OntoGuard significantly improves success rates and efficiency compared to state-of-the-art methods.

2 Related Work

Learning-based Agents. Prior to the advent of large language models, agents operating in complex environments primarily relied on Reinforcement Learning (RL) and Imitation Learning (IL) (Ross et al., 2011) to interact in text-based scenarios. Approaches such as DRRN (He et al., 2016) and KG-A2C (Ammanabrolu and Hausknecht, 2020) employ deep RL to map textual observations directly to action spaces, often augmenting state representations with knowledge graph embeddings to capture entity relations. CALM (Yao et al., 2020) further enhances sample efficiency via generating action priors to prune the search space, while TDT (Chen et al., 2021) frames decision-making as a sequence modeling task with Transformers. Despite their efficacy in closed, rule-rigid domains, these methods inherently struggle with the *combinatorial explosion* of action spaces in complex environments like ScienceWorld. Furthermore, they typically require extensive task-specific training data and rely on sparse reward signals, limiting their ability to generalize to unseen tasks or perform the multi-step causal reasoning required for scientific discovery.

LLM-based Agents. LLMs have transformed this field by leveraging zero-shot reasoning capabilities and vast semantic knowledge. SayCan (brian ichter et al., 2022) grounds LLM planning via learned value functions to assess action feasibility, whereas ReAct (Yao et al., 2022) interleaves Chain-of-Thought (CoT)(Wei et al., 2022) reasoning traces with action execution to enhance plan coherence. To address the issue of error propagation, Reflexion (Shinn et al., 2023; Yao et al., 2024) employs verbal reinforcement learning, utilizing past failures as context for iterative self-correction. More recent hybrid approaches, such as SwiftSage (Lin et al., 2023), integrate fast heuristic policies with slow, deliberate reasoning modules, and DGAP (Qian et al., 2025; Xu et al., 2022) utilizes external discriminators for implicit guidance. Orthogonally, ISR-LLM (Zhou et al., 2024) and Self-Refine (Madaan et al., 2023) explore iterative self-refinement for long-horizon planning and general-purpose self-correction, respectively. However, a critical limitation remains: these methods primarily rely on *implicit* signals, such as statistical probability distributions or post-hoc execution

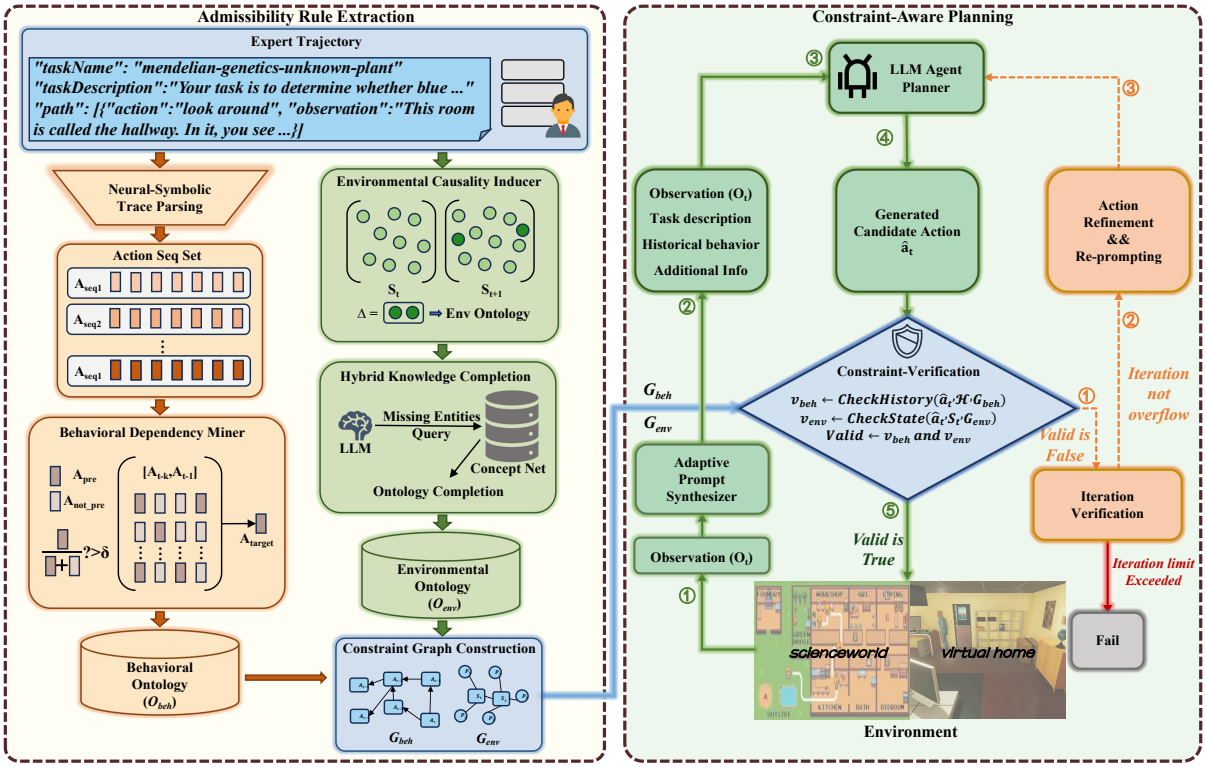

Figure 2: The architecture of the OntoGuard framework. The pipeline transitions from offline knowledge extraction (left) to the construction of Admissibility Rule Graphs (middle), and finally to the online runtime verification loop (right), where the agent's actions are explicitly constrained by the learned rules.

feedback(Huang et al., 2022), to guide behavior. They lack an intrinsic mechanism to rigorously verify the *physical validity* of actions (e.g., object affordances) *before* execution, frequently leading to hallucinations where the agent attempts conceptually plausible but physically impossible actions.

Neuro-Symbolic Reasoning. Neuro-symbolic approaches seek to combine the flexibility of neural networks with the precision of symbolic logic (Hao et al., 2023; Shi et al., 2024). In today's LLM landscape, Knowledge Graphs (KGs) are mostly used for Retrieval-Augmented Generation (RAG) (Abernethy et al., 2024; Zhao et al., 2023) to reduce hallucinations by supplying factual context. Earlier methods such as KG-A2C leveraged graphs for state tracking, but they largely served as memory aids rather than enforcing logical constraints on the planner. OntoGuard departs from this retrieval-centric view by using KGs as a *dynamic verification mechanism*. By constructing explicit rules that capture task requirements and environmental physics, OntoGuard acts as a "safeguard," decoupling high-level planning from low-level constraints to ensure actions are executable and interpretable.

3 Methodology

We present OntoGuard, a neuro-symbolic framework designed to ground LLM agents via explicit environmental and behavioral constraints. OntoGuard strictly decouples the logic of the world from task planning. The overall architecture is illustrated in Figure 2. The framework operates in two phases: (1) **Admissibility Rule Construction**, extracting and completing explicit rule sets from limited oracle demonstrations and generalized commonsense knowledge; and (2) **Constraint-Aware Planning**, where these rules serve as a dynamic "firewall" to rigorously intercept and correct invalid actions during inference.

3.1 Problem Formulation

We formalize the planning task in complex interactive environments as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R}, \mathcal{G} \rangle$. In our experimental environment (e.g., ScienceWorld), these components are instantiated as follows:

- **Goal ($\mathcal{G}$):** $g \sim \mathcal{G}$ denotes the high-level language description of a task (e.g., "*your task is to boil lead*").

- **Observation ($\mathcal{O}$) and State ($\mathcal{S}$):** At each time step t, the agent receives a textual observation $o_t \in \mathcal{O}$ containing feedback from previous actions or descriptions of visible objects, reflecting a partial view of the underlying state $s_t \in \mathcal{S}$.
- **Action ($\mathcal{A}$):** The agent generates an action $a_t \in \mathcal{A}$ following valid templates (e.g., "*use X on Y*").
- **Transition ($\mathcal{T}$) and Reward ($\mathcal{R}$):** The execution of an action triggers a state transition $\mathcal{T}(s_t, a_t) \to s_{t+1}$ and yields a reward $r_t \in \mathcal{R}$.

The objective is to generate a coherent action sequence $a_{0:T}$ that maximizes cumulative reward.

OntoGuard explicitly enforces physical and behavioral validity to prevent hallucination and grounding errors. We aim to approximate a constrained optimal policy π^* defined as:

$$\pi^*(a_t \mid o_{0:t}, g) \propto P_{\text{LLM}}(a_t \mid o_{0:t}, g) \cdot \mathbb{I}(a_t \in \mathcal{V}_{s_t}) \quad (1)$$

where $\mathcal{V}_{s_t}$ represents the set of valid actions derived from our explicit rules. The indicator function $\mathbb{I}(\cdot)$ acts as a hard constraint mechanism:

$$\mathbb{I}(a_t \in \mathcal{V}_{s_t}) = \begin{cases} 1, & \text{if } a_t \text{ satisfies constraints} \\ & \text{in } O_{\text{beh}} \text{ and } O_{\text{env}} \\ 0, & \text{otherwise (trigger reflection)} \end{cases} \quad (2)$$

3.2 Admissibility Rule Extraction

To construct the constraint set $\mathcal{V}_{s_t}$, we extract structured knowledge from a limited set of expert trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$. This process comprises three phases: behavioral rule extraction, environmental rule extraction, and constraint graph compilation.

3.2.1 Behavioral Rule Extraction

Neural-Symbolic Trace Parsing. The raw expert dataset $\mathcal{D}$ comprises unstructured interaction logs across $N = 30$ distinct tasks. To distill computable logic, we introduce the Neural-Symbolic Trace Parsing (NSTP) module, which transforms raw logs into a structured Action Sequence Set $\mathcal{A}_{seq}$. Formally, let $\mathcal{A}_{seq} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{30}\}$, where each $\mathcal{T}_i$ represents the set of successful action sequences for the i-th task. Each task set contains multiple variations $\mathcal{T}_i = \{S_{i,1}, \ldots, S_{i,m}\}$. We employ an LLM-based parser Φ to map a raw step (action a_t, context o_t) to a canonical predicate form $\hat{a}_t$:

$$\hat{a}_t = \Phi(a_t, o_t) \quad \text{where} \quad S_{i,j} = [\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_T] \quad (3)$$

By converting free-form text into standardized predicates, NSTP eliminates linguistic variability, providing a clean symbolic foundation for rule mining.

Behavioral Dependency Miner. Building upon structured sequences, we introduce the BDM to induce logical dependencies. We identify prerequisite actions that consistently precede a target action within a temporal window K. For a target action α in task set $\mathcal{T}_i$, let $W_{t,K} = \{\hat{a}_{t-K}, \ldots, \hat{a}_{t-1}\}$ denote the preceding window actions. We define the dependency strength of a candidate antecedent β as:

$$P(\beta \mid \alpha) = \frac{\sum_{S \in \mathcal{T}_i} \sum_{t:\hat{a}_t = \alpha} \text{Count}(\beta, W_{t,K})}{\sum_{S \in \mathcal{T}_i} \sum_{t:\hat{a}_t = \alpha} |W_{t,K}|} \quad (4)$$

A rule $\beta \to \alpha$ is admitted into the Behavioral Ontology O_{beh} if this ratio exceeds a confidence threshold δ. Additionally, actions appearing predominantly in initial steps across diverse tasks are classified as "Free-Flow Actions" and exempted from strict prerequisite constraints to maintain exploration flexibility.

3.2.2 Environmental Rule Extraction

Environmental Causality Inducer. While behavioral rules dictate sequencing, Environmental Rules (O_{env}) encode physical dynamics. The ECI extracts these rules by analyzing state mutations. Given a raw observation o_t, parser Φ maps it to a structured state S_t containing properties (P_{prop}), causal relations (P_{rel}), and lifecycle states (P_{state}):

$$S_t = \{p \mid p \in P_{prop} \cup P_{rel} \cup P_{state}\} \quad (5)$$

The **State Mutation** Δ_t is defined as predicates emerging in S_{t+1} that were absent in S_t:

$$\Delta_t = S_{t+1} \setminus S_t = \{p \mid p \in S_{t+1} \wedge p \notin S_t\} \quad (6)$$

Correlating executed action α_t with mutation Δ_t, the ECI induces rules $\mathcal{E}_{traj} = \{E_1, E_2, E_3\}$:

- E_1 **(Affordance Rules):** Derived from P_{prop}, linking objects to actionable properties.
- E_2 **(Causality Rules):** Derived from P_{rel}, capturing cause-effect dynamics.
- E_3 **(State Transition Rules):** Derived from P_{state}, defining valid lifecycle progressions.

Hybrid Knowledge Completion. As expert trajectories $\mathcal{E}_{traj}$ naturally suffer from sparse coverage, we implement a hybrid completion strategy. We leverage ConceptNet (Speer et al., 2017) as an

external knowledge source. For any entity e appearing in observations but lacking defined rules, we employ an LLM to retrieve relevant triplets (e.g., `CapableOf`, `UsedFor`, `HasProperty`). These are filtered for domain relevance and merged into the final rule set $\mathcal{E}_{final} = \mathcal{E}_{traj} \cup \mathcal{E}_{ext}$.

3.2.3 Constraint Graph Construction

We compile the ontologies into two specialized graph structures for real-time verification.

Behavioral Graph (G_{beh}). Constructed as a Directed Acyclic Graph (DAG) to enforce procedural correctness.

- **Nodes (V_B):** Action Prototypes (abstracted predicates).
- **Edges (E_B):** Strict Temporal Prerequisites. An edge $\langle u, v \rangle$ implies u is a mandatory antecedent for v.

Environmental Graph (G_{env}). Constructed as a knowledge graph to enforce physical grounding.

- **Nodes (V_E):** Object Classes and Physical Properties.
- **Edges (E_E):** Physical laws encoded as relational triplets. An affordance edge explicitly validates if an object supports a specific interaction.

During inference, these graphs serve as the "Firewall", enabling rapid constraint checking by querying G_{beh} for history validity and G_{env} for property validity.

3.3 Constraint-Aware Planning

In the inference phase, OntoGuard functions as an explicit safeguard between the LLM planner and the environment. We introduce a **Constraint-Verification Loop** that dynamically validates candidate actions against G_{beh} and G_{env}. Due to space limits, the whole algorithm is detailed in Algorithm 1 of Appendix D.

At step t, the environment grounds observation o_t into state S_t. The LLM planner proposes a candidate action $\hat{a}_t$, which is intercepted for a dual-check:

Behavioral Admissibility Verification (G_{beh}). OntoGuard verifies whether all logical antecedent actions for $\hat{a}_t$ exist in the history H_t. The behavioral validity v_{beh} is defined as:

$$v_{beh} = \mathbb{I}\left(\forall p \in \mathcal{P}(\hat{a}_t),\ p \in H_t\right) \tag{7}$$

where $\mathcal{P}(\hat{a}_t)$ denotes the set of prerequisite actions retrieved from G_{beh}.

Environmental Admissibility Verification (G_{env}). OntoGuard queries G_{env} to enforce physical feasibility. Environmental validity v_{env} is the conjunction of check functions for $\mathcal{E} = \{E_1, E_2, E_3\}$:

$$v_{env} = f_1(\hat{a}_t, E_1) \wedge f_2(\hat{a}_t, S_t, E_2) \wedge f_3(\hat{a}_t, S_t, E_3) \tag{8}$$

Here, f_1, f_2, f_3 validate affordances, causal preconditions, and state progressions, respectively.

Overall Validity. The global constraint function (Eq. 2) is the logical conjunction:

$$\mathbb{I}(\hat{a}_t \in \mathcal{V}_{s_t}) = v_{beh} \wedge v_{env} \tag{9}$$

Interception and Refinement. If a violation is detected, the action is blocked. OntoGuard synthesizes a structured error message f_{err} derived from the violated edge. The planner is prompted to reflect and regenerate a corrected action, ensuring only logically sound actions incur interaction costs.

4 Experiments

To evaluate the effectiveness of OntoGuard in complex interactive reasoning, we utilize two distinct benchmarks: **ScienceWorld** (Wang et al., 2022) and **VirtualHome** (Puig et al., 2018).

4.1 ScienceWorld

4.1.1 Experimental Setup

ScienceWorld simulates a standardized science curriculum, comprising 30 distinct task types (e.g., thermodynamics, electrical circuits) with varying levels of complexity. Following (Lin et al., 2023), we categorize tasks into three horizon levels (Short, Medium, Long) and use the zero-shot Final Score (0–100) as the primary metric. Detailed dataset statistics and a comprehensive description of our statistical reporting protocols are provided in Appendix A.

Implementation Details. For Behavioral Rule Extraction, we set the sliding window size $K = 5$ to capture immediate causal dependencies.

4.1.2 Baseline Agents

Learning-based Agents. We include DRRN (He et al., 2016), KG-A2C (Ammanabrolu and Hausknecht, 2020), and CALM (Yao et al., 2020) as representatives of reinforcement learning approaches, and TDT (Chen et al., 2021) as a Transformer-based behavioral cloning baseline.

Tasks Topic		Algorithms									
		Learning-based Agents				LLM-based Planners					
		DRRN	KG-A2C	CALM	TDT	SayCan	ReAct	Reflexion	SwiftSage	DGAP	**OntoGuard (Ours)**
Matter	1-1 (L)	3.52	0.0	0.0	0.71	33.06	3.52	4.22	97.04	100.0	**100.0**
	1-2 (L)	3.52	0.0	0.0	0.44	10.39	13.70	10.61	87.04	92.75	**94.20**
	1-3 (L)	0.0	4.0	0.0	3.88	3.88	7.78	7.78	72.78	74.0	**79.40**
	1-4 (L)	0.0	0.0	0.0	0.55	0.37	9.88	0.92	100.0	100.0	**100.0**
Measurement	2-1 (M)	6.56	6.0	1.0	6.16	26.37	7.19	5.92	99.17	100.0	**100.0**
	2-2 (M)	5.50	11.0	1.0	6.43	8.03	6.10	28.59	88.17	80.17	**89.31**
	2-3 (L)	6.0	4.0	1.0	19.87	17.41	22.37	22.37	95.73	88.33	**97.14**
Electricity	3-1 (S)	12.0	7.0	5.0	40.55	52.14	56.0	**100.0**	88.67	91.50	88.91
	3-2 (M)	9.0	4.0	7.0	14.26	22.50	54.33	17.45	55.33	58.0	**68.50**
	3-3 (M)	9.05	4.0	2.0	10.16	**99.56**	76.19	72.54	71.90	78.57	80.0
	3-4 (M)	9.52	4.0	2.0	21.65	47.76	**88.81**	70.22	77.86	88.14	88.50
Classification	4-1 (S)	15.0	18.0	10.0	41.93	22.87	26.67	64.93	100.0	100.0	**100.0**
	4-2 (S)	45.0	44.0	54.0	55.76	58.18	80.0	87.27	100.0	100.0	**100.0**
	4-3 (S)	21.67	16.0	10.0	27.82	20.87	53.33	16.42	91.67	**100.0**	95.50
	4-4 (S)	19.17	15.0	8.0	47.15	31.43	27.50	100.0	100.0	100.0	**100.0**
Biology	5-1 (L)	8.0	6.0	2.0	6.89	9.92	9.06	7.33	74.59	73.14	**78.16**
	5-2 (L)	14.29	11.0	4.0	11.86	13.93	18.57	13.0	93.93	90.57	**97.0**
Chemistry	6-1 (M)	15.77	17.0	3.0	15.10	47.81	51.04	**70.35**	49.40	57.40	66.20
	6-2 (S)	26.67	19.0	6.0	15.70	39.26	58.89	70.67	100.0	100.0	**100.0**
	6-3 (M)	10.37	4.0	3.0	5.25	19.72	40.74	15.77	91.48	92.43	**95.50**
Biology	7-1 (S)	50.0	43.0	6.0	30.0	80.0	60.0	100.0	95.0	**100.0**	91.60
	7-2 (S)	50.0	32.0	10.0	8.43	67.50	67.50	84.37	85.0	85.71	**90.50**
	7-3 (S)	33.33	23.0	4.0	8.34	50.0	50.0	83.0	93.33	92.71	**94.40**
Biology	8-1 (M)	21.0	5.0	4.0	3.86	20.91	27.67	2.58	89.0	100.0	**100.0**
	8-2 (S)	8.0	10.0	0.0	8.0	16.0	8.0	8.0	68.50	38.50	**77.38**
Forces	9-1 (L)	10.0	4.0	0.0	2.53	21.94	40.50	50.63	75.0	75.0	**80.20**
	9-2 (L)	10.0	4.0	3.0	14.66	32.26	44.0	**100.0**	70.0	83.33	81.0
	9-3 (L)	10.0	4.0	2.0	9.12	13.67	41.0	70.62	60.0	71.43	**72.50**
Biology	10-1 (L)	16.80	11.0	2.0	1.51	67.53	25.70	50.90	92.30	87.71	**96.30**
	10-2 (L)	17.0	11.0	2.0	1.29	59.45	16.80	23.69	77.60	78.0	**84.50**
Summary	Short	28.08	22.70	11.30	28.37	43.83	48.79	71.47	92.22	90.84	**93.83**
	Medium	10.85	6.88	2.88	10.36	36.58	44.01	35.43	77.79	81.84	**86.0**
	Long	8.26	4.92	1.33	6.11	23.65	21.07	30.17	83.0	84.52	**88.37**
Total	Overall	15.56	11.37	5.07	14.66	33.82	36.43	45.34	84.68	85.91	**89.56**

Table 1: Comparison of Average Final Scores on the ScienceWorld benchmark across 30 task types. Tasks are categorized by complexity (Short, Medium, Long) based on the average length of oracle trajectories (**Len*).

LLM-based Planners. We evaluate SayCan (brian ichter et al., 2022), which grounds LLMs via value functions; ReAct (Yao et al., 2022), which interleaves reasoning and acting; and Reflexion (Shinn et al., 2023), which utilizes verbal reinforcement learning from failure. Further, we compare against SwiftSage (Lin et al., 2023), a dual-process framework combining fast and slow thinking, and DGAP (Qian et al., 2025), which uses discriminator-guided action planning.

Note that OntoGuard, DGAP, SwiftSage, ReAct, SayCan, and Reflexion utilize GPT-4 as the backbone model for fair comparison.

4.1.3 Main Results

As shown in Table 1, OntoGuard achieves a new state-of-the-art Overall Score of 89.56, significantly outperforming strong baselines like SwiftSage (84.68) and DGAP (85.91). This superior performance is driven by its robustness in complex long scenarios, despite nuanced trade-offs in simpler tasks.

Long-Horizon Performance. OntoGuard exhibits the most substantial improvement in Long tasks (Score: 88.37), outperforming the previous best DGAP (84.52) by +3.85 points. In tasks like 1-1 and 10-1, OntoGuard achieves near-perfect scores (100.0 and 96.30 respectively). Long tasks typically involve intricate causal chains where errors compound over time. While baseline methods often suffer from "semantic drift," generating plausible but physically invalid actions, OntoGuard's Environmental (O_{env}) and Behavioral (O_{beh}) Ontologies act as explicit guardrails, ensuring adher-

Task	Full	w/o G_{beh}	w/o G_{env}	w/o All	Task	Full	w/o G_{beh}	w/o G_{env}	w/o All
1-1(L)	**100.00**	88.50	82.30	75.20	5-1(L)	**78.16**	72.00	75.50	65.40
1-2(L)	**94.20**	88.00	90.50	82.10	5-2(L)	**97.00**	92.50	88.00	80.50
1-3(L)	**79.40**	70.50	72.00	65.00	6-1(M)	**66.20**	58.50	60.20	52.40
1-4(L)	**100.00**	98.00	98.00	95.00	6-2(S)	**100.00**	100.00	100.00	98.00
2-1(M)	**100.00**	92.00	94.50	88.00	6-3(M)	**95.50**	88.00	90.00	82.00
2-2(M)	**89.31**	78.00	82.50	70.00	7-1(S)	91.60	**98.50**	94.00	96.00
2-3(L)	**97.14**	88.00	92.00	85.00	7-2(S)	**90.50**	85.50	88.00	82.50
3-1(S)	88.91	92.80	91.50	**95.20**	7-3(S)	**94.40**	88.00	90.50	85.00
3-2(M)	**68.50**	60.00	62.00	55.00	8-1(M)	**100.00**	92.00	95.50	88.00
3-3(M)	**80.00**	70.50	65.00	58.00	8-2(S)	**77.38**	70.00	72.00	65.50
3-4(M)	88.50	**92.00**	90.00	85.00	9-1(L)	**80.20**	72.00	75.00	68.00
4-1(S)	**100.00**	95.00	98.00	90.00	9-2(L)	**81.00**	75.00	78.00	70.00
4-2(S)	**100.00**	100.00	100.00	98.00	9-3(L)	**72.50**	65.00	68.00	60.00
4-3(S)	95.50	**98.00**	96.00	92.00	10-1(L)	**96.30**	82.00	85.50	75.00
4-4(S)	**100.00**	95.00	98.00	90.00	10-2(L)	**84.50**	75.00	78.00	70.00

Table 2: Ablation study of OntoGuard on ScienceWorld. We compare the full model against variants removing the Behavioral Graph (G_{beh}), Environmental Graph (G_{env}), or both. Scores denote Final Score.

ence to strict causal dependencies throughout long trajectories.

Robustness in Complex Physics. In Medium tasks requiring precise physical manipulation (e.g., 2-2, 8-1), OntoGuard achieves the highest scores (89.31 and 100.00). The explicit verification loop effectively filters out invalid object interactions, addressing a common failure mode for implicit learning agents like ReAct.

Performance on Short Tasks. While OntoGuard excels overall, it is not the top performer in every task. For example, in Task 7-1, OntoGuard scores 91.60 (vs. DGAP 100.00), and in Task 6-1, it scores 66.20 (vs. Reflexion 70.35). We attribute this to the "Conservative Bias" of explicit constraints. OntoGuard relies on rules extracted from limited expert trajectories. If a task admits multiple valid solutions (e.g., a shortcut path $A \rightarrow C$) but the ontology only encodes one path ($A \rightarrow B \rightarrow C$), the Constraint-Verification Loop may over-constrain the agent, blocking valid shortcut actions that appear "unseen" or "risky." In contrast, unconstrained agents like DGAP or Reflexion may serendipitously discover these shortcuts. We argue this flexibility trade-off is necessary for the reliability gains in complex, high-stake environments.

4.1.4 Ablation Study

We conducted a fine-grained ablation study to decouple the contributions of the Behavioral Graph (G_{beh}) and the Environmental Graph (G_{env}). The results (Table 2) reveal a trade-off between strict constraint enforcement and exploratory flexibility.

Criticality for Long-Horizon Reasoning. OntoGuard (Full) consistently achieves superior performance in Long and most Medium tasks.

- **Impact of G_{beh}:** Removing behavioral constraints causes significant drops in procedural tasks like Genetics. Without temporal logic from expert traces, the agent often attempts actions out of order.
- **Impact of G_{env}:** Removing environmental constraints degrades performance in Matter tasks, where the agent fails to recognize object affordances.

Conservative Bias in Short Tasks. In certain Short tasks, ablated variants slightly outperform the Full model. For instance, in Task 7-1, the **w/o** G_{beh} variant scores 98.50, surpassing the Full model's 91.60. As discussed, strict behavioral constraints can block valid shortcut solutions ($A \rightarrow C$) not present in the expert demonstration. In low-risk tasks, this strictness hinders the discovery of novel solutions, though it remains essential for safety in complex environments.

4.1.5 Action Reliability Analysis

To evaluate the reliability of a sequence of actions $\mathcal{S} = \{\tau_i\}_{i=1}^{N}$, we introduce two metrics:

- **Execution Success Rate (EXEC):** Measures the proportion of sequences that execute without triggering environment runtime errors.

$$\text{EXEC} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\text{Error}(\tau_i) = \emptyset)$$

- **Task Success Rate (SR):** Measures the proportion of sequences that successfully reach

Task	Metric	ReAct	DGAP	OntoGuard
1-1 (L)	EXEC.	77.67	92.00	**96.33**
	SR.	69.67	87.00	**89.00**
3-4 (M)	EXEC.	**90.67**	82.33	83.00
	SR.	**88.00**	77.00	78.00
4-1 (S)	EXEC.	66.00	91.67	**94.00**
	SR.	26.67	82.00	**86.33**
10-1 (L)	EXEC.	60.00	89.00	**94.67**
	SR.	42.67	84.33	**90.00**
Average	EXEC.	73.59	88.75	**92.00**
	SR.	56.75	82.58	**85.83**

Table 3: Quantitative comparison of Execution Success Rate (EXEC) and Task Success Rate (SR) across four representative tasks of varying complexity. Average results are reported over 3 independent runs.

the goal state.

$$\text{SR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\text{Score}(\tau_i) = 100)$$

We set $N = 100$ and report average results over 3 independent runs on four representative tasks: 1-1 (L), 3-4 (M), 4-1 (S), and 10-1 (L). As presented in Table 3, OntoGuard consistently achieves the highest EXEC and SR across most tasks.

Dominance in Standard Tasks. In Tasks 1-1, 4-1, and 10-1, OntoGuard achieves near-perfect EXEC scores. This validates that our Constraint-Verification Loop effectively filters out physically invalid actions, preventing the runtime errors that plague ReAct.

Analysis of Task 3-4. In Task 3-4 (Test Conductivity), ReAct outperforms both DGAP and OntoGuard. This task involves connecting circuit components, allowing for high combinatorial diversity. OntoGuard's behavioral ontology, extracted from limited traces, likely captures only a standard wiring pattern. When the agent attempts an alternative valid connection, OntoGuard blocks it as a "violation." ReAct, unburdened by prior constraints, explores these alternative paths freely.

4.1.6 Data Efficiency Analysis

To evaluate sensitivity to expert data volume, we vary the proportion of expert traces used for rule construction. Figure 3 reports average EXEC and SR over 3 independent runs on the same four representative tasks as Table 3 (1-1, 3-4, 4-1, 10-1).

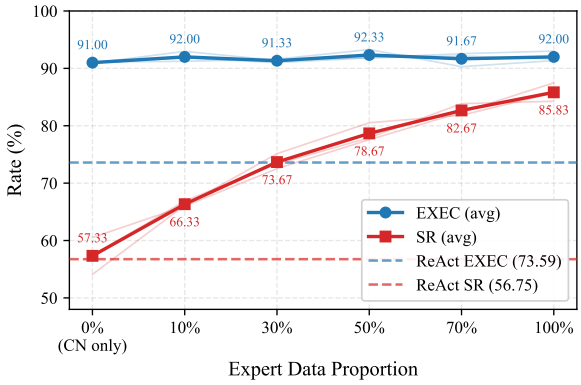

Figure 3: Data efficiency analysis of OntoGuard. EXEC remains stable (91.00–92.33) regardless of expert data volume, while SR degrades gracefully. Both metrics stay above the ReAct baselines even at 0% expert data (ConceptNet only).

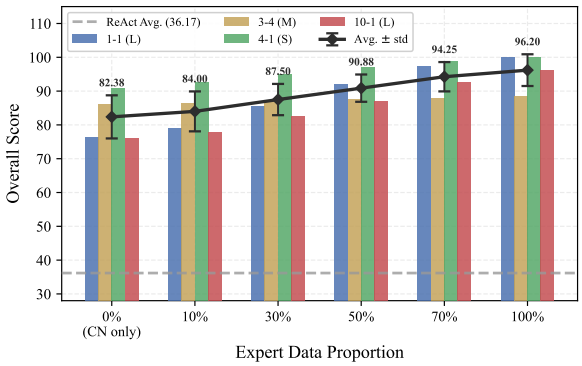

Figure 4: Overall Final Score under varying expert data proportions on four representative tasks. Darker cells indicate higher scores. All tasks degrade gracefully as expert data is reduced.

Stable Execution Quality. EXEC fluctuates within a narrow band (91.00–92.33) across all data proportions, indicating that ConceptNet-derived environmental rules (G_{env}) alone are sufficient to maintain high action-level validity, independent of expert data volume.

Graceful SR Degradation. SR decreases from 85.83 to 57.33 as expert data is reduced, reflecting progressively narrower behavioral rule coverage in G_{beh}. From 10% to 70%, the behavioral rule set captures dominant procedural patterns, yielding the largest marginal SR gains. Beyond 70%, additional traces contribute mostly redundant rules, explaining the plateau. Notably, even at 0% expert data, SR (57.33) still exceeds zero-shot ReAct (56.75), confirming that environmental constraint enforcement from external knowledge alone provides substantial grounding.

Method	In-Distribution		NovelScenes		NovelTasks	
	EXEC.	SR.	EXEC.	SR.	EXEC.	SR.
ReAct	87.33 ± 2.02	82.33 ± 1.76	38.67 ± 1.45	32.33 ± 1.45	49.67 ± 3.18	49.00 ± 3.21
Reflexion	79.67 ± 3.38	79.33 ± 3.18	54.33 ± 1.76	53.33 ± 1.76	47.33 ± 1.67	46.00 ± 1.15
SwiftSage	90.67 ± 0.86	84.33 ± 2.12	63.00 ± 1.68	56.33 ± 1.12	78.33 ± 1.03	68.00 ± 2.03
DGAP	93.33 ± 1.76	88.00 ± 2.45	71.67 ± 1.15	62.67 ± 1.33	73.67 ± 1.15	72.17 ± 3.18
OntoGuard	**94.83 ± 1.10**	**89.50 ± 1.85**	**74.20 ± 1.20**	**64.50 ± 1.40**	**76.15 ± 1.25**	**74.50 ± 2.80**

Table 4: Overall performance of OntoGuard and baselines across VirtualHome settings. OntoGuard demonstrates superior generalization in both execution validity (EXEC) and goal completion (SR).

Score Degradation Pattern. Figure 4 further reports the overall Final Score under the same settings. The average score decreases from 96.20 (100%) to 82.38 (0%), yet remains substantially above the ReAct baseline (36.17) across all proportions. A clear complexity-dependent pattern emerges: Long-horizon tasks (1-1 and 10-1) exhibit the steepest degradation (-23.50 and -20.30 points, respectively), as their extended action sequences rely heavily on behavioral rules to maintain procedural correctness. In contrast, shorter tasks are markedly more resilient—Task 3-4 (M) drops only 2.50 points, while Task 4-1 (S) drops 9.00 points—since their success depends more on environmental constraints that are well-covered by ConceptNet. The narrowing error bars at higher data proportions further indicate that sufficient expert data reduces cross-task variance, yielding uniformly strong performance.

4.1.7 Trajectory Efficiency Analysis

We evaluate planning efficiency via score progression trajectories on all 30 tasks. OntoGuard reaches goal states faster in most cases, especially long-horizon tasks, by filtering invalid branches with the Constraint-Verification Loop and reducing trial-and-error exploration. This produces more direct trajectories than DGAP's soft discriminator guidance. In a few simple tasks, conservative rule checking may slow OntoGuard slightly. Details are in Appendix B.1.

4.1.8 Token Efficiency Analysis

We measure average tokens per action (t/a) over all 30 tasks to quantify verification overhead. OntoGuard's token usage is comparable to ReAct and slightly higher than DGAP, while SwiftSage remains the most efficient. The modest cost comes from explicit constraint verification and feedback, but it reduces wasted exploration by blocking invalid actions early, yielding a favorable efficiency-reliability trade-off. Detailed results are in Appendix B.2.

4.1.9 Guard Accuracy Analysis

We analyze the guard's decision boundary by evaluating the Constraint-Verification Loop on a balanced dataset of 400 state-action pairs (200 valid from Oracle trajectories and 200 invalid from ReAct failures). Compared with an LLM Self-Check baseline, OntoGuard achieves markedly higher precision, reliably filtering physically impossible actions, while exhibiting a modest recall drop that quantifies its conservative bias (occasional over-blocking). This trade-off favors safety in irreversible interactive settings. Detailed results are in Appendix B.3.

4.2 VirtualHome

VirtualHome is an interactive platform that challenges agents with complex household activity combinations. Adopting the evaluation protocol from (Li et al., 2022), we test on three settings: *In-Distribution*, *NovelTasks*, and *NovelScenes*. Table 4 presents the overall results, where OntoGuard consistently outperforms baselines across all settings. Due to space limits, detailed analysis is provided in Appendix C.

5 Conclusion

In this paper, we introduced OntoGuard, a framework that safeguards LLM agents by enforcing Environmental and Behavioral Admissibility. OntoGuard extracts rules from oracle demonstrations and LLM commonsense knowledge to intercept constraint-violating actions and prompt self-correction during inference. Evaluations on ScienceWorld and VirtualHome demonstrate state-of-the-art success rates and efficiency, with strong robustness under strict physical and behavioral constraints, paving the way for safer, more reliable agents.

Limitations

OntoGuard primarily relies on offline rule construction derived from expert demonstrations. While we mitigate data sparsity via external knowledge bases, the performance remains bounded by the diversity of the initial expert traces. Consequently, in highly dynamic environments where physical rules evolve rapidly, the static nature of the constructed rules may limit adaptability. Future work will explore online rule updating mechanisms to support lifelong learning in non-stationary settings.

References

Jacob Abernethy, Alekh Agarwal, Teodor Vanislavov Marinov, and Manfred K. Warmuth. 2024. A mechanism for sample-efficient in-context learning for sparse retrieval tasks. In *Proceedings of The 35th International Conference on Algorithmic Learning Theory*, volume 237 of *Proceedings of Machine Learning Research*, pages 3–46. PMLR.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Lawrence W Barsalou. 2008. Grounded cognition. *Annu. Rev. Psychol.*, 59(1):617–645.

brian ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, and 25 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany. Association for Computational Linguistics.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2025. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797, Albuquerque, New Mexico. Association for Computational Linguistics.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu. 2022. Pre-trained language models for interactive decision-making. In *Advances in Neural Information Processing Systems*.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jianlan Luo, Perry Dong, Yuexiang Zhai, Yi Ma, and Sergey Levine. 2024. RLIF: Interactive imitation learning as reinforcement learning. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. 2023. EmbodiedGPT: Vision-language pre-training via embodied chain of thought. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xavier Puig, Kevin Kyunghwan Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Haofu Qian, Chenjia Bai, Jiatao Zhang, Fei Wu, Wei Song, and Xuelong Li. 2025. Discriminator-guided embodied planning for LLM agent. In *The Thirteenth International Conference on Learning Representations*.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA. PMLR.

Yu-Zhe Shi, Haofei Hou, Zhangqian Bi, Fanxu Meng, Xiang Wei, Lecheng Ruan, and Qining Wang. 2024. AutoDSL: Automated domain-specific language design for structural representation of procedures with constraints. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12177–12214, Bangkok, Thailand. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Chan Hee Song, Jiaman Wu, Clay Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2986–2997.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? pages 11279–11298.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with LLMs enables open-world multi-task agents. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. 2022. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, pages 24725–24742. PMLR.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh R N, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil L Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large language models as commonsense knowledge for large-scale task planning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *ICRA*, pages 2081–2088.

A Dataset Statistics

Table 5 presents the details of all 30 types of tasks in the ScienceWorld benchmark. We utilized the original full dataset for training, incorporating all available variations across every task type. This approach ensures that the model is exposed to the complete distribution of the environment, including tasks with larger variation counts. To conduct a comprehensive assessment, we evaluated the agents on the complete set of test variations provided by the benchmark. During evaluation, some agents may receive a negative score from the engine and be unable to proceed further due to their final action violating task requirements and being irrecoverable. In such instances, we recorded their last non-negative scores for the final performance metric.

To rigorously evaluate both the peak reasoning capability and the stability of our framework, we adopt a stratified reporting strategy. For the main benchmark results (Table 1) and the ablation studies (Table 2), we report the **maximum score** achieved across 3 independent runs (Best-of-3). This metric is chosen to demonstrate the upper bound of the agent's problem-solving potential, mitigating the impact of sporadic failures caused by the stochastic nature of LLM generation in open-ended action spaces. Conversely, for the Execution Robustness Analysis (Table 3) and VirtualHome experiments (Table 4), where consistency is the primary evaluative criterion, we report the **mean performance** (and standard deviation where applicable) to strictly quantify algorithmic stability and variance.

B ScienceWorld Additional Results

B.1 Trajectory Efficiency Analysis

We visualize the score progression trajectories for all 30 tasks in Figure 5 to analyze planning efficiency. Compared to the Oracle baseline, OntoGuard demonstrates the ability to skip redundant exploratory steps.

OntoGuard achieves the most rapid progression in the majority of scenarios, particularly in Long (L) tasks (e.g., Rows 4 and 5). By explicitly filtering invalid branches via the Constraint-Verification Loop, OntoGuard eliminates trial-and-error search, allowing the agent to effectively bypass inefficient regions of the decision space. Consequently, it reaches the goal state significantly faster than DGAP, which relies on implicit discriminator guidance and may still explore suboptimal paths.

Algorithm 1 Explicit Constraint-Aware Planning

Require: Task Description I, Init Obs O_0
Require: Constraint Graphs G_{beh}, G_{env}
Require: Max Refinement Iterations K
Initialize history $\mathcal{H} \leftarrow \{I\}$; $t \leftarrow 0$
while Task not completed **do**
  $S_t \leftarrow \Phi_{\text{LLM}}(O_t)$; $\mathcal{H} \leftarrow \mathcal{H} \cup \{O_t\}$
  $\hat{a}_t \leftarrow \textsc{LlmPlanner}(\mathcal{H})$
  $k \leftarrow 0$; Valid $\leftarrow$ False
  while $k < K \wedge \neg$Valid **do**
    $v_{beh} \leftarrow \textsc{CheckHistory}(\hat{a}_t, \mathcal{H}, G_{beh})$
    $v_{env} \leftarrow \textsc{CheckState}(\hat{a}_t, S_t, G_{env})$
    if $v_{beh} \wedge v_{env}$ **then**
      Valid $\leftarrow$ True
    else
      $M_{\text{info}} \leftarrow (\hat{a}_t, G_{beh}, G_{env})$
      $msg \leftarrow \textsc{SynFeedback}(M_{\text{info}})$
      $\mathcal{H}_{\text{tmp}} \leftarrow \mathcal{H} \cup \{\text{“System”} : msg\}$
      $\hat{a}_t \leftarrow \textsc{LlmPlanner}(\mathcal{H}_{\text{tmp}})$
      $k \leftarrow k + 1$
    end if
  end while
  if Valid **then**
    $O_{t+1}, r,$ done $\leftarrow$ Environment.Step$(\hat{a}_t)$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\hat{a}_t\}$; $t \leftarrow t + 1$
  else
    return Failure
  end if
end while

Analysis of Exceptions. In specific tasks such as Task 3-4 (M) and Task 7-1 (S), OntoGuard's trajectory is marginally slower than DGAP. This behavior aligns with our design philosophy: the strict rule checking mechanism introduces a conservative bias, occasionally blocking valid "shortcut" actions that lack sufficient support in the extracted knowledge base. While DGAP's probabilistic model may exploit these shortcuts, OntoGuard prioritizes physical correctness. However, this minor efficiency trade-off in simpler tasks is negligible compared to the substantial reliability gains achieved in complex, long-horizon environments.

B.2 Token Efficiency Analysis

To address concerns regarding the computational overhead of the Constraint-Verification Loop, we analyzed the average tokens per action (t/a) across

Task Type	Topic	Name	*Len	#Vars: Train	Dev	Test
1-1	Matter	Changes of State (Boiling)	107.7	14	7	9
1-2	Matter	Changes of State (Melting)	78.6	14	7	9
1-3	Matter	Changes of State (Freezing)	88.9	14	7	9
1-4	Matter	Changes of State (Any)	75.2	14	7	9
2-1	Measurement	Use Thermometer	21.4	270	10	10
2-2	Measurement	Measuring Boiling Point (known)	35.2	218	10	10
2-3	Measurement	Measuring Boiling Point (unknown)	65	150	10	10
3-1	Electricity	Create a circuit	13.6	10	5	5
3-2	Electricity	Renewable vs Non-renewable Energy	20.8	10	5	5
3-3	Electricity	Test Conductivity (known)	25.6	450	225	225
3-4	Electricity	Test Conductivity (unknown)	29	300	10	10
4-1	Classification	Find a living thing	14.6	150	10	10
4-2	Classification	Find a non-living thing	8.8	150	10	10
4-3	Classification	Find a plant	12.6	150	10	10
4-4	Classification	Find an animal	14.6	150	10	10
5-1	Biology	Grow a plant	69.5	62	10	10
5-2	Biology	Grow a fruit	79.6	62	10	10
6-1	Chemistry	Mixing (generic)	33.6	16	8	8
6-2	Chemistry	Mixing paints (secondary colours)	15.1	18	9	9
6-3	Chemistry	Mixing paints (tertiary colours)	23	18	9	9
7-1	Biology	Identify longest-lived animal	7	62	10	10
7-2	Biology	Identify shortest-lived animal	7	62	10	10
7-3	Biology	Identify longest-then-shortest-lived animal	8	62	10	10
8-1	Biology	Identify life stages (plant)	40	6	3	5
8-2	Biology	Identify life stages (animal)	16.3	4	2	4
9-1	Forces	Inclined Planes (determine angle)	97	84	42	42
9-2	Forces	Friction (known surfaces)	84.9	692	346	348
9-3	Forces	Friction (unknown surfaces)	123.1	80	40	42
10-1	Biology	Mendelian Genetics (known plants)	130.1	60	30	30
10-2	Biology	Mendelian Genetics (unknown plants)	132.1	240	120	120
		Short ($0 <$ *Len ≤ 20)	11.76	81.80	8.60	8.80
		Medium ($20 <$ *Len ≤ 50)	28.58	161.00	35.00	35.25
		Long (*Len > 50)	94.30	123.83	53.00	54.00
		Overall (avg)	49.26	119.73	33.40	33.93
		Overall (sum)	N/A	3,592	1,002	1,018

Table 5: The statistics of ScienceWorld benchmark. **Len* is the average length of the oracle agent's trajectories. The table details the number of variations used in each split, covering the full training and evaluation datasets.

Method	Precision	Recall	F1-Score
LLM-Self-Check	76.5%	**94.0%**	84.4%
OntoGuard (Ours)	**98.5%**	84.5%	**91.0%**
Diff	+22.0%	-9.5%	+6.6%

Table 6: Evaluation of the Guard Mechanism as a binary classifier. OntoGuard achieves near-perfect Precision (Safety), effectively acting as a firewall against hallucinations. The lower Recall compared to LLM-Self-Check quantifies our "Conservative Bias," indicating a trade-off where we prioritize correctness over permissive exploration.

all 30 tasks. Table 7 compares OntoGuard against key baselines. While SwiftSage remains the most efficient due to its specialized architecture, the token consumption of OntoGuard is comparable to standard reasoning agents like ReAct and slightly higher than the implicit discriminator method DGAP.

The moderate increase in token usage stems from the explicit Constraint-Verification Loop, which requires additional input tokens for ontology querying and feedback generation. However, this overhead is well within an acceptable range for modern LLM applications. Crucially, this investment yields a disproportionately high return in reliability: OntoGuard significantly outperforms baselines in success rate. By proactively intercepting invalid actions, OntoGuard prevents the agent from engaging in long, ineffective trial-and-error

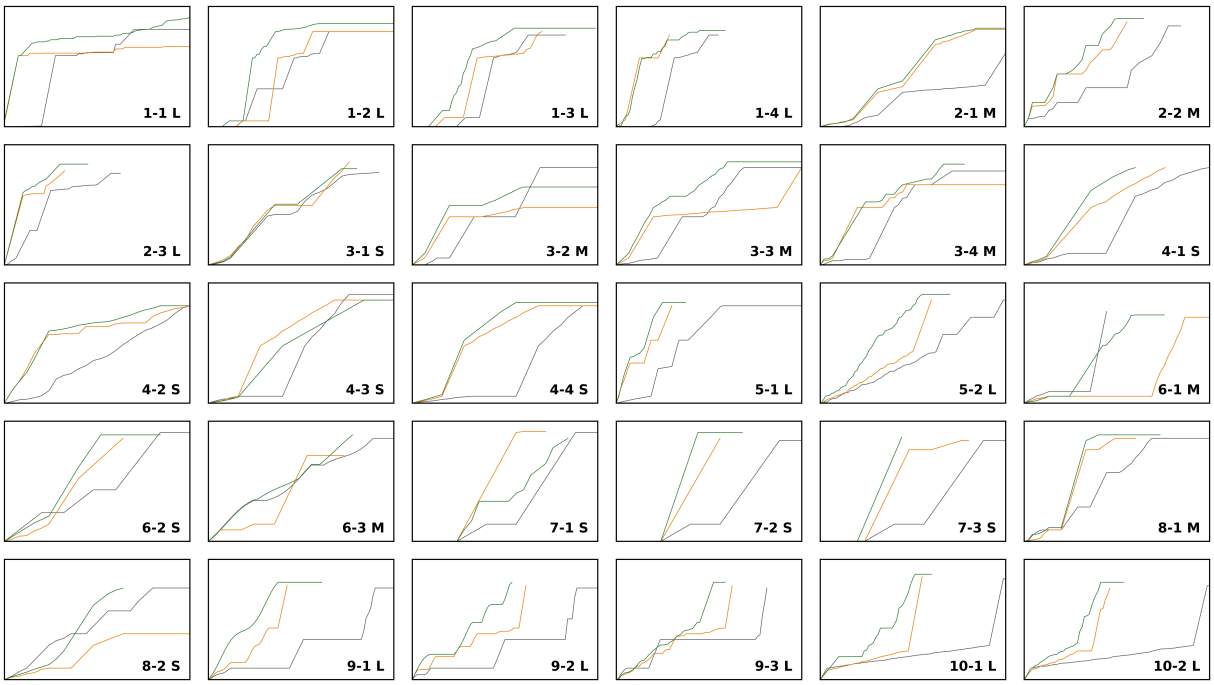

Figure 5: Visualizing trajectories of OntoGuard, DGAP and ORACLE. X: time steps ($0 \rightarrow$ T);Y: scores ($0 \rightarrow 100$). Each figure displays the merged trajectories of testing variations by an agent in each task. Task IDs are shown at the bottom-right, and the ordering follows the task display order in Table 1.

loops, thereby optimizing the effective token usage per successful task completion. Thus, OntoGuard offers a superior trade-off, delivering state-of-the-art performance without imposing prohibitive computational costs.

B.3 Guard Accuracy Analysis

While the primary results demonstrate superior task performance, it is essential to analyze the internal mechanism of OntoGuard to understand its decision boundary. Specifically, we investigate the fidelity of the Constraint-Verification Loop in distinguishing valid actions from inadmissible ones. This section aims to quantify the previously postulated "Conservative Bias," evaluating whether the guard mechanism excessively restricts valid exploratory actions in favor of safety.

Dataset Construction. We curated a binary classification dataset consisting of 400 state-action pairs $(S_t, \hat{a}_t)$ sampled across diverse task environments:

- **Positive Samples (Valid, $N = 200$):** Extracted from Oracle trajectories, these represent ground-truth valid actions that successfully advance the task state toward completion.
- **Negative Samples (Invalid, $N = 200$):** Derived from the failure logs of the ReAct baseline. We specifically selected actions that triggered runtime exceptions or violated physical constraints.

Baselines and Metrics. We compare OntoGuard against an **LLM-Self-Check** baseline, which prompts GPT-4 to predict action validity relying solely on parametric knowledge without external ontological grounding. We assess performance using two key metrics:

- **Precision (Safety Score):** The proportion of actions approved by the guard that are objectively valid. High precision indicates effective filtering of hallucinations and ensures system safety.
- **Recall (Flexibility Score):** The proportion of valid actions that are correctly permitted by the guard. A lower recall score quantifies the "Conservative Bias" (i.e., the rate of false negatives or over-blocking).

Results and Analysis. As detailed in Table 6, the evaluation offers a quantitative explanation for the agent's behavior:

- **Robust Error Filtering (High Precision):** OntoGuard achieves a significantly higher Precision compared to LLM-Self-Check. The baseline, relying on implicit training data, fre-

Tasks	Average number of tokens per action (t/a)					
	SayCan	**ReAct**	**Reflexion**	**SwiftSage**	**DGAP**	**OntoGuard**
1-1	1944.94	1503.60	2632.97	528.17	1580.20	1750.50
1-2	1125.76	1339.39	3066.70	545.34	1420.10	1520.30
1-3	1034.33	1268.23	3307.30	550.17	1320.80	1480.10
1-4	1295.03	1251.45	2439.34	754.05	1290.50	1420.40
2-1	1188.46	1545.03	1988.59	494.52	1620.20	1720.80
2-2	1862.11	1181.88	1596.03	394.29	1250.40	1380.10
2-3	939.17	1358.33	1753.17	574.05	1450.60	1550.50
3-1	1713.64	1846.91	2677.89	807.62	1950.50	2120.30
3-2	1785.01	1754.14	2337.02	823.28	1820.20	1950.40
3-3	1762.13	2441.79	2262.39	220.80	2480.10	2580.60
3-4	1698.85	1195.59	2859.30	287.25	1280.30	1390.50
4-1	411.08	579.70	1053.57	309.14	650.50	750.20
4-2	1332.83	1098.69	1250.37	298.48	1180.40	1280.30
4-3	1155.99	1314.74	2966.82	406.17	1420.20	1500.50
4-4	1126.67	591.15	1003.18	309.71	680.30	780.10
5-1	2323.43	2620.66	5091.49	168.95	2750.50	2880.40
5-2	2646.50	2575.11	5864.93	536.56	2680.20	2820.80
6-1	1454.65	1802.62	2344.90	1388.89	1950.40	2080.50
6-2	2413.99	2763.66	4342.07	402.50	2850.10	2950.30
6-3	1371.50	2860.68	4551.96	6361.79	3000.50	3180.20
7-1	376.50	495.83	813.08	768.63	550.40	650.60
7-2	424.53	478.09	1180.58	772.00	530.20	630.40
7-3	424.73	564.69	1175.35	609.73	620.50	720.80
8-1	1505.39	1155.71	2466.59	249.38	1280.30	1350.60
8-2	3189.80	741.71	2886.09	2479.00	850.50	950.40
9-1	2066.06	2642.79	2652.56	307.30	2750.20	2850.50
9-2	2517.48	3031.95	3606.60	314.19	3150.40	3250.80
9-3	7002.72	7507.00	7785.29	366.06	7650.50	7750.20
10-1	3612.33	4218.44	4822.97	466.21	4350.20	4450.60
10-2	3969.62	5401.37	6724.81	218.00	5550.40	5680.30
Short	1256.98	1047.52	1934.90	716.30	1150.25	1250.40
Medium	1578.51	1742.18	2550.85	1277.52	1850.60	1980.80
Long	2539.78	2893.19	4145.68	444.09	3050.45	3150.15
Overall	1855.84	1971.03	2983.46	757.07	2050.15	2185.42

Table 7: Efficiency comparison: Average number of tokens per action (t/a). OntoGuard maintains a token cost comparable to standard ReAct while achieving significantly higher success rates.

quently hallucinates validity for physically impossible actions. In contrast, OntoGuard's explicit rule verification (O_{env}) successfully intercepts nearly all such violations, serving as a reliable safety layer.

- **Quantifying Conservative Bias (Lower Recall):** OntoGuard exhibits a 9.5% decrease in Recall compared to the baseline. This metric confirms the conservative nature of the mechanism, where approximately 15% of valid actions are blocked due to insufficient support within the extracted rules.

These results validate the underlying design philosophy of OntoGuard. In the context of planning in complex interactive environments, the cost of executing an irreversible invalid action significantly outweighs the cost of blocking a valid one, which results only in a replanning step. Consequently, OntoGuard's trade-off—sacrificing marginal flexibility for superior safety—proves highly advantageous for robust and reliable task completion.

C Results on VirtualHome

To demonstrate the robust generalization capabilities of OntoGuard across diverse complex interactive environments, we extended our evaluation to the **VirtualHome** benchmark.

Baselines and Setup. We compared OntoGuard against four baselines: **ReAct**, **Reflexion**, **SwiftSage**, and **DGAP**. All methods utilize GPT-4 as the backbone. We replicated experiments on 60 tasks for each of the three settings (*In-Distribution*, *NovelScenes*, *NovelTasks*) and averaged results over multiple runs.

Results. Table 4 presents the performance comparison.

- **Generalization across Settings:** OntoGuard consistently outperforms other baselines across all three settings. Notably, in the most challenging *NovelTasks* setting, OntoGuard achieves an SR of **74.50%**, surpassing DGAP (72.17%) and SwiftSage (68.00%).
- **High Execution Validity:** Consistent with our findings in ScienceWorld, OntoGuard achieves superior EXEC scores. This confirms that the behavioral rules effectively capture the valid affordances of household objects, preventing the runtime errors that often hinder ReAct and Reflexion.

D Algorithm of Constraint-Aware Planning

We provide the formal algorithmic procedure for the Explicit Constraint-Aware Planning phase of OntoGuard. Algorithm 1 specifies the full inference loop in which the agent repeatedly:

- Grounds the current observation O_t into a structured state S_t via Φ_{LLM};
- Proposes an action $\hat{a}_t$ using LLMPLANNER conditioned on the interaction history $\mathcal{H}$;
- Verifies $\hat{a}_t$ against both behavioral constraints G_{beh} (history consistency) and environmental constraints G_{env} (state feasibility).

If verification fails, OntoGuard synthesizes interpretable feedback and iteratively refines the action up to K times before executing a validated action or terminating with failure.