

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 AGENT-ENVIRONMENT ALIGNMENT VIA AUTOMATED INTERFACE GENERATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Large language model (LLM) agents have shown impressive reasoning capabilities in interactive decision-making tasks. These agents interact with environment through intermediate interfaces, such as predefined action spaces and interaction rules, which mediate the perception and action. However, mismatches often happen between the internal expectations of the agent regarding the influence of its issued actions and the actual state transitions in the environment, a phenomenon referred to as **agent-environment misalignment**. While prior work has invested substantially in improving agent strategies and environment design, the critical role of the interface still remains underexplored. In this work, we empirically demonstrate that agent-environment misalignment poses a significant bottleneck to agent performance. To mitigate this issue, we propose **ALIGN**, an Auto-Aligned Interface Generation framework that alleviates the misalignment by enriching the interface. Specifically, the ALIGN-generated interface enhances both the static information of the environment and the step-wise observations returned to the agent. Implemented as a lightweight wrapper, this interface achieves the alignment without modifying either the agent logic or the environment code. Experiments across multiple domains including embodied tasks, web navigation and tool-use, achieve consistent performance improvements, with up to a 45.67% success rate improvement observed in ALFWorld. Meanwhile, ALIGN-generated interface can generalize across different agent architectures and LLM backbones without interface regeneration.

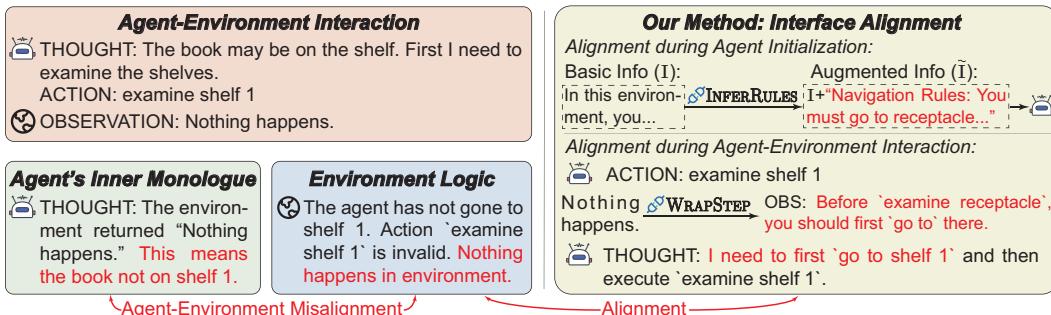


Figure 1: **Illustration of agent-environment misalignment and our proposed solution.** **Left:** The agent and the environment have a misalignment in their interpretation of the same observation, where the agent's understanding of the observation differs from the environment's underlying logic. **Right:** ALIGN bridges this gap via an automatically generated interface comprising two modules, **INFERRULES** and **WRAPSTEP**: (1) During initialization, **INFERRULES** augments the basic information with explicit environment constraints. (2) During interaction, **WRAPSTEP** translates the raw observation into an informative observation that better conveys the environment state transitions.

1 INTRODUCTION

Large Language Model (LLM) agents have demonstrated promising performance in interactive tasks such as embodied tasks (Driess et al., 2023; Lin et al., 2023; Wang et al., 2024a), web

navigation tasks (Chae et al., 2025; He et al., 2024a; Qi et al., 2024), and tool-use tasks (Wang et al., 2024b; Paranjape et al., 2023; Schick et al., 2023). In these tasks, **agents** typically interact with the **environment** through manually designed **interfaces** such as predefined action spaces and interaction rules. While substantial efforts have been devoted to improving agents and environments, comparatively little attention has been paid to the interface between them, leading to a problem we term **agent-environment misalignment**, which significantly impacts the agent performance.

The agent-environment misalignment refers to the discrepancy between the interpretation of the agent to the observation following an action and the underlying logic of the environment. As illustrated in Figure 1 (left), in ALFWorld (Shridhar et al., 2021), issuing *examine receptacle* fails unless the agent first executes “go to receptacle”. Consequently, the environment responds with the observation “Nothing happens.”. At this point, the agent interprets the observation to mean that there is nothing on shelf 1, which is inconsistent with the underlying reason for the environment providing it. To assess the impact of this misalignment, we conduct preliminary experiments, which reveal that simply revising the observation for an invalid “*examine receptacle*” action to “You need to first go to receptacle before you can examine it” increases the success rate of a vanilla Qwen2.5-7B-Instruct (Team, 2024) agent on ALFWorld from 13.4% to 31.3%¹. Such phenomenon suggests that the agent-environment misalignment significantly hinders task success, and can be alleviated by improving interface design. From the perspective of the agent, poorly designed interfaces impose unnecessary cognitive overhead. Furthermore, from an evaluation perspective, inadequate interfaces can obscure an accurate assessment of the true reasoning capabilities of agents. Therefore, we argue that the problem of agent-environment misalignment warrants greater attention.

However, addressing the agent-environment misalignment is challenging. On one hand, current benchmarks primarily focus on advance agent intelligence by constructing increasingly complex and challenging environments (Jimenez et al., 2024; Wang et al., 2025b; Wei et al., 2025; Xie et al., 2024; Zhou et al., 2024a), often overlooking the importance of improving interface design. This oversight extends across multiple domains of interactive tasks, such as, ALFWorld, OSWorld (Xie et al., 2024), and M³ToolEval (Wang et al., 2024b). They all exhibit similar deficiencies: failing to provide agent-parseable observations for environmental constraints violation in embodied tasks, positional inaccuracies in operating system tasks or parameter format errors in multi-turn tool-use tasks, respectively. On the other hand, although some recent work (Agashe et al., 2024; Yang et al., 2024a; Zheng et al., 2024) has begun to consider interface design, these efforts often rely on manual, environment-specific tailoring, which introduces two critical issues: (1) they are highly labor-intensive and (2) whether human-designed interfaces are optimal for agents remains an open question.

Furthermore, in addition to studies that explicitly optimize interface design, it is common in agent-focused research for researchers to manually re-engineer environment interfaces to align with their specific methods. For instance, for the same environment ALFWorld, Zhou et al. (2024b) manually maintains the environment’s state information in JSON format; Ma et al. (2024) introduces a new action *check_valid_actions* to enable agents to retrieve all valid actions; and Chen et al. (2024a) re-implements the environment by wrapping it into a new class *InteractEnv*. However, such ad-hoc customization pose a significant challenge to the field: it compromises the direct comparability across different approaches. Moreover, these modifications are often tailored to the specific methods proposed, making it difficult for the research community to determine whether performance variations stem from novel agent architectures or from the non-standardized, customized interfaces. Therefore, we believe that manually re-engineering environment interfaces is not an optimal approach to alleviating the agent-environment misalignment problem.

Distinct from the aforementioned works, we propose to **automatically generate interfaces for bridging the agent-environment misalignment**. In this work, we introduce **ALIGN** (Auto-Aligned Interface Generation), a framework that automatically generate aligned interfaces for environments. The generated interface consists of two modules: **INFERRULES** and **WRAPSTEP**. The former automatically discovers and provides the agent with static information about environmental rules or internal constraints, facilitating *static alignment*, while the latter enhances the interaction by offering more detailed observations for agent-issuing actions, enabling *dynamic alignment*, as shown in Figure 1 (right). Owing to the powerful reasoning and coding capabilities of current advanced LLMs, we utilize these models to analyze existing agent-environment misalignments and automatically generate the interface. Moreover, we employ LLMs to conduct experimental verification to mitigate

¹Experimental details are provided in Appendix C.1.

108 hallucination issues (Bang et al., 2023; Xu et al., 2024). Specifically, our LLM-based system
 109 autonomously validate both proposed misalignments and generated interface through direct interaction
 110 with the environment, ensuring that identified issues genuinely exist and are properly addressed by
 111 the interface. The generated interface acts as a lightweight wrapper, providing richer context and
 112 explicit constraint hints, enabling different LLM agents to align with the environment directly.

113 To evaluate the effectiveness of ALIGN, we conduct experiments on four representative benchmarks
 114 across three domains: embodied tasks, web navigation, and tool-use tasks. Our results demonstrate
 115 consistent performance improvements across all four benchmarks when using the ALIGN-generated
 116 interface, with notably gains of 45.67% in average success rate on ALFWorld. Moreover, the
 117 performance of GPT-4.1-based agents on ALFWorld are improved from 73.88% to 93.28% with
 118 ALIGN, highlighting the efficiency of our approach in mitigating the agent-environment misalignment
 119 to unleash the agent’s true capabilities.

120 Our key contributions can be summarized as follows:

- 121 • We identify and characterize the **agent-environment misalignment** problem, empirically showing
 122 its prevalence across diverse domains and its role as a key bottleneck to agent performance.
- 123 • We introduce **ALIGN**, the first framework that automatically generates aligned interfaces to
 124 alleviate agent-environment misalignment, without modifying agent logic or environment code.
- 125 • We demonstrate the effectiveness and generalizability of **ALIGN** across three domains, with up
 126 to a 45.67% success rate improvement on ALFWorld.

128 2 RELATED WORK

130 **Agent-environment interface** The agent-environment interface defines how agents interact with the
 131 environment. In reinforcement learning, researchers construct unified interaction interfaces (Bonnet
 132 et al., 2024; Brockman et al., 2016; Kolve et al., 2017; Towers et al., 2024) to standardize the
 133 application and evaluation of different algorithms. With the increasing capability of LLMs to perform
 134 human-like actions (Guo et al., 2024; Liu et al., 2024; Ma et al., 2024), interface design has been
 135 proven to largely influence the performance of LLM-based agents (Xie et al., 2024; Rawles et al.,
 136 2024). SWE-agent (Yang et al., 2024a) proposes agent-computer interfaces for coding agents and
 137 recent efforts aim to improve generalization (Agashe et al., 2024; Qin et al., 2025; Niu et al., 2024) and
 138 enhance interfaces with auxiliary tools (Bula et al., 2025; Gou et al., 2024; Lei et al., 2025; Lu et al.,
 139 2024; Yang et al., 2023a). Nevertheless, current agent-environment interfaces are mostly manually
 140 crafted and tailored for specific environments or agent frameworks, limiting their generalization and
 141 scalability. Therefore, we propose automated interface generation to empower agents with effective,
 142 generalizable and automatic interface alignment.

143 **Methods aligning agents with environments** LLM agents have exhibited strong potential for
 144 real-world interaction and task completion Yao et al. (2023); Shinn et al. (2023); Liu et al. (2024).
 145 Current research in this area can be broadly categorized into training-based methods and training-
 146 free methods. Training-based methods consists of fine-tuning LLMs with expert-level interaction
 147 trajectories Zeng et al. (2024); Chen et al. (2023; 2025); Fu et al. (2025); Chen et al. (2024b) and
 148 enhancing environment-aligned planning and acting via reinforcement learning Bai et al. (2025);
 149 Yang et al. (2024b); Qi et al. (2024); Feng et al. (2024); Zhou et al. (2024c); Wang et al. (2025a).
 150 Though effective, these methods suffer from high computational costs and limited generalization
 151 towards unseen environments. Another approach constructs training-free multi-agent frameworks
 152 for task decomposition and experience accumulation (Chen et al., 2024a; He et al., 2024b; Sun
 153 et al., 2024; Yang et al., 2023b; Zhou et al., 2024b). However, static agent pipelines lack flexibility
 154 and experience injected through prompting often fails to capture environment dynamics and is not
 effectively utilized by LLMs, resulting in insufficient alignment between agents and environments.

155 3 METHOD

156 3.1 PROBLEM FORMULATION

158 **Environment and Agent.** In interactive decision-making tasks, we define the environment \mathcal{E} as a
 159 tuple $(\mathcal{S}, \mathcal{A}, T, F, \mathcal{I})$, where \mathcal{S} denotes the set of all possible states of the environment; \mathcal{A} denotes
 160 the set of actions the agent can invoke; $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ defines how the environment state evolves in

162 response to agent actions; $F : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{O}$ provides textual feedback that reflects the consequences
 163 of the action in the current state, where \mathcal{O} is all possible observations; \mathcal{I} encodes the *environment*
 164 *foundational information description*, a fixed, declarative representation of the environment’s basic
 165 introduction, which is exposed to the agent at initialization.

166 An agent π interacts with environment \mathcal{E} by receiving task description and observations, then
 167 producing actions $a_t \in \mathcal{A}$. The interaction generates trajectory $\tau = [(s_0, a_0, o_0), \dots, (s_T, a_T, o_T)]$,
 168 culminating in task completion feedback.

170 **Formal Definition of Interface.** Existing works typically assume the agent interacts directly with
 171 \mathcal{E} . However, we argue that this interaction is mediated by an **Interface**, denoted as Φ , which acts
 172 as a translation layer between the agent’s cognitive space and the environment’s execution space.
 173 Formally, we define the interface as a tuple of mapping functions: $\Phi = \langle f_{\text{info}}, f_{\text{act}}, f_{\text{obs}} \rangle$ where each
 174 component serves a distinct role:

- 175 • **Information Augmenter** $f_{\text{info}} : \mathcal{I} \rightarrow \tilde{\mathcal{I}}$ exposes implicit environment logic (e.g., constraints,
 176 admissible action sequences) into an explicit descriptive context $\tilde{\mathcal{I}}$ provided at agent initialization.
- 177 • **Action Transducer** $f_{\text{act}} : \mathcal{A}_{\text{agent}} \rightarrow \mathcal{A}_{\text{env}} \cup \{\perp\}$ maps the agent’s output to an executable
 178 environment command. If the output cannot be parsed, it returns an invalid symbol \perp .
- 179 • **Observation Transducer** $f_{\text{obs}} : \mathcal{S} \times \mathcal{A}_{\text{env}} \times \mathcal{O}_{\text{raw}} \rightarrow \mathcal{O}_{\text{agent}}$ transforms the raw feedback \mathcal{O}_{raw}
 180 (from F) into an informative observation $\mathcal{O}_{\text{agent}}$ that better conveys the actual state transitions
 181 and their causes.

183 At each timestep t , the agent receives $\tilde{o}_t \in \mathcal{O}_{\text{agent}}$ (processed by f_{obs}) and generates $a_t \in \mathcal{A}_{\text{agent}}$,
 184 which is then executed as $a_t^{\text{env}} = f_{\text{act}}(a_t) \in \mathcal{A}_{\text{env}}$.

185 **Agent-Environment Misalignment.** We analyze the misalignment problem through the lens of the
 186 interface Φ . Ideally, Φ should be *lossless*, maximizing the mutual information between the agent’s
 187 belief state and the ground-truth environment state. However, manually designed interfaces often
 188 exhibit **Semantic Gaps**, leading to misalignment through two primary mechanisms:

- 190 • **State Aliasing via Lossy Observations (f_{obs}):** A poorly designed f_{obs} may map distinct error
 191 states (e.g., “action invalid due to wrong location” vs. “action invalid due to missing object”) to
 192 the same generic observation (e.g., “Nothing happens.”). This creates state aliasing, preventing
 193 the agent from diagnosing failures and correcting its policy.
- 194 • **Under-specified Constraints (f_{info}):** When critical transitions T rely on preconditions (e.g.,
 195 “open” requires “go to” first) that are not explicitly encoded in $\tilde{\mathcal{I}}$ by f_{info} , the agent operates
 196 under a hallucinated world model where such constraints appear absent.

197 Therefore, we define **Agent-Environment Misalignment** as the discrepancy between the agent’s
 198 expected state transition $s_{t+1}^{\text{expected}}$ (derived from its internal world model based on $\tilde{\mathcal{I}}$ and prior ob-
 199 servations $[\tilde{o}_0, \tilde{o}_1, \dots, \tilde{o}_{t+1}]$) and the actual transition $s_{t+1}^{\text{actual}} = T(s_t, a_t^{\text{env}})$, caused by insufficient
 200 expressiveness of the interface Φ .

202 **Scope of This Work.** While a complete interface theoretically includes all three components, we
 203 observe that misalignment in existing benchmarks primarily stems from information loss in f_{info}
 204 and f_{obs} , rather than from action space incompatibility. Therefore, ALIGN focuses on *automatically*
 205 *optimizing* these two components, treating f_{act} as a fixed identity mapping throughout this work:
 206 $f_{\text{act}}(a) = a$. This design choice allows us to address the core misalignment issues without modifying
 207 the agent’s action generation logic or the environment’s execution layer.

208 3.2 ALIGN OVERVIEW

209 To alleviate agent-environment misalignment, we introduce **ALIGN** (Auto-Aligned Interface
 210 Generation), a framework that automatically generates an optimized interface Φ^* to bridge the
 211 semantic gaps identified in Section 3.1. Specifically, ALIGN focuses on learning improved f_{info} and
 212 f_{obs} functions that minimize information loss during agent-environment interaction.

214 **Interface Instantiation.** As illustrated in Figure 2, ALIGN instantiates the theoretical interface
 215 components through two learnable modules implemented as a lightweight Python wrapper, without
 modifying the environment code or agent logic.

216 **INFERRULES** (implements f_{info}): Transforms raw environment
 217 information \mathcal{I} into augmented information $\tilde{\mathcal{I}}$ that
 218 explicitly exposes interaction rules and constraints. Formally:
 219 $\text{INFERRULES} : (\text{task}, o_0, \mathcal{I}) \rightarrow \tilde{\mathcal{I}}$, where $\tilde{\mathcal{I}}$ includes the
 220 constraints automatically extracted, such as precondition dependencies
 221 or action ordering requirements.

222 **WRAPSTEP** (implements f_{obs}): Intercepts the raw observation
 223 function F and augments its output to resolve state aliasing. Given the current state s_t and agent action a_t , formally:
 224 $\text{WRAPSTEP} : (F, s_t, a_t) \rightarrow \tilde{o}_t$, where \tilde{o}_t encapsulates both
 225 $F(s_t, a_t)$ and additional diagnostic or corrective information.

226 Together, these modules form an intermediate interface wrapper
 227 layer that intercepts and transforms environment information
 228 before it reaches the agent. This design allows the base agent
 229 π to remain unchanged, while still benefiting from contextual
 230 clarity and enriched observation that help avoid misaligned
 231 actions. From the perspective of the agent, interaction now
 232 occurs with an *augmented environment*, denoted as $\tilde{\mathcal{E}} = (\mathcal{S}, \mathcal{A}, T, \tilde{F}, \mathcal{I} \cup \tilde{\mathcal{I}})$, where \tilde{F} is defined
 233 as $\tilde{F}(s_t, a_t) := \text{WRAPSTEP}(F, s_t, a_t)$. This formulation does not alter the internal structure or
 234 transition dynamics of the original environment \mathcal{E} . Instead, it constructs an externally wrapped
 235 interface that provides the agent with a richer and more interpretable view of its operating context.
 236 The interface is denoted as $\Phi := \{\text{INFERRULES}, \text{WRAPSTEP}\}$ in the remainder of this paper.

237 As shown in Figure 3, the ALIGN integrates two cooperative modules, **Analyzer** and **Optimizer**,
 238 to generate aligned interfaces. The framework operates through iterative optimization, with each
 239 iteration comprising three stages: in Stage 1, the Analyzer identifies agent-environment misalignments
 240 by analyzing past interaction trajectories; in Stage 2, the Optimizer generates, validates and refines
 241 a new interface based on the detected misalignments; and in Stage 3, the agent interacts with the
 242 environment wrapped with the newly generated interface, and the failed task trajectories are fed back
 243 to Analyzer for analysis in the next iteration.

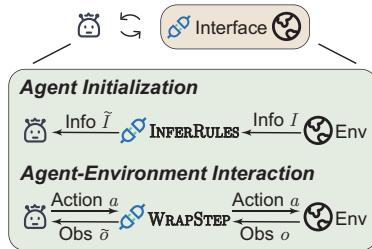
244 3.3 ALIGN FRAMEWORK

245 To automate the generation of interfaces that bridge the agent-environment misalignments, ALIGN
 246 need to solve two key challenges: (1) how to analyze and identify existing agent-environment
 247 misalignments, and (2) how to generate an interface that addresses these misalignments. The overall
 248 algorithm process of ALIGN is outlined in Algorithm 1 in Appendix B.

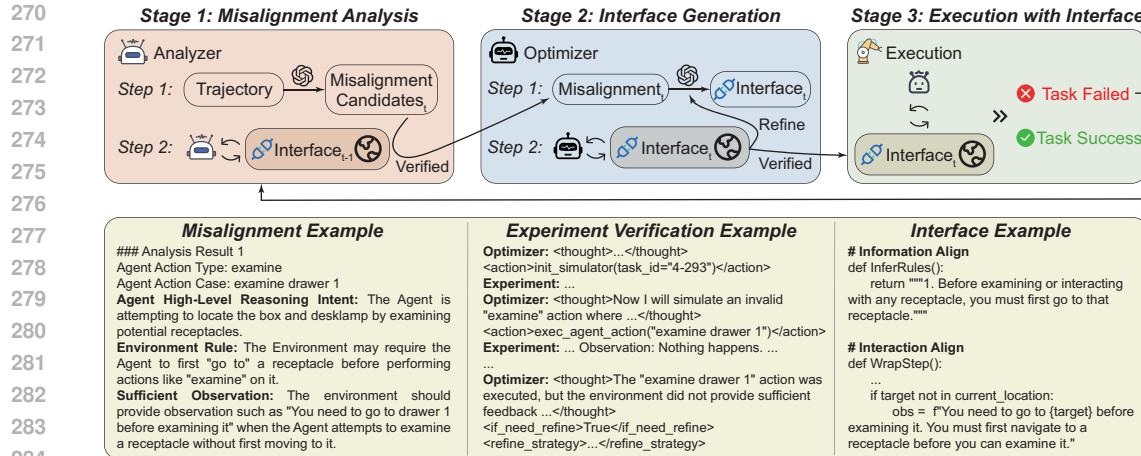
249 **Misalignment Analysis** We represent each agent-environment misalignment using structured text,
 250 as shown in the bottom left of Figure 3. The ‘‘Agent High-Level Reasoning Intent’’ and ‘‘Environment
 251 Rule’’ respectively depict the agent’s expectations of the action and the environment’s observation
 252 rules, together representing a misalignment. The ‘‘Sufficient Observation’’ represents the observation
 253 the environment should provide to resolve the misalignment. To analyze and identify these
 254 misalignments, we designed the Analyzer module based on LLMs. In each iteration, the Analyzer
 255 takes the failed interaction trajectory $\tau^{(i-1)}$ in the previous iteration, the set of currently identified
 256 misalignments \mathcal{M} , and the interface $\Phi^{(i-1)}$ from the previous round as input, and generates a new
 257 set of misalignments $\mathcal{M}^{(i)}$. Detailed prompts for this process are provided in Appendix E.4.

258 **Interface Generation** Once the new set of misalignments $\mathcal{M}^{(i)}$ is identified, we employ the
 259 Optimizer module to generate a new interface. We represent the two modules of the interface,
 260 **INFERRULES** and **WRAPSTEP**, as Python functions, as shown in the bottom right of Figure 3, to
 261 leverage the powerful code generation capabilities of LLMs. In each iteration, the Optimizer takes
 262 the newly identified misalignments $\mathcal{M}^{(i)}$ and the previous interface $\Phi^{(i-1)}$ as input, and generates a
 263 new interface $\Phi^{(i)}$. The detailed prompts for this process are provided in Appendix E.4.

264 **Experimental Verification** Given the hallucination (Bang et al., 2023; Xu et al., 2024) issues
 265 of LLMs, we incorporate an experimental verification procedure. Specifically, after the Analyzer
 266 generates $\mathcal{M}^{(i)}$, it will interact with the environment wrapped by the previous interface $\Phi^{(i-1)}$ to



267 Figure 2: Overview of the ALIGN-generated interface. The interface
 268 wraps the original environment \mathcal{E} to create an augmented environment
 269 $\tilde{\mathcal{E}}$. **INFERRULES** enriches static information ($\mathcal{I} \rightarrow \tilde{\mathcal{I}}$) at
 270 agent initialization, while **WRAPSTEP** augments step-wise observations
 271 ($F \rightarrow \tilde{F}$) during interaction.



285 **Figure 3: ALIGN framework.** In each iteration, ALIGN progresses through three stages. **Stage 1:** the Analyzer identifies potential agent-environment misalignments and validates them through experiments; **Stage 2:** the Optimizer generates a new interface based on the previous interface and identified misalignments, followed by verification and refinement; **Stage 3:** the agent interacts with the updated interface-wrapped environment, with trajectories of failed tasks fed back to the Analyzer for analysis in the next iteration. At the bottom of the figure, examples for misalignment, verification of interface integrity by Optimizer, and the ALIGN-generated interface are provided.

292 validate whether the identified misalignments do indeed exist and can be resolved by the proposed
293 "Sufficient Observation". And after the Optimizer generates the new interface $\Phi^{(i)}$, it will interact with
294 the environment wrapped by this new interface to ensure that the generated interface can resolve the
295 identified misalignments. If the Optimizer finds that the proposed interface is insufficient to address
296 the discovered misalignments, it will provide a refinement strategy and regenerate the interface. This
297 iterative process continues until the interface passes the validation, ensuring that the misalignments
298 identified are appropriately addressed. An example of this process is provided in the bottom center of
299 Figure 3. To facilitate this interaction with the interface-wrapped environment, we designed a set of
300 encapsulated tools for both the Analyzer and Optimizer to use, as described in Appendix E.3.

301 After each iteration, the agent interacts with the environment wrapped by the new generated interface
302 $\Phi^{(i)}$, and trajectories of the failed tasks are returned to Analyzer for further analysis. The algorithm
303 continues iteratively until one of the following holds: (1) the pre-defined maximum number of
304 iterations is reached; (2) no failed trajectories are produced; (3) no new misalignments are identified.

4 EXPERIMENT

4.1 EXPERIMENTAL SETTINGS

310 **Evaluation Protocol** To validate the effectiveness of ALIGN, we assess the performance of various
311 agents in the original, unmodified environments. Subsequently, ALIGN is utilized to generate
312 interfaces for these environments with the respective agents. Afterward, the agents are re-evaluated
313 in the same environments, wrapped with the ALIGN-generated interfaces. **During the interface**
314 **generation and refinement process, only tasks from the training set are used. The interface logic**
315 **is fixed and remains unchanged during testing.** This design enables us to observe and measure the
316 changes in agent performance before and after the interface alignment.

317 **Benchmarks** We conduct experiments on four representative benchmarks across three domains:
318 embodied tasks, web navigation and tool-use. Among them, (1) ALFWorld (Shridhar et al., 2021)
319 focuses on embodied AI agents performing household tasks through textual interactions in simu-
320 lated environments; (2) ScienceWorld (Wang et al., 2022) evaluates the abilities to conduct scientific
321 experiments and apply scientific reasoning of agents in an interactive text-based environment; (3) Web-
322 Shop (Yao et al., 2022) simulates e-commerce scenarios where agents navigate product catalogs
323 and complete purchasing tasks; and (4) M³ToolEval (Wang et al., 2024b) is specifically designed to
324 evaluate agent performance in multi-turn tool-use tasks.

324
 325 **Table 1: Effect of ALIGN-generated interfaces on four benchmarks.** For every agent we report its
 326 score without the interface (w/o ALIGN) and with the interface (w/ ALIGN); the value in parentheses
 327 is the absolute improvement. Metrics are task-success rate (%) for ALFWorld and M³ToolEval, and
 328 scores for ScienceWorld and WebShop.

329 330 331 332 333 334 335 336 337 338 339	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377		340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
			340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	
340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377		340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
			340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	
340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377

Agent Methods To verify the capability of ALIGN to enhance performance across diverse agent architectures, we evaluate five representative methods: (1) Vanilla Agent: Base implementation without specialized prompting strategies; (2) ReAct (Yao et al., 2023): Leveraging the reasoning capabilities of LLMs through interleaved reasoning and action steps; (3) Self-Consistency (Wang et al., 2023): Utilizing probabilistic outputs from LLMs to generate multiple solution paths and select the most consistent one; (4) Self-Refine (Madaan et al., 2023): Employing an iterative self-critic and refine mechanism where agents critique and refine their previous solutions; and (5) Planning Agent: Inspired by RAP Hao et al. (2023), this approach leverages the planning capabilities of LLMs to decompose complex tasks into manageable sub-tasks.

Implementation Details Unless otherwise noted, we use Qwen2.5-7B-Instruct (Team, 2024) as the base model of agents. The Optimizer for interface generation uses Gemini 2.5 Pro (Google, 2025), while other steps the Analyzer and Optimizer use GPT-4.1 (OpenAI, 2025). Implementation details of benchmark task splits and hyper-parameters can be found in Appendix E.

4.2 MAIN RESULTS

Table 1 summarizes the task success rates or scores of five representative agent methods in the environment without (w/o) or with (w/) ALIGN-generated interface. The interfaces generated and the misalignments analyzed can be found in Appendix F and the token consumption analysis can be found in Appendix D. Our empirical investigation yields three principal findings:

(1) ALIGN consistently enhances performance across different domains. All evaluated agent methods demonstrate significant performance improvements when utilizing ALIGN-generated interfaces. Specifically, the five agent methods exhibit mean improvements of 45.67% in task-success rate for ALFWorld, 10.07 points for ScienceWorld, 6.59 points for WebShop, and 6.39% in task-success rate for M³ToolEval. These consistent improvements substantiate the effectiveness of ALIGN.

(2) Agent-environment misalignment is a pervasive phenomenon impeding the agent performance. The observed performance enhancements provide empirical evidence that numerous errors in baseline configurations originate from implicit constraints or under-specified observation, rather than from intrinsic reasoning deficiencies. This finding suggests that when these environmental constraints are explicitly surfaced, agents can execute their intended tasks with substantially improved reliability. Consequently, we posit that agent-environment misalignment is pervasive in interactive decision-making tasks, and addressing this problem is crucial for advancing agent performance.

(3) Alignment between agent and environment can facilitate identification of additional performance-influencing factors. While the Self-Consistency agent achieves a 69.40% success rate in ALFWorld with ALIGN, the performance of Self-Refine agent remains comparatively sub-optimal (40.30%), indicating potential deficiencies in the critic and self-refinement capabilities of the Qwen2.5-7B-Instruct model. These limitations are similarly manifested in the M³ToolEval results. Furthermore, the relatively modest performance improvements in ScienceWorld suggest that Qwen2.5-7B-Instruct may exhibit insufficient scientific causal reasoning capabilities. These

378 observations indicate that properly aligning agent and environment enables more precise isolation
 379 and analysis of other factors influencing agent performance beyond alignment considerations.
 380

381 **4.3 INTERFACE QUALITY ANALYSIS**
 382

383 **Table 2: Impact of the ALIGN-generated interface on consecutive invalid actions.** The metric
 384 reports the fraction (%) of consecutive invalid actions. Lower values indicate more desirable behavior.
 385 Δ denotes the relative reduction with respect to the **w/o ALIGN** setting.

Method	ALFWorld			ScienceWorld		
	w/o ALIGN	w/ ALIGN	Δ	w/o ALIGN	w/ ALIGN	Δ
Vanilla	77.91	26.59	66%	49.12	24.47	50%
ReAct	82.23	38.63	53%	46.61	29.99	36%
Self-Consistency	77.71	15.08	81%	51.10	31.51	38%
Self-Refine	90.38	45.84	49%	58.02	29.48	49%
Planning	74.09	19.14	74%	68.67	20.94	70%
Average	80.46	28.51	65%	54.70	27.28	49%

395 **Influence on Agent Decision** To quantitatively assess the influence of ALIGN-generated interfaces on
 396 agent decision beyond end-task performance metric, we introduce a metric that measures the frequency
 397 of *consecutive invalid actions* by calculating the proportion of the actions that occur within sequences
 398 of two or more consecutive invalid steps. Lower values of this metric indicate: (1) enhanced agent
 399 awareness of implicit preconditions, and (2) improved recovery capability following isolated errors.
 400 Table 2 presents the results for five agent methods implemented on ALFWorld and ScienceWorld.
 401 The empirical results demonstrate a substantial reduction in consecutive invalid actions frequency
 402 across all agent methods when utilizing ALIGN-generated interfaces. Specifically, we observe a mean
 403 reduction of 65% in ALFWorld and 49% in ScienceWorld. These findings provide robust evidence
 404 that ALIGN effectively enriches the information conveyed by the observation, preventing agents from
 405 entering repetitive error cycles, which aligns with the findings documented in Section 4.2.
 406

407 **Comparison with Agentic Systems and Human-designed Interfaces** To further assess the effectiveness
 408 of our automatically generated interfaces, we compare ALIGN against (1) agentic frameworks
 409 equipped with carefully designed reasoning, planning and memory modules and (2) human-designed
 410 interfaces. The experimental setup and results are presented in Appendix C.2. As shown in Table
 411 7, even without bespoke reasoning, planning, or memory modules, a vanilla agent that directly
 412 outputs the next action yields a 6.71 percentage points higher success rate than the best agentic
 413 system when paired with ALIGN-generated interfaces, indicating agent-environment misalignment
 414 substantially constrains the performance of LLM-based agents in interactive tasks. Moreover, using
 415 interfaces automatically generated by ALIGN yields a 13.44 percentage points higher success rate
 416 than human-designed interfaces, further validating the effectiveness of our method (Table 8).
 417

418 **4.4 GENERALIZATION AND GENERALITY STUDY**
 419

420 **Generalization Study** To evaluate the generalization capabilities of ALIGN, we performed the
 421 following two experiments, with the results presented in Table 3 and detailed results in Appendix C.3.

422 (1) ALIGN can generalize to different agent architectures. Panel (a) of Table 3 applies interfaces
 423 generated with the Vanilla agent to the other four agents. Across all four environments every
 424 target agent shows consistent growth, demonstrating that ALIGN captures genuine and previously
 425 unexposed environment constraints. This also reinforces the earlier conclusion that agent-environment
 426 misalignment is a pervasive source of error independent of the agent’s reasoning style.

427 (2) ALIGN can generalize to larger and heterogeneous LLMs. Panel (b) of Table 3 examines whether
 428 an interface generated with Qwen2.5-7B-Instruct can extend to larger or architecturally different
 429 model backbones. The results demonstrate that ALIGN-generated interfaces lead to performance
 430 improvements across base models of varying sizes and architectural families, which indicates that our
 431 method possesses strong generalization capabilities. We also observe that this generalization is not
 432 uniformly robust across all model families and datasets. For instance, Llama3.1-8B-Instruct (Meta,
 433 2025a) shows only a marginal gain of +0.33 on the WebShop benchmark. This limited improvement
 434 may be attributed to the inherent reasoning capabilities of the model itself.

432
 433
 434
 435
 436
 437
 438
 439
Table 3: Generalization of ALIGN-generated interfaces across agents and models. Mean performance improvements from applying ALIGN-generated interfaces in the four environments across different settings. (a) Cross-agent transfer: interfaces generated with a Vanilla agent improve other agent methods. (b) Cross-model transfer: interfaces generated with Qwen2.5-7B-Instruct can generalize to other LLMs.

(a) Interface source: Vanilla agent				
Target method	ALF.	Sci.	Web.	M ³ T.
ReAct	+39.56	+12.29	+7.87	+5.56
Self-Consistency	+51.49	+15.30	+3.00	+8.33
Self-Refine	+34.33	+14.11	+6.17	+4.17
Planning	+41.05	+9.66	+3.26	+11.11

(b) Interface source: Qwen2.5-7B-Instruct agent				
Target LLM	ALF.	Sci.	Web.	M ³ T.
Qwen2.5-14B-Instruct	+17.46	+4.61	+4.66	+6.11
Llama3.1-8B-Instruct	+5.97	+10.27	+0.33	+0.83
Llama3.3-70B-Instruct	+5.82	+3.99	+5.68	+1.67

500
Table 4: Generality of ALIGN. Task success rates (SR) without and with ALIGN-generated interfaces in ALFWorld across two settings. (a) Using GPT-4.1 series models as the base model of agents; (b) Using GiGPO-Qwen2.5-7B-Instruct evaluated under different agent architectures.

(a) GPT-4.1 series		
Base Model	Interface	SR (%)
GPT-4.1-mini	w/o ALIGN	28.36
	w/ ALIGN	64.93 (+36.57)
GPT-4.1	w/o ALIGN	73.88
	w/ ALIGN	93.28 (+19.40)

(b) GiGPO-Qwen2.5-7B-Instruct		
Agent Method	Interface	SR (%)
Vanilla	w/o ALIGN	35.04
	w/ ALIGN	55.97 (+20.93)
Training Config	w/o ALIGN	89.55
	w/ ALIGN	92.54 (+2.99)

453
 454 Taken together, these results show that ALIGN-generated interfaces can generalize (1) across agent
 455 policies and (2) across model scales and families, validating the practicality of ALIGN.

456
Generality Study In this work, our empirical observations indicate that the root cause of agent-
 457 environment misalignment lies in the robustness of the interface itself, making it a universal issue that
 458 affects agents irrespective of the underlying model capability. To further validate this claim and assess
 459 the generality of ALIGN, we conduct experiments on both closed-source LLMs and domain-specific
 460 models trained within the environment. For the former, we use the GPT-4.1 series; for the latter, we
 461 use GiGPO-Qwen2.5-7B-Instruct-ALFWorld (Feng et al., 2025), a state-of-the-art model specifically
 462 post-trained on ALFWorld via reinforcement learning. Detailed experimental setup and full results
 463 are provided in Appendix C.4. As the results reported in Panel (a) of Table 4 shown, applying the
 464 ALIGN-generated interface substantially improves the performance of the GPT-4.1-based agent
 465 from 73.88% to 93.28%. Meanwhile, as the results reported in Panel (b) of Table 4 shown, the
 466 ALIGN-generated interface also enhances the performance of the domain-specific model under both
 467 our Vanilla Agent setting and its original training configuration, from 35.04% to 55.97% and 89.55%
 468 to 92.54%, respectively. These findings demonstrate that the fundamental and pervasive nature of
 469 agent-environment misalignment stems from deficiencies in the environment’s interface rather than
 470 solely from the reasoning limitations of any given model, and further corroborate the generality of
 471 our method across both frontier and domain-specialized models.

4.5 ABLATION STUDY

472
Ablation on Interface Components Starting
 473 from the full ALIGN interface, we conduct
 474 two ablations: (1) w/o INFERRULES and (2)
 475 w/o WRAPSTEP. Table 5 reports the change
 476 relative to the full interface on ALFWorld and
 477 ScienceWorld, with the full results presented
 478 in Appendix C.5. Both ablations reduce
 479 performance: w/o INFERRULES averages -6.72
 480 percentage points on ALFWorld and -2.05
 481 on ScienceWorld, while removing WRAP-
 482 STEP yields a larger decline of -31.79 per-
 483 centage points and -7.84, respectively. These
 484 decreases confirm that each interface compo-
 485 nent contributes meaningfully. Moreover, the

486
Table 5: Ablation on Interface components. Values
 487 represent the change in success rate (%) on ALF-
 488 World and score on ScienceWorld. Negative values
 489 mean performance drops from the *Full* interface.

Method	w/o INFERRULES		w/o WRAPSTEP	
	ALF.	Sci.	ALF.	Sci.
Vanilla	-8.96	-3.35	-33.58	-4.72
ReAct	-5.22	-2.08	-17.91	-6.44
Self-Consistency	-1.49	-2.30	-37.27	-10.59
Self-Refine	-7.46	-1.72	-34.33	-7.59
Planning	-10.45	-0.78	-26.87	-9.86
<i>Mean</i>	<i>-6.72</i>	<i>-2.05</i>	<i>-31.79</i>	<i>-7.84</i>

486 much larger drop w/o WRAPSTEP shows the critical role of fine-grained and enriched observation
 487 during interaction. This also suggests that rich, LLM-friendly observation should be prioritized by
 488 future environment designers when constructing environments.

489 **Ablation on Experimental Verification** To assess whether
 490 the experimental verification procedure in Section 3.3 is in-
 491 dispensable, we ablated it and re-ran ALIGN with the Vanilla
 492 agent on ALFWorld. As a surrogate, we employed a multi-
 493 sampling strategy in each iteration: the Analyzer sampled six
 494 candidate misalignments and selected the one it judged most
 495 accurate; the Optimizer then sampled six candidate interfaces
 496 and likewise chose its top candidate. Within this multi-sampling process, we controlled stochasticity
 497 via decoding temperature; specifically, we evaluated $T \in \{0.2, 0.5\}$ under the prompts listed in
 498 Appendix E.4. The resulting task success rates over three iterations are summarized in Table 6.
 499 Without the ability to execute experiments, task success rate deteriorates sharply, a result of the
 500 limited single-shot reliability of LLMs in both diagnosing misalignments and synthesizing correct
 501 interfaces, which underscores the necessity of the experimental verification procedure design.

502 5 CONCLUSION

503 In this work, we introduce **ALIGN**, a novel framework that automatically generates aligned interfaces
 504 to alleviate the **agent-environment misalignment**, a pervasive and underexplored source of failure
 505 in interactive decision-making tasks. By diagnosing implicit constraints through the Analyzer and
 506 synthesizing aligned interface via the Optimizer, ALIGN improves agent performance significantly on
 507 four representative benchmarks across three domains: embodied tasks, web navigation, and tool-use.
 508 Our results demonstrate that ALIGN not only boosts performance across multiple agent methods but
 509 also generalizes effectively to unseen models and strategies, offering a robust, plug-and-play solution
 510 that decouples agent designs from manual environment-specific alignment. These findings suggest
 511 that automatic interface generation is a promising direction for building more reliable, reusable,
 512 and interpretable LLM-based agents. Future research should explore richer forms of interface
 513 representation, expand evaluations to more domains, and develop finer-grained metrics to quantify
 514 interface quality and its impact on agent behavior.

517 LIMITATIONS AND FUTURE WORK

518 Despite the effectiveness of ALIGN in alleviating agent-environment misalignment, this work
 519 represents an initial exploration into automated interface generation. Several important directions
 520 remain open for further investigation:

521 **Toward more diverse and complex environments.** Our current evaluation focuses on environments
 522 with discrete, text-based action spaces across three domains: embodied tasks, web navigation, and
 523 tool-use. ALIGN’s applicability to more complex settings remains to be explored. Future work could
 524 investigate more complex environments like extending ALIGN to multimodal domains such as GUI
 525 agents, where interfaces must process visual observations alongside textual feedback.

526 **Beyond information and observation augmentation.** As formalized in Section 3.1, a complete
 527 interface comprises three components: f_{info} , f_{obs} , and f_{act} . This work focuses on optimizing f_{info}
 528 and f_{obs} to alleviate the agent-environment misalignment. However, f_{act} also plays a critical role
 529 in interactive tasks. Constraining agents to predefined action spaces may force them to deviate
 530 from their natural output distributions, potentially degrading performance. Automatically generating
 531 and optimizing f_{act} to bridge the gap between an agent’s preferred action representation and the
 532 environment’s expected format remains an important direction.

533 **Metrics for interface quality.** This paper evaluates interface effectiveness primarily through task
 534 success rates and consecutive invalid actions. More comprehensive metrics are needed to quantify
 535 interface influence on agent behavior. Promising directions include: (1) developing finer-grained
 536 behavioral diagnostics measuring specific aspects of agent understanding, such as exploratory actions
 537 or strategy diversity; (2) employing LLM-as-a-Judge (Zheng et al., 2023) paradigms to evaluate
 538 whether interfaces successfully convey environment constraints.

Table 6: Task success rate (%) on ALFWorld across iterations without experimental verification procedure.

Temp.	Iter0	Iter1	Iter2	Iter3
0.2	13.43	22.39	0.00	0.00
0.5	13.43	23.88	1.49	0.75

540 REPRODUCIBILITY STATEMENT
541

542 We present the framework and algorithm design of our method in Section 3 and Appendix B, and
543 the implementation details of the experiments in Appendix C and Appendix E. Meanwhile, the
544 code necessary to reproduce the proposed methods and the main experiments has been provided as
545 supplemental material. The supplemental material also includes the corresponding experimental logs.
546

547 REFERENCES
548

549 Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: an
550 open agentic framework that uses computers like a human. *CoRR*, abs/2410.08164, 2024. doi: 10.
551 48550/ARXIV.2410.08164. URL <https://doi.org/10.48550/arXiv.2410.08164>.

552 Hao Bai, Yifei Zhou, Erran Li Li, Sergey Levine, and Aviral Kumar. Digi-Q: Transforming VLMs to
553 device-control agents via value-based offline RL, 2025.

554 Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia,
555 Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask,
556 multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In
557 Jong C. Park, Yuki Arase, Baotian Hu, Wei Lu, Derry Wijaya, Ayu Purwarianti, and Adila Alfa
558 Krisnadhi (eds.), *Proceedings of the 13th International Joint Conference on Natural Language
559 Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational
560 Linguistics, IJCNLP 2023 -Volume 1: Long Papers, Nusa Dua, Bali, November 1 - 4, 2023*, pp. 675–
561 718. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.IJCNLP-MAIN.45.
562 URL <https://doi.org/10.18653/v1/2023.ijcnlp-main.45>.

563 Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth,
564 Vincent Coyette, Laurence Illing Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub,
565 Matthew Macfarlane, Andries P. Smit, Nathan Grinsztajn, Raphaël Boige, Cemlyn N. Waters,
566 Mohamed A. Mimouni, Ulrich A. Mbou Sob, Ruan de Kock, Siddarth Singh, Daniel Furelos-
567 Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable
568 reinforcement learning environments in JAX. In *The Twelfth International Conference on Learning
569 Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL
570 <https://openreview.net/forum?id=C4CxQmp9wc>.

571 Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and
572 Wojciech Zaremba. OpenAI gym. *CoRR*, abs/1606.01540, 2016. URL <http://arxiv.org/abs/1606.01540>.

573 Timothy Bula, Saurabh Pujar, Luca Buratti, Mihaela Bornea, and Avirup Sil. SeaView: Software
574 engineering agent visual interface for enhanced workflow. *arXiv preprint arXiv:2504.08696*, 2025.

575 Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim,
576 Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and
577 leveraging environment dynamics in web navigation. In *The Thirteenth International Conference
578 on Learning Representations*, 2025. URL <https://openreview.net/forum?id=moWiYJuSGF>.

579 Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao.
580 FireAct: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023. doi: 10.48550/ARXIV.
581 2310.05915. URL <https://doi.org/10.48550/arXiv.2310.05915>.

582 Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual:
583 Constructing instruction manuals by LLM agents via interactive environmental learning.
584 In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M.
585 Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/0142921fad7ef9192bd87229cdafa9d4-Abstract-Conference.html.

594 Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen,
 595 and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large
 596 language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the*
 597 *Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting,*
 598 *August 11-16, 2024*, pp. 9354–9366. Association for Computational Linguistics, 2024b. doi:
 599 10.18653/V1/2024.FINDINGS-ACL.557. URL <https://doi.org/10.18653/v1/2024.findings-acl.557>.

600

601 Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. ATLaS: Agent tuning
 602 via learning critical steps. *CoRR*, abs/2503.02197, 2025. doi: 10.48550/ARXIV.2503.02197. URL
 603 <https://doi.org/10.48550/arXiv.2503.02197>.

604

605 Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter,
 606 Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar,
 607 Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc
 608 Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied
 609 multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara
 610 Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine*
 611 *Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of*
 612 *Machine Learning Research*, pp. 8469–8488. PMLR, 2023. URL <https://proceedings.mlr.press/v202/driess23a.html>.

613

614 Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for
 615 LLM agent training. *CoRR*, abs/2505.10978, 2025. doi: 10.48550/ARXIV.2505.10978. URL
 616 <https://doi.org/10.48550/arXiv.2505.10978>.

617

618 Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li.
 619 AGILE: A novel reinforcement learning framework of LLM agents. In Amir Globersons, Lester
 620 Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang
 621 (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural*
 622 *Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 -*
 623 *15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/097c514162ea7126d40671d23e12f51b-Abstract-Conference.html.

624

625 Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang,
 626 Jingang Wang, Xunliang Cai, and Weiran Xu. AgentRefine: Enhancing agent generalization
 627 through refinement tuning. *CoRR*, abs/2501.01702, 2025. doi: 10.48550/ARXIV.2501.01702.
 628 URL <https://doi.org/10.48550/arXiv.2501.01702>.

629

630 Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li,
 631 and Yongfeng Zhang. Openagi: When LLM meets domain experts. In Alice Oh, Tris-
 632 tan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.),
 633 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural*
 634 *Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December*
 635 *10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1190733f217404edc8a7f4e15a57f301-Abstract-Datasets_and_Benchmarks.html.

636

637 Google. Gemini 2.5 Pro preview model card, 2025. URL <https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro-preview.pdf>.

638

639 Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and
 640 Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents.
 641 *CoRR*, abs/2410.05243, 2024. doi: 10.48550/ARXIV.2410.05243. URL <https://doi.org/10.48550/arXiv.2410.05243>.

642

643

644 Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf
 645 Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress
 646 and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial*
 647 *Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pp. 8048–8057. ijcai.org, 2024.
 648 URL <https://www.ijcai.org/proceedings/2024/890>.

648 Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhitong
 649 Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan
 650 Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in
 651 Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 8154–8173.
 652 Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.507.
 653 URL <https://doi.org/10.18653/v1/2023.emnlp-main.507>.

654 Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan,
 655 and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In
 656 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting
 657 of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok,
 658 Thailand, August 11-16, 2024*, pp. 6864–6890. Association for Computational Linguistics, 2024a.
 659 doi: 10.18653/V1/2024.ACL-LONG.371. URL <https://doi.org/10.18653/v1/2024.acl-long.371>.

660 Kaiyu He, Mian Zhang, Shuo Yan, Peilin Wu, and Zhiyu Zoey Chen. IDEA: Enhancing the rule
 661 learning ability of large language model agent through induction, deduction, and abduction, 2024b.
 662 URL <https://arxiv.org/abs/2408.10455>.

663 Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R.
 664 Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The
 665 Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria,
 666 May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=VTF8yNQM66>.

667 Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-
 668 THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017. URL [http://arxiv.org/abs/1712.05474](https://arxiv.org/abs/1712.05474).

669 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.
 670 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
 671 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating
 672 Systems Principles*, 2023.

673 Xuanyu Lei, Zonghan Yang, Xinrui Chen, Peng Li, and Yang Liu. Scaffolding coordinates to
 674 promote vision-language coordination in large multi-modal models. In Owen Rambow, Leo
 675 Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert
 676 (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING
 677 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 2886–2903. Association for Computational
 678 Linguistics, 2025. URL <https://aclanthology.org/2025.coling-main.195/>.

679 Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula,
 680 Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. SwiftSage: A generative
 681 agent with fast and slow thinking for complex interactive tasks. In Alice Oh, Tristan
 682 Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in
 683 Neural Information Processing Systems 36: Annual Conference on Neural Information
 684 Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 -
 685 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/4b0eea69deea512c9e2c469187643dc2-Abstract-Conference.html.

686 Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding,
 687 Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui
 688 Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang.
 689 AgentBench: Evaluating llms as agents. In *The Twelfth International Conference on Learning
 690 Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL
 691 <https://openreview.net/forum?id=zAdUB0aCTQ>.

692 Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. OmniParser for pure vision
 693 based GUI agent. *CoRR*, abs/2408.00203, 2024. doi: 10.48550/ARXIV.2408.00203. URL
 694 <https://doi.org/10.48550/arXiv.2408.00203>.

702 Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong
 703 Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board
 704 of multi-turn LLM agents. In Amir Globersons, Lester Mackey, Danielle Belgrave,
 705 Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances
 706 in Neural Information Processing Systems 38: Annual Conference on Neural Infor-
 707 mation Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 -
 708 15, 2024*. 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

711 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegr-
 712 effe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bod-
 713 hisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and
 714 Peter Clark. Self-Refine: Iterative refinement with self-feedback. In Alice Oh, Tris-
 715 tan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Ad-
 716 vances in Neural Information Processing Systems 36: Annual Conference on Neural Infor-
 717 mation Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 -
 718 16, 2023*. 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

720 Meta. Model cards and prompt formats Llama 3.1, 2025a. URL https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_1/.

721 Meta. Model cards and prompt formats Llama 3.3, 2025b. URL https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/.

722 Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and
 723 Qi Wang. ScreenAgent: A vision language model-driven computer control agent. In *Proceedings
 724 of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju,
 725 South Korea, August 3-9, 2024*, pp. 6433–6441. ijcai.org, 2024. URL <https://www.ijcai.org/proceedings/2024/711>.

726 OpenAI. Introducing GPT-4.1 in the api, 2025. URL <https://openai.com/index/gpt-4-1/>.

727 Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer,
 728 and Marco Túlio Ribeiro. ART: automatic multi-step reasoning and tool-use for large language
 729 models. *CoRR*, abs/2303.09014, 2023. doi: 10.48550/ARXIV.2303.09014. URL <https://doi.org/10.48550/arXiv.2303.09014>.

730 Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and
 731 Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer,
 732 Jeff Han, Jürgen Steimle, and Nathalie Henry Riche (eds.), *Proceedings of the 36th Annual ACM
 733 Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA,
 734 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763.
 735 URL <https://doi.org/10.1145/3586183.3606763>.

736 Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue
 737 Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL:
 738 Training LLM web agents via self-evolving online curriculum reinforcement learning. *CoRR*,
 739 abs/2411.02337, 2024. doi: 10.48550/ARXIV.2411.02337. URL <https://doi.org/10.48550/arXiv.2411.02337>.

740 Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang,
 741 Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu
 742 Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei
 743 Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang,
 744 Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: pioneering automated GUI
 745 interaction with native agents. *CoRR*, abs/2501.12326, 2025. doi: 10.48550/ARXIV.2501.12326.
 746 URL <https://doi.org/10.48550/arXiv.2501.12326>.

756 Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth
 757 Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert
 758 Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. *AndroidWorld: A dynamic*
 759 *benchmarking environment for autonomous agents.* *CoRR*, abs/2405.14573, 2024. doi: 10.48550/
 760 ARXIV.2405.14573. URL <https://doi.org/10.48550/arXiv.2405.14573>.

761 Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Ham-
 762 bro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. *Toolformer: Lan-*
 763 *guage models can teach themselves to use tools.* In Alice Oh, Tristan Naumann,
 764 Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in*
 765 *Neural Information Processing Systems 36: Annual Conference on Neural Information*
 766 *Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16,*
 767 *2023, 2023.* URL http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

768

770 Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. *Agentsquare: Automatic*
 771 *LLM agent search in modular design space.* In *The Thirteenth International Conference on*
 772 *Learning Representations, ICLR 2025, Singapore, April 24-28, 2025.* OpenReview.net, 2025. URL
 773 <https://openreview.net/forum?id=mPdmDYIQ7f>.

774 Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. *Re-*
 775 *flexion: language agents with verbal reinforcement learning.* In Alice Oh, Tristan Nau-
 776 *mann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), Advances*
 777 *in Neural Information Processing Systems 36: Annual Conference on Neural Informa-*
 778 *tion Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16,*
 779 *2023, 2023.* URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

780

781 Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J.
 782 *Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning.* In
 783 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria,*
 784 *May 3-7, 2021.* OpenReview.net, 2021. URL <https://openreview.net/forum?id=0IOX0YcCdTn>.

785

786

787 Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu,
 788 Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao,
 789 and Zhiyong Wu. *OS-Genesis: Automating GUI agent trajectory construction via reverse task*
 790 *synthesis.* *CoRR*, abs/2412.19723, 2024. doi: 10.48550/ARXIV.2412.19723. URL <https://doi.org/10.48550/arXiv.2412.19723>.

791

792 Qwen Team. *Qwen2.5: A party of foundation models*, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.

793

794

795 Mark Towers, Ariel Kwiatkowski, Jordan K. Terry, John U. Balis, Gianluca De Cola, Tristan Deleu,
 796 Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea
 797 Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. *Gymnasium: A standard*
 798 *interface for reinforcement learning environments.* *CoRR*, abs/2407.17032, 2024. doi: 10.48550/
 799 ARXIV.2407.17032. URL <https://doi.org/10.48550/arXiv.2407.17032>.

800

801 Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan,
 802 and Anima Anandkumar. *Voyager: An open-ended embodied agent with large language models.*
 803 *Trans. Mach. Learn. Res.*, 2024, 2024a. URL <https://openreview.net/forum?id=ehfRiF0R3a>.

804

805 Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. *ScienceWorld:*
 806 *Is your agent smarter than a 5th grader?* In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang
 807 (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing,*
 808 *EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 11279–11298.
 809 Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.775.
 810 URL <https://doi.org/10.18653/v1/2022.emnlp-main.775>.

810 Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji.
 811 Executable code actions elicit better LLM agents. In *Forty-first International Conference on*
 812 *Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b. URL
 813 <https://openreview.net/forum?id=jJ9BoXAfFa>.

814

815 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha
 816 Chowdhery, and Denny Zhou. Self-Consistency improves chain of thought reasoning in language
 817 models. In *The Eleventh International Conference on Learning Representations, ICLR 2023,*
 818 *Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.

819

820 Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu,
 821 Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun
 822 Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution
 823 in ILM agents via multi-turn reinforcement learning, 2025a. URL <https://arxiv.org/abs/2504.20073>.

824

825 Ziyue Wang, Yurui Dong, Fuwen Luo, Minyuan Ruan, Zhili Cheng, Chi Chen, Peng Li, and Yang
 826 Liu. How do multimodal large language models handle complex multimodal reasoning? placing
 827 them in an extensible escape game, 2025b. URL <https://arxiv.org/abs/2503.10042>.

828

829 Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won
 830 Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet
 831 challenging benchmark for browsing agents, 2025. URL <https://arxiv.org/abs/2504.12516>.

832

833 Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang
 834 He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language
 835 models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna,*
 836 *Austria, May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=OqTMUPuLuC>.

837

838 Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing
 839 Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio
 840 Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal
 841 agents for open-ended tasks in real computer environments. In Amir Globersons, Lester
 842 Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng
 843 Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference*
 844 *on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada,*
 845 *December 10 - 15, 2024*. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

846

847

848 Ziwei Xu, Sanjay Jain, and Mohan S. Kankanhalli. Hallucination is inevitable: An innate limitation
 849 of large language models. *CoRR*, abs/2401.11817, 2024. doi: 10.48550/ARXIV.2401.11817. URL
 850 <https://doi.org/10.48550/arXiv.2401.11817>.

851

852 Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-Mark
 853 prompting unleashes extraordinary visual grounding in GPT-4V. *CoRR*, abs/2310.11441, 2023a.
 854 doi: 10.48550/ARXIV.2310.11441. URL <https://doi.org/10.48550/arXiv.2310.11441>.

855

856 John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik
 857 Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated
 858 software engineering. In Amir Globersons, Lester Mackey, Danielle Belgrave, An-
 859 gela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in*
 860 *Neural Information Processing Systems 38: Annual Conference on Neural Information*
 861 *Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15,*
 862 *2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

863

864 Zeyuan Yang, Peng Li, and Yang Liu. Failures pave the way: Enhancing large language models
 865 through tuning-free rule accumulation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.),
 866 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing,
 867 EMNLP 2023, Singapore, December 6-10, 2023*, pp. 1751–1777. Association for Computational
 868 Linguistics, 2023b. doi: 10.18653/V1/2023.EMNLP-MAIN.109. URL <https://doi.org/10.18653/v1/2023.emnlp-main.109>.

870 Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. ReAct meets ActRe: When
 871 language agents enjoy training data autonomy. *CoRR*, abs/2403.14589, 2024b. doi: 10.48550/
 872 ARXIV.2403.14589. URL <https://doi.org/10.48550/arXiv.2403.14589>.

873 Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scal-
 874 able real-world web interaction with grounded language agents. In Sammi Koyejo, S. Mo-
 875 hamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural
 876 Information Processing Systems 35: Annual Conference on Neural Information Process-
 877 ing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9,
 878 2022*. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

879 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan
 880 Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh Interna-
 881 tional Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.
 882 OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

883 Junchi Yu, Ran He, and Zhitao Ying. Thought propagation: an analogical approach to complex
 884 reasoning with large language models. In *The Twelfth International Conference on Learning
 885 Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL
 886 <https://openreview.net/forum?id=SBoRhRCzM3>.

887 Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning:
 888 Enabling generalized agent abilities for LLMs. In Lun-Wei Ku, Andre Martins, and Vivek
 889 Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok,
 890 Thailand and virtual meeting, August 11-16, 2024*, pp. 3053–3077. Association for Computational
 891 Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.181. URL <https://doi.org/10.18653/v1/2024.findings-acl.181>.

892 Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web
 893 agent, if grounded. In *Forty-first International Conference on Machine Learning, ICML 2024,
 894 Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=piecdKJ2D1B>.

895 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao
 896 Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez,
 897 and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In Alice Oh,
 898 Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.),
 899 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural
 900 Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December
 901 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

902 Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,
 903 Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic
 904 web environment for building autonomous agents. In *The Twelfth International Conference on
 905 Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a.
 906 URL <https://openreview.net/forum?id=oKn9c6ytLx>.

907 Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang.
 908 WALL-E: world alignment by rule learning improves world model-based LLM agents. *CoRR*,
 909 abs/2410.07484, 2024b. doi: 10.48550/ARXIV.2410.07484. URL <https://doi.org/10.48550/arXiv.2410.07484>.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024c. URL <https://openreview.net/forum?id=b6rA0kAHT1>.

A LLM USAGE STATEMENT

Throughout the completion of this work, the LLM was employed solely for the purpose of refining sentences and improving grammatical accuracy during the manuscript writing process.

B FORMALIZATION OF THE ALIGN ALGORITHM

The formalization of the ALIGN algorithm is outlined in Algorithm 1.

Algorithm 1 ALIGN: Auto-Aligned Interface Generation

```

Require: Environment  $\mathcal{E}$ , Agent  $\pi$ , Task training set  $\mathcal{T}_{\text{train}}$ , Maximum iterations  $K$ 
1: Initialize misalignment set  $\mathcal{M} \leftarrow \emptyset$ , interface  $\Phi^{(0)} \leftarrow \{\text{INFERRULES}^{(0)}, \text{WRAPSTEP}^{(0)}\}$ , where
    $\text{INFERRULES}^{(0)}$  and  $\text{WRAPSTEP}^{(0)}$  are identity functions
2: for  $i = 1, 2, \dots, K$  do
3:    $\tilde{\mathcal{E}}^{(i-1)} \leftarrow$  Environment  $\mathcal{E}$  wrapped with interface  $\Phi^{(i-1)}$ 
4:    $\tau_{\text{fail}}^{(i-1)} \leftarrow$  Failed trajectories from agent  $\pi$  interacting with  $\tilde{\mathcal{E}}^{(i-1)}$  on  $\mathcal{T}_{\text{train}}$ 
5:   if  $\tau_{\text{fail}}^{(i-1)} = \emptyset$  then
6:     break ▷ No more failures in the training set
7:   end if
   // Stage 1: Misalignment Analysis
8:    $\mathcal{M}^{(i)} \leftarrow \text{Analyzer}(\tau_{\text{fail}}^{(i-1)}, \mathcal{M}, \Phi^{(i-1)})$ 
9:   if  $\mathcal{M}^{(i)} = \emptyset$  then
10:    break ▷ No new misalignments identified
11:   end if
12:    $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}^{(i)}$ 
   // Stage 2: Interface Generation
13:    $\Phi^{(i)} \leftarrow \text{Optimizer}(\mathcal{M}^{(i)}, \Phi^{(i-1)})$ 
14: end for
15: return final interface  $\Phi^{(i)}$ 

```

C SUPPLEMENTARY EXPERIMENTAL SETUP AND DETAILED RESULTS

C.1 PRELIMINARY EXPERIMENTS

To preliminarily assess the significance of agent-environment misalignment, we conducted exploratory experiments on the ALFWorld. We employed the vanilla Qwen2.5-7B-Instruct agent with a temperature setting of 0.0. The deployment protocol, prompt template, followed the same configuration described in Appendix E and Appendix E.4.

During the experiments, we introduced a minor modification to the environment: if the agent issued the action *examine receptacle* and the environment returned the default observation ‘‘Nothing happens.’’, we replaced it with ‘‘You need to first go to receptacle before you can examine it.’’ This simple adjustment increased the agent’s task success rate from 13.4% to 31.3%.

C.2 INTERFACE QUALITY ANALYSIS EXPERIMENTS

To further assess the quality of the ALIGN-generated interface, we first compare our method with human-designed agentic system. Our experiments are conducted on ALFWorld using the AgentSquare (Shang et al., 2025) framework. To maximize the advantages of the agentic system, we adopt gpt-4.1-2025-04-14 as the base model, select OPENAGI (Ge et al., 2023) for the planning

972
973
974
Table 7: Experimental results of the comparison between agents with ALIGN-generated interface
and agents with human-designed reasoning, planning and memory module.

Agent Framework	Interface	Memory Module	pick and place	pick clean and place	pick heat and place	pick cool and place	look at / examine in light	pick two obj and place	Success Rate (%)
AgentSquare	/	Generative	95.83	87.10	69.57	95.24	83.33	88.24	86.57
AgentSquare	/	DiLu	91.67	87.10	52.17	95.24	83.33	70.59	80.60
AgentSquare	/	TP	87.50	51.61	4.35	61.90	27.78	47.06	47.76
AgentSquare	/	VOYAGER	95.83	83.87	52.17	90.48	83.33	64.71	79.10
Vanilla Agent	w/o ALIGN	/	100.00	93.55	13.04	71.43	61.11	100.00	73.88
Vanilla Agent	w/ ALIGN	/	100.00	100.00	78.26	100.00	77.78	100.00	93.28

985
986
987
988
989
module, Self-Refine (Madaan et al., 2023) for the reasoning module, and evaluate memory using Generative (Park et al., 2023), DiLu (Wen et al., 2024), TP (Yu et al., 2024), and VOYAGER (Wang et al., 2024a). For our approach, we employ a gpt-4.1-2025-04-14-based vanilla agent, where the interface is generated with the gpt-4.1-2025-04-14-mini-based vanilla agent by ALIGN (the experimental setup is same as Appendix C.4). The results are reported in Table 7.

990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
Furthermore, we compare the ALIGN-generated interface against the human-designed interface. We adopt the following configurations for comparison with our method: (1) **Few-shot**: Settings identical to those in the ReAct (Yao et al., 2023); (2) **Valid Actions**: Supplying the agent with all valid actions at every response turn, analogous to the `check_valid_actions` configuration in Agent-Board (Ma et al., 2024); (3) **Human-Designed Interface**: Interfaces manually crafted by Ph.D. students after inspecting ALFWorld experiments, examining trajectories, and running experiments themselves. The design logic includes: executing “go to” prior to each action; automatically checking object labels; converting “put” to “move” when appropriate; returning the action space upon invalid actions; issuing reminders when “clean with” is applied to non-sinkbasin objects; and other hand-engineered rules. We use Qwen2.5-7B-Instruct as the base model. Experimental results are reported in Table 8.

1006 C.3 GENERALIZATION STUDY EXPERIMENTS

1008
1009
1010
Detailed results of the generalization study are provided for the cross-method experiments in Table 9 and for the cross-model experiments in Tables 10, 11, and 12.

1011
1012
1013
Table 9: **Generalization of ALIGN-generated interfaces generated with Vanilla agents to other agent methods.** For each agent we report its score without the interface (w/o ALIGN) and with the interface (w/ ALIGN); the value in parentheses is the *absolute* improvement.

Base Method: Vanilla		Embodied		Web	Tool-use
Method	Interface	ALFWorld	ScienceWorld	WebShop	M ³ ToolEval
ReAct	w/o ALIGN	19.40	20.03	37.20	9.72
	w/ ALIGN	58.96 (+39.56)	32.32 (+12.29)	45.07 (+7.87)	15.28 (+5.56)
Self-Consistency	w/o ALIGN	11.94	14.07	56.23	11.11
	w/ ALIGN	63.43 (+51.49)	29.37 (+15.30)	59.23 (+3.00)	19.44 (+8.33)
Self-Refine	w/o ALIGN	3.73	14.87	44.80	5.55
	w/ ALIGN	38.06 (+34.33)	28.98 (+14.11)	50.97 (+6.17)	9.72 (+4.17)
Planning	w/o ALIGN	9.70	17.13	46.95	11.11
	w/ ALIGN	50.75 (+41.05)	26.79 (+9.66)	50.21 (+3.26)	22.22 (+11.11)

571
572
573
574
575
Table 8: Experimental results of the comparison between agents with ALIGN-generated interface and agents with human-designed interfaces.

Experimental Setting	Success Rate (%)
w/o Interface	13.43
Few-shot	44.78
Valid Actions	44.03
Human Designed Interface	47.01
ALIGN-generated Interface	60.45

576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
75100
75101
75102
75103
75104
75105
75106
75107
75108
75109
75110
75111
75112
75113
75114
75115
75116
75117
75118
75119
75120
75121
75122
75123
75124
75125
75126
75127
75128
75129
75130
75131
75132
75133
75134
75135
75136
75137
75138
75139
75140
75141
75142
75143
75144
75145
75146
75147
75148
75149
75150
75151
75152
75153
75154
75155
75156
75157
75158
75159
75160
75161
75162
75163
75164
75165
75166
75167
75168
75169
75170
75171
75172
75173
75174
75175
75176
75177
75178
75179
75180
75181
75182
75183
75184
75185
75186
75187
75188
75189
75190
75191
75192
75193
75194
75195
75196
75197
75198
75199
75200
75201
75202
75203
75204
75205
75206
75207
75208
75209
75210
75211
75212
75213
75214
75215
75216
75217
75218
75219
75220
75221
75222
75223
75224
75225
75226
75227
75228
75229
75230
75231
75232
75233
75234
75235
75236
75237
75238
75239
75240
75241
75242
75243
75244
75245
75246
75247
75248
75249
75250
75251
75252
75253
75254
75255
75256
75257
75258
75259
75260
75261
75262
75263
75264
75265
75266
75267
75268
75269
75270
75271
75272
75273
75274
75275
75276
75277
75278
75279
75280
75281
75282
75283
75284
75285
75286
75287
75288
75289
75290
75291
75292
75293
75294
75295
75296
75297
75298
75299
75300
75301
75302
75303
75304
75305
75306
75307
75308
75309
75310
75311
75312
75313
75314
75315
75316
75317
75318
75319
75320
75321
75322
75323
75324
75325
75326
75327
75328
75329
75330
75331
75332
75333
75334
75335
75336
75337
75338
75339
75340
75341
75342
75343
75344
75345
75346
75347
75348
75349
75350
75351
75352
75353
75354
75355
75356
75357
75358
75359
75360
75361
75362
75363
75364
75365
75366
75367
75368
75369
75370
75371
75372
75373
75374
75375
75376
75377
75378
75379
75380
75381
75382
75383
75384
75385
75386
75387
75388
75389
75390
75391
75392
75393
75394
75395
75396
75397
75398
75399
75400
75401
75402
75403
75404
75405
75406
75407
75408
75409
75410
75411
75412
75413
75414
75415
75416
75417
75418
75419
75420
75421
75422
75423
75424
75425
75426
75427
75428
75429
75430
75431
75432
75433
75434
75435
75436
75437
75438
75439
75440
75441
75442
75443
75444
75445
75446
75447
75448
75449
75450
75451
75452
75453
75454
75455
75456
75457
75458
75459
75460
75461
75462
75463
75464
75465
75466
75467
75468
75469
75470
75471
75472
75473
75474
75475
75476
75477
75478
75479
75480
75481
75482
75483
75484
75485
75486
75487
75488
75489
75490
75491
75492
75493
75494
75495
75496
75497
75498
75499
75500
75501
75502
75503
75504
75505
75506
75507
75508
75509
75510
75511
75512
75513
75514
75515
75516
75517
75518
75519
75520
75521
75522
75523
75524
75525
75526
75527
75528
75529
75530
75531
75532
75533
75534
75535
75536
75537
75538
75539
75540
75541
75542
75543
75544
75545
75546
75547
75548
75549
75550
75551
75552
75553
75554
75555
75556
75557
75558
75559
75560
75561
75562
75563
75564
75565
75566
75567
75568
75569
75570
75571
75572
75573
75574
75575
75576
75577
75578
75579
75580
75581
75582
75583
75584
75585
75586
75587
75588
75589
75590
75591
75592
75593
75594
75595
75596
75597
75598
75599
75600
75601
75602
75603
75604
75605
75606
75607
75608
75609
75610
75611
75612
75613
75614
75615
75616
75617
75618
75619
75620
75621
75622
75623
75624
75625
75626
75627
75628
75629
75630
75631
75632
75633
75634
75635
75636
75637
75638
75639
75640
75641
75642
75643
75644
75645
75646
75647
75648
75649
75650
75651
75652
75653
75654
75655
75656
75657
75658
75659
75660
75661
75662
75663
75664
75665
75666
75667
75668
75669
75670
75671
75672
75673
75674
75675
75676
75677
75678
75679
75680
75681
75682
75683
75684
75685
75686
75687
75688
75689
75690
75691
75692
75693
75694
75695
75696
75697
75698
75699
75700
75701
75702
75703
75704
75705
75706
75707
75708
75709
75710
75711
75712
75713
75714
75715
75716
75717
75718
75719
75720
75721
75722
75723
75724
75725
75726
75727
75728
75729
75730
75731
75732
75733
75734
75735
75736
75737
75738
75739
75740
75741
75742
75743
75744
75745
75746
75747
75748
75749
75750
75751
75752
75753
75754
75755
75756
75757
75758
75759
75760
75761
75762
75763
75764
75765
75766
75767
75768
75769
75770
75771
75772
75773
75774
75775
75776
75777
75778
75779
75780
75781
75782
75783
75784
75785
75786
75787
75788
75789
75790
75791
75792
75793
75794
75795
75796
75797
75798
75799
75800
75801
75802
75803
75804
75805
75806
75807
75808
75809
75810
75811
75812
75813
75814
75815
75816
75817
75818
75819
75820
75821
75822
75823
75824
75825
75826
75827
75828
75829
75830
75831
75832
75833
75834
75835
75836
75837
75838
75839
75840
75841
75842
75843
75844
75845
75846
75847
75848
75849
75850
75851
75852
75853
75854
75855
75856
75857
75858
75859
75860
75861
75862
75863
75864
75865
75866
75867
75868
75869
75870
75871
75872
75873
75874
75875
75876
75877
75878
75879
75880
75881
75882
75883
75884
75885
75886
75887
75888
75889
75890
75891
75892
75893
75894
75895
75896
75897
75898
75899
75900
75901
75902
75903
75904
75905
75906
75907
75908
75909
75910
75911
75912
75913
75914
75915
75916
75917
75918
75919
75920
75921
75922
75923
75924
75925
75926
75927
75928
75929
75930
75931
75932
75933
75934
75935
75936
75937
75938
75939
75940
75941
75942
75943
75944
75945
75946
75947
75948
75949
75950
75951
75952
75953
75954
75955
75956
75957
75958
75959
75960
75961
75962
75963
75964
75965
75966
75967
75968
75969
75970
75971
75972
75973
75974
75975
75976
75977
75978
75979
75980
75981
75982
75983
75984
75985
75986
75987
75988
75989
75990
75991
75992
75993
75994
75995
75996
75997
75998
75999
75100
75101
75102
75103
75104
75105
75106
75107
75108
75109
75110
75111
75112
75113
75114
75115
7511

1026
 1027 **Table 10: Generalization of ALIGN-generated interfaces generated with Qwen2.5-7B-Instruct**
 1028 **to Qwen2.5-14B-Instruct.** For each agent we report its score without the interface (w/o ALIGN)
 1029 and with the interface (w/ ALIGN); the value in parentheses is the *absolute* improvement.

Base Model: Qwen2.5-14B-Instruct		Embodied		Web	Tool-use
Method	Interface	ALFWorld	ScienceWorld	WebShop	M ³ ToolEval
Vanilla	w/o ALIGN	48.51	22.58	53.67	13.89
	w/ ALIGN	52.24 (+3.73)	37.58 (+15.00)	58.40 (+4.73)	18.06 (+4.17)
ReAct	w/o ALIGN	54.48	31.24	39.73	15.28
	w/ ALIGN	70.15 (+15.67)	29.79 (-1.45)	42.17 (+2.44)	26.39 (+11.11)
Self-Consistency	w/o ALIGN	43.28	25.60	52.63	13.89
	w/ ALIGN	72.39 (+29.11)	26.68 (+1.08)	51.07 (-1.56)	27.78 (+13.89)
Self-Refine	w/o ALIGN	5.22	18.97	41.00	15.28
	w/ ALIGN	14.18 (+8.96)	20.72 (+1.75)	39.93 (-1.07)	16.67 (+1.39)
Planning	w/o ALIGN	49.25	21.46	31.72	25.00
	w/ ALIGN	79.10 (+29.85)	28.13 (+6.67)	50.47 (+18.75)	25.00 (0.00)

1043
 1044 **Table 11: Generalization of ALIGN-generated interfaces generated with Qwen2.5-7B-Instruct**
 1045 **to Llama3.1-8B-Instruct.** For each agent we report its score without the interface (w/o ALIGN)
 1046 and with the interface (w/ ALIGN); the value in parentheses is the *absolute* improvement.

Base Model: Llama3.1-8B-Instruct		Embodied		Web	Tool-use
Method	Interface	ALFWorld	ScienceWorld	WebShop	M ³ ToolEval
Vanilla	w/o ALIGN	5.22	23.59	35.17	5.56
	w/ ALIGN	14.18 (+8.96)	36.40 (+12.81)	24.00 (-11.17)	1.39 (-4.17)
ReAct	w/o ALIGN	1.49	22.42	27.12	12.50
	w/ ALIGN	15.67 (+14.18)	28.74 (+6.32)	27.10 (-0.02)	22.22 (+9.72)
Self-Consistency	w/o ALIGN	5.22	25.21	29.80	4.17
	w/ ALIGN	11.94 (+6.72)	34.83 (+9.62)	15.83 (-13.97)	2.78 (-1.39)
Self-Refine	w/o ALIGN	0.00	22.34	27.70	1.39
	w/ ALIGN	0.75 (+0.75)	31.33 (+8.99)	37.43 (+9.73)	1.39 (0.00)
Planning	w/o ALIGN	6.72	13.33	23.67	4.17
	w/ ALIGN	5.97 (-0.75)	26.95 (+13.62)	40.77 (+17.10)	4.17 (0.00)

1062
 1063 **Table 12: Generalization of ALIGN-generated interfaces generated with Qwen2.5-7B-Instruct**
 1064 **to Llama3.3-70B-Instruct.** For each agent we report its score without the interface (w/o ALIGN)
 1065 and with the interface (w/ ALIGN); the value in parentheses is the *absolute* improvement.

Base Model: Llama3.3-70B-Instruct		Embodied		Web	Tool-use
Method	Interface	ALFWorld	ScienceWorld	WebShop	M ³ ToolEval
Vanilla	w/o ALIGN	52.99	55.77	51.67	37.50
	w/ ALIGN	43.28 (-9.71)	57.74 (+1.97)	62.07 (+10.40)	33.33 (-4.17)
ReAct	w/o ALIGN	45.52	56.50	58.22	34.72
	w/ ALIGN	47.01 (+1.49)	58.28 (+1.78)	53.83 (-4.39)	43.06 (+8.34)
Self-Consistency	w/o ALIGN	54.48	56.66	50.37	36.11
	w/ ALIGN	65.67 (+11.19)	59.24 (+2.58)	55.63 (+5.26)	34.72 (-1.39)
Self-Refine	w/o ALIGN	38.06	56.97	38.40	1.39
	w/ ALIGN	46.27 (+8.21)	60.17 (+3.20)	47.85 (+9.45)	0.00 (-1.39)
Planning	w/o ALIGN	58.96	48.75	54.90	33.33
	w/ ALIGN	76.87 (+17.91)	59.17 (+10.42)	62.60 (+7.70)	40.28 (+6.95)

1080 C.4 GENERALITY STUDY EXPERIMENTS
10811082 Table 13: Experimental results for GPT-4.1 series agents with ALIGN on ALFWorld.
1083

Base Model	Interface	pick and place	pick clean and place	pick heat and place	pick cool and place	look at / examine in light	pick two obj and place	Success Rate (%)
gpt-4.1-mini	w/o ALIGN	58.33	22.58	8.70	9.52	22.22	52.94	28.36
	w/ ALIGN	95.83	87.10	26.09	80.95	27.78	52.94	64.93
gpt-4.1	w/o ALIGN	100.00	93.55	13.04	71.43	61.11	100.00	73.88
	w/ ALIGN	100.00	100.00	78.26	100.00	77.78	100.00	93.28

1090
1091 For the validation on closed-source LLMs, we selected the GPT-4.1 family. Specifically, we exper-
1092 imented with gpt-4.1-mini-2025-04-14 and gpt-4.1-2025-04-14. First, we used gpt-4.1-mini-2025-
1093 04-14 as the base model to instantiate a Vanilla Agent and synthesize interface with ALIGN. We
1094 then applied the same interface to an agent powered by gpt-4.1-2025-04-14. All other experimental
1095 settings were identical to those in the main experiments. The results are presented in Table 13.

1096 For domain-specific models trained within the environment, we used GiGPO-Qwen2.5-7B-Instruct-
1097 ALFWorld, a state-of-the-art model post-trained on ALFWorld via reinforcement learning (Feng
1098 et al., 2025). We reused the interface produced in our main experiment (generated with the base
1099 Qwen2.5-7B-Instruct model under the Vanilla Agent method). At evaluation time, we considered
1100 two configurations: (1) our Vanilla Agent setting, and (2) a configuration that matches the logic and
1101 prompt setting used during training in the original paper.

1102 C.5 ABLATION STUDY EXPERIMENTS
1103

1104 The full result of interface ablation experiment can be found in Table 14.
1105

1106 Table 14: Ablation study on the components of ALIGN. Values represent task success rates (%) or
1107 scores. For ablated conditions (w/o INFERRULES, w/o WRAPSTEP), performance changes from the
1108 ‘Full’ are shown in parentheses.
1109

Method	Interface	Embodied		Webshop	M ³ ToolEval
		ALFWorld	ScienceWorld		
Vanilla	Full	60.45	27.69	61.23	20.83
	w/o INFERRULES	51.49 (-8.96)	24.34 (-3.35)	51.03 (-10.20)	18.06 (-2.77)
	w/o WRAPSTEP	26.87 (-33.58)	22.97 (-4.72)	61.23 (-0.00)	11.11 (-9.72)
ReAct	Full	63.43	28.97	42.93	18.06
	w/o INFERRULES	58.21 (-5.22)	26.89 (-2.08)	35.97 (-6.96)	9.72 (-8.34)
	w/o WRAPSTEP	45.52 (-17.91)	22.53 (-6.44)	47.60 (+4.67)	19.44 (+1.38)
Self-Consistency	Full	69.40	25.41	61.10	16.67
	w/o INFERRULES	67.91 (-1.49)	23.11 (-2.30)	55.67 (-5.43)	13.89 (-2.78)
	w/o WRAPSTEP	23.13 (-17.91)	14.82 (-10.59)	60.67 (-0.43)	15.28 (-1.39)
Self-Refine	Full	40.30	22.99	52.30	6.94
	w/o INFERRULES	32.84 (-7.46)	21.27 (-1.72)	46.33 (-5.97)	6.94 (-0.00)
	w/o WRAPSTEP	5.97 (-34.33)	15.40 (-7.59)	47.80 (-4.50)	6.94 (-0.00)
Planning	Full	52.99	26.34	54.67	18.06
	w/o INFERRULES	42.54 (-10.45)	25.56 (-0.78)	48.18 (-6.49)	16.67 (-1.39)
	w/o WRAPSTEP	26.12 (-26.87)	16.48 (-9.86)	52.87 (-1.80)	16.67 (-1.39)

1127 D TOKEN CONSUMPTION ANALYSIS
1128

1130 The average token consumption per iteration in the main experiment described in Section 4.1 is
1131 shown in Table 15.
1132

1133 Due to the “Experimental Verification” setup, the Analyzer and Optimizer need to interact with the
environment multiple times, and all previous interaction histories are included as new prompt inputs to

1134 Table 15: The average token consumption per iteration in the main experiment described in Sec-
 1135 tion 4.1.

		ALFWorld	ScienceWorld	WebShop	M ³ ToolEval
Analyzer	Input Token (M)	0.2770	0.4333	0.1783	0.1094
	Output Token (M)	0.0040	0.0036	0.0048	0.0016
	Total Token (M)	0.2809	0.4370	0.1831	0.1109
Optimizer	Input Token (M)	0.2619	0.2288	0.0669	0.1100
	Output Token (M)	0.0087	0.0172	0.0040	0.0118
	Total Token (M)	0.2706	0.2460	0.0709	0.1217
Total	Total Token (M)	0.5515	0.6830	0.2540	0.2326

1146
 1147 the LLM in each round of interaction. Additionally, when the Optimizer identifies that the generated
 1148 interface is imperfect, it needs to refine the previously generated interface and conduct experimental
 1149 verification again, leading to increased token consumption. However, as LLM capabilities continue
 1150 to improve and hallucination issues decrease, this cost will gradually reduce. Furthermore, it is worth
 1151 noting that:

- 1153 • The INFERRULES wrapper and WRAPSTEP wrapper are implemented as python logic code,
 1154 which does not involve calls to models or agents, therefore not incurring additional token
 1155 consumption. On the contrary, as demonstrated in our experiments in Section 4.3, **using**
 1156 **ALIGN-generated interfaces can help agents reduce repetitive meaningless actions, thereby**
 1157 **reducing the number of LLM calls and decreasing token consumption** compared to not
 1158 using ALIGN-generated interfaces.
- 1159 • Except when the Optimizer generates interface codes requiring the cutting edge LLMs (such as
 1160 Gemini 2.5 Pro), weaker and more cost-effective LLMs (such as GPT-4.1-mini) can be used at
 1161 other times, which will significantly reduce the operational costs of ALIGN.
- 1162 • ALIGN-generated interfaces can generalize to different agent architectures and base LLMs.
 1163 This means that for each environment, using the ALIGN method to generate an interface only
 1164 once can bring performance improvements to different agents, regardless of agent version
 1165 updates. This also means that **the cost of interface generation is a one-time expense**, rather
 1166 than requiring the generation of new interfaces for each task execution. Therefore, from an
 1167 amortization perspective, the method’s cost becomes increasingly economical as the environment
 1168 is utilized more frequently, with the one-time interface design cost being distributed across
 1169 multiple uses and becoming proportionally smaller with increased usage.

E IMPLEMENTATION DETAILS

E.1 BENCHMARKS TASK SPLITS

1175 The task splits of benchmarks we use are as follows:

1176 (1) ALFWorld (Shridhar et al., 2021): We adhere to the original dataset partitioning presented in the
 1177 paper, wherein the tasks from the “eval_out_of_distribution” category are used as the test set, and the
 1178 “train” category is designated as the training set. In each iteration, we randomly select three tasks
 1179 from the training set of each task type to serve as the training data for the agent’s interaction.

1180 (2) ScienceWorld (Wang et al., 2022): We follow the original partitioning of the train and test sets as
 1181 described in the paper. For efficiency reasons, during testing, we select at most the first five tasks
 1182 from the 30 available task types for evaluation. In each iteration, we randomly select one task from
 1183 the training set of each task type to be used as the training data for the agent’s interaction.

1184 (3) WebShop (Yao et al., 2022): In alignment with the setup of Yao et al. (2023), we use tasks with
 1185 IDs ranging from 0 to 49 (50 tasks in total) as the test set, and tasks with IDs from 50 to 199 (150
 1186 tasks in total) as the training set. In each iteration, we randomly select 20 tasks from the training set
 1187 to serve as the training data for the agent’s interaction.

1188 (4) M³ToolEval (Wang et al., 2024b): Since M³ToolEval does not provide a distinct training set
 1189 division, we select two tasks from each task type in the original dataset as the training set, with the
 1190 remaining tasks used as the test set. In each iteration, the entire training set is utilized for the agent's
 1191 interaction.
 1192

1193 E.2 HYPERPARAMETER AND EXPERIMENT SETTING

1195 For all the agents, we deploy them uniformly using vllm (Kwon et al., 2023) across 8 Nvidia A100
 1196 80GB GPUs, with the inference temperature set to 0.0. The models utilized contain Qwen2.5-7B-
 1197 Instruct² (Team, 2024), Qwen2.5-14B-Instruct³ (Team, 2024), Llama3.1-8B-Instruct⁴ (Meta, 2025a)
 1198 and Llama3.3-70B-Instruct⁵ (Meta, 2025b).

1199 In ALIGN, we use Gemini 2.5 Pro (gemini-2.5-pro-exp-03-25)(Google, 2025) for Optimizer to
 1200 generate new interface, with the temperature set to 0.2. For other scenarios requiring the use of an
 1201 LLM, we employ GPT-4.1 (gpt-4.1-2025-04-14)(OpenAI, 2025). We set $K = 8$ during experiments.
 1202

1203 E.3 TOOLS FOR EXPERIMENTAL VERIFICATION

1205 In order to implement the experimental verification process mentioned in Section 3.3, we have
 1206 encapsulated the following tools for Analyzer and Optimizer to interact with the interface-wrapped
 1207 environment:

- 1208 (1) `init_simulator(task_id, interface)`: Initializes an experimental task, specifying
 1209 the task ID and the interface code.
- 1210 (2) `reset_simulator()`: Resets the experimental task.
- 1212 (3) `run_task()`: Runs the current task until completion, returning the interaction trajectory.
- 1213 (4) `exec_agent_action(agent_action)`: Executes a specific action and returns the en-
 1214 hanced observation after the interface processing.
- 1216 (5) `get_agent_action()`: Based on the current trajectory, returns the next action to be issued
 1217 by the agent.
- 1218 (6) `change_obs(obs)`: Modifies the observation of the previous action execution.

1220 E.4 PROMPT TEMPLATES

1222 We present the prompt template of the Analyzer and Optimizer for ALFWorld. For the prompt
 1223 templates of other benchmarks, please refer to the supplemental materials. For the WebShop and
 1224 M³ToolEval environments, no “Gold Action and Observation Sequence” is provided.

1225 Analyzer Prompt Template of Misalignment Analysis

1227 User message:

1228 In modern benchmarks evaluating LLM Agent reasoning capabilities, human designers create
 1229 an Environment with a set of rules defining how tasks are accomplished. These rules, referred
 1230 to as the Environment's World Model, specify the sequence of actions required to achieve
 1231 specific outcomes. For example, the Environment's World Model might dictate that certain
 1232 actions (e.g., operating on a receptacle) can only be performed after prerequisite actions (e.g.,
 1233 moving to the receptacle).

1234 Meanwhile, the Agent operates based on its own World Model, which it constructs
 1235 by interpreting the task and environment prompts. The Agent first determines its high-level
 1236 reasoning intent—its understanding of what needs to be done—and then selects actions
 1237

1238 ²<https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>

1239 ³<https://huggingface.co/Qwen/Qwen2.5-14B-Instruct>

1240 ⁴<https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>

1241 ⁵<https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct>

1242
1243
1244
1245
1246
1247

according to its internal World Model. However, because the Environment’s World Model is manually crafted and may not be fully conveyed through prompts, the Agent’s World Model might differ, leading to unexpected behavior. For instance, the Agent might choose an action that aligns with its intent but violates the Environment’s rules, or it might misinterpret feedback due to insufficient information from the Environment.

1248
1249
1250
1251
1252

We define a misalignment between the Environment’s World Model and the Agent’s World Model as a situation where:

- The Environment provides feedback that does not sufficiently clarify its World Model, leaving the Agent unable to adjust its understanding of the rules.

1253
1254
1255
1256

Your task is to analyze the logs from a recent task to determine whether such a misalignment occurred, preventing a fair assessment of the Agent’s capabilities. And this misalignment has not been fixed by current ‘WrapStep’ function. Your analysis will guide us in addressing this issue moving forward.

1257
1258
1259

Experimental Environment Evaluation Template

1260
1261
1262
1263

```
““python
{{ experimental_template }}
```

1264
1265
1266
1267
1268

In this template, the function ‘InferRules’ is used to define the environment rules. The function ‘WrapStep’ handles post-processing of the Agent’s actions (e.g., splitting them into multiple steps, performing pre-checks, returning more detailed feedback, etc.). This function should not interfere with the Agent’s own reasoning. The current implementation is as follows:

1269
1270
1271
1272
1273

```
““python
{{ Interface }}
```

1274
1275

Environment Logs

1276
1277
1278

```
““txt
{{ logs }}
```

1279
1280
1281

Here, each ‘Observation’ is the feedback returned to the Agent after it executes an action.

1282
1283
1284

Gold Action and Observation Sequence

1285
1286
1287
1288

```
““txt
{{ gold_action_obs_sequence }}
```

1289
1290
1291

Environment Logics and Misalignment Analyzed in the Previous Steps

1292
1293
1294
1295

```
{{ environment_logics }}
```

Your Task

1296
 1297 Determine whether, during this task, there was a misalignment between the Envi-
 1298 ronment's World Model and the Agent's World Model that hindered a fair assessment of the
 1299 Agent's capabilities. Choose exactly one of the following outputs:
 1300
 1301 If there is NO misalignment (i.e., the Agent's failures stem from its own errors or
 1302 limitations, not a mismatch with the Environment's World Model), output:
 1303 <analysis_result> No Misalignment </analysis_result>
 1304
 1305 If there IS a misalignment (i.e., the Environment's World Model conflicts with the
 1306 Agent's World Model), output:
 1307 <analysis_result> Found Misalignment </analysis_result>
 1308 <environment_logic_and_misalignments> the new environment rules and misalignments
 1309 identified by you, which have not been fixed by current 'WrapStep' function.
 1310 </environment_logic_and_misalignments>
 1311
 1312 The format of the environment logic and misalignment is as follows:
 1313 “txt
 1314 ### Analysis Result 1
 1315 Analysis Task ID: xxx
 1316 Agent Action Type: xxx # The type of action the Agent attempted to perform, such as
 1317 "examine", "move object to receptacle", etc.
 1318 Agent Action Case: xxx # The specific action the Agent attempted to perform.
 1319 Agent High-Level Reasoning Intent: xxx # The Agent's high-level reasoning intent, which
 1320 may be a general description of the action it was trying to perform.
 1321 Environment World Model Rule: xxx # The rule from the Environment's World Model that
 1322 don't align the Agent's World Model.
 1323 Sufficient Environment Feedback: xxx # to offer the Agent adequate information to bridge
 1324 gaps in understanding the environment's world model. such as "The environment should
 1325 provide 'xxx' feedback when the Agent attempts to operate on a receptacle without first
 1326 going to it."
 1327 Type: "Bug of current WrapStep function" or "Need to add new logic in the WrapStep
 1328 function"
 1329
 1330 ### Analysis Result 2
 1331 ...
 1332
 1333 Note: You should not generate duplicate misalignment analysis results as the ones
 1334 already provided in the 'Environment Logics and Misalignment Analyzed in the Previous
 1335 Steps' section.
 1336

Analyzer Prompt Template of Experimental Verification

User message:

Now you should conduct simulation experiments in the simulator to verify that the environment rules you hypothesized and Misalignment you identified truly exists. You must perform sufficient experiments to confirm or refute your suspicion.

Here are the operations you can use:

1. init_simulator(task_id: str)
 - Initializes a new simulator for the specified 'task_id'.
 - 'task_id' must be in the format 'int-int' where the first int $\in [0, 5]$.
 - The different task types are mapped as follows:

```

1350
1351 0: 'pick_and_place',
1352 1: 'pick_clean_and_place',
1353 2: 'pick_heat_and_place',
1354 3: 'pick_cool_and_place',
1355 4: 'look_at_or_examine_in_light',
1356 5: 'pick_two_obj_and_place'
1357
1358 - All subsequent operations occur within this initialized simulator.
1359
1360 2. reset_simulator()
1361 - Resets the current simulator to its initial state.
1362
1363 3. execute_agent_action(agent_action: str)
1364 - Executes an agent action using the 'WrapStep' function.
1365
1366 4. change_last_action_observation(obs: str)
1367 - Updates the last observation returned by the simulator to the specified 'obs'.
1368 - This is useful for simulating the agent's next action in a different environment feedback
1369 context.
1370
1371 5. get_next_agent_action()
1372 - Retrieves the next action that the real Agent would perform under the current simulation
1373 conditions.
1374 - Note: The Agent's choice of the next action is based on the current environment state,
1375 including the outcomes of any previous 'step()' or 'get_next_agent_action()' call, along with
1376 the latest observations.
1377
1378 If you believe you have reached a conclusion from your experiments, provide it in
1379 this format:
1380
1381 <thought> Your reasoning here </thought>
1382 <environment_logic_and_misalignments> the new environment rules and misalignments
1383 identified by you, which have not been fixed by current 'WrapStep' function. </environment_logic_and_misalignments>
1384
1385 The format of the environment logic and misalignment is as follows:
1386
1387 "txt
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3046
3047
3047
3048
3048
3049
3049
3050
3050
3051
3051
3052
3052
3053
3053
3054
3054
3055
3055
3056
3056
3057
3057
3058
3058
3059
3059
3060
3060
3061
3061
3062
3062
3063
3063
3064
3064
3065
3065
3066
3066
3067
3067
3068
3068
3069
3069
3070
3070
3071
3071
3072
3072
3073
3073
3074
3074
3075
3075
3076
3076
3077
3077
3078
3078
3079
3079
3080
3080
3081
3081
3082
3082
3083
3083
3084
3084
3085
3085
3086
3086
3087
3087
3088
3088
3089
3089
3090
3090
3091
3091
3092
3092
3093
3093
3094
3094
3095
3095
3096
3096
3097
3097
3098
3098
3099
3099
3100
3100
3101
3101
3102
3102
3103
3103
3104
3104
3105
3105
3106
3106
3107
3107
3108
3108
3109
3109
3110
3110
3111
3111
3112
3112
3113
3113
3114
3114
3115
3115
3116
3116
3117
3117
3118
3118
3119
3119
3120
3120
3121
3121
3122
3122
3123
3123
3124
3124
3125
3125
3126
3126
3127
3127
3128
3128
3129
3129
3130
3130
3131
3131
3132
3132
3133
3133
3134
3134
3135
3135
3136
3136
3137
3137
3138
3138
3139
3139
3140
3140
3141
3141
3142
3142
3143
3143
3144
3144
3145
3145
3146
3146
3147
3147
3148
3148
3149
3149
3150
3150
3151
3151
3152
3152
3153
3153
3154
3154
3155
3155
3156
3156
3157
3157
3158
3158
3159
3159
3160
3160
3161
3161
3162
3162
3163
3163
3164
3164
3165
3165
3166
3166
3167
3167
3168
3168
3169
3169
3170
3170
3171
3171
3172
3172
3173
3173
3174
3174
3175
3175
3176
3176
3177
3177
3178
3178
3179
3179
3180
3180
3181
3181
3182
3182
3183
3183
3184
3184
3185
3185
3186
3186
3187
3187
3188
3188
3189
3189
3190
3190
3191
3191
3192
3192
3193
3193
3194
3194
3195
3195
3196
3196
3197
3197
3198
3198
3199
3199
3200
3200
3201
3201
3202
3202
3203
3203
3204
3204
3205
3205
3206
3206
3207
3207
3208
3208
3209
3209
3210
3210
3211
3211
3212
3212
3213
3213
3214
3214
3215
3215
3216
3216
3217
3217
3218
3218
3219
3219
3220
3220
3221
3221
3222
3222
3223
3223
3224
3224
3225
3225
3226
3226
3227
3227
3228
3228
3229
3229
3230
3230
3231
3231
3232
3232
3233
3233
3234
3234
3235
3235
3236
3236
3237
3237
3238
3238
3239
3239
3240
3240
3241
3241
3242
3242
3243
3243
3244
3244
3245
3245
3246
3246
3247
3247
3248
3248
3249
3249
3250
3250
3251
3251
3252
3252
3253
3253
3254
3254
3255
3255
3256
3256
3257
3257
3258
3258
3259
3259
3260
3260
3261
3261
3262
3262
3263
3263
3264
3264
3265
3265
3266
3266
3267
3267
3268
3268
3269
3269
3270
3270
3271
3271
3272
3272
3273
3273
3274
3274
3275
3275
3276
3276
3277
3277
3278
3278
3279
3279
3280
3280
3281
3281
3282
3282
3283
3283
3284
3284
3285
3285
3286
3286
3287
3287
3288
3288
3289
3289
3290
3290
3291
3291
3292
3292
3293
3293
3294
3294
3295
3295
3296
3296
329
```

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

If you need to carry out more operations in the simulator, respond in the following format, specifying exactly one operation per turn:

<thought> Your reasoning here, you should consider all hypotheses if the simulation result is not as expected </thought>

<action> The single operation you wish to perform (e.g., init_simulator(task_id="x-y"), step(action="x"), execute_agent_action(agent_action="x"), etc.) </action>

Note:

You should verify the correctness of the following, step by step, through your experiments:

1. environment_rules: Use ‘execute_agent_action’ to confirm that the environment rules you hypothesized are indeed correct, and current ‘WrapStep’ function is not sufficient.

2. agent_intent_description: Obtain the Agent’s intended behavior (e.g., via ‘get_next_agent_action’) and simulate it by using ‘WrapStep’ to confirm whether it aligns with your description.

3. identified_misalignment: Through chaning the environment feedback, you can verify whether the misalignment you identified is indeed correct and the environment feedback you hypothesized is indeed sufficient. You can use ‘WrapStep’ to simulate the agent’s action, then use ‘change_last_action_observation’ to change the environment feedback, and finally use ‘get_next_agent_action’ to check whether the agent can correctly identify the next action.

Analyzer Prompt Template of Reranking Misalignments Analysis (Ablation Study)

User message:

In modern benchmarks evaluating LLM Agent reasoning capabilities, human designers create an Environment with a set of rules defining how tasks are accomplished. These rules, referred to as the Environment’s World Model, specify the sequence of actions required to achieve specific outcomes. For example, the Environment’s World Model might dictate that certain actions (e.g., operating on a receptacle) can only be performed after prerequisite actions (e.g., moving to the receptacle).

Meanwhile, the Agent operates based on its own World Model, which it constructs by interpreting the task and environment prompts. The Agent first determines its high-level reasoning intent—its understanding of what needs to be done—and then selects actions according to its internal World Model. However, because the Environment’s World Model is manually crafted and may not be fully conveyed through prompts, the Agent’s World Model might differ, leading to unexpected behavior. For instance, the Agent might choose an action that aligns with its intent but violates the Environment’s rules, or it might misinterpret feedback due to insufficient information from the Environment.

We define a misalignment between the Environment’s World Model and the Agent’s World Model as a situation where:

- The Environment provides feedback that does not sufficiently clarify its World Model, leaving the Agent unable to adjust its understanding of the rules.

Now other human experts have analyzed the logs from a recent task and identified some potential misalignments. Your task is to review these misalignments and choose the most appropriate one.

Experimental Environment Evaluation Template

```
““python
{{ experimental_template }}““
```

1458
 1459 In this template, the function ‘InferRules‘ is used to define the environment rules.
 1460 The function ‘WrapStep‘ handles post-processing of the Agent’s actions (e.g., splitting them
 1461 into multiple steps, performing pre-checks, returning more detailed feedback, etc.). This
 1462 function should not interfere with the Agent’s own reasoning. The current implementation
 1463 is as follows:

1464 ““python
 1465 {{ Interface }}
 1466 ““

1468 _____
 1469 ### Environment Logs

1470 ““txt
 1471 {{ logs }}
 1472 ““

1474 Here, each ‘Observation‘ is the feedback returned to the Agent after it executes an
 1475 action.

1477 _____
 1478 ### Gold Action and Observation Sequence

1480 ““txt
 1481 {{ gold_action_obs_sequence }}
 1482 ““

1484 _____
 1485 ### Environment Logics and Misalignment Analyzed in the Previous Steps

1486 {{ environment_logics }} Note: These logics may not be accurate. They are the
 1487 environment rules that were previously hypothesized and may contain errors.

1489 _____
 1490 ### Your Task

1491 Choose the most appropriate misalignment analyzed by human experts from the list
 1492 below:

1494 {{ new_environment_logics }}

1496 You should respond in format as follows:
 1497 ““

1498 <review> Your review of each expert output one by one </review>
 1499 <expert_id> id of the selected expert output, only the number </expert_id>
 1500 ““

1503 Optimizer Prompt Template of Interface Generation

1504
 1505 **User message:**

1506 In modern benchmarks evaluating LLM Agent reasoning capabilities, human designers create
 1507 an Environment with a set of rules defining how tasks are accomplished. These rules, referred
 1508 to as the Environment’s World Model, specify the sequence of actions required to achieve
 1509 specific outcomes. For example, the Environment’s World Model might dictate that certain
 1510 actions (e.g., operating on a receptacle) can only be performed after prerequisite actions (e.g.,
 1511 moving to the receptacle).

1512
1513
1514
1515
1516
1517
1518
1519
1520

Meanwhile, the Agent operates based on its own World Model, which it constructs by interpreting the task and environment prompts. The Agent first determines its high-level reasoning intent—its understanding of what needs to be done—and then selects actions according to its internal World Model. However, because the Environment’s World Model is manually crafted and may not be fully conveyed through prompts, the Agent’s World Model might differ, leading to unexpected behavior. For instance, the Agent might choose an action that aligns with its intent but violates the Environment’s rules, or it might misinterpret feedback due to insufficient information from the Environment.

1521
1522
1523
1524
1525

We define a misalignment between the Environment’s World Model and the Agent’s World Model as a situation where:

- The Environment provides feedback that does not sufficiently clarify its World Model, leaving the Agent unable to adjust its understanding of the rules.

1526
1527
1528

Your task is to refine the environment’s behavior based on the misalignment identified by the AnalysisAgent, ensuring the Agent’s true intentions are executed and its reasoning capabilities are fairly assessed.

1529
1530
1531

Experimental Environment Evaluation Template

1532
1533
1534
1535

```
““python
{{ experimental_template }}
```

1536
1537
1538
1539
1540

In this template, the function ‘InferRules’ is used to define the environment rules. The function ‘WrapStep’ handles post-processing of the Agent’s actions (e.g., splitting them into multiple steps, performing pre-checks, returning more detailed feedback, etc.). This function should not interfere with the Agent’s own reasoning. There current implementation is as follows:

1541
1542
1543
1544
1545

```
““python
{{ WrapStep }}
```

1546
1547
1548
1549
1550

Environment Logics and Misalignment Analyzed by AnalysisAgent Previously

```
{{ last_environment_logics }}
```

1551
1552
1553
1554

New Environment Logics and Misalignment Analyzed by AnalysisAgent

```
{{ new_environment_logics }}
```

1555
1556

Your Task

1557
1558
1559
1560
1561
1562

Based on the misalignments identified by the AnalysisAgent, you need to refine and enhance the ‘InferRules’ function and ‘WrapStep’ function to align the Environment’s World Model with the Agent’s actions and provide clearer feedback. Your output should present the new versions of these functions, ensuring the Agent’s high-level reasoning intent is preserved.

Please ensure you follow these requirements:

1563
1564
1565

1. **Function Signature**

The function signature must be:

```

1566
1567     ````python
1568     def InferRules(init_obs, task)
1569     - init_obs: str, the initial observation from the environment, containing all receptacles.
1570     - task: str, the task description.
1571
1572     def WrapStep(env, init_obs, task, agent_action: str, logger)
1573     ````

1574     2. **Return Values**
1575     The ‘InferRules‘ function’s return value must be a string that describes the environment rules.
1576
1577     The ‘WrapStep‘ function’s return value must be three items:
1578     ````python
1579     obs: str, reward: bool, done: bool
1580     ````

1581     3. **‘env.step‘ Usage**
1582     The only permitted usage pattern for ‘env.step‘ is:
1583     ````python
1584     obs, reward, done, info = env.step([agent_action])
1585     obs, reward, done = obs[0], info[‘won’][0], done[0]
1586     ````

1587     No alternative usage forms are allowed. Each call to env.step causes an irreversible change to
1588     the environment state; actions must therefore be chosen carefully.
1589
1590     4. **Package Imports**
1591     You may import other packages if necessary, but you must include all imports in your code.
1592
1593     5. **Multiple Calls and Conditional Returns**
1594     You are free to call ‘env.step‘ multiple times or return different ‘obs‘ depending on
1595     ‘agent_action‘ or the outcomes of these calls.
1596
1597     6. **You can use logger.debug**
1598     You can use ‘logger.debug‘ to log any information you find useful. The logging will be
1599     captured and returned to you in the future for further analysis.
1600
1601     7. Do not modify any aspects not explicitly identified by the AnalysisAgent in the
1602     “New Environment Logics and Misalignment Analyzed by AnalysisAgent” section.
1603
1604     8. You must use the following approach when addressing the identified misalign-
1605     ment:
1606     - For each action defined in environment, provide clear, informative, and sufficient feedback
1607     from the environment whenever an invalid action is attempted, guiding the Agent toward
1608     understanding and adhering to the environment’s rules.
1609
1610     9. **Output Format**
1611     You must provide the output strictly in the following format:
1612     <thought>YOUR_THOUGHT_PROCESS_HERE</thought>
1613     <code>YOUR_CODE_HERE</code>
1614
1615     Please ensure your final answer follows these guidelines so that we can accurately
1616     bridge the misalignment and allow the environment to execute the Agent’s true intentions.
1617
1618
1619

```

1620
1621

Optimizer Prompt Template of Experimental Verification

1622

User message:

1623

Now you should conduct simulation experiments in the simulator to verify if the ‘InferRules’ and ‘WrapStep’ function you provided is correct for the new environment logics and misalignment analyzed by the AnalysisAgent.

1624

1625

1626

1627

1628

You must perform sufficient experiments to confirm or refute your suspicion. Here are the operations you can use:

1629

1. init_simulator(task_id: str)

- Initializes a new simulator for the specified ‘task_id’.
- ‘task_id’ must be in the format ‘int-int’ where the first int $\in [0, 5]$.
- The different task types are mapped as follows:

1630

0: ‘pick_and_place’,
1: ‘pick_clean_and_place’,
2: ‘pick_heat_and_place’,
3: ‘pick_cool_and_place’,
4: ‘look_at_or_examine_in_light’,
5: ‘pick_two_obj_and_place’

1631

- All subsequent operations occur within this initialized simulator.

1632

2. reset_simulator()

- Resets the current simulator to its initial state.

1633

3. execute_agent_action(agent_action: str)

- Executes an agent action using the ‘WrapStep’ function you generated.

1634

4. change_last_action_observation(obs: str)

- Updates the last observation returned by the simulator to the specified ‘obs’.
- This is useful for simulating the agent’s next action in a different environment feedback context.

1635

5. get_next_agent_action()

- Retrieves the next action that the real Agent would perform under the current simulation conditions.
- Note: The Agent’s choice of the next action is based on the current environment state, including the outcomes of any previous ‘step()’ or ‘get_next_agent_action()’ call, along with the latest observations.

1636

6. run_task(task_id: str)

- Runs the entire task in the simulator and returns the running log.

1637

- After running the whole task, you need to call ‘init_simulator’ or ‘reset_simulator’ to reinitialize the simulator for further operations.

1638

If you believe you have reached a conclusion from your experiments, provide it in this format:

1639

<thought> Your reasoning here </thought>

1640

<if_need_refine> True/False </if_need_refine>

1641

<refine_strategy> Your strategy for refining the WrapStep function, if if_need_refine is True </refine_strategy>

1642

1643

If you need to carry out more operations in the simulator, respond in the following format, specifying exactly one operation per turn:

1644

1674
 1675 <thought> Your reasoning here, you should consider all hypotheses if the simula-
 1676 tion result is not as expected </thought>
 1677 <action> The single operation you wish to perform (e.g., init_simulator(task_id="x-y"),
 1678 step(action="x"), execute_agent_action(agent_action="x"), etc.) </action>
 1679
 1680

Optimizer Prompt Template of Reranking Interface Generation (Ablation Study)

User message:

In modern benchmarks evaluating LLM Agent reasoning capabilities, human designers create an Environment with a set of rules defining how tasks are accomplished. These rules, referred to as the Environment’s World Model, specify the sequence of actions required to achieve specific outcomes. For example, the Environment’s World Model might dictate that certain actions (e.g., operating on a receptacle) can only be performed after prerequisite actions (e.g., moving to the receptacle).

Meanwhile, the Agent operates based on its own World Model, which it constructs by interpreting the task and environment prompts. The Agent first determines its high-level reasoning intent—its understanding of what needs to be done—and then selects actions according to its internal World Model. However, because the Environment’s World Model is manually crafted and may not be fully conveyed through prompts, the Agent’s World Model might differ, leading to unexpected behavior. For instance, the Agent might choose an action that aligns with its intent but violates the Environment’s rules, or it might misinterpret feedback due to insufficient information from the Environment.

We define a misalignment between the Environment’s World Model and the Agent’s World Model as a situation where:

- The Environment provides feedback that does not sufficiently clarify its World Model, leaving the Agent unable to adjust its understanding of the rules.

Now other human experts have generated a set of code patches to address the misalignment between the Environment’s World Model and the Agent’s World Model. Your task is to evaluate these patches and select the best one.

Experimental Environment Evaluation Template

```
```python
{{ experimental_template }}
```

In this template, the function ‘InferRules’ is used to define the environment rules. The function ‘WrapStep’ handles post-processing of the Agent’s actions (e.g., splitting them into multiple steps, performing pre-checks, returning more detailed feedback, etc.). This function should not interfere with the Agent’s own reasoning. The current implementation is as follows:

```
```python
{{ WrapStep }}
```

Environment Logics and Misalignment Analyzed by AnalysisAgent Previously

```
{{ last_environment_logics }}
```

1728
 1729 ### New Environment Logics and Misalignment Analyzed by AnalysisAgent
 1730
 1731 {{ new_environment_logics }}
 1732
 1733
 1734 ### Your Task

1735 Choose the best code from the following options to address the misalignment be-
 1736 tween the Environment’s World Model and the Agent’s World Model:

1738 {{ code_patches }}

1740 You should respond in format as follows:

1741 “

1742 <review> Your review of each code one by one </review>

1743 <code_id> id of the selected result, only the number </code_id>

1744 “

1745
 1746
 1747 We present the prompt template of the Vanilla agent in ALFWorld to illustrate the usage of the
 1748 INFERRULES. For the prompt templates of other agent methods and benchmarks, please refer to the
 1749 supplemental materials.

Vanilla Agent Prompt Template in ALFWorld

System message:

You are an AI assistant solving tasks in a household environment. Your goal is to break down complex tasks into simple steps and plan your actions accordingly.

Action Space

In this environment, you have a set of high-level actions at your disposal, each corresponding to a typical household activity. These actions are:

- look: look around your current location
- inventory: check your current inventory
- go to (receptacle): move to a receptacle
- open (receptacle): open a receptacle
- close (receptacle): close a receptacle
- take (object) from (receptacle): take an object from a receptacle
- move (object) to (receptacle): place an object in or on a receptacle
- examine (something): examine a receptacle or an object
- use (object): use an object
- heat (object) with (receptacle): heat an object using a receptacle
- clean (object) with (receptacle): clean an object using a receptacle
- cool (object) with (receptacle): cool an object using a receptacle
- slice (object) with (object): slice an object using a sharp object

Although each action may internally consist of multiple embodied steps (e.g., walking to the sink, turning a knob, etc.), from your perspective you need only provide one high-level action at a time.

Instructions

Single Action per Turn

At each step, you must respond with exactly one action (i.e., the next “thought”). Use the format:

```

1782
1783 ACTION [object/receptacle specifier]
1784 ACTION [object/receptacle specifier]
1785 For example:
1786 take apple from table
1787 or
1788 go to kitchen.

1789 Environment Feedback
1790 After you provide your single action, the environment will automatically execute it and return
1791 the resulting observation. You then decide on your next action based on the updated state.
1792
1793 Reasoning (Chain of Thought)
1794 You may use hidden reasoning to figure out the best next step. However, only output the
1795 single action that represents your decision. Do not reveal your entire chain of thought.
1796
1797 Continue Until Task Completion
1798 You will iterate this process—receiving the environment’s feedback, deciding on the next
1799 action, and outputting a single action—until the task is finished.

1800 # Environment Rule
1801
1802 {InferRules(init_obs, task)}
1803
1804 User message:
1805 # Task
1806
1807 {initial_obs}

1808 Begin by examining the environment or taking any initial steps you find relevant.
1809 Remember, provide only one action each time.
1810
1811
1812

```

E.5 INITIALIZED INTERFACE

Initialized interface we used in ALFWorld:

```

1816 def InferRules(init_obs, task):
1817     """
1818     Contains the rules for environment and task execute logic for
1819     different task types.
1820     """
1821     return "There is no rule for this environment."
1822
1823 def WrapStep(env, init_obs, task, agent_action: str, logger):
1824     """
1825     Process the agent action and return the next observation, reward,
1826     and done status.
1827     """
1828     obs, reward, done, info = env.step([agent_action])
1829     obs, reward, done = obs[0], info['won'][0], done[0]
1830     return obs, reward, done

```

Initialized interface we used in ScienceWorld:

```

1831 def InferRules(init_obs, task):
1832     """
1833     Contains the rules for environment and task execute logic for
1834     different task types.
1835     """
1836     return "There is no rule for this environment."

```

```

1836
1837     def WrapStep(env, init_obs, task, agent_action: str, logger):
1838         """
1839             Process the agent action and return the next observation, done
1840             status and score(returned by the environment).
1841         """
1842         obs, _, done, info = env.step(agent_action)
1843         return obs, done, info["score"]
1844
1845
1846     Initialized interface we used in WebShop:
1847
1848
1849     def InferRules(init_obs, task):
1850         """
1851             Contains the rules for environment and task execute logic.
1852         """
1853         return "There is no rule for this environment."
1854
1855     def WrapStep(env, init_obs, task, agent_action: str, logger):
1856         """
1857             Process the agent action and return the next observation, reward,
1858             and done status.
1859         """
1860         obs, reward, done = env.step(agent_action)
1861         return obs, reward, done

```

1862 Initialized interface we used in M³ToolEval:

```

1863
1864
1865     def InferRules(task_name, task_type_idx):
1866         """
1867             Contains the rules for environment and task execute logic for
1868             different task types.
1869         """
1870         return "There is no rule for this environment."
1871
1872     def WrapStep(env, task_name, instruction, agent_action: str, logger):
1873         """
1874             Process the agent action and return the next observation, reward,
1875             and done status.
1876         """
1877         obs, reward, done = env.step(agent_action)
1878         return obs, reward, done

```

F CASE STUDY

F.1 MISALIGNMENTS ANALYZED BY ANALYZER

1886 We present the misalignments analyzed by Analyzer with Vanilla agent. For the misalignments
1887 analyzed by Analyzer with other agent methods, please refer to the supplemental materials.

1890	# ALFWORLD	
1891		
1892	Agent Action Type: heat object with receptacle	
1893	Agent Action Case: heat mug 1 with stoveburner 1	
1894	Agent High-Level Reasoning Intent: The Agent intended to heat the mug using the stoveburner to fulfill the “put a hot mug in cabinet” task requirement.	
1895	The Environment requires heating the mug specifically by the microwave, and the Agent must be at and open the microwave before heating. Heating with the stoveburner or heating without opening the microwave results in no effect.	
1896	The environment feedback “Nothing happens.” after heating with stoveburner or heating without opening the microwave is insufficient to clarify the correct heating method and prerequisites.	
1897		
1898		
1899		
1900		
1901		
1902		
1903		
1904	# SCIENCEWORLD	
1905		
1906	Agent Action Type: pick up OBJ from CONTAINER / take OBJ from CONTAINER	
1907	Agent Action Case: pick up orange seed from seed jar, take orange seed from seed jar, take seed from seed jar, pick up seed from seed jar	
1908	Agent High-Level Reasoning Intent: Agent intends to retrieve a seed from the “seed jar” container using common interaction verbs and syntax (“pick up X from Y”, “take X from Y”).	
1909		
1910		
1911	Environment Rule: The environment does not support the “take OBJ from CONTAINER” syntax. Furthermore, for the “seed jar”, the “pick up OBJ from CONTAINER” syntax is also invalid. The required procedure to access the seeds involves picking up the entire container first (“pick up seed jar”) and then likely using a “move” command later. Direct retrieval from the container using “pick up” or “take with from” is disallowed.	
1912		
1913		
1914		
1915		
1916		
1917		
1918	Sufficient Environment Feedback: The current generic feedback provided by “process_agent_action” for “No known action” is insufficient. Sufficient feedback should diagnose the invalid syntax or procedure, e.g., “The action ‘take X from Y’ is not valid. To get items from the ‘seed jar’, try picking up the ‘seed jar’ first using ‘pick up seed jar’.” Simulation confirmed this guides the agent correctly.	
1919		
1920		
1921		
1922		
1923		
1924		
1925		
1926		
1927		
1928		
1929		
1930		
1931		
1932		
1933		
1934		
1935		
1936		
1937		
1938		
1939		
1940		
1941		
1942		
1943		

1944	# WEBSHOP	
1945	Agent Action Type:	click
1946	Agent Action Case:	click[1 ounce (pack of 21)] (or similar option clicks like flavor, color, etc.)
1947	Agent High-Level Reasoning Intent:	The Agent intended to select a specific product configuration (e.g., size) required by the task before proceeding to purchase or further inspection.
1948	Environment Rule:	When an Agent clicks on a product option (e.g., size, color, flavor), the internal state of the environment updates to reflect this selection. This selection affects the final product configuration (and potentially price, availability, description shown) when subsequent actions like “Buy Now” or viewing details are taken. The visual representation of the page should ideally reflect this selected state.
1949	Sufficient Environment Feedback:	The environment currently returns only a confirmation message (e.g., “You have clicked [Option Name].”). This is insufficient as it doesn’t show the agent the result of its action in the context of the full page. Sufficient feedback would involve returning the complete observation of the item page *after* the option click, reflecting the updated state (e.g., showing the selected size/flavor visually marked, potentially an updated price, updated product title/description if applicable, and all other page elements like “[Buy Now]”). This allows the agent to verify the selection and its consequences (like price changes) before taking the next step.
1950		
1951		
1952		
1953		
1954		
1955		
1956		
1957		
1958		
1959		
1960		
1961		
1962		
1963		
1964		
1965		
1966		
1967		
1968		
1969	# M³TOOLEVAL	
1970		
1971	Agent Action Type:	web_browsing
1972	Agent Action Case:	click_url(“/about”)
1973	Agent High-Level Reasoning Intent:	The Agent is attempting to use the click_url tool to navigate to the “/about” page by following the clickable link, as per the tool description and the rendered webpage content.
1974	Environment Rule:	The Environment’s World Model expects tool invocations to match the tool name exactly, with arguments provided after the tool name and separated by a comma (i.e., “Action: click_url, /about End Action” instead of “Action: click_url(‘/about’) End Action”). However, the Agent, following the tool signature format shown in the tool descriptions (e.g., click_url(url: str)), invokes the tool as click_url(“/about”), which is a common programming convention.
1975	Sufficient Environment Feedback:	The feedback “Cound not find tool with name click_url(‘/about’)” is insufficient for the Agent to understand the correct invocation format. The environment should provide feedback such as: “Tool arguments should be provided after the tool name, separated by a comma, e.g., ‘Action: click_url, /about End Action’ instead of ‘Action: click_url(‘/about’) End Action’.”
1976		
1977		
1978		
1979		
1980		
1981		
1982		
1983		
1984		
1985		
1986		
1987		
1988		
1989		
1990		
1991	F.2 INTERFACES GENERATED BY ALIGN	
1992		
1993	We present the ALIGN-generated interface with Vanilla agent in ALFWorld to illustrate the interface case. For the ALIGN-generated interface with other agent methods and other environments, please refer to the supplemental materials.	
1994		
1995		
1996	ALIGN-generated interface with Vanilla agent in ALFWorld:	
1997	<pre>def InferRules(init_obs, task):</pre>	

```

1998     """
1999     Provides the rules for environment and task execute logic for
2000     different task types.
2001
2002     Args:
2003         init_obs: Initial observation string containing information
2004             about the environment
2005             task: The specific task description
2006
2007     Returns:
2008         A string describing the environment rules
2009     """
2010     return """
2011     1. Navigation and Location Rules:
2012         - You must go to a receptacle before you can examine it, open it,
2013             close it, or interact with objects in/on it.
2014         - You can only interact with objects and receptacles that are at
2015             your current location.
2016         - If you try to interact with a receptacle or object that is not
2017             at your current location, you will be informed that you need to go
2018             to that location first.
2019         - After successfully going to a location, you are at that
2020             location until you go somewhere else.
2021
2022     2. Object Interaction Rules:
2023         - To take an object, it must be present at your current location
2024             and visible (not inside a closed receptacle).
2025         - Once you take an object, it goes into your inventory and is no
2026             longer at its original location.
2027         - To move an object to a receptacle, you must have the object in
2028             your inventory and be at the target receptacle.
2029         - To use, heat, clean, cool, or slice objects, you must have the
2030             required objects in your inventory or be at their location.
2031         - You cannot take an object that is already in your inventory.
2032
2033     3. Container Rules:
2034         - Some receptacles can be opened and closed (like refrigerators,
2035             microwaves, cabinets, etc.).
2036         - You must open a closed container before you can take objects
2037             from it or put objects into it.
2038         - Objects inside closed containers are not visible or accessible
2039             until the container is opened.
2040
2041     4. Action Sequence Requirements:
2042         - Some tasks require a specific sequence of actions - for example
2043             , to heat food, you need to:
2044                 a) Go to the microwave
2045                 b) Open the microwave
2046                 c) Place the food inside
2047                 d) Close the microwave
2048                 e) Use the microwave
2049             - The environment will guide you if you're missing a prerequisite
2050                 step for an action.
2051
2052     5. Feedback Interpretation:
2053         - If an action cannot be performed, the environment will explain
2054             why and what prerequisites are needed.
2055         - The environment will inform you if you try to take an object
2056             that's already in your inventory.
2057         - The environment will inform you if you try to move an object
2058             that's not in your inventory.
2059         - Pay attention to the feedback to understand the current state
2060             of the environment and what actions are possible next.
2061         - When you successfully go to a location, the environment will
2062             describe what's there.

```

```

2052     """
2053
2054     def WrapStep(env, init_obs, task, agent_action: str, logger):
2055         """
2056         Process the agent action and return the next observation, reward,
2057         and done status.
2058
2059         Args:
2060             env: The environment object
2061             init_obs: Initial observation string containing information
2062             about the environment
2063             task: The specific task description
2064             agent_action: The action string from the agent
2065             logger: Logger object for debugging information
2066
2067         Returns:
2068             obs: Observation string after the action
2069             reward: Boolean indicating if a reward was received
2070             done: Boolean indicating if the task is complete
2071         """
2072
2073         # Track the agent's current location using an attribute on the env
2074         # object
2075         if not hasattr(env, '_current_location'):
2076             env._current_location = None
2077
2078         # Track container states (open/closed) using an attribute on the env
2079         # object
2080         if not hasattr(env, '_container_states'):
2081             env._container_states = {}
2082
2083         action_item = {
2084             'matched': False,
2085             'action': None,
2086             'object': None,
2087             'receptacle': None,
2088             'object2': None
2089         }
2090
2091         # Parse the agent action
2092         # Simple actions without parameters
2093         if agent_action.lower() == 'look' or agent_action.lower() == 'inventory':
2094             action_item['matched'] = True
2095             action_item['action'] = agent_action.lower()
2096
2097         # Pattern: go to (receptacle)
2098         elif agent_action.lower().startswith('go to '):
2099             receptacle = agent_action[6:].strip()
2100             action_item['matched'] = True
2101             action_item['action'] = 'go to'
2102             action_item['receptacle'] = receptacle
2103
2104         # Pattern: open/close (receptacle)
2105         elif agent_action.lower().startswith('open ') or agent_action.lower().startswith('close '):
2106             action = 'open' if agent_action.lower().startswith('open ') else 'close'
2107             receptacle = agent_action[len(action)+1:].strip()
2108             action_item['matched'] = True
2109             action_item['action'] = action
2110             action_item['receptacle'] = receptacle
2111
2112         # Pattern: take (object) from (receptacle)
2113         elif 'take' in agent_action.lower() and 'from' in agent_action.lower():

```

```

2106     parts = agent_action.split(' from ')
2107     if len(parts) == 2:
2108         obj = parts[0][5:].strip() # Remove 'take ' prefix
2109         receptacle = parts[1].strip()
2110         action_item['matched'] = True
2111         action_item['action'] = 'take from'
2112         action_item['object'] = obj
2113         action_item['receptacle'] = receptacle
2114
2115     # Pattern: move (object) to (receptacle)
2116     elif 'move ' in agent_action.lower() and ' to ' in agent_action.
2117     lower():
2118         parts = agent_action.split(' to ')
2119         if len(parts) == 2:
2120             obj = parts[0][5:].strip() # Remove 'move ' prefix
2121             receptacle = parts[1].strip()
2122             action_item['matched'] = True
2123             action_item['action'] = 'move to'
2124             action_item['object'] = obj
2125             action_item['receptacle'] = receptacle
2126
2127     # Pattern: examine (something)
2128     elif agent_action.lower().startswith('examine '):
2129         something = agent_action[8:].strip()
2130         action_item['matched'] = True
2131         action_item['action'] = 'examine'
2132
2133     # Determine if it's a receptacle or object by checking if it
2134     # appears in the initial observation
2135     if something.lower() in init_obs.lower():
2136         action_item['receptacle'] = something
2137     else:
2138         action_item['object'] = something
2139
2140     # Pattern: use (object)
2141     elif agent_action.lower().startswith('use '):
2142         obj = agent_action[4:].strip()
2143         action_item['matched'] = True
2144         action_item['action'] = 'use'
2145         action_item['object'] = obj
2146
2147     # Pattern: heat/clean/cool (object) with (receptacle)
2148     elif any(agent_action.lower().startswith(action) for action in [
2149         'heat ', 'clean ', 'cool ']) and ' with ' in agent_action.lower():
2150         for action in ['heat ', 'clean ', 'cool ']:
2151             if agent_action.lower().startswith(action):
2152                 parts = agent_action.split(' with ')
2153                 if len(parts) == 2:
2154                     obj = parts[0][len(action):].strip()
2155                     receptacle = parts[1].strip()
2156                     action_item['matched'] = True
2157                     action_item['action'] = action.strip()
2158                     action_item['object'] = obj
2159                     action_item['receptacle'] = receptacle
2160                 break
2161
2162     # Pattern: slice (object) with (object)
2163     elif agent_action.lower().startswith('slice ') and ' with ' in
2164     agent_action.lower():
2165         parts = agent_action.split(' with ')
2166         if len(parts) == 2:
2167             obj = parts[0][6:].strip() # Remove 'slice ' prefix
2168             obj2 = parts[1].strip()
2169             action_item['matched'] = True
2170             action_item['action'] = 'slice'

```

```

2160         action_item['object'] = obj
2161         action_item['object2'] = obj2 # Using object2 for the tool
2162         used for slicing
2163
2164         # If action wasn't matched, provide feedback
2165         if not action_item['matched']:
2166             return f"I don't understand the action '{agent_action}'. Please
2167             use one of the allowed actions from the action space.", False, False
2168
2169         logger.debug(f"parsed action: {action_item}")
2170
2171         # Get the current observation to check location
2172         test_obs, _, _, _ = env.step(['look'])
2173         test_obs = test_obs[0]
2174         logger.debug(f"Current observation after 'look': {test_obs}")
2175
2176         # Get inventory to check what objects the agent has
2177         inventory_obs, _, _, _ = env.step(['inventory'])
2178         inventory_obs = inventory_obs[0]
2179         logger.debug(f"Current inventory observation: {inventory_obs}")
2180
2181         # Improved function to check if an object is in inventory
2182         def is_in_inventory(object_name):
2183             object_name_lower = object_name.lower()
2184             logger.debug(f"Checking if '{object_name_lower}' is in inventory
2185             ")
2186
2187             # Extract inventory items from the observation
2188             inventory_items = []
2189
2190             # Check for common inventory patterns
2191             if "carrying:" in inventory_obs.lower():
2192                 carrying_section = inventory_obs.lower().split("carrying:")
2193                 carrying_section = carrying_section[1].strip()
2194                 inventory_items = [item.strip() for item in carrying_section
2195                 .split(',')]

2196                 elif "inventory:" in inventory_obs.lower():
2197                     inventory_section = inventory_obs.lower().split("inventory:")
2198                     inventory_section = inventory_section[1].strip()
2199                     inventory_items = [item.strip() for item in inventory_section
2200                     .split(',')]

2201                     # Also check line by line for inventory items
2202                     inventory_lines = inventory_obs.lower().split('\n')
2203                     for line in inventory_lines:
2204                         line = line.strip()
2205                         if line and not line.startswith(("you are", "carrying:", "inventory")):
2206                             inventory_items.append(line)

2207
2208                     logger.debug(f"Extracted inventory items: {inventory_items}")

2209                     # Check if object_name or its base name (without numbers) is in
2210                     inventory
2211                     base_name = ''.join([c for c in object_name_lower if not c.
2212                     isdigit()]).strip()

2213                     for item in inventory_items:
2214                         # Check for exact match

```

```

2214         if object_name_lower == item or f"{{object_name_lower}} (in
2215             your inventory)" == item:
2216                 logger.debug(f"Found exact match '{item}' in inventory")
2217                 return True
2218
2219             # Check for base name match (without numbers)
2220             if base_name != object_name_lower and (base_name == item or
2221                 f"{{base_name}} (in your inventory)" == item):
2222                     logger.debug(f"Found base name match '{item}' in
2223                         inventory")
2224                     return True
2225
2226             # Check if item contains the object name
2227             if object_name_lower in item:
2228                 logger.debug(f"Found partial match '{item}' containing
2229                         '{{object_name_lower}}' in inventory")
2230                 return True
2231
2232             # Check if item contains the base name
2233             if base_name != object_name_lower and base_name in item:
2234                 logger.debug(f"Found partial match '{item}' containing
2235                         base name '{{base_name}}' in inventory")
2236                 return True
2237
2238         # Direct check for common patterns in the full inventory text
2239         patterns = [
2240             f"carrying: {{object_name_lower}}",
2241             f"{{object_name_lower}} (in your inventory)",
2242             f"you are carrying: {{object_name_lower}}",
2243             f"inventory: {{object_name_lower}}"
2244         ]
2245
2246         if base_name != object_name_lower:
2247             patterns.extend([
2248                 f"carrying: {{base_name}}",
2249                 f"{{base_name}} (in your inventory)",
2250                 f"you are carrying: {{base_name}}",
2251                 f"inventory: {{base_name}}"
2252             ])
2253
2254         for pattern in patterns:
2255             if pattern in inventory_obs.lower():
2256                 logger.debug(f"Found pattern '{pattern}' in inventory
2257                         text")
2258                 return True
2259
2260         logger.debug(f"'{{object_name_lower}}' not found in inventory")
2261         return False
2262
2263     # Helper function to check if we're at a location
2264     def is_at_location(location_name):
2265         location_name_lower = location_name.lower()
2266
2267         # If we've already tracked this location, use the tracked value
2268         if env._current_location and location_name_lower in env.
2269             _current_location.lower():
2270                 logger.debug(f"Using tracked location: '{env.
2271             _current_location}'")
2272                 return True
2273
2274
2275         # Check if location is mentioned in current observation after "You are in"
2276         if "you are in" in test_obs.lower() and location_name_lower in
2277             test_obs.lower():

```

```

2268     logger.debug(f"Location '{location_name_lower}' mentioned in
2269     observation after 'You are in'")
2270     return True
2271
2272     # Check if the location is in the first line of the observation
2273     (common format)
2274     first_line = test_obs.split('\n')[0].lower()
2275     if location_name_lower in first_line:
2276         logger.debug(f"Location '{location_name_lower}' found in
2277         first line of observation")
2278         return True
2279
2280     # Check if the observation mentions items at/on the location
2281     location_patterns = [
2282         f"on the {location_name_lower}",
2283         f"in the {location_name_lower}",
2284         f"at the {location_name_lower}"
2285     ]
2286
2287     for pattern in location_patterns:
2288         if pattern in test_obs.lower():
2289             logger.debug(f"Found pattern '{pattern}' in observation")
2290     return True
2291
2292     logger.debug(f"Not at location '{location_name_lower}'")
2293     return False
2294
2295     # Handle go to action
2296     if action_item['action'] == 'go to':
2297         receptacle = action_item['receptacle']
2298         receptacle_lower = receptacle.lower()
2299
2300         # Check if we're already at this location
2301         if is_at_location(receptacle_lower):
2302             env._current_location = receptacle
2303             return f"You are already at the {receptacle}. You can
2304             interact with it directly.", False, False
2305
2306         # Execute the go to action
2307         obs, reward, done, info = env.step([agent_action])
2308         obs, reward, done = obs[0], info['won'][0], done[0]
2309
2310         # Update the current location if the action was successful
2311         if obs and "nothing happens" not in obs.lower():
2312             env._current_location = receptacle
2313
2314             # If the observation doesn't clearly indicate arrival,
2315             # enhance it
2316             if not any(phrase in obs.lower() for phrase in [f"you arrive
2317             at", f"you are at", f"you see {receptacle_lower}"]):
2318                 obs = f"You arrive at the {receptacle}. {obs}"
2319             else:
2320                 # Provide more informative feedback
2321                 obs = f"Cannot go to {receptacle}. It might not be a valid
2322                 location or not accessible from here."
2323
2324             return obs, reward, done
2325
2326     # Handle examine, open, close, take from, move to actions that
2327     # require being at location
2328     if action_item['action'] in ['examine', 'open', 'close', 'take from',
2329     'move to']:
2330         receptacle = action_item['receptacle'].lower() if action_item['
2331         receptacle'] else ""

```

```

2322     logger.debug(f"Action: {action_item['action']} with receptacle:
2323     {receptacle}")
2324
2325     # Skip location check for examining objects in inventory
2326     if action_item['action'] == 'examine' and action_item['object'] and
2327     is_in_inventory(action_item['object']):
2328         # Execute the examine action directly
2329         obs, reward, done, info = env.step([agent_action])
2330         obs, reward, done = obs[0], info['won'][0], done[0]
2331         return obs, reward, done
2332
2333     # Check if we need to be at a receptacle and if we're there
2334     if receptacle and not is_at_location(receptacle):
2335         action_name = action_item['action']
2336         if action_name == 'examine':
2337             return f"You must go to the {action_item['receptacle']} first
2338             before examining it.", False, False
2339         elif action_name == 'take from':
2340             return f"You need to go to the {action_item['receptacle']} first
2341             before taking objects from it.", False, False
2342         elif action_name == 'move to':
2343             return f"You need to go to the {action_item['receptacle']} first
2344             before placing objects on/in it.", False, False
2345         else: # open or close
2346             return f"You need to go to the {action_item['receptacle']} first
2347             before you can {action_name} it.", False, False
2348
2349     # Handle open and close actions to track container states
2350     if action_item['action'] in ['open', 'close']:
2351         receptacle = action_item['receptacle']
2352
2353         # Execute the action
2354         obs, reward, done, info = env.step([agent_action])
2355         obs, reward, done = obs[0], info['won'][0], done[0]
2356
2357         # Check for "Nothing happens" and provide more informative
2358         # feedback
2359         if obs.strip() == "Nothing happens.":
2360             if action_item['action'] == 'open':
2361                 return f"Unable to open {receptacle}. It might already
2362                 be open or not be openable.", reward, done
2363             else: # close
2364                 return f"Unable to close {receptacle}. It might already
2365                 be closed or not be closable.", reward, done
2366
2367         # Update container state tracking
2368         if "successfully" in obs.lower() or "already" in obs.lower():
2369             env._container_states[receptacle.lower()] = 'open' if
2370             action_item['action'] == 'open' else 'closed'
2371
2372         return obs, reward, done
2373
2374     # Check if taking an object that's already in inventory
2375     if action_item['action'] == 'take from':
2376         object_name = action_item['object']
2377         if is_in_inventory(object_name):
2378             return f"You already have the {object_name} in your
2379             inventory. No need to take it again.", False, False
2380
2381     # Check if moving an object that's not in inventory
2382     if action_item['action'] == 'move to':
2383         object_name = action_item['object']
2384         if not is_in_inventory(object_name):
2385             return f"You don't have the {object_name} in your inventory.
2386             You need to take it first.", False, False

```

```

2376
2377     # Execute the action in the environment
2378     logger.debug(f"Executing action in environment: {agent_action}")
2379     obs, reward, done, info = env.step([agent_action])
2380     obs, reward, done = obs[0], info['won'][0], done[0]
2381     logger.debug(f"Environment response: {obs}")

2382     # Handle special case for "Nothing happens" response
2383     if obs.strip() == "Nothing happens." and action_item['action'] == 'take from':
2384         object_name = action_item['object']
2385         receptacle_name = action_item['receptacle']

2386         # Check if it might be because the object is already in
2387         # inventory
2388         if is_in_inventory(object_name):
2389             return f"You already have the {object_name} in your
2390             inventory. No need to take it again.", reward, done

2391         # Check if it might be because the container is closed
2392         receptacle_state = env._container_states.get(receptacle_name.
2393             lower())
2394         if receptacle_state == 'closed':
2395             return f"You need to open the {receptacle_name} first before
2396             taking objects from it.", reward, done

2397         # Otherwise, the object might not be there
2398         return f"There is no {object_name} at the {receptacle_name} to
2399             take. It might be elsewhere or already taken.", reward, done

2400     # Handle special case for "Nothing happens" response for move action
2401     if obs.strip() == "Nothing happens." and action_item['action'] == 'move to':
2402         object_name = action_item['object']
2403         receptacle_name = action_item['receptacle']

2404         # Double-check if the object is in inventory
2405         if is_in_inventory(object_name):
2406             # If object is in inventory but move fails, check if
2407             # receptacle is closed
2408             receptacle_state = env._container_states.get(receptacle_name.
2409                 lower())
2410             if receptacle_state == 'closed':
2411                 return f"You need to open the {receptacle_name} first
2412                 before placing objects in it.", reward, done
2413             else:
2414                 return f"Unable to move {object_name} to {
2415                 receptacle_name}. Make sure the receptacle is open if it's a
2416                 container.", reward, done
2417             else:
2418                 # If object is not in inventory, provide clear feedback
2419                 return f"You don't have the {object_name} in your inventory.
2420                 You need to take it first before moving it.", reward, done

2421     # Handle other "Nothing happens" cases with more informative
2422     # feedback
2423     if obs.strip() == "Nothing happens.":
2424         if action_item['action'] == 'open':
2425             return f"Unable to open {action_item['receptacle']}. It
2426             might already be open or not be openable.", reward, done
2427             elif action_item['action'] == 'close':
2428                 return f"Unable to close {action_item['receptacle']}. It
2429                 might already be closed or not be closable.", reward, done
2430             elif action_item['action'] == 'examine':
2431                 if action_item['object']:

```

```

2430
2431         return f"Unable to examine {action_item['object']}. Make
2432         sure you have it in your inventory or it's visible at your location
2433         .", reward, done
2434     else:
2435         return f"Unable to examine {action_item['receptacle']}.
2436         Make sure you're at the right location and it's visible.", reward,
2437         done
2438     elif action_item['action'] == 'use':
2439         return f"Unable to use {action_item['object']}. Make sure
2440         you have it in your inventory or it's at your current location and
2441         usable.", reward, done
2442     elif action_item['action'] in ['heat', 'clean', 'cool', 'slice'
2443     ]:
2444         return f"Unable to {action_item['action']} {action_item['
2445         object']}. Make sure you have all required objects and are at the
2446         right location.", reward, done
2447     elif action_item['action'] == 'go to':
2448         # This case should be handled earlier, but as a fallback
2449         return f"Cannot go to {action_item['receptacle']}. It might
2450         not be a valid location in this environment.", reward, done
2451     else:
2452         # Generic clarification for other actions
2453         return f"Action '{agent_action}' resulted in no effect.
2454         Check if you have all prerequisites or if the action is valid in
2455         this context.", reward, done
2456
2457     # For successful move actions, verify the object was actually in
2458     # inventory
2459     if "successfully" in obs.lower() and "place" in obs.lower() and
2460     action_item['action'] == 'move to':
2461         object_name = action_item['object']
2462         # If the environment says the move was successful, we should
2463         # trust that and not override
2464         return obs, reward, done
2465
2466     return obs, reward, done
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483

```