

000 001 002 003 004 005 LUMINA: LONG-HORIZON UNDERSTANDING 006 FOR MULTI-TURN INTERACTIVE AGENTS 007 008 009

010 **Anonymous authors**
011 Paper under double-blind review
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027

ABSTRACT

028 Language models can excel at a variety of tasks (e.g., mathematical reasoning and
029 coding) which are fundamental to solving more general goal-oriented feedback-
030 driven agentic problems. However, based on recent findings, two key points are
031 evident: (a) agentic problems require a variety of skills such as long-context rea-
032 soning, planning and decision making, and efficient exploration; (b) even large
033 frontier models under-perform in these family of tasks, especially in problems re-
034quiring long-horizon understanding. For example, GPT-4o has a 48.8% success
035 rate on the AppWorld benchmark. In this paper, our goal is to understand the
036 relation between the two, by examining which skills are necessary for solving
037 multi-turn problems. We work towards this goal using an oracle counter-factual
038 framework that allows us to answer the question: what if the agent could leverage
039 a specific oracle skill to achieve its goal? To enable this framework, we intro-
040 duce a set of procedurally-generated game-like tasks whose complexity can be
041 controlled. For these controlled environments, we can provide accurate oracle in-
042 terventions to guide the agent towards the goal. Our findings suggest that while
043 most interventions (e.g., planning) are generally beneficial, for some interventions
044 the utility depends on the intricacies of the benchmark (e.g., ability to track state
045 while iteratively modifying python lists).
046

1 INTRODUCTION

047 Large Language Models (LLMs) have demonstrated exceptional performance across a wide range of
048 tasks, including natural conversations (Touvron et al., 2023; Achiam et al., 2023), question answer-
049 ing (Yang et al., 2018), competitive coding (Austin et al., 2021; White et al., 2024), and mathemat-
050 ical reasoning (Comanici et al., 2025; Guo et al., 2025). Consequently, given their general-purpose
051 capabilities, a natural question that is actively being explored is whether language models can be
052 leveraged as *multi-turn* agents, i.e., whether they can iteratively perceive, reason, and take strate-
053 gic actions towards achieving a distant goal. Such tasks introduce a host of new challenges, such
054 as maintaining coherence over multiple turns and long contexts, reasoning over multiple dynamic
055 pathways, recovering from errors, identifying the right tools for the task, and efficiently tracking
056 state and progress without explicit feedback.
057

058 How capable are current language models as agents? To answer this, the community is actively
059 working on numerous benchmarks to quantitatively analyze multi-turn agent capabilities across var-
060 ious domains, ranging from function calling (Patil et al., 2025), web navigation (Koh et al., 2024),
061 interactive coding (Trivedi et al., 2024), human interaction (Liu et al., 2023), and game-playing
062 (Guertler et al., 2025). Ongoing results suggest that there is plenty of progress to be made, e.g., on
063 Appworld (Trivedi et al., 2024), GPT-4o has a success rate of 48.8% and open-weight models such as
064 LLama3-70B (Grattafiori et al., 2024) achieve 24.4%. As many of the benchmark analysis reports,
065 pushing progress requires enabling numerous skills, such as efficient task decomposition, planning,
066 state tracking, and information gathering. However, it is largely unclear on which of these skills (or
067 combination thereof) are the bottleneck to make progress towards capable multi-turn agents.
068

069 In this paper, our goal is to critically examine and understand skills that enable progress towards
070 general-purpose long-horizon agents. To help us understand, we propose an oracle intervention
071 framework that helps us evaluate the importance of skills by asking counterfactual questions. The
072 framework helps us gauge agent’s performance improvement when assisted by a skill-specific or-
073

acle or a combination of multiple skills. Using this framework, we investigate three oracle interventions: planning, tracking belief state, and context reformulation. However, constructing oracle interventions on real-world benchmarks (which typically involve human annotation and verification) is cumbersome and not straight-forward. Since multi-turn tasks admit numerous valid dynamic paths towards reaching the goal, annotating oracle solutions becomes intractable. To make oracle interventions tractable, we additionally propose a set of procedurally-generated game-playing environments where optimal actions and strategies can be reliably computed at any step of the roll-out. Specifically, we consider three environments: `ListWorld` (to evaluate multi-turn list modification capabilities), `TreeWorld` (to evaluate multi-turn graph traversal), and `GridWorld` (to evaluate 2D spatial navigation). All the environments are configurable, and importantly, enable us to inject accurate oracle information at any point of the agent’s trajectory.

Our framework enables us to examine multi-turn agents at long-horizon tasks along multiple dimensions (e.g., task complexity, model size, influence of specific oracle skills) and helps us provide a number of insights. First, we observe a significant discrepancy with the low success rates over long-horizon trajectories in spite of high accuracy per-step (i.e., whether action is one of the optimal actions). For example, in tree search problems, we observe very low success rates ($<10\%$ in this case) despite high per-step accuracy ($>80\%$). This indicates that a dominant factor in difficulty of multi-turn environments is the fact that success in the task requires many correct steps, and even a small probability of error in each step prohibitively hinders task success. Apart from compounding errors (Sinha et al., 2025; Li et al., 2025), we also attribute making terminal errors (e.g., premature termination) contributing to the discrepancy. Second, we can leverage oracle interventions to understand contributions of specific skills that best contribute to improving success over multi-turn tasks. Here, we find that although oracle interventions generally help improve success rates, the degree to which they help significantly depends on other factors. For instance, while optimally pruning context (containing action-observation interaction history) helps smaller models ($\leq 8B$ parameters), it also shows to be counter-productive for larger models. Another factor that determines improvements is unsurprisingly the task itself: tasks that rely on accurately tracking belief state (e.g., those involving tracking hidden state) benefit the most from relevant state-tracking skills, while other tasks such as spatial navigation benefit heavily from planning. Overall, our findings present a double-edged picture: while improving specific skills (enabled by oracles interventions in our case) generally help the LLM-based agents in multi-turn, fully bridging the gap likely requires exploiting environment and model-specific understanding.

2 RELATED WORKS

LLMs and Agents Agent-based systems have a long history (Russell et al., 1995) and can be defined by an agent (the policy) perceiving and interacting with an environment towards achieving a goal and, in turn, receiving a reward. Recent literature demonstrates that capable language models can serve many roles within such systems, such as modeling the policy (Huang et al., 2022; Yao et al., 2023b), the environment as a world model (Hao et al., 2023), or the reward (Zheng et al., 2023; Zhang et al., 2025). We specifically focus on leveraging the LLM as a policy, which based on trajectory auto-regressively samples the next action. A notable and representative example is ReAct-based prompting (Yao et al., 2023b), which interleaves chain-of-thought thinking and taking task-specific actions at each step. ReAct prompting has been shown to be highly successful in a variety of domains, ranging from playing games (Wang et al., 2023a) to interactive coding agents (Trivedi et al., 2024). In this work, we use ReAct-based prompting to elicit dynamic reasoning and planning behavior from an LLM.

LLM Agent capabilities What makes for a good Language model agent? While models need to be fundamentally capable of language understanding and complex reasoning, a number of skills are required beyond this. The policy interacts with the environment with a set of admissible actions (e.g., tools, function calls) and hence need to be capable of calling functions (Qin et al., 2023; Patil et al., 2024) with appropriate arguments. Since tasks admit multiple paths towards the goal, the agents need to also be capable of multi-path reasoning (Besta et al., 2024; Yao et al., 2023a) and re-planning (Song et al., 2023) to revise actions in light of dynamic environmental feedback. Since decision making involves reflecting short-term (e.g., episode history in context window) and long-term (e.g., external storage) memory (Song et al., 2023; Huang et al., 2023; Wang et al., 2023b)

108 also plays an important role. The sequence of actions taken by the agent also modify the (hidden)
 109 state in the environment, and the models need to be adept in tracking its state (Ebrahimi et al., 2024;
 110 Vodrahalli et al., 2024). As enumerated above, a number of capabilities appear to play a crucial role
 111 towards enabling agentic use-cases.
 112

113 **Characterizing Bottlenecks for Agents** How can we assess the effectiveness of models in agentic
 114 tasks? One approach is to conduct holistic evaluations, while another is to analyze the contribution
 115 of specific capability dimensions, such as long-context reasoning, to marginal performance gains.
 116 Towards a holistic evaluation, a number of benchmarks exist (Trivedi et al., 2024; Patil et al., 2024) to
 117 evaluate an agent’s capabilities. These benchmarks generally show a common trend: large models
 118 tend to significantly outperform smaller models. This motivates the question in our work: what
 119 is the bottleneck that leads to the performance discrepancies. Towards understanding this, a few
 120 works exist on characterizing bottlenecks of capabilities of language models over multi-turn tasks.
 121 Abdulhai et al. (2023) work towards this goal by understanding the influence of RL algorithms in
 122 strategic game-playing tasks (e.g., maze). Cemri et al. (2025) study error taxonomy of multi-agent
 123 systems across a range of popular benchmarks, such as AppWorld (Trivedi et al., 2024). Concurrent
 124 to our work, Sinha et al. (2025) study bottlenecks by isolating execution capabilities in long-horizon
 125 tasks. Similar to prior works, our goal too is to characterize bottlenecks of multi-turn agents by
 126 isolating capabilities. However, in contrast, we study the bottlenecks in procedurally-generated
 127 game-playing environments, which allows us to enable oracle interventions.
 128

3 FORMULATION: LUMINA

130 In this section, we begin by detailing the underlying process that requires an agent to perform se-
 131 quential decision-making to complete the task. To better enable the agent complete the task, we then
 132 elaborate on how to augment information at each turn using oracle interventions.
 133

134 **POMDP Tasks** We study tasks that can be modeled as a Partially-observable Markov Decision
 135 Process: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, H, S_{\text{Goal}} \rangle$ where \mathcal{S} is the hidden state space, \mathcal{A} the agent’s action
 136 space, and \mathcal{O} is the observation space. Furthermore, $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function
 137 (deterministic in our case) and $\Omega : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{O}$ is the observation function. The termination can be
 138 performed either by the agent (e.g., DONE action) or by the environment (e.g., $t \geq \text{horizon } H$). For
 139 simplicity, we consider terminal reward function: the agent receives reward of 1 if it terminates at
 140 the goal state S_{Goal} within at most H steps. The objective of the agent is to maximize the probability
 141 of success, which we refer to as the *success rate* and denote by $J(\pi_\theta)$ for an agent π_θ .
 142

143 **Base (ReAct) Policy Agent** Towards taking sequential decisions to solve the task (represented as
 144 text x , we consider a stochastic policy modeled by a ReAct (Yao et al., 2023b) LLM agent π_θ where
 145 at each step t :

$$a_t \sim \pi_\theta(\cdot | x, h_{t-1}).$$

146 Here, $h_{t-1} := (a_1, o_1, \dots, a_{t-1}, o_{t-1})$ is the history of the past interactions between the agent and
 147 the environment. Action a_t consists of both a chain-of-thought text (which is irrelevant to the envi-
 148 ronment) and one of the allowed operations.
 149

150 **Oracle Interventions** To help our understanding and isolate factors to determine what the bottle-
 151 neck is, we consider oracle interventions to assist policy π_θ by augmenting auxiliary information.
 152 First, we establish the existence of an oracle \mathcal{O} that, given the prompt x and the context h_{t-1} is able
 153 to accurately recover the belief state of the POMDP. Then, we consider the policy π at every step is
 154 conditioned on **oracle-augmented history** \tilde{h}_t :
 155

$$a_t \sim \pi_\theta(\cdot | x, \tilde{h}_{t-1}),$$

$$\tilde{h}_{t-1} = \mathcal{O}^{\text{history}}(h_{t-1} \oplus \mathcal{O}^{\text{plan}}(x, h_{t-1}) \oplus \mathcal{O}^{\text{state}}(x, h_{t-1})).$$

156 Generally, our oracle formulation accommodates appending (\oplus) a hint for the next step of an optimal
 157 plan (via $\mathcal{O}^{\text{plan}}$) and a summary of the belief state (via $\mathcal{O}^{\text{state}}$) to the history, as well as representing
 158 the context compactly by pruning the history of redundant information (via $\mathcal{O}^{\text{history}}$). We elaborate
 159 on the details of each of these in the following paragraphs.
 160

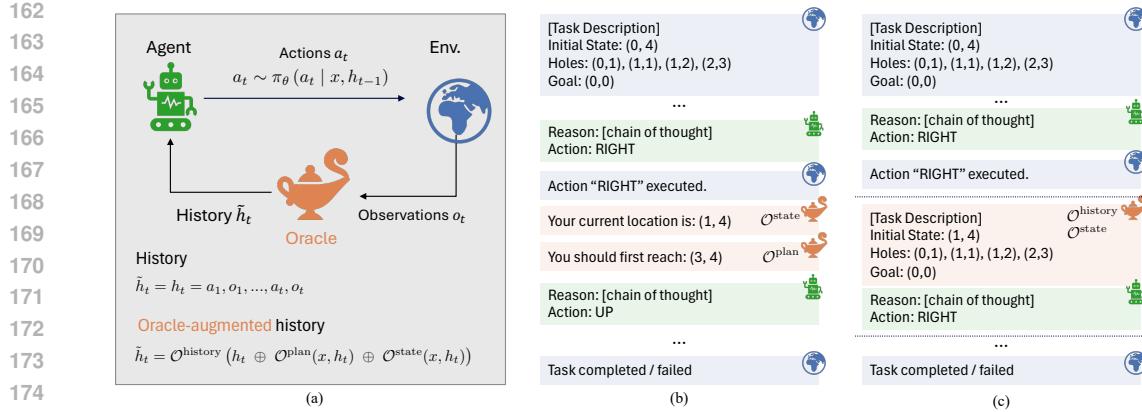


Figure 1: Formulation. (a) **Oracle-augmented history.** Within multi-turn tasks, we study LLM-based agents π_θ when additionally assisted by an oracle. We can leverage one or more oracles to modify the history h_t (context for language model). (b) **GridWorld example.** In this example, the agent needs to navigate from an initial 2D location to a goal location. We can use oracle $\mathcal{O}^{\text{state}}$ to summarize the current location (instead of the model reflectively reasoning at each turn). Similarly, we can also use $\mathcal{O}^{\text{plan}}$ to hint way points to reach the goal. (c) **History pruning.** Since we consider Markov decision processes, $\mathcal{O}^{\text{history}}$ can be used to rewrite the task description such that the actions can be taken independent to previous steps.

Planning $\mathcal{O}^{\text{plan}}$ As apparent from the POMDP formulation, solving multi-turn environments requires reasoning about many steps into the future and devising a plan towards completing the task. At every step t , the planning intervention $\mathcal{O}^{\text{plan}}(x, h_{t-1})$ is the description of a single-turn subtask. This subtask is designed not to require planning (no reasoning about the future steps needed). Most importantly, the action that accomplishes this subtask is one of the optimal actions in the environment at that moment.

State Tracking $\mathcal{O}^{\text{state}}$ Solving partially-observable long-horizon tasks requires the agent to accurately track its knowledge about the hidden state of the environment at every step. This is highly challenging, since at each turn the agent needs to collect the information implicitly from its history of interactions with the environment and reason about the environment transitions. Consequently, we consider an oracle belief function that accurately summarizes the current knowledge of the agent (e.g., current location in GridWorld) in a compact form that is easy to parse.

History Pruning $\mathcal{O}^{\text{history}}$ It is well-known that the performance of LLMs degrade as the size of the context (history h_t in our case) grows. More relevant to us, existing work (Laban et al., 2025; Vodrahalli et al., 2024) highlights that the performance at the same task drops merely due to the presence of distractors. This is a common problem in multi-turn LLM agentic tasks, where the history contains overcomplete information to guide the agent towards the goal, and since the size of the context (history h_t) grows at each turn, makes decision making more error prone. To mitigate the influence of distractors, we consider an oracle $\mathcal{O}^{\text{history}}$ that reduces the contents of the context into a compact form. In this work, we consider the simple implementation that can be done when state tracking is present. In this case, we drop the old history h_{t-1} , since when the compact summary $\mathcal{O}^{\text{state}}(x, h_{t-1})$ is given, h_{t-1} is not necessary.

Step vs Task Metric J_{step} In multi-turn environments, the success rate can often be low because agents are required to take the correct action at each step consistently. This challenge is compounded by the unique difficulties inherent in such environments, such as planning and state tracking. For example, even if the environment involves solving single-turn step-by-step reasoning problems, the performance can suffer due to the dependency on the chain, irrespective of the absence of planning. However, there are cases where this setup can be advantageous by offering the agent retries; for instance, in scenarios where solving a fraction of tasks is enough for success. To measure this aspect, we define an objective over each step. We call a step accurate if the action taken is optimal for that step. We refer to this metric as *step accuracy*. The distinction between step accuracy and success rate lies in how forgiving the environment is to mistakes.



Figure 2: Environments. In this work, we study the influence of oracle interventions in three unique environments. In all cases, the agent reasons (shown in gray italic) and performs an action (shown in monospace), and the environment provides minimal but sufficient feedback to help the agent progress towards the goal. **(a) List World:** which requires modifying a python list using only `pop (idx)` actions; **(b) Tree World:** where the task is to iteratively search over a tree to find a specific node; and **(c) Grid World:** where the agent needs to move from an initial location to a goal location.

4 EXPERIMENTAL RESULTS

In this section, we first present the programmatically generated environments and tasks used in our experiments. Then, we walk through the implementation details and conclude by reporting our findings.

4.1 PROCEDURALLY-GENERATED MULTI-TURN ENVIRONMENTS

Our goal is to characterize bottlenecks of multi-turn agents by leveraging oracle interventions. Existing benchmarks fall short for this task, since data are predominantly human-generated and are rarely accompanied with trajectory-level annotation. Some works (e.g., Trivedi et al. (2024)) investigate marginal gains through oracle interventions, however in a very narrow scope that is admissible within the dataset. Consequently, we propose procedurally-generated multi-turn environments with the following requirements: (a) Minimal external knowledge: such that all necessary information can be specified in the prompt; (b) Simple action space: to prevent failures from constructing complex function calls; (c) Variable complexity: to enable us analyze success by varying the complexity of the task in a procedural manner; (d) Compositionality: such that tasks can be programmatically and accurately broken down into clear subproblems; (d) No data contamination: since the tasks are novel and can additionally be randomly re-generated, there is little risk from contamination; and most importantly, (e) Oracle interventions: since we know the underlying process at any instant, we can faithfully construct various flavors of oracle interventions.

General Framework All our environments can be cast as a Partially Observable Markov Decision Process (POMDP). Given an initial task x that can be communicated verbally, the agent needs to complete the task with a finite turn budget $T_{\max} \geq mT^* + n$, where T^* is the number of actions required by an optimal policy. To achieve the goal, the agent interacts with the environments using a simple set of actions (e.g., up, down). All environments have a common terminating action `done`, which the agent needs to invoke once it completes the objective. The environment provides minimal essential feedback (e.g., ‘move successful’) at each turn.

List World Inspired by Vodrahalli et al. (2024), we introduce ListWorld to evaluate the ability of an LLM agent to sequentially modify and track the state of an initial object. Specifically, the task of the agent is to prune an initial `input` Python list to a smaller `target` list. The agent needs to prune using a single action: `pop (index)`. The agent has to pop the elements from left to right: once an element is popped, it becomes illegal to pop the elements before it. We control the task

270 complexity by varying the number of elements that need to be pruned (i.e., `len(initial)` –
 271 `len(target)`). To complete the task successfully, the agent at each turn needs to: (a) determine
 272 current list by accounting initial list and historical actions; (b) find the candidates to prune; and
 273 (c) pop the corresponding candidate. This introduces a subtle challenge of understanding partial
 274 changes to the index-value mappings after every successful pop operation. Furthermore, the agent
 275 needs to carefully reflect before each action, since pruning an unnecessary element places the agent
 276 into an irrecoverable state leading to immediate termination.

277

278 **Tree World** In this environment, we study an agent’s ability to sequentially explore and gather
 279 information at each turn. Specifically, the task is tree traversal: the agent needs to navigate from a
 280 source node to a target node to find the node containing a particular value. The nodes (except source
 281 and target) and edges are unknown to the agent. For simplicity and ease of analysis, we consider
 282 traversing from the root to a leaf node of a tree. The agent needs to traverse the tree using a single
 283 action: `get_children(node_id)`. Efficiently completing this task requires the agent to keep
 284 track of the frontier of unexplored nodes and sequentially explore them. We vary the complexity
 285 of the task by controlling the number of nodes in an m -ary tree. This task is partially inspired by
 286 interactive coding problems (Trivedi et al., 2024), which require an agent to navigate a new library
 287 documentation (tree topology) to find the right function to invoke.

288

288 **Grid World** We also consider a 2D grid world environment to study an agent’s ability to plan and
 289 spatially navigate towards a goal. Our grid world takes the form of an $N \times N$ grid with holes, where
 290 stepping into the holes incurs an additional cost. The environment is fully observable, with each task
 291 stating the agent’s start position and the goal position. The agent needs to navigate to the goal using
 292 one of four actions (`up()`, `down()`, `left()`, or `right()`) and reach the goal within a specified
 293 cost budget. To successfully complete the task, the agent needs to: (a) understand and reason spatial
 294 structure of the environment; (b) reason carefully to plan a trajectory avoiding holes; (c) keep track
 295 of progress towards the goal.

296

297 **Oracle Specifications** In all the environments above, the oracle is designed either to *augment*
 298 the information provided (e.g., by providing a plan or summarizing the belief state) or *truncate* to
 299 sufficient information necessary to achieve the goal. When we truncate the context, we signal it by
 300 rewriting the task specification and providing the current state as the initial state (e.g., current 2D
 301 location in Grid world) and discarding the previous history. A specific benefit in our environments
 302 is that we evaluate the optimal plan from *any* state that the environment is in.

303

4.2 SETUP: LLM AGENTS

304

305 In this section, we walk through the models we used for evaluation, how the prompts were designed
 306 to elicit closed-loop interactions with the environment, and also discuss evaluation metrics.

307

308 **Models** We ran all our experiments using the Qwen-3 (Yang et al., 2025) family of models as the
 309 policy model π_θ . This family of models is appealing for our understanding multi-turn scenarios
 310 for two reasons: (i) it enables us to study performance over a range of different model sizes (we
 311 focus on 4B - 32B); and (ii) the models are already pre-trained on multi-turn interaction cycles,
 312 during the RL stage (Yang et al., 2025), and hence they are well-suited for our analysis. We report
 313 all findings by running inference in non-thinking mode (but with reasoning traces using chain-of-
 314 thought prompting) with the context length limit set to 32K and recommended sampling temperature
 315 of 0.7. We also ran preliminary experiments with thinking mode, but we found lower success rates
 316 with most failure cases due to hitting token limits. We use the ReAct (Yao et al., 2023b) framework
 317 to perform the roll-outs by interleaving reasoning traces with actions. The number of turns for which
 318 the roll-outs are performed is example dependent. Since many of our experiments are long-horizon
 319 and do not fit into the context length, we report results for such scenarios (involving contexts $> 32K$)
 320 using YaRN (Peng et al., 2023) encoding, following the official recommendation.

321

322 **Prompting** Across all environments, we engineer environment-specific prompts to ensure best
 323 success of the pre-trained models. This helps us ensure that at evaluation time, errors can be at-
 tributed with high confidence to limitations of the model rather than prompt construction. Specif-
 ically, we: (a) use in-context example trajectories, which demonstrate task-specific reasoning and

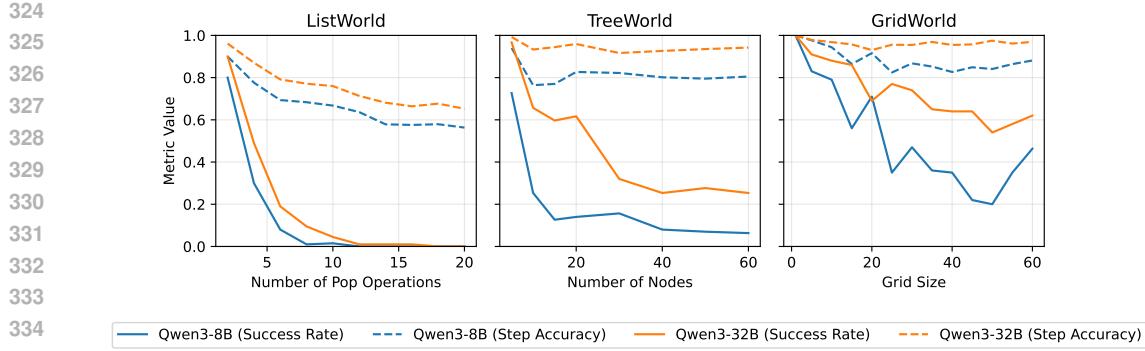


Figure 3: Success rate and step accuracy of Qwen3-8B and Qwen3-32B models by task horizon in ListWorld (left), TreeWorld (middle), and GridWorld (right).

actions; (b) ensure the in-context examples reflect the information augmented by the oracles during roll-outs. We found the latter to be especially important, as models underperformed if the format of in-context examples did not appear consistent with environmental feedback during roll-out.

Evaluation Metrics Our primary evaluation metric is the success rate, i.e., whether the model completes the task within the specified horizon budget. In addition, in some cases we report the step accuracy. In this context, a step is “accurate”, if it is optimal. Since for the environments we consider, optimal policies are generally not unique. A step is accurate if it is aligned with at least one optimal policy of the current state.

4.3 RESULTS AND ANALYSIS

We examine the performance of Qwen-3 models on our environments. Figure 3 shows the success rate of the Qwen3-8B and Qwen3-32B models on our three environments as a function of the task horizon. We observe that despite the strong performance on problems with a short horizon, the success rate drops drastically as we increase the horizon. This is aligned with the notorious challenging nature of long-horizon tasks. We now investigate the main driving forces behind this phenomenon with our framework.

The dependence of task success on being correct over many steps is the main reason for low success rate in multi-turn environments. The dashed lines in Fig. 3 show the step accuracy of the model for each value of task horizon. We can see that the step accuracy is much larger than the task success rate. Even in cases where the success rate is almost zero, the step accuracy stays above 60%. This means that even in the most challenging problem instances, the model is taking the correct action in the majority of steps. However, as the number of required steps increases, the agent becomes more likely to fail due to the occasional wrong actions it takes. In order to solve these tasks reliably, the agent needs to be almost perfect at all steps. As we have discussed, this is highly challenging in multi-turn environments due to challenges of state-tracking, planning, and growing prompt length.

To better understand the impact of each challenge of multi-turn environments, we conduct a thorough evaluation of the models across all sizes in the presence or absence of our oracle interventions. Each of these interventions removes one of the aforementioned challenges, and their impact on the agent’s success rate allows us to understand the importance of each one.

Planning and state tracking interventions can be activated or deactivated independently. In cases where state-tracking intervention is active, the provided state contains sufficient information for the agent’s decision-making. In these cases, we can choose to apply history pruning to reduce the content in the model’s context window. Therefore, we have six possible configurations for the oracle intervention. Figure 4 provides the success rate of 4B, 8B, 14B, and 32B models in each environment for all six oracle configurations, averaged over complexity levels. For each environment, we also provide the aggregate results averaged over the four model sizes. The addition of each intervention

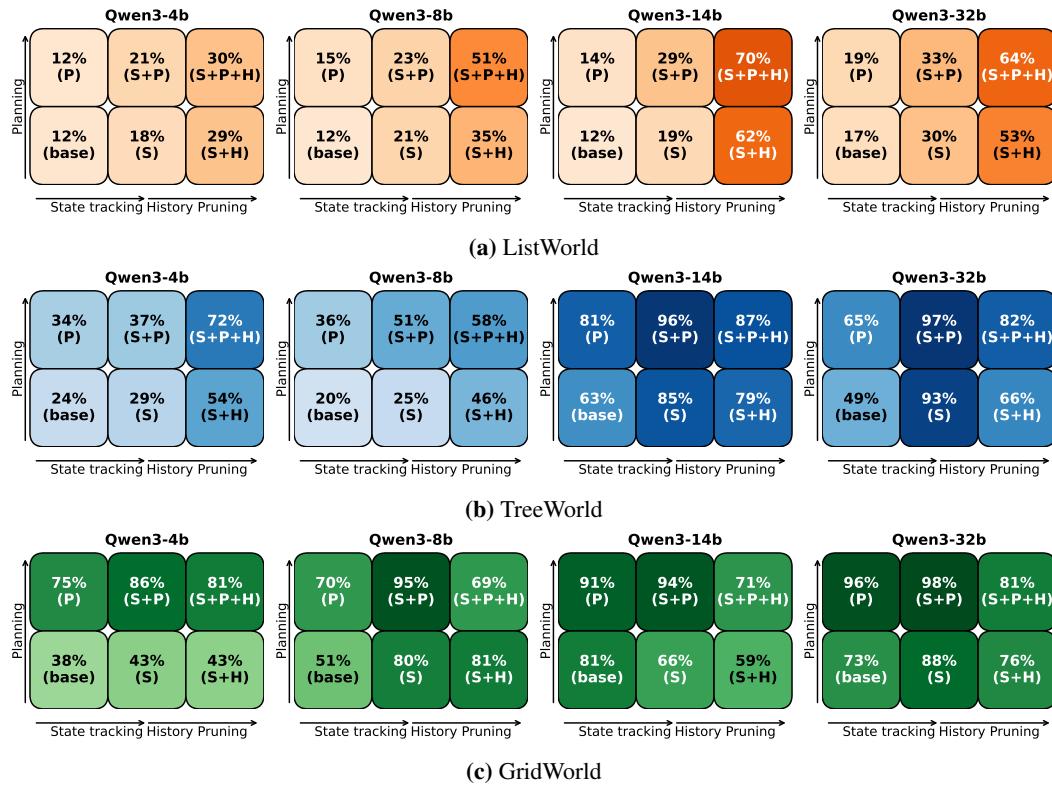


Figure 4: Influence of oracle interventions. Results are averaged over all horizon lengths. The labels indicate the active oracles (S: state tracking, P: planning, and H: history pruning) compared to the base model.

improves the success rate in most cases. The exact comparison of the impact of each challenge varies among the environments and model sizes.

To better understand how the bottleneck factor of models varies among different sizes, we look into the impact of each oracle intervention at that size. For a fixed benchmark, the success rate of bigger models is generally larger than the smaller models. Since the success rate is upper bounded by 100%, it means that the oracle interventions have less room to improve the success rate of bigger models and their impact will naturally be smaller. To enable an insightful comparison of the challenges across size categories, we utilize the programmability of our environments to adjust for this natural imbalance. We pick a longer horizon for larger models such that all models' success rates become similar. For ListWorld and TreeWorld, we challenge each model to succeed only 30% of the times. In GridWorld, larger models never reach this low success rate, and we pick the complexity such that success rate becomes about 75% for all model sizes. We present this comparison in Figure 5.

In the left plot of Figure 5, we provide the change in success rate due to each intervention for each model size. We average over the choice of environment and the presence/absence of other interventions. **The most drastic difference between small and larger models is in the context processing.** While the 4B and 8B models immensely benefit from removing the irrelevant parts of the context, the 14B and 32B even suffer from this removal. This indicates that for larger models, the growing context is less of an issue, and it even helps the agent. The second observation we make is that state tracking becomes more and more a bottleneck in larger models. We interpret this as other challenges are better overcome by size but state tracking remains challenging. We don't observe any specific trends in planning and find the difference between task success rate and step accuracy uniform among sizes.

Figure 5 (right) compares the impact of each intervention in different environments. We use the same method to adaptively choose the task complexity and average over the choice of model size and the presence of other interventions. **We observe that the relative severity of challenges significantly varies among environments.** In ListWorld, history pruning and choosing step accuracy introduce

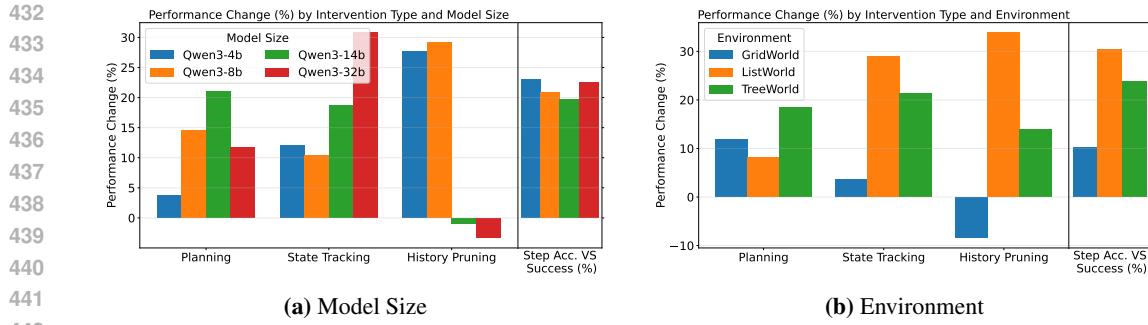


Figure 5: Performance change from interventions. The impact of each intervention and varies depending on the model size (a) and environment (b). The relationship between step accuracy and success (%) also varies.

the largest boost in performance. This can be explained by the fact that in complex cases, the task’s list is significantly long and keeping it in the recent messages helps the model to attend to it. Also, among our environments, ListWorld is the only environment that is completely irrecoverable. A single wrong action leads to task failure, hence the largest drop from step accuracy to task success rate is observed in ListWorld.

Gridworld’s most demanding skill is shown to be planning, which is reasonable due to the navigation nature of the task. We find state tracking and history pruning to have a minimal benefit and even a negative impact in GridWorld. We hypothesize that this is due to the simple state update rule that can be effectively done from the context. Lastly, TreeWorld appears to be mostly demanding state tracking, perhaps due to the need to backtrack during the tree traversal. The drop from step accuracy to task success rate is also large, likely due to frequent agent’s premature termination.

5 CONCLUSION

In this paper, we worked towards understanding the discrepancy of LLM performance between excelling at a range of complex single-turn reasoning tasks and underperforming in multi-turn closed-loop feedback-driven tasks. Our key insight was to ground the discrepancy in terms of additional skills that are required in agent-specific use cases, such as planning, learning from errors and environmental feedback, and tracking state. To enable this grounding, we proposed a simple oracle intervention framework, where the oracle complements the LLM policy by augmenting and pruning the information exposed to the agent at each turn. To support oracle interventions, we additionally propose three procedurally-generated environments (List world, Tree world, Grid world), which lets us control task complexity, and more importantly, we can at any turn estimate the set of optimal actions that can be used to guide the agent. Our findings indicate that while the skills (planning, state tracking, and history representation in our case) enable the LLM policy to generally improve, the effectiveness of each skill is also significantly influenced by the model size and the environment.

Limitations and Future Work This paper presents the first step towards explaining the performance degradation of capable LLMs in multi-turn long-horizon agent tasks. While we find valuable insights, many important steps remain to fully understand the performance degradation. First, we rely on prompt-based mechanisms to elicit agent-like behavior. While such mechanisms have been shown to be a strong baseline, performances are also influenced by the prompt itself, and as a result post-training is appealing to discount influence of prompt design. Second, we run our analysis on simple programmable environments and benefit from being able to accurately and efficiently construct and isolate oracle interventions. While it is insightful to study this on real-world applications, annotating oracle is often intractable or ill-defined.

ETHICS STATEMENT

Our study focuses on leveraging LLMs in multi-turn feedback-driven scenarios. Specific to the study in this paper, we used well-established publicly-available open-weight models (however whose training is proprietary) and run our experiments on synthetic game-like programmable environments.

486 In this regard, we believe this is no conflict, since the environments do not contain any sensitive or
 487 personal information.
 488

489 **REPRODUCIBILITY STATEMENT**
 490

491 To ensure reproducibility, we run our analysis with open-weight models that are popularly used in
 492 the research community. Additionally, we provide relevant details and parameters in Section 4.2 and
 493 4.3.
 494

495 **REFERENCES**
 496

497 Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu,
 498 and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language
 499 models. *arXiv preprint arXiv:2311.18232*, 2023. pages 3

500 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-
 501 man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical
 502 report. *arXiv preprint arXiv:2303.08774*, 2023. pages 1

503 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan,
 504 Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language
 505 models. *arXiv preprint arXiv:2108.07732*, 2021. pages 1

506 Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawska, Lukas Gian-
 507 inazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczek, et al. Graph of
 508 thoughts: Solving elaborate problems with large language models. In *AAAI*, 2024. pages 2

509 Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt
 510 Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm
 511 systems fail? *arXiv preprint arXiv:2503.13657*, 2025. pages 3

512 Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit
 513 Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the
 514 frontier with advanced reasoning, multimodality, long context, and next generation agentic capa-
 515 bilities. *arXiv preprint arXiv:2507.06261*, 2025. pages 1

516 MohammadReza Ebrahimi, Sunny Panchal, and Roland Memisevic. Your context is not an array:
 517 Unveiling random access limitations in transformers. *arXiv preprint arXiv:2408.05506*, 2024.
 518 pages 3

519 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad
 520 Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd
 521 of models. *arXiv preprint arXiv:2407.21783*, 2024. pages 1

522 Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. Textarena.
 523 *arXiv preprint arXiv:2504.11442*, 2025. pages 1

524 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
 525 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
 526 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. pages 1

527 Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu.
 528 Reasoning with language model is planning with world model. In *EMNLP*, 2023. pages 2

529 Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot
 530 planners: Extracting actionable knowledge for embodied agents. In *International conference on
 531 machine learning*, pp. 9118–9147. PMLR, 2022. pages 2

532 Ziheng Huang, Sebastian Gutierrez, Hemanth Kamana, and Stephen MacNeil. Memory sandbox:
 533 Transparent and interactive memory management for conversational agents. In *Adjunct Proceed-
 534 ings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–3,
 535 2023. pages 2

540 Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-
 541 trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint*
 542 *arXiv:2410.21276*, 2024. pages 13

543

544 Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham
 545 Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating
 546 multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024. pages 1

547

548 Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn
 549 conversation. *arXiv preprint arXiv:2505.06120*, 2025. pages 4

550

551 Yubo Li, Xiaobin Shen, Xinyu Yao, Xueying Ding, Yidi Miao, Ramayya Krishnan, and Rema Pad-
 552 man. Beyond single-turn: A survey on multi-turn interactions with large language models. *arXiv*
 553 *preprint arXiv:2504.04717*, 2025. pages 2

554

555 Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding,
 556 Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint*
 557 *arXiv:2308.03688*, 2023. pages 1

558

559 Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model
 560 connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–
 561 126565, 2024. pages 2, 3

562

563 Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and
 564 Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic
 565 evaluation of large language models. In *ICML*, 2025. pages 1

566

567 Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window
 568 extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023. pages 6

569

570 Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru
 571 Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world
 572 apis. *arXiv preprint arXiv:2307.16789*, 2023. pages 2

573

574 Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence.*
 575 *Prentice-Hall, Egnlewood Cliffs*, 25(27):79–80, 1995. pages 2

576

577 Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of
 578 diminishing returns: Measuring long horizon execution in llms. *arXiv preprint arXiv:2509.09677*,
 579 2025. pages 2, 3

580

581 Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su.
 582 Llm-planner: Few-shot grounded planning for embodied agents with large language models. In
 583 *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3009, 2023.
 584 pages 2

585

586 Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej,
 587 Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical
 588 report. *arXiv preprint arXiv:2503.19786*, 2025. pages 13

589

590 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
 591 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and
 592 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. pages 1

593

594 Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank
 595 Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of
 596 apps and people for benchmarking interactive coding agents. *ACL*, 2024. pages 1, 2, 3, 5, 6

597

598 Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripurani, Kelvin Xu, Sanil Jain, Rakesh Shivanna,
 599 Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, et al. Michelangelo: Long
 600 context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*,
 601 2024. pages 3, 4, 5

594 Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan,
 595 and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models.
 596 *arXiv preprint arXiv:2305.16291*, 2023a. pages 2
 597

598 Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe,
 599 explain, plan and select: Interactive planning with large language models enables open-world
 600 multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023b. pages 2

601 Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-
 602 Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. Livebench: A challenging, contamination-
 603 free llm benchmark. *arXiv preprint arXiv:2406.19314*, 4, 2024. pages 1
 604

605 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,
 606 Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint*
 607 *arXiv:2505.09388*, 2025. pages 6

608 Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov,
 609 and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question
 610 answering. *arXiv preprint arXiv:1809.09600*, 2018. pages 1

611 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik
 612 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Ad-*
 613 *vances in neural information processing systems*, 36:11809–11822, 2023a. pages 2
 614

615 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
 616 React: Synergizing reasoning and acting in language models. In *International Conference on*
 617 *Learning Representations (ICLR)*, 2023b. pages 2, 3, 6

618 Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal.
 619 Generative verifiers: Reward modeling as next-token prediction. In *ICLR*, 2025. pages 2
 620

621 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,
 622 Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and
 623 chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023. pages
 624 2

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

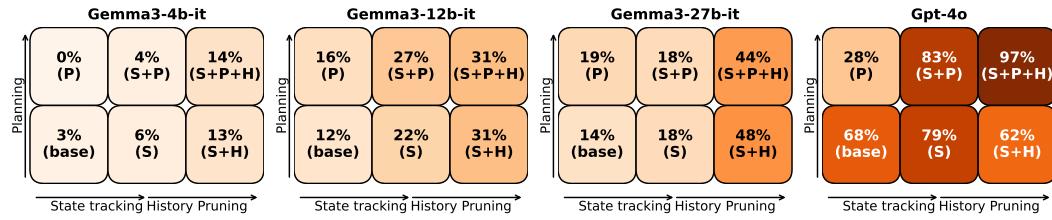
645

646

647

648 A ADDITIONAL EXPERIMENTS

650
 651 In Figure 6, we provide the success rate of GPT-4o (Hurst et al., 2024) and Gemma 3 models (Team
 652 et al., 2025) on ListWorld for different oracle configurations. Gemma 3 results are aligned with the
 653 findings in Section 4.3 with the Qwen 3 models. Planning intervention shows minimal benefits while
 654 state tracking and history pruning significantly help the model. Both Qwen 3 and Gemma 3 models
 655 across all sizes benefit from history pruning in ListWorld, but GPT-4o, which is much larger, starts
 656 to show signs of performance degradation with history pruning. This further confirms our finding
 657 that history pruning benefits smaller models but can hurt models larger than a threshold.



658
 659 **Figure 6: Influence of oracle interventions in ListWorld.** Results are averaged over all horizon lengths. The
 660 labels indicate the active oracles (S: state tracking, P: planning, and H: history pruning) compared to the base
 661 model.
 662
 663

664 B PROMPTS

665 B.1 PROMPT: LIST WORLD

666 We used the following prompt for List World experiments
 667

```
668 You are an assistant designed to **modify lists** through a sequence of
669 simple commands that you can execute at every turn. You will be given
670 an initial list and a target list. Your task is to modify the initial
671 list to a target list using the following functions, provided in JSON
672 format:
673 ````json
674 {
675   "pop": {
676     "name": "pop",
677     "type": "function",
678     "description": "Removes an element from the environment's list.
679     Index of the elements after the removed one will be reduced by
680     1.",
681     "parameters": {
682       "type": "object",
683       "properties": {
684         "id": {
685           "type": "integer",
686           "description": "The ID of the element to remove."
687         }
688       },
689       "required": [
690         "id"
691       ]
692     }
693   },
694   "done": {
695     "name": "done",
696     "type": "function",
697     "description": "Terminates the task. Should be called when no
698     more operations are needed.",
699     "parameters": {
700       "type": "object",
701     }
702   }
703 }
```

```

702         "properties": {}
703     }
704   }
705 }
706 ```
707 The initial list has all the elements of the target list in the same
708 order, but contains some extra elements that need to be removed. To
709 remove them, you should use the 'pop' function with the index of the
710 element you want to remove. Remember that once an element is removed,
711 the index of elements after it will decrease by 1. The most important
712 rule is that you can **only pop the elements from left to right**. Once
713 you pop an element, the elements before it (thoses with a smaller
714 index) can no longer be removed, and you will get an error if you try
715 to do so. Once you are done and have turned the initila list to the
716 target list, you should call the 'done' function.
717
718 You must reason step-by-step, choosing actions based on the current
719 state of the list. Avoid redundant queries and aim for efficiency.
720 Provide your response as a single python function call enclosed in a
721 code block:
722 ````python
723 function_call(arg1=val1, arg2=val2, ...)
724 ````

725 === Starting new task ===
726 Initial list: ${initial_list}
727 Target list: ${target_list}
728 Your task is to modify the initial list to target list by calling the
729 'pop' function. Call 'done' function once you are done. Remember, **only
730 pop the elements from left to right**.
731
732
733 B.2 PROMPT: TREE WORLD
734
735 We used the following prompt for tree world:
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755

```

```

SYSTEM_PROMPT = f"""
You are a reasoning agent searching a tree for a
node with a specific value (which may or may not be reachable by you).
Each node has two attributes: (1) "id" (unique string) and (2) "value"
(unique integer). In each task, you are provided with a partial
information about some of the nodes in the tree. For each node included
in this information, you are given the id and the value. For some of
the nodes the list of the node's children ids is also provided. The
format in that case is: (id=<node_id>, value=<node_value>) -> [<
child_id1>, <child_id2>, ...]. Note that an empty list indicates that
the node is a leaf and has no children. For other nodes, the children
are not given to you. In that case, you are given: \"UNKNOWN\" in place
of the children ids.

A target value will be given to you at the beginning of each task. Your
job is to try to find a node with this value and report its id. You
should do this using the following functions, provided in JSON format:
````json
{
 "get_children": {
 "name": "get_children",
 "type": "function",
 "description": "Returns the list of children nodes for a given
node ID (i.e., the outgoing edges)",
 "parameters": {
 "type": "object",
 "properties": {
 "id": {
 "type": "string",

```

```

756 "description": "The ID of the node whose children are to
757 be retrieved."
758 }
759 },
760 "required": [
761 "id"
762],
763 "returns": {
764 "type": "array",
765 "items": {
766 "type": "object",
767 "properties": {
768 "id": {
769 "type": "string"
770 },
771 "val": {
772 "type": "integer"
773 }
774 },
775 "required": [
776 "id",
777 "val"
778]
779 },
780 },
781 "found": {
782 "name": "found",
783 "type": "function",
784 "description": "Indicates that the node with the specified ID
785 contains the target value.",
786 "parameters": {
787 "type": "object",
788 "properties": {
789 "id": {
790 "type": "string",
791 "description": "The ID of the node that contains the
792 target value."
793 }
794 },
795 "required": [
796 "id"
797]
798 },
799 "returns": {
800 "type": "string",
801 "description": "Confirmation that the node with the given ID
802 contains the target value."
803 }
804 },
805 "unreachable": {
806 "name": "unreachable",
807 "type": "function",
808 "description": "Indicates that the node with the target value is
809 impossible to reach.",
810 "parameters": {
811 "type": "object",
812 "properties": {}
813 },
814 "returns": {
815 "type": "string",
816 }
817 }
818 }

```

```
810 "description": "Confirmation that the target value was not
811 possible to find in the tree."
812 }
813 }
814 ```
815 You can ask for the ids and values of the children of a node by calling
816 the 'get_children' function with the node's id.
817 If you find the target node (the node with the target value), return
818 its id using the 'found' function. After calling 'found' the task will
819 terminate and you succeed if you have reported the correct id.
820 If you believe it is impossible to find the target node, call the 'unreachable' function. After calling 'unreachable' the task will
821 terminate and you succeed if it was impossible for you to find the
822 target node.
823
824 You must reason step-by-step, choosing actions based on the current
825 state of the search. Avoid redundant queries and aim for efficiency.
826 Provide your response as a single python function call enclosed in a
827 code block:
828 ```python
829 function_call(arg1=val1, arg2=val2, ...)
830
831 === Starting new task ===
832 Your task is as follows: Find the id of the node with value **${target_node_val}**
833 Once you find the target node containing this value, return its id by
834 calling 'found' function. If you think it is impossible to find this
835 node, call the 'unreachable' function.
836 Provide all responses as a single python function call enclosed in a
837 code block:
```

### B.3 PROMPT: GRID WORLD

We used the prompt below for grid world:

You are an intelligent agent playing a grid world navigation game. Your goal is to move from the given start position to the goal position using the fewest possible moves. The game board is a 2D grid with the following properties:

- The top-left corner is coordinate (0, 0), and the bottom-right corner is (size-1, size-1).
- You will be given:
  - \* The size of the board (N x N)
  - \* Your starting position (row\_index, column\_index)
  - \* The goal position (row\_index, column\_index)
  - \* A list of hole positions (each a coordinate)
  - \* The maximum number of moves allowed
- You can move using these actions: 'up()', 'down()', 'left()', 'right()'
- \*Only\* if you have reached the goal, call 'done()' to terminate the game. Once you terminate the game, you are not allowed any more moves.
- You can reason, but always end by specifying a single action within triple fenced blocks. Example

```
'''python
up()
'''
or
'''python
done()
'''
- Each move costs **1 move**.
```

```

864 - If you move into a hole, you incur a **penalty of 3 additional moves
865 ** (because it is hard to get out of a hole).
866 - You must stay within the grid boundaries.
867 - Your objective: **Reach the goal in as few moves as possible without
868 exceeding the maximum allowed moves.**
869 - After each move, you will receive the updated position and remaining
870 moves.
871 - In the triple fenced blocks, do not write anything except the next
872 action in the required format.

873 === Your Task ===
874 The grid world game is set up as follows:
875 - Board size: ${size} x ${size}
876 - Start position: ${start}
877 - Goal position: ${goal}
878 - Holes at: ${holes}
879 - Your move budget is: ${max_moves}

880 Your task: Navigate from the start to the goal using the fewest moves
881 possible. Remember:
882 - You can move using the following actions: 'up()', 'down()', 'left()', 'right()'
883 - If you reached the goal, terminate by performing action 'done()'
884 - Each action must be in a triple-fenced Python code block, like:
885 ````python
886 right()
887 ````

888 - Avoid holes if possible, as they cost extra moves.
889 - Do not exceed the maximum allowed moves.

890 Begin your first move now.
891 """
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

```