Do Think Tags Really Help LLMs Plan?
A Critical Evaluation of ReAct-Style Prompting

Anonymous ACL submission

Abstract

The reasoning abilities of Large Language Models (LLMs) remain a topic of debate and are critically tested in sequential decision-making problems. ReAct, a recently popular method, has gained popularity for claiming to enhance LLM reasoning abilities while directly prompting them by "*interleaving reasoning trace with action execution*" in text-based planning domains such as AlfWorld and WebShop. However, given the different components of ReAct-style prompting, it remains unclear what the source of improvement in LLM performance is. In this paper, we critically examine the claims of ReAct-style prompting for sequential decision-making problems. By introducing systematic variations to the input prompt, we perform a sensitivity analysis along the original claims of ReAct. Contrary to these claims and common use cases that utilize ReAct-style prompting, we find that the performance is minimally influenced by the interleaved reasoning trace or by the content of these generated reasoning traces. Instead, the performance of LLMs is primarily driven by the unreasonably high degree of similarity between input example tasks and queries, implicitly forcing the prompt designer to provide instance-specific examples, which significantly increases the cognitive burden on the human. Our empirical results, in the same suite of domains as ReAct, show that the perceived reasoning abilities of LLMs stem from the exemplar-query similarity and approximate retrieval rather than any inherent reasoning abilities.

1 Introduction

Large Language Models (LLMs) have seen rapid advancements specifically in Natural Language Processing and Understanding (NLP & NLU). LLMs have unparalleled capabilities in text generation, summarization, translation, and question answering to name a few. (Bubeck et al., 2023). Motivated by these capabilities of LLMs, there has also been a rush to look for other emergent abilities–especially for reasoning and planning. A popular way to improve LLM performance on reasoning/planning tasks has been in-context prompting or prompt engineering (Sahoo et al., 2024) to include instructions (Giray, 2023), syntax structure (Marvin et al., 2023), criticism and plan guidance with verification (Kambhampati et al., 2024), etc. Among these approaches, ReAct (Yao et al., 2022b), presented at ICLR 2023, stands out which claims to improve LLM planning abilities through the use of reasoning traces interleaved with action execution given as plan guidance. Although their original study includes multiple domain tasks such as question answering in HotPotQA and FEVER datasets (Yang et al., 2018; Thorne et al., 2018), in this paper we are particularly interested in planning tasks. where ReAct claims superior performance.

In our initial experiments with ReAct for planning, we found that the system is overly dependent on a high degree of syntactic similarity between the example prompt and the query, and is extremely brittle to minor perturbations to the input prompt. For example, when provided with an explicit set of examples of *pick-and-place-object* task and asked to plan for a *pick-and-place-two-objects* task, it should be trivial to generalize the solution of the examples to the queried task. Unfortunately, even such a minor variation to the original ReAct agent setup disrupts its performance.

Given the seemingly widespread adoption of ReAct methodology (as of this writing, it has 1,862 citations), the brittleness we witnessed calls for a systematic study of the factors contributing to the performance of ReAct-based LLM Agents. Moreover, recent studies have highlighted similar case studies on the original Chain-of-Thought idea (Stechly et al., 2024a; Sprague et al., 2024). Based on the claims of (Yao et al., 2022b), we isolate three possible reasons for the claimed performance of ReAct framework: 1) the utility of interleaving reasoning trace during action execution, 2) the utility of providing plan guidance, and, 3) the significance of example prompt provided to the the LLM.

The brittleness of ReAct becomes apparent when considering the variability in prompt designs. Depending on the domain and task, prompt designers may provide abstract guidance, task-specific instructions within the same domain, analogical examples, or global reasoning traces. ReAct's inability to robustly adapt to such variations underscores its limitations in handling diverse input prompts, thereby necessitating a closer examination of its design and implementation. In this work, we systematically evaluate the brittleness of ReAct by studying which potential factors contribute to its performance. This analysis is conducted by investigating the following research questions : **RQ1:** Does the agent performance depend on interleaving reasoning trace with action execution? **RQ2:** How does the nature of the reasoning trace or guidance information affect the performance of LLM Agents? **RQ3:** How does the similarity between the example $\langle$problem, solution$\rangle$and the query $\langle$problem, ? $\rangle$, which are present in the prompt, affect LLM Agent performance?

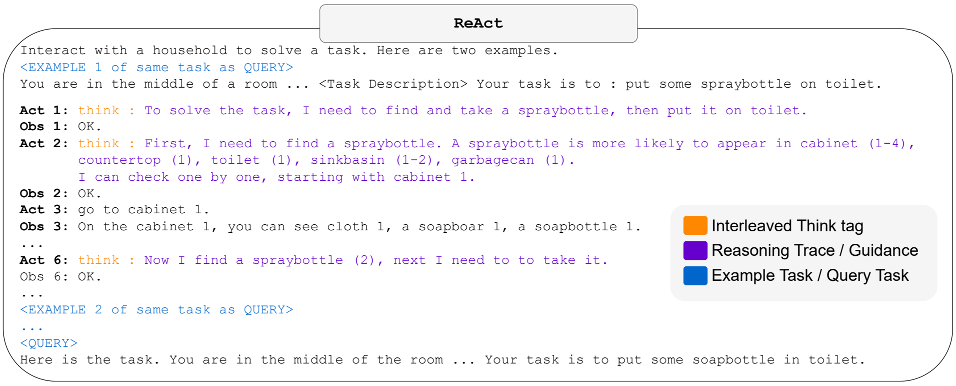
We conduct extensive experiments on the AlfWorld and WebShop domain using various LLM Models, including GPT-3.5-turbo, GPT-3.5-instruct, GPT-4-0314(Old Variant), GPT-4-0613 (Latest Variant), GPT-4o, Claude-Opus and Llama 3.1-8b. Through our comprehensive empirical study, we answer each of the research questions above. (RQ1) We find that LLM performance in-fact improves when the reasoning trace is **not** interleaved with action execution. (RQ2) Moreover, providing weaker guidance or placebo-guidance (where the text provides no information about the task) has comparable performance to strong reasoning trace-based guidance. Answers to RQ1 and RQ2 suggest that the source of ReAct LLM agent performance is **not** the interleaving aspect or the content of the reasoning trace. Finally, in (RQ3), we see that variations to the example prompt such that it belongs to different task within the same domain, or has a different goal or plan attributes than the queried problem; causes the performance of ReAct-LLM Agent to plummet.

Our findings highlight that the benefits of ReAct-style prompting are present when prompt engineers can curate highly instance-specific examples. This may not scale for domains with a large number of problem instance classes, and it places an undue burden on prompt engineers to provide instance specific examples. Finally, our experiments call into question claims of enhanced “emergent reasoning" of LLMs with prompt engineering efforts; corroborating contemporary research (Verma et al., 2024; Valmeekam et al., 2024; Stechly et al., 2024b; Ullman, 2023; Schaeffer et al., 2023; McCoy et al., 2023; Stechly et al., 2024a; Sprague et al., 2024) questioning reasoning abilities of LLMs.

2 Related Work

Large Language Models have been shown to be successful in a plethora of natural language tasks (Kocoń et al., 2023; Gilardi et al., 2023; Zhu et al., 2023; Bubeck et al., 2023; Bhattacharjee et al., 2024). However, there are two schools of thought when it comes to utilizing off-the-shelf LLMs for planning and reasoning tasks. Works such as Chain of Thought, ReAct, and others that followed (Wei et al., 2022; Yao et al., 2023; Long, 2023; Yao et al., 2024; Besta et al., 2024; Fu et al., 2024; Aksitov et al., 2023), have argued about the reasoning abilities of LLMs by proposing prompting methods. On the other hand, (Valmeekam et al., 2024; Stechly et al., 2024b) have refuted these claims by showing the inability of LLMs to solve deterministic planning and classical reasoning problems.

In particular, for investigating the use of LLMs in solving decision making problems, AlfWorld (Shridhar et al., 2020) is a popular domain that was originally proposed for training text-based Reinforcement Learning agents. Lately, works such as ReAct, Reflexion, and their other variants (Yao et al., 2022b; Shinn et al., 2023) have argued on the prowess of LLMs' reasoning abilities on AlfWorld. Furthermore, there have been several extensions to ReAct that boost their generalization abilities across more domains including multi-modal domains (Yang et al., 2023; Castrejon et al., 2024), autonomous vehicles (Cui et al., 2024), table question answering (Zhang et al., 2023), etc. While the effectiveness of ReAct is celebrated across different areas, these works only depend on anthropomorphization of LLMs (Min et al., 2022; Peng et al., 2024) for using ReAct based prompting with no justification on the source of improvement in performance. This motivates our work in investigating the components of ReAct with respect to sequential decision-making problems and analyzing the role each component plays.



ReAct
🟧 Interleaved Think tag
🟪 Reasoning Trace / Guidance
🟦 Example Task / Query Task

```
Interact with a household to solve a task. Here are two examples.
<EXAMPLE 1 of same task as QUERY>
You are in the middle of a room ... <Task Description> Your task is to : put some spraybottle on toilet.

Act 1: think : To solve the task, I need to find and take a spraybottle, then put it on toilet.
Obs 1: OK.
Act 2: think : First, I need to find a spraybottle. A spraybottle is more likely to appear in cabinet (1-4),
       countertop (1), toilet (1), sinkbasin (1-2), garbagecan (1).
       I can check one by one, starting with cabinet 1.
Obs 2: OK.
Act 3: go to cabinet 1.
Obs 3: On the cabinet 1, you can see cloth 1, a soapboar 1, a soapbottle 1.
...
Act 6: think : Now I find a spraybottle (2), next I need to to take it.
Obs 6: OK.
...
<EXAMPLE 2 of same task as QUERY>
...
<QUERY>
Here is the task. You are in the middle of the room ... Your task is to put some soapbottle in toilet.
```

Figure 1: An example of ReAct in AlfWorld. We highlight the main components of ReAct, i.e., Interleaved reasoning and acting, the reasoning trace / plan guidance and the example and query task.

3 Preliminaries

3.1 Domains

AlfWorld: (Shridhar et al., 2020) is a synthetic text-based game built on top of a STRIPS-style PDDL domain description (Fikes and Nilsson, 1971). ReAct (Yao et al., 2022b) defines six tasks (or problem classes) within this domain namely - Put, Clean, Heat, Cool, Examine, and PutTwo. Each problem class consists of several problem instances, such as *put a spraybottle on toilet* (see Fig. 1 is an example instance of Put class. Since AlfWorld is a partially observable environment, each of these problem instances can be solved by navigating and interacting with the environment simulator via text actions. For example, this task can be solved by the following actions- *go to cabinet 2, take spraybottle 2 from cabinet 2, go to toilet 1, put spraybottle 2 in/on toilet 1*.

WebShop: (Yao et al., 2022a) is an online shopping website environment with 1.18M real-world products and 12K human instructions. The agent is provided with an initial human instruction (for example, "I am looking for a nightstand with drawers. It should have a nickel finish, and priced lower than $140"). The agent's task is to crawl the shopping environment using actions such as `search ‘nightstand drawers’`, `choose ‘white buttons’`, `back to search`, etc. For this work, we randomly sample 500 test instructions from the environment and evaluate the success rate of the agent's task completion.

3.2 ReAct

ReAct (Yao et al., 2022b) claims to increase LLM's performance on text-based planning tasks such as AlfWorld and WebShop primarily by augmenting the original action space of the agent with a *think* action. The *think* action tag provided by ReAct is claimed to comprise of **Re**asoning + **Act**ion trace that is provided in the solution for the example problems (exemplars) as part of the prompt. During execution, the LLM is expected to generate a *think* action tag for the queried problem instance that is semantically similar to the one provided for the examples in the prompt.

Location of *THINK* tag In ReAct, the integration of the *think* tag within actions serves to expand the action space. This allows the language model (LLM) agent to execute a *think* action, prompting an *‘OK’* response. Through analysis of example prompts in ReAct experiments, we identify various instances of the `think` action. Typically, it appears after stating the problem instance, reiterating the task, and providing problem-specific guidance. However, the authors offer no structured guidelines for its implementation, placement, or guidance. This observation aligns with feedback from the paper's reviewers (OpenReview, 2024) citing inconsistencies in the prompting format.

Content of *THINK* tag In ReAct, the *think* action consistently provides the decision-making agent with success-oriented guidance for task completion. For instance, upon encountering a `spraybottle`, the prompt might include: `think: Now I find a spraybottle (2). Next, I need to take it.` This guidance exposes forthcoming actions and sub-tasks for the agent.

Few shot *EXAMPLEs* In the AlfWorld domain (wihch is a PDDL domain), ReAct authors (Yao et al., 2022b) classify six problem classes or tasks: `Put, Clean, Heat, Cool, Examine, PutTwo`. Despite representing different tasks, they share the same environment dynamics and action space, allowing for very similar execution trace. For instance, a `Heat` task might involve `Putting` an item into a microwave. In ReAct experiments, authors provide two example problem-solution pairs (referred to as exemplars in our work) before querying the LLM agent with a problem instance. Authors force ReAct agent to use examples and queries belonging to the same problem class without motivating this design decision. However, the queried problem may differ in objects or locations from the exemplars.

4 Critically Evaluating ReAct Prompting

The subsequent sub-sections explore the design of exemplar prompt variations to investigate our research questions concerning the claims of ReAct. Each variation modifies the base ReAct prompt, and we use the AlfWorld domain as the running example for discussing these variations (see Appendix for WebShop prompts).

4.1 RQ1 : Interleaving *think*ing with acting

Does the agent performance depend on interleaving reasoning trace with action execution?

To answer this research question, we propose collating the guidance information contained within the multiple *think* tags present in the examples of the input prompt into a single *think* tag appended after the example problem is specified. This approach can be interpreted as Chain-of-Thought (Kojima et al., 2022; Wei et al., 2022), where guidance information is generated before action execution.

Variation 1: Exemplar-based CoT AlfWorld is a partially observable environment where an agent can only observe objects after reaching that location. Hence, we remove specific location and object identifiers to modify the *think* actions that are originally interleaved with other actions in the environment (see RQ1 a in Figure 2). Finally, we append all the *think* actions together at the beginning of the example problem. **Intuition:** Problem-specific guidance for a sequential decision-making agent can be given step-by-step (as in ReAct) or all at once. Note, that this variation is possible since AlfWorld is not a dynamically changing environment in which case providing information on the task will not be possible.

Variation 2: Anonymized Exemplar-CoT We take one step further and modify the *think* tag to remove references to specific locations and objects, making it more general (see RQ1 b in Figure 2). Similarly, in WebShop we anonymize specific item options as `desired options`. **Intuition:** Exemplars can be made more general by providing abstract guidance and exploiting LLMs ability to identify necessary semantic entity relations.

4.2 RQ2 : Plan Guidance following *think* tag

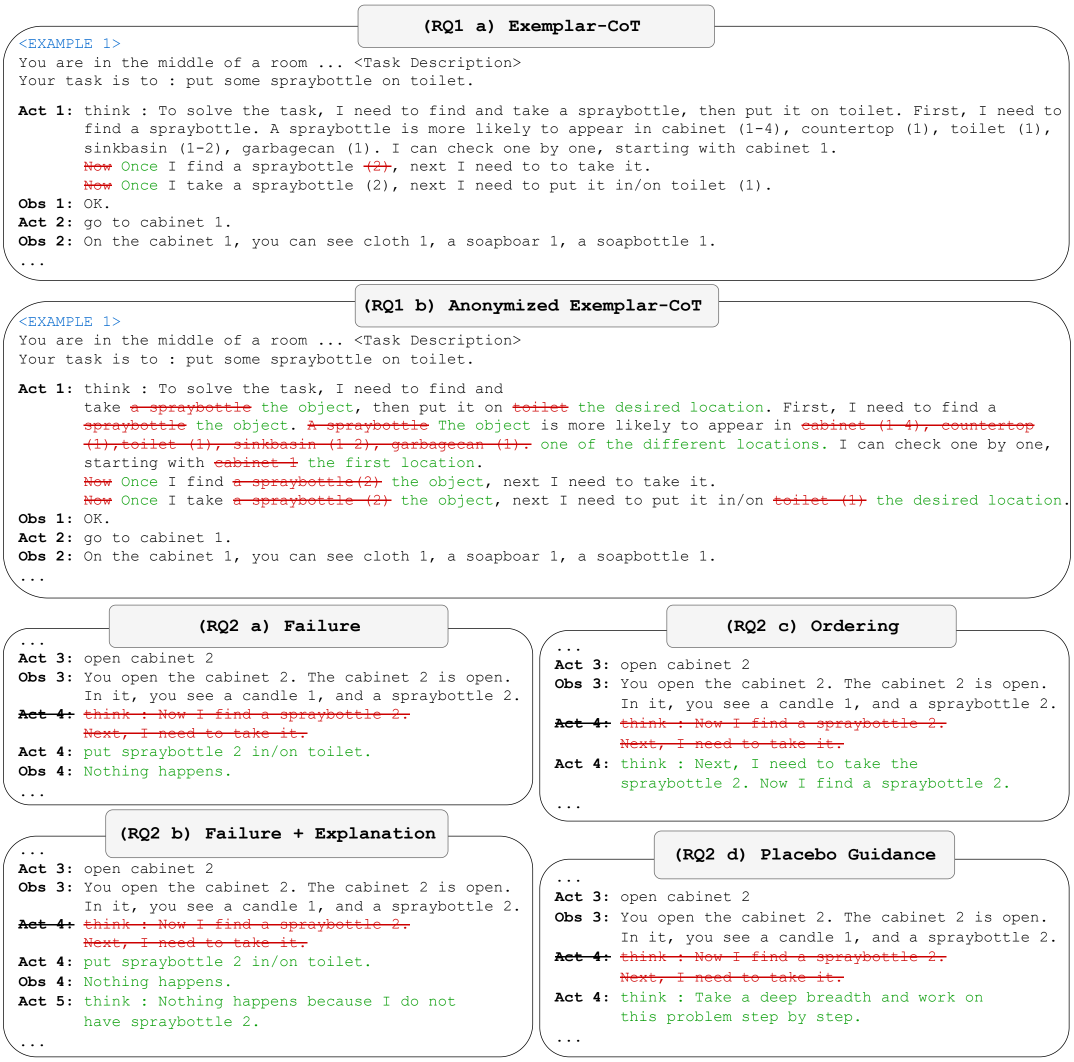
How does the nature of the reasoning trace affect the performance of LLM?

ReAct claims to use reasoning trace as the guidance information following the *think* tag. For instance, in ReAct (Yao et al., 2022b), thoughts are to (1) decompose the goal (2) track sub-goal completion (3) determine the next sub-goal and (4) reason via common-sense where to find and object and what to do with it. It is, however, unclear what is the motivation to use these as the reasoning trace. The potential anthropomorphization of large language models (LLMs) may suggest that their thought processes are similar to the abstract plans humans make, and that they must be prompted in the same manner. However, it is unclear why this assumption should hold true.

Variation 1: Failure We note that none of the examples used in ReAct prompting for any task consist of invalid actions. We inject two invalid actions in the execution trace : the first that attempts to execute the action pertinent to the task (such as *put spraybottle 2 in/on toilet*) when not possible and, second, executes some other invalid action. We include the expected simulator response, *Nothing happens.*, when invalid actions are taken. **Intuition:** Reasoning trace can be about *what to do* such as future subgoals, or *what not to do* such as mistakes in hindsight. This should be weaker guidance than in base ReAct as the exemplars do not point out what to do next.

Variation 2: Failure + Explanation We place *think* actions after invalid actions injected in Failure Variation which consist of explanations for the failure (see RQ2 b in Figure 2). **Intuition:** We can augment mistakes with explanations to avoid similar failures. This is a stronger guidance signal than Failure, however, the exemplars still not provide information on what to do next.

Variation 3: Guidance Ordering LLMs are known to be susceptible to minor syntactic perturbations to inputs. We test whether it is true for guidance information given as prompt as well (see RQ2 c in Figure 2). We identify chain of sub-tasks in a reasoning trace $S_1 \rightarrow S_2 \cdots S_n$ and reverse it to be $S_n \rightarrow S_{n-1} \cdots S_1$. **Intuition:** LLM agent



(RQ1 a) Exemplar-CoT

<EXAMPLE 1>
You are in the middle of a room ... <Task Description>
Your task is to : put some spraybottle on toilet.

Act 1: think : To solve the task, I need to find and take a spraybottle, then put it on toilet. First, I need to find a spraybottle. A spraybottle is more likely to appear in cabinet (1-4), countertop (1), toilet (1), sinkbasin (1-2), garbagecan (1). I can check one by one, starting with cabinet 1.
~~Now~~ Once I find a spraybottle ~~(2)~~, next I need to to take it.
~~Now~~ Once I take a spraybottle (2), next I need to put it in/on toilet (1).
Obs 1: OK.
Act 2: go to cabinet 1.
Obs 2: On the cabinet 1, you can see cloth 1, a soapboar 1, a soapbottle 1.
...

(RQ1 b) Anonymized Exemplar-CoT

<EXAMPLE 1>
You are in the middle of a room ... <Task Description>
Your task is to : put some spraybottle on toilet.

Act 1: think : To solve the task, I need to find and take ~~a spraybottle~~ the object, then put it on ~~toilet~~ the desired location. First, I need to find a ~~spraybottle~~ the object. ~~A spraybottle~~ The object is more likely to appear in ~~cabinet (1-4), countertop (1),toilet (1), sinkbasin (1-2), garbagecan (1).~~ one of the different locations. I can check one by one, starting with ~~cabinet 1~~ the first location.
~~Now~~ Once I find ~~a spraybottle(2)~~ the object, next I need to take it.
~~Now~~ Once I take ~~a spraybottle (2)~~ the object, next I need to put it in/on ~~toilet (1)~~ the desired location.
Obs 1: OK.
Act 2: go to cabinet 1.
Obs 2: On the cabinet 1, you can see cloth 1, a soapboar 1, a soapbottle 1.
...

(RQ2 a) Failure

...
Act 3: open cabinet 2
Obs 3: You open the cabinet 2. The cabinet 2 is open. In it, you see a candle 1, and a spraybottle 2.
~~**Act 4:** think : Now I find a spraybottle 2. Next, I need to take it.~~
Act 4: put spraybottle 2 in/on toilet.
Obs 4: Nothing happens.
...

(RQ2 b) Failure + Explanation

...
Act 3: open cabinet 2
Obs 3: You open the cabinet 2. The cabinet 2 is open. In it, you see a candle 1, and a spraybottle 2.
~~**Act 4:** think : Now I find a spraybottle 2. Next, I need to take it.~~
Act 4: put spraybottle 2 in/on toilet.
Obs 4: Nothing happens.
Act 5: think : Nothing happens because I do not have spraybottle 2.
...

(RQ2 c) Ordering

...
Act 3: open cabinet 2
Obs 3: You open the cabinet 2. The cabinet 2 is open. In it, you see a candle 1, and a spraybottle 2.
~~**Act 4:** think : Now I find a spraybottle 2. Next, I need to take it.~~
Act 4: think : Next, I need to take the spraybottle 2. Now I find a spraybottle 2.
...

(RQ2 d) Placebo Guidance

...
Act 3: open cabinet 2
Obs 3: You open the cabinet 2. The cabinet 2 is open. In it, you see a candle 1, and a spraybottle 2.
~~**Act 4:** think : Now I find a spraybottle 2. Next, I need to take it.~~
Act 4: think : Take a deep breadth and work on this problem step by step.
...

Figure 2: Example of prompt variations considered for RQ1 and RQ2.

should be invariant to the syntax of reasoning trace if the semantic information is preserved. This does not change the reasoning trace from the perspective of information content.

Variation 4: Placebo Guidance It is unclear to what extent LLM agent uses the supposed helpful thoughts for the decision making task. In this variation we replace *think* tag guidance with a placebo thought that does not contain any task relevant information (see RQ2 c in Figure 2), but has been widely used as prompt engineering trick (Kojima et al., 2022). **Intuition:** According to claims of ReAct, we expect the performance to get worse when the guidance does not have any information useful for task success.

4.3 RQ3: *EXAMPLEs* - *QUERY* Similarity

How does the similarity between the prompt examples and the query problem affect LLM performance?

RQ3 investigates the role of example similarity to the query in LLM agent's performance. Establishing problem similarity can be challenging, especially where minor variations to the problem can have varied interpretations (such as an analogy to a different task altogether). Our work explores this challenge in a systematic way. During example prompt construction, prompt designers may use synonyms to refer to objects (`Domain`), come up with examples where the agent task is the same as query but the goals are different (`Instance`), or provide optimal solutions as the examples (`Optimal`) preventing LLM to obtain information regarding exploration strategy. Furthermore, given that the domain has the same underlying action dynamics and that the tasks reuse several actions, prompt designers may choose to provide query specific example prompts (as in base `ReAct`), provide one of a different task and one of the same task (`One`), provide both examples to be of a different task (`Both`), or take an exhaustive approach and provide one example of all tasks (`All`).

Variation 1: Synonyms - `Domain` For this variation, we replace the object and location names in the example prompts with their synonyms. For example, *~~spraybottle~~* → *aerosolbottle*, *~~cabinet~~* → *cupboard*, and, *~~microwave~~* → *oven*. We make 36 such changes to object and location names across all the examples.[1] **Intuition:** Exemplar guidance maybe specified with alternate synonymous object and location names. Reasoning agents should be invariant to variable name substitution for closed world dynamics such as PDDL-based AlfWorld.

Variation 2: Problem Level - `Instance` We inject instance-level changes to the examples provided in the prompts. We change the goal location in exemplar problem to ensure that it does not match with any of the goal locations in query problem. We also add repetitive yet futile actions in the exemplar execution trace which does not effect the solution. **Intuition:** Ensuring a different goal location in exemplar from the queried problem is a natural use-case. Moreover, exemplars may contain arbitrary exploration strategies such as action repetition (Sharma et al., 2017).

Variation 3: Problem Level - `Both, One, All` Recall that the environment dynamics for all the tasks are the same. In fact, several tasks subsume the use of our tasks such as `Heat` requires the agent to `Put` an food in the microwave. In general, all the tasks share a large portion of actions (such as exploring cabinets and locations, picking objects etc.). Motivated by how tight relationship of these tasks we come up with three variations. First, `One`, uses one exemplar of an arbitrarily picked task and the other exemplar of the same task as the query. Second, `Both`, uses both exemplars from an arbitrarily picked task. Finally, `All`, uses a total of six exemplars (this is the only variation where we provide more than the standard two examples as in ReAct) corresponding to each task under consideration. Remember, this includes the query task which is always present at the end in the input prompt. **Intuition:** With a very similar action execution trace (such as exploration, picking and placing objects) across tasks, and shared dynamics, LLM agent should be minimally affected by the use of exemplars of a different task.

Table 1: Average Success % of LLM for RQ1 and RQ2 on six AlfWorld tasks.

Model / Prompt	**Act**	**ReAct**	**RQ1**		**RQ2**			
			CoT	**Anon. CoT**	**Placebo**	**Order**	**Failure**	**Explanation**
GPT-3.5-Turbo	34.3	27.6	46.6	41	30	28.3	43.3	41.6
GPT-3.5-Instruct	44	50.7	61.9	50.7	41	42.5	47	44.7
GPT-4-0314 (Old)	-	23.3	43.3	33.3	36.6	30	50	36.6
GPT-4-0613 (Latest)	70.0	26.7	40.0	26.6	36.6	30	60	36.6
Claude-Opus	43.3	56.6	50	46.6	30	50	53.3	30

Table 2: Average Success % of LLM for RQ1 and RQ2 on WebShop tasks.

Model / Prompt	**Act**	**ReAct**	**RQ1**		**RQ2**		
			CoT	**Anon. CoT**	**Placebo**	**Failure**	**Explanation**
GPT-3.5-Turbo	1.12	1.04	2.20	1.88	1.52	3.48	3.48
GPT-3.5-Instruct	7.24	7.16	7.52	6.12	7.40	7.20	7.24
GPT-4-0613 (Latest)	8	4	8	8	6	8	8
GPT-4o	4.64	2.24	4.68	4.52	4.08	4.68	4.68
Claude-Opus	4	4	4	2	4	2	4
LLAMA-3.1-8B	1.44	3.16	3.28	3.92	2.04	1.20	2.16

Variation 4: Exploration Strategy - `Optimal` As noted before, ReAct does not explain the choice of exemplars used. An important ingredient to the exemplars is the exploration strategy used. In this variation we provide exemplars which serendipitously take the optimal actions (as if the environment were fully observable) and therefore the example plan is the shortest possible. **Intuition:** Exploration strategy exposed in exemplars (that too for the same problem task) should not impact ReAct's performance if the LLM agent is reasoning instead of retrieval (or pattern matching).

5 Results

While the original ReAct experiments were carried out on PaLM (currently decommissioned), we reproduce their results with newer set of models. We use GPT-3.5-Turbo, GPT-3.5-Instruct, GPT-4, GPT-4o, and Claude-Opus, which are all newer models than those benchmarked in ReAct (Yao et al., 2022b). Note, that despite using newer models, our results shed doubts on the reproducibility and consistency across models of the original paper's results. As noted, we use the setup in (Yao et al., 2022b) for all our experiments. In AlfWorld, GPT3.5-(Turbo, Instruct) results are on 134 instances across six tasks, GPT-4/Claude-Opus on 60 instances (10 for each task) due to cost considerations. In WebShop, GPT3.5(Turbo, Instruct), GPT-4o, and LLAMA-3.1-8B results are on 500 samples, while GPT-4/Claude-Opus are on 50 instances due to cost considerations.

5.1 Utility of Interleaving THINK tags (RQ1)

From Table 1(RQ1) note that the exemplar CoT and the anonymized exemplar CoT performs signif-

---

[1] The object names / location are unchanged for the problem query and subsequent interaction with the simulator.

icantly better than base ReAct for all GPT-X family of models. Moreover, the performance dips slightly for Claude-Opus along these variations. This refutes ReAct's first claim on the importance of interleaving reasoning trace generation with action execution. Even in the case of the Claude where there is a slight dip in performance, the models seems to be performing at reasonably high success rate which questions the importance of interleaved reasoning and action execution. We omit LLAMA-3.1-8B and GPT-4o (See B.4) for AlfWorld as they achieve zero performance over baselines and all the variations. From Table 2, we find a similar pattern: CoT and Anon. CoT variants perform closely or better than the baseline ReAct. A surprising result consistent in both the domains is the performance of `Act` baseline (where *think* tags are absent and actions are generated directly). `Act` baseline is weaker only for two models `GPT-3.5-Instruct`, `Claude-Opus` for both the domains, which further questions the utility of using ReAct prompting.

5.2 Utility of THINK tag Content (RQ2)

Recall that reasoning trace guidance pertains to the prospective actions or behaviors an agent should execute (foresight guidance). This type of guidance is more informative compared to other variations, such as hindsight guidance, which focuses on past errors without providing future solution steps, and placebo guidance, which is entirely unrelated to the task. ReAct claims that reasoning trace is crucial for LLM agent performance, which would predict a decline in performance with hindsight guidance and a collapse with placebo guidance. Therefore, a practitioner would expect all the rows to be a dark shade of `red`. In contrast, our findings in Table 1 indicate that hindsight guidance (`Failure, Explanation`) actually improves the performance of the GPT family of models. The Claude-Opus model's performance remains stable with hindsight (`Failure`) guidance and declines with placebo guidance. This refutes ReAct's claim that task-specific reasoning trace is the source of LLM performance. Our argument that LLM agent's performance is only slightly affected by the reasoning trace explains the indifference to ordering perturbation as well. If the LLM is not utilizing the reasoning trace for decision making, change in ordering would not affect the agent's performance. Our arguments hold for the WebShop domain as well, where all of the variants perform closely or better than the baseline ReAct.[2]

Table 3: Average Success % of LLM for RQ3 on six AlfWorld tasks. OC: Out of context limit

Model / Prompt	Act	ReAct	RQ3					
			Domain	Instance	Optimal	All	One	Both
GPT-3.5-Turbo	34.3	27.6	1.6	30	20.1	32	28.3	1.6
GPT-3.5-Instruct	44	50.7	47.6	42.5	39.5	OC	17.9	5.2
GPT-4-0314 (Old)	–	23.3	13.3	23.3	50	23.3	16.6	0
GPT-4-0613 (Latest)	70.0	26.7	10.0	20.0	53.3	23.3	20	3.3
Claude-Opus	43.3	56.6	50	46.6	43.3	50	60	6.6

5.3 Utility of Example-Query Similarity(RQ3)

Intuitively, the similarity of `Domain` examples is closest with base ReAct, followed by `Instance` and `Optimal` variations. Finally, `All` contains an overload of information followed by `One` and `Both` which has the same action space but uses different tasks as exemplars. Recall that AlfWorld being a PDDL domain has a shared environment dynamics across all tasks with upto 80% of actions shared across execution traces. While ReAct does not investigate impact of varied exemplars, given the popular usage, one expects LLMs to be robust to such changes especially in a common-sense household domain. Table 3 shows the severe brittleness of ReAct based LLM agent to even minor variations (such as `Domain, Instance`). Specifically, performance of GPT-3.5-Turbo and GPT-4 plummnts for `Domain`. Claude-Opus which was more robust in RQ1, RQ2, is also impacted severely by `Domain`, `Instance` variations. Furthermore, when we do not expose the exploration strategy and only provide Optimal exemplars, the performance of LLM agents further drops (except in GPT4).

Overloading the LLMs with more exemplars `All` does not impact its performance. We posit, this is because the query-task exemplar is still part of the large input prompt. Among the two exemplars, as provided in ReAct, when one of them is of a different task (`One`) then the performance significantly reduces for LLMs. When both of the exemplars are of a different task then the performance collapses to single digit success rates for all the models. This is a key result of this work highlighting the severe dependence of LLMs on the similarity of the exemplars to the query task. Through sensitivity analysis using our RQ3 variations we could find parts of the input (the task similarity of the exemplar with query) which is the source of ReAct performance. Essentially, the LLM is mimicking / performing

---

[2] In all our experiments, we note that LLMs replicate the exact steps as shown for the examples in the prompts. Hence, they do not output what ReAct claims as think tags if those tags are not present in the original prompt.

approximate retrieval from the context presented to it. Moreover, our results corroborate the line of research that questions the inability of LLMs to reason or plan (Verma et al., 2024; Valmeekam et al., 2024; Stechly et al., 2024b; Ullman, 2023; Schaeffer et al., 2023; McCoy et al., 2023; Stechly et al., 2024a; Sprague et al., 2024).

The reported success-rate from the ReAct paper (Yao et al., 2022b) on the WebShop domain is 40%. Due to the absence of the exact queries used in the paper, we randomly sampled queries from the WebShop dataset comprising 12K records. This approach possibly resulted in the decoupling of any relationship between the exemplars and the queries. Referring to Table 2, it is evident that the performance of the WebShop ReAct agent significantly declined, reaching single digit percentages (as well as other variants). This mirrors the trends observed in the `Both` variant of the Alfworld in Table 3, further supporting our arguments. We perform some additional experiments and error analysis to support our RQ3 arguments and are presented in Appendix Section C.

5.4 Discussion

Pitfalls of ReAct-Style Prompting: Recall, that ReAct claims an improved performance for text-based planning domains, namely - AlfWorld and WebShop, where the presence of a *think* tag acts as guidance for the LLM to generate the next set of actions during the LLM-environment interaction. Through our sensitivity analysis, we dissect each component of ReAct-style prompting in a critical effort to understand the factor that leads to the observed success rates in these domains. With variations on the placement (RQ1) and content (RQ2) of the *think* tag, we eliminate it as the primary cause of any improvement. Furthermore, slight variations in exemplar tasks (RQ3) lead to a stark decline in success rate, clearly indicating the dependence of performance on the highly curated instance-specific examples by domain experts. While some works on In-Context Learning point out the impact of well-curated examples (Min et al., 2022; Peng et al., 2024), our work specifically highlights exemplar-query similarity as the cause of ReAct's performance and rejects contemporary belief that the heavy-lifting of LLM reasoning & planning is done through the *think* tag.

Relevance of ReAct to newer LLMs: ReAct uses the `Act` baseline in their work to showcase improvements due to the presence of the proposed *think* tag. For AlfWorld, ReAct reports 45% success rate for `Act` baseline and 71% success rate for `ReAct` prompting using the PaLM model. For WebShop, ReAct reports 30.1% success rate for `Act` baseline and 40% success rate for `ReAct` prompting on PaLM. However, we note from our results on both domains that the `Act` baseline performs much better than ReAct for several LLMs, which questions on the compatibility of ReAct to newer-age LLMs. ReAct performs worse with newer models as compared to the results they report on the currently decommissioned PaLM. This observation also questions the contemporary belief that such prompting strategies are generalizable throughout different LLM families, including newer models.

We re-iterate our key result, given any LLM model, the success rates plummet with our RQ3 variations showing a consistent pattern of dependence on the provided examples irrespective of the LLM. Moreover, the performance of all the LLMs remain quite high (if not better) when we vary the location and content of the *think* tags. This highlights the need for higher rigor in agentic LLM experimentation and in-depth evaluation seeking source of improvements.

6 Conclusion

ReAct based prompt engineering methods have been claimed to improve planning abilities of Large Language Models. In this study, we critically examine ReAct along three dimensions, informed by its claims and our hypotheses regarding its performance sources. Contrary to ReAct's claims, our findings reveal that its performance is **neither** due to interleaving reasoning trace and guidance information generation with action execution, **nor** due to the specific nature of the guidance information. Instead, we identify that the true source of LLM performance in sequential decision-making tasks, such as AlfWorld, is the high degree of similarity between exemplar problems (few-shot) and the query task. Our findings caution against an uncritical adoption of ReAct-style frameworks for their putative abilities to enhance performance in domains requiring planning. To conclude, we believe that these results will be helpful for practitioners and future works, particularly when designing prompts for decision-making problems, and benefit from avoiding putting any efforts into constructing non-trivial problem-specific reasoning traces.

Limitations

In this section, we aim to highlight some of the limitations of the sensitivity analysis done in this work on the popular ReAct-style prompting method. 1) We deliberately confined the scope of our experiments to be deep along two domains considered as common-sense planning domains by the contemporary research community. 2) Moreover, while restricting our discussion to sequential decision making problems (AlfWorld and WebShop), we admit that prompting solutions to reasoning problems other than ReAct must be re-evaluated with similar scrutiny but are beyond the scope of this work. 3) We do not aim to benchmark or analyze any single LLM's reasoning abilities on decision-making tasks, but rather intend to understand the robustness/brittleness of various LLMs with respect to different components in the ReAct-style prompting method for these tasks. Since we do not draw any insights on the performance of different LLMs in this work, we tried establishing baseline ReAct results for each of these models and analyzed the gain or drop in performance across our different prompt variations. While these can be seen as a potential limitations, we believe that our examination of ReAct can be easily translated to other domains and prompt strategies of interest by the consumer of this research.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, et al. 2023. Rest meets react: Self-improvement for multi-step reasoning llm agent. arXiv preprint arXiv:2312.10003.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 17682–17690.

Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu. 2024. Towards llm-guided causal explainability for black-box text classifiers.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712.

Lluis Castrejon, Thomas Mensink, Howard Zhou, Vittorio Ferrari, Andre Araujo, and Jasper Uijlings. 2024. Hammr: Hierarchical multimodal react agents for generic vqa. arXiv preprint arXiv:2404.05465.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, and Ziran Wang. 2024. Receive, reason, and react: Drive as you say, with large language models in autonomous vehicles. IEEE Intelligent Transportation Systems Magazine.

Richard E Fikes and Nils J Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence, 2(3-4):189–208.

Dayuan Fu, Jianzhao Huang, Siyuan Lu, Guanting Dong, Yejie Wang, Keqing He, and Weiran Xu. 2024. Preact: Predicting future in react enhances agent's planning ability. arXiv preprint arXiv:2402.11534.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd workers for text-annotation tasks. Proceedings of the National Academy of Sciences, 120(30):e2305016120.

Louie Giray. 2023. Prompt engineering with chatgpt: a guide for academic writers. Annals of biomedical engineering, 51(12):2629–2633.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. arXiv preprint arXiv:2402.01817.

Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, et al. 2023. Chatgpt: Jack of all trades, master of none. Information Fusion, 99:101861.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. Advances in neural information processing systems, 35:22199–22213.

Jieyi Long. 2023. Large language model guided tree-of-thought. arXiv preprint arXiv:2305.08291.

Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. 2023. Prompt engineering in large language models. In International Conference on Data Intelligence and Cognitive Informatics, pages 387–402. Springer.

R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths. 2023. Embers of autoregression: Understanding large language models through the problem they are trained to solve. arXiv preprint arXiv:2309.13638.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? arXiv preprint arXiv:2202.12837.

OpenReview. 2024. ReAct: Synergizing Reasoning and Acting in Language Models. https://openreview.net/forum?id=WE_vluYUL-X.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. arXiv preprint arXiv:2401.12087.

Shamik Roy, Sailik Sengupta, Daniele Bonadiman, Saab Mansour, and Arshit Gupta. 2024. Flap: Flow adhering planning with constrained decoding in llms. arXiv preprint arXiv:2403.05766.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927.

Rylan Schaeffer, Kateryna Pistunova, Samar Khanna, Sarthak Consul, and Sanmi Koyejo. 2023. Invalid logic, equivalent gains: The bizarreness of reasoning in language model prompting. arXiv preprint arXiv:2307.10573.

Sahil Sharma, Aravind Srinivas, and Balaraman Ravindran. 2017. Learning to repeat: Fine grained action repetition for deep reinforcement learning. arXiv preprint arXiv:1702.06054.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. arXiv preprint arXiv:2303.11366.

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

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. arXiv preprint arXiv:2409.12183.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024a. Chain of thoughtlessness: An analysis of cot in planning. arXiv preprint arXiv:2405.04776.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024b. On the self-verification limitations of large language models on reasoning and planning tasks. arXiv preprint arXiv:2402.08115.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. arXiv preprint arXiv:1803.05355.

Tomer Ullman. 2023. Large language models fail on trivial alterations to theory-of-mind tasks. arXiv preprint arXiv:2302.08399.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2024. On the planning abilities of large language models-a critical investigation. Advances in Neural Information Processing Systems, 36.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. 2024. Theory of mind abilities of large language models in human-robot interaction: An illusion? In Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, pages 36–45.

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

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. Mm-react: Prompting chatgpt for multimodal reasoning and action. arXiv preprint arXiv:2303.11381.

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

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations.

Yao Yao, Zuchao Li, and Hai Zhao. 2023. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. arXiv preprint arXiv:2305.16582.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2023. Reactable: Enhancing react for table question answering. arXiv preprint arXiv:2310.00815.

Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. 2023. Can chatgpt reproduce human-generated labels? a study of social computing tasks. arXiv preprint arXiv:2304.10145.

A Resources Used

In this work we leverage OpenAI API and Claude API for prompting the Language Models. We use `gpt-4-0613` for GPT4, `gpt-3.5-turbo-0125`, `gpt-3.5-turbo-instruct`, `claude-3-opus-20240229`, `claude-3-sonnet-20240229` and `claude-3-haiku-20240307` for AlfWorld in April-May 2024. Results using `GPT-4-0613` (for both AlfWorld and WebShop) and all the results on Webshop were completed in Sep 2024. As an estimate, for AlfWorld, ReAct and corresponding experiments use approximately 14M input tokens (due to repeated prompting after each action execution) and 150K output tokens for 134 problem instances as used by ReAct.

B Additional Considerations

B.1 Fine-grained performance on each task on AlfWorld

Figures 3, 4 and 5 shows a radar chart highlighting the failure cases of various LLMs with different ReAct based prompting variations for RQ1, RQ2 and RQ3 respectively on AlfWorld domain. Note that GPT-4 in the figures refer to GPT-4-0314 (Old Variant).

B.2 Failure Rates

We report failure rates in the radar chart as in Figs. 3, 4, 5 and 6 instead of success rates. We attempted to visualize the severe brittleness given by the larger area of the shaded region. Since, for various of our RQ variations the LLMs performance was very low, we decided to report failure rate given as (100 - Success Rate %) instead.

B.3 Performance of Claude-Haiku

We skip on mentioning the performance of Claude-Haiku, since it was not able to generate syntactically correct actions for any of the instances. We found that following our instruction to generate specific actions as in the exemplar was difficult. We improved the prompt to have specific instructions for generating actions (See D.3) but it did not yield any improvements for Claude-Haiku. However, the instruction did help with Claude-Sonnet and Claude-Opus. We find that Claude-Sonnet follows a similar pattern as GPT-3.5-Instruct as presented in our results, and decided to focus ourselves on the strongest/largest Claude model (Claude-Opus) for our evaluation.

B.4 Performance of GPT-4o and LLAMA-3.1 in AlfWorld

In our experiments, we found that GPT-4o obtained zero success-rate across all variations (including Act and ReAct and our proposed variations). Upon closer inspection, we found that GPT-4o requires significant effort in instruction tuning for AlfWorld, specifically, it would start emitting justifications for why a previously taken action was unsuccessful rather than generate syntactically accepted `think` tags and environment actions. Even for action generation, GPT-4o would pre-pend the actions with justifications, thereby expecting the users to write specific parsers. While stronger parsers maybe possible to implement, we hold GPT-4o to the same expectations as other models (which do not get the benefit of stronger parsers) and report our findings on a consistent evaluation.

LLAMA-3.1-8B obtains zero success-rate across all variations in Alfworld as well. We observe that the LLAMA model would generate incorrect actions and would repeat those actions exhausting the iteration budget.

B.5 Why is GPT-o1 missing in our evaluation bed?

We take up a large set of large langauge models in our evaluation and perform extensive evaluations for each of those. This is already in contrast with the baseline ReAct work which only showcased results on a single PaLM model. Our variations are agnostic to the underlying language model and code-base implements the standard `text-in-text-out` interface. However, ReAct-based agents are costly to evaluate. As of this submission, GPT-o1 incurs higher input/output token costs and also charges for generated thought tokens and would require a substantially larger budget to investigate than we can support.

B.6 Extension to other Models

We are in the process of experimenting with GPT-4o and Google Gemini models, APIs for which were released in May 2024 which does not allow enough time for thorough and verified evaluation before the conference submission. For completeness, however, we will experiment with these APIs as they become accessible and append our results.

B.7 Main Results on Exemplar CoT variant

While this work does not investigates effectiveness of exemplar Chain of Thought as presented in

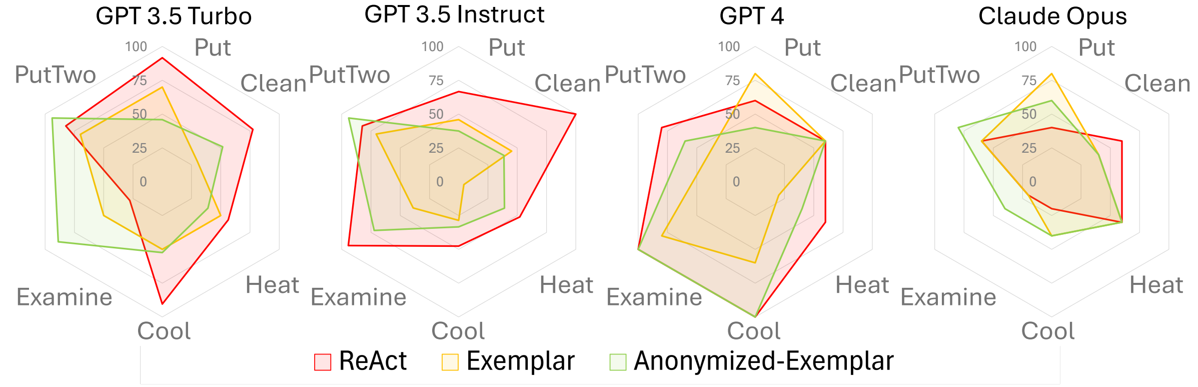


Figure 3: The radar chart shows the **failure rates** of various LLMs with different ReAct-based prompt settings for RQ1 (Base React, Global, Anonymized) across six Alfworld tasks (hexagon vertices). Higher values / Larger shaded region indicate worse performance.

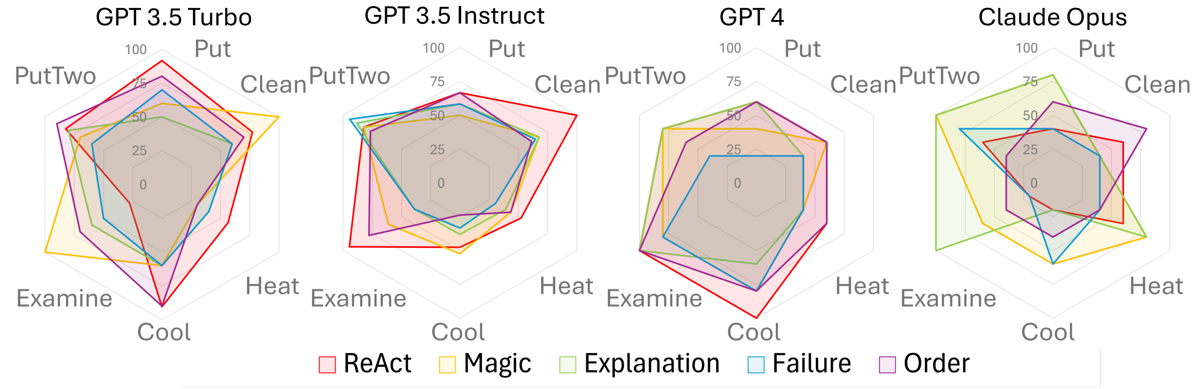


Figure 4: The radar chart shows the **failure rates** of various LLMs with different ReAct-based prompt settings for RQ2 (Base React, Magic, Failure, Failure+Explanation, Ordering) across six Alfworld tasks (hexagon vertices). Higher values / Larger shaded region indicate worse performance.

RQ1, we do however test the main results of the work with Exemplar CoT to identify whether our findings hold true there as well. That is, we test `RQ3-Both, RQ3-One`. For `GPT-3.5-Turbo` we find that the average performance drops from 46.6% (RQ3-Exemplar CoT) as in Table 1 to 28.3% in `One` and 10.4% in `Both` variation cases, and remains at 40.3% for `All` variation.

C Experiment Design

Each of the variations proposed along RQ1, RQ2 and RQ3 modifies the few-shot examples only. Remaining aspects such as the query problem or the interaction with the simulator is directly inherited from the ReAct code-base (Yao et al., 2022b) at publicly available at `https://github.com/ysymyth/ReAct`. Our code can be found in the attached supplementary material.

Except `All` RQ3 variation, all other settings use the standard two examplars for prompting the LLM. Depending on the variation we change the content of the exemplar. Full prompts can be found in the attached supplementary code.

C.1 Running the experiments

In our experiments, according to the variation style we take the exemplar prompts and use the same exemplar prompts across the instances of the query task. Other than `RQ3-Both, One` we use the exemplar of the same task as the query as done in ReAct (and still find brittleness of ReAct). For `RQ3 - Both, One` we use exactly two exemplars but of a different task than query. Finally, `RQ3-All` is the only variation that provides six exemplars (instead of two) and we force the exemplar of the query-task to be appended at the end in the prompt. This was the best performing prompting strategy (on GPT-3.5-Turbo) amongst when the query-task

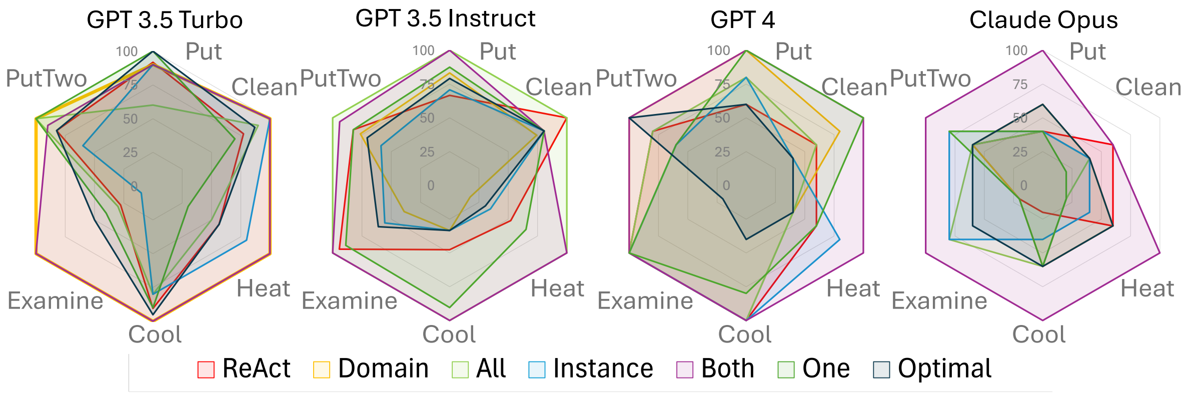


Figure 5: The radar chart shows the **failure rates** of various LLMs with different ReAct-based prompt settings for RQ3 (Base React, Domain, Instance, All, Both, One) across six Alfworld tasks (hexagon vertices). Higher values / Larger shaded region indicate worse performance.

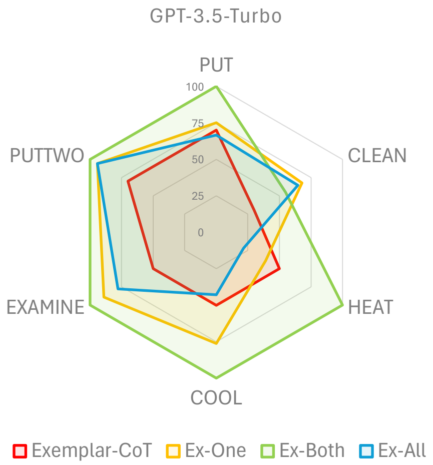


Figure 6: The radar chart shows the failure rates of GPT-3.5-Turbo with our RQ3 variants on ReAct across six Alfworld tasks. Higher values / Larger shaded region indicate worse performance.

exemplar was placed at the beginning, at position 4 (middle) and at the end.

C.2 Hyperparameters

We use $temperature = \tau = 0$ for all of the GPT and Claude models and set `max-tokens = 100` which is borrowed from ReAct's hyperparameters. Rest of the parameters are kept to be default as specified in the respective model's API documentation.

C.3 Additional Experiments and Error Analysis

Unrolling and Subtask Similarity We perform additional experiments where the query task is to essentially repeat the task in the exemplar (`Unrolling`). For instance, in AlfWorld, the exemplar is `Put` and the query is `PutTwo` to put two objects at given location. In this case, the LLM has to unroll the given advice and repeat exemplar task execution to solve the query. The success rate of GPT-3.5-Instruct (the best performing GPT model in our experiments) drops down from 52% to 9%. Similarly, we experiment with a `Subtask Similarity` variation where the exemplar task subsumes execution of the query task. For instance, the `Heat` task requires the agent to pick and place object in the microwave (which is an instantiation of `Put` task). One would expect that `Heat` is a good exemplar for `Put`, however, the performance of GPT-3.5-instruct model goes from 18% to 0% in this case. These results further underscore the brittleness and the need for instance-specific exemplars in ReAct.

Thought operationalization ability of LLMs Given the free form nature of thought generation and arbitrary nature of thought (about subtask, common-sense next steps etc.), checking whether the generated thoughts are in-fact reasonable is a challenging problem. For completeness, we find that 40% of the times after generation of a *think* tag, subsequent environment action taken by the LLM was invalid (for GPT-3.5-instruct) in AlfWorld. It is much higher (80% for GPT-3.5-Turbo, 90% for Claude-Haiku) for weaker LLM models. This further highlights the inability of LLMs to operationalize its generated thought as also seen in (Roy et al., 2024). From manual inspection we find that the typical thoughts would enlist all possible locations as next locations to visit for most of the tasks. As demonstrated in Section 5.2, the performance of LLMs actually decreases when provided with foresight guidance, as seen with the base ReAct model. A detailed investigation into the validity of the generated reasoning traces is beyond the scope of this work and is suggested as future research.

D Example Prompts

The full list of curated variations can be found in the supplementary materials. However, for completeness we reference the prompt used for base ReAct (as in (Yao et al., 2022b)) and our variations along RQ1, RQ2 and RQ3 for the `Put` task.

D.1 Synonym Substitution mapping for `Domain`

We make the following substitutions to the object names / locations in the exemplar prompt in the `Domain` variation. Note that these substitutions are done only to the exemplar, and the query problem and subsequent interaction with the AlfWorld simulator uses the original vocabulary mapping.

```
spraybottle -> aerosolbottle
cabinet -> cupboard
countertop -> worktop
sinkbasin -> sinkbowl
toilet -> lavatory
toiletpaperhanger -> toiletpaperholder
towelholder -> towelrack

microwave -> oven
shelf -> rack
drawer -> deskdrawer
stoveburner -> hob
diningtable -> table
garbagecan -> trashbin

fridge -> refrigerator
peppershaker -> pepperpot
room -> livingroom
bread -> breadloaf
pan -> fryingpan
pot -> saucepan
book -> notebook

creditcard -> amexcard
mirror -> lookingglass
dresser -> chestofdrawers
sofa -> couch
cellphone -> mobilephone
coffeemachine -> coffeemaker
knife -> kitchenknife
spatula -> turner
soapbottle -> liquidsoapdispenser
saltshaker -> saltpot
statue -> sculpture
vase -> flowerpot
dishsponge -> spongewipe
desklamp -> tablelamp
sidetable -> nightstand
```

D.2 For `All, Both, One`

All: We take the exemplar prompt for each task and concatenate it together.

Both : We use the following mapping generated arbitrarily, to replace the exemplar prompt for the query task.

```
VARIATION_MAPPING = {
    'put':'examine',
    'clean':'cool',
    'heat':'put',
    'cool':'puttwo',
    'examine':'clean',
    'puttwo':'heat'
}
```

One : We use the first example from the above mapping, and the second example from the same task as the query.

D.3 Building the input prompt

We inherit ReAct's way of stiching together the exemplars and the query using the following prompt template as :

`Interact with a household to solve a task. Here are two examples.’ + eg1 + eg2 + INSTRUCTION + Here is the task. (TASK)´`

In the case of GPT models `INSTRUCTION` is an empty string. Claude Models gave 0% success rate (regardless of the model size, i.e. Haiku, Sonnet and Opus) with the original ReAct prompt design. We identified that Claude required an explicit instruction to generate only the actions for the simulator. We added a task independent INSTRUCTION prompt as :

`INSTRUCTION = " Only provide the one action at a time. Be concise, and do not provide any extra information. Always start with the action. For example, ’pick up apple’ is correct, but ’I want to pick up the apple’ is not."`

Interestingly, GPT-3-Turbo performance dropped to 1% (averaged across 6 independent runs over 134 examples) with the inclusion of this INSTRUCTION tag. This further highlights how brittle prompt engineering is, and cautions us against misplacing our trust with ReAct in enhancing reasoning abilities of LLMs.

Note that, we did experiment with both prompt variations (with our explicit instruction and without) for Claude models and picked the best-performing variant (which had the instructions) for our reported results.

D.4 RQ1

Prompt 1: rq1-anon-exemplar

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take the object, then put it on the desired location. First I need to find the object. The object is more likely to appear in one of the different locations. I can check one by one, starting with the first location. Once I find the object, next I need to take it. Once I take the object, next I need to put it in/on the desired location.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

Prompt 2: rq1-exemplar

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take an apple, then put it in sidetable. First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with fridge 1. Once I find an apple, next I need to take it. Once I find an apple, next I need to put it in/on sidetable.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

Prompt 3: base-react

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with fridge 1.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Now I find a apple (3). Next, I need to put it in/on sidetable 1.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

D.5 RQ2

Prompt 4: rq2-Explanation

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> go to fridge 1
The fridge 1 is closed.
> close cabinet 2.
Nothing Happens.
> think: Nothing happens because you are not at cabinet 2.
OK.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> put apple 3 in/on sidetable.
Nothing Happens.
> think: Nothing happens because I do not have apple 3.
OK.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

Prompt 5: rq2-ordering

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to put an apple in sidetable, for which I need to find and take an apple.
OK.
> think: An apple is more likely to appear in sidetables (1), fridges (1), countertops (1), diningtables (1-3), garbagecan (1), sinkbasins (1). First I need to find an apple. I can check one by one, starting with fridge 1.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Next, I need to take the apple.Now I find an apple (3).
OK.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Next, I need to put the apple in/on sidetable 1. Now I find an apple (3).
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

Prompt 6: rq2-Placebo

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: Take a deep breath and work on this problem step-by-step.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Take a deep breath and work on this problem step-by-step.
OK.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Take a deep breath and work on this problem step-by-step.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

Prompt 7: rq2-Failure

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> go to fridge 1
The fridge 1 is closed.
> close cabinet 2.
Nothing Happens.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> put apple 3 in/on sidetable.
Nothing Happens.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

D.6 RQ3

Prompt 8: rq3-Domain

```
You are in the middle of a livingroom. Looking quickly around you, you see a cupboard 4, a cupboard 3, a cupboard 2, a cupboard 1, a coffeemaker 1, a worktop 1, a table 3, a table 2, a table 1, a deskdrawer 1, a refrigerator 1, a trashbin 1, a oven 1, a nightstand 1, a sinkbowl 1, a hob 4, a hob 3, a hob 2, a hob 1, and a toaster 1.
Your task is to: find some apple and put it in nightstand.
> think: To solve the task, I need to find and take an apple, then put it in nightstand.
OK.
> think: First I need to find an apple. An apple is more likely to appear in refrigerators (1), tables (1-3), nightstands (1), worktops (1), sinkbowls (1), trashbin (1). I can check one by one, starting with refrigerator 1.
OK.
> go to refrigerator 1
The refrigerator 1 is closed.
> oballpointpen refrigerator 1
You oballpointpen the refrigerator 1. The refrigerator 1 is oballpointpen. In it, you see a lettuce 2, a mug 2, a saucepanato 2, and a tomato 1.
> go to table 1
On the table 1, you see nothing.
> go to table 2
On the table 2, you see a pineapple 1.
> go to table 3
On the table 3, you see a bowl 2, a bowl 1, a breadloaf 2, and a breadloaf 1.
> go to nightstand 1
On the nightstand 1, you see a cup 1, a lettuce 1, a peppersaucepan 3, a saucepanato 1, and a saltpot 1.
> go to worktop 1
On the worktop 1, you see a butterknife 1, a spongewipe 1, and a liquidsoapdispenser 1.
> go to sinkbowl 1
On the sinkbowl 1, you see a cup 2, a egg 1, a mug 1, and a saucepanato 3.
> go to trashbin 1
On the trashbin 1, you see a apple 3, and a egg 3.
> think: Now I find an apple (3). Next, I need to take it.
> take apple 3 from trashbin 1
You pick up the apple 3 from the trashbin 1.
> think: Now I find an apple (3). Next, I need to put it in/on nightstand 1.
OK.
> go to nightstand 1
On the nightstand 1, you see a cup 1, a lettuce 1, a peppersaucepan 3, a saucepanato 1, and a saltpot 1.
> put apple 3 in/on nightstand 1
You put the apple 3 in/on the nightstand 1.
```

Prompt 9: rq3-Instance

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take an apple, then put it in sidetable.
OK.
> think: First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with fridge 1.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dishsponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Now I find an apple (3). Next, I need to take it.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Now I find an apple (3). Next, I need to put it in/on sidetable 1.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```

Prompt 10: rq3-Optimal

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take an apple, then put it in sidetable.
OK.
> think: First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with garbagecan 1.
OK.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Now I find an apple (3). Next, I need to take it.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Now I find an apple (3). Next, I need to put it in/on sidetable 1.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.
```