DKPROMPT: Domain Knowledge Prompting Vision-Language Models

Xiaohan Zhang Zainab Altaweel* Yohei Hayamizu* Yan Ding Saeid Amiri Shiqi Zhang
State Univeristy of New York at Binghamton
{xzhan244, zaltaweel, yhayami1, yding25, samiri1, zhangs}@binghamton.edu

1. Introduction

Prompting foundation models such as large language models (LLMs) and vision-language models (VLMs) requires extensive domain knowledge and manual efforts, resulting in the so-called "prompt engineering" problem. One can provide examples explicitly [1] or implicitly [4], or encourage intermediate reasoning steps [8, 9] to improve the performance of foundation models. However, those methods as applied to LLMs and VLMs still lack the theoretical guarantee and provable correctness. Our idea is to leverage the foundation of classical AI, i.e., knowledge representation and reasoning, to develop a prompting strategy that enables the VLMs to verify the correctness of an agent's behavior at execution time, in the real world.

Given the natural connection between planning symbols and human language, this paper investigates how pre-trained VLMs can assist the robot in realizing symbolic plans generated by classical planners, while avoiding the engineering efforts of checking the outcomes of each action. Specifically, we propose a novel closed-loop task planning and execution framework called DKPROMPT, which prompts VLMs using domain knowledge in PDDL, generating visually grounded, provably correct task plans. DKPROMPT leverages VLMs to detect action failures and verify action affordances towards successful plan execution (Figure 2). We take the advantage of the domain knowledge encoded in classical planners, including the actions defined by their effects and preconditions. By simply querying current observations against the action knowledge, similar to applying VLMs to Visual Question Answering (VQA) tasks, DKPROMPT can trigger the robot to repeat an unsuccessful action recovering from previous failures, or call the symbolic planner to generate a new valid plan.

We conducted quantitative evaluations in the OmniGibson simulator, where we reused some tasks from the Behavior-1K benchmark [5]. We provided predefined parameterized actions for DKPROMPT, as well as other baselines, and these actions are *imperfect* by nature, frequently causing situations (Figure 1). Experimental results demonstrate that DKPROMPT is able to recover from action failures and re-plan when situations occur. As a result, our approach outperforms competitive baselines from the literature, achieving the highest task completion rate.

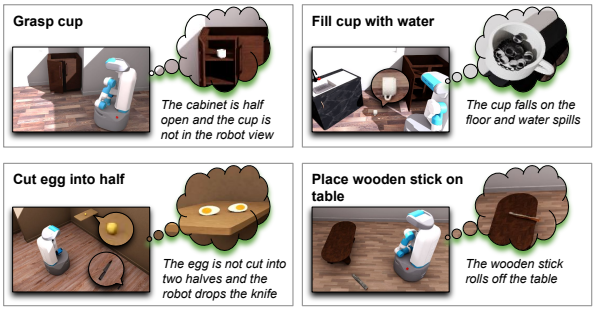

Figure 1. A few unforeseen situations during action execution. In the top-left example, the robot "opened" the cabinet door to get prepared for grasping the cup. It was expected that the cup in white would have been in the robot's view after the "opening" action, while a situation occurred, i.e., the cabinet was only half-open. DKPROMPT prompts vision-language models (VLMs) using domain knowledge to detect and address such situations. The goal is to compute visually grounded, provably correct plans.

2. DKPROMPT

Before every action execution, DKPROMPT extracts knowledge about action preconditions from the planner's domain description. For instance, as indicated in Figure 2, action `graspfrom(a, o, r)` has preconditions of `open(r)`, `inview(a, o)`, `inside(o, r)`, and `handempty(a)`, meaning that to grasp an object o from a receptacle r, r should be open (not closed), o should be in the agent's current first person view, o should be inside r, and the agent's hand should be empty. Then, we simply convert each action precondition into a natural language query by using manually defined templates, such as *"Is <o> inview agent?"* and *"Is <o> inside <r>?"* Paring each natural language query with the current observation from the robot's first-person view, we call the VLM to get answers indicating if the precondition is satisfied.

According to the results (i.e., "yes", "no", or "skip" if unsure) from the VLM, DKPROMPT will update the

* indicates equal contribution.

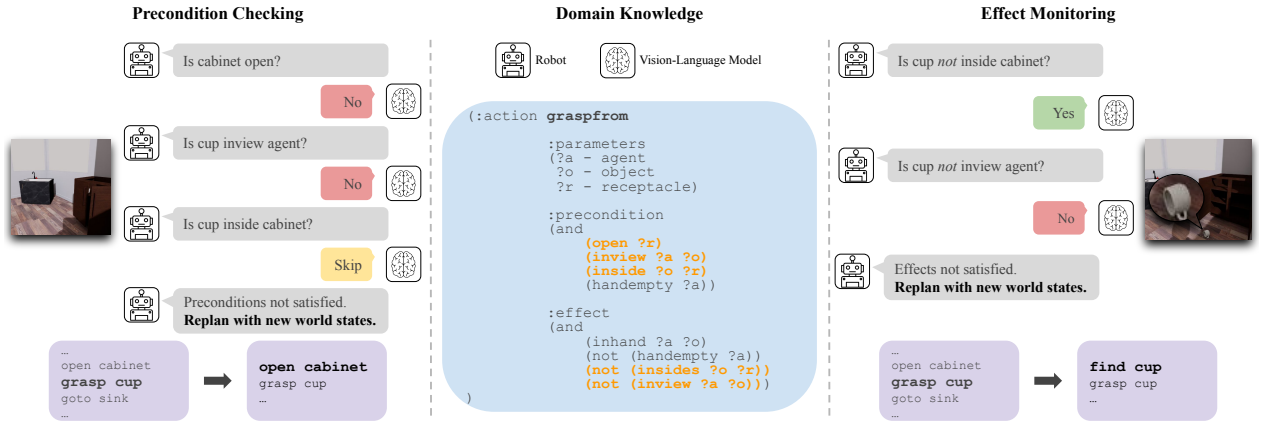

Figure 2. An overview of DKPROMPT. By simply querying the robot's current observation against the domain knowledge (i.e., action preconditions and effects) as VQA tasks, DKPROMPT can call the classical planner to generate a new valid plan using updated world states. Note that DKPROMPT only queries about predicates. The left shows how DKPROMPT checks every precondition of the action to be executed next, and the right shows how it verifies the expected action effects are all in place after action execution. Replanning is triggered when preconditions or effects are unsatisfied after updating the planner's action knowledge.

current state information in the classical planning system. Figure 2 (Left) shows an example where the robot wants to `graspfrom(cup, cabinet)` but fails to detect "cabinet is open", "cup is inview of agent", and is suspecious about if "cup is in the cabinet" (the VLM answers "skip" to this question) given the current observation. As a result, DKPROMPT will update the current state by changing `open(cabinet)` to `closed(cabinet)`, and removing `inview(agent, cup).inside(cup, cabinet)` will remain the same because we do not update the state if the VLM answers "skip", indicating the agent holds a positive attitude that situations will not commonly occur. We then provide the updated world state to the classical planner as the "new" initial state to re-generate a plan. In the above example, instead of `graspfrom(cup, cabinet)`, the robot will now take the action of `open(cabinet)` again according to the newly-generated action plan. After every action execution, DKPROMPT extracts knowledge about action effects from the planner's domain description, illustrated in Figure 2 (Right). It queries action effects by using the VLM. If the effects are not satisfied, the robot will update its belief on the current states and re-plan accordingly.

3. Experiments

Our hypothesis is DKPROMPT produces the highest task completion rate because of its effectiveness in plan monitoring and online re-planning using domain knowledge and perception. Figure 3 presents the main experimental results and details the comparative success rates of various methods from the literature [2, 3]. Important findings are: 1) By incorporating domain knowledge for prompting, DKPROMPT is significantly better than methods that query about actions solely by their names (e.g., Suc.Aff.-QA), indicating that action knowledge is more informative for pretrained VLMs to reason over; 2) Though action effects might have more direct impacts on the overall success rate than preconditions, considering both of them is always a good practice in closed-loop systems; 3) Our evaluation benchmark (designed with open-world situations and parameterized actions) presents a significant challenge, both for the community focused on robotic planning (see the open-loop baseline) and for assessing the vision-language understanding abilities of large-scale models (see the results of GPT-4 [6], Gemini 1.5 [7], and Claude 3).

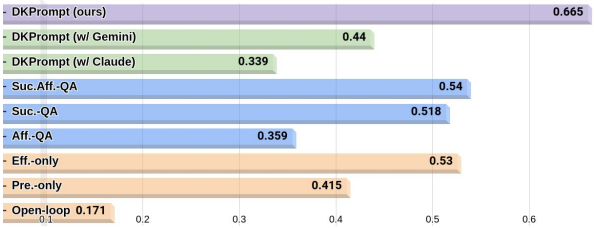

Figure 3. DKPROMPT (with GPT-4) is shown as the purple bar achieving the best performance in task completion rate. Green bars are evaluations of DKPROMPT with other VLMs. Blue bars are from the literature, where Suc.-QA asks if actions are successful, Aff.-QA asks if actions are executable, and Suc.Aff.-QA asks both. Orange bars are ablative versions of our approach, where Eff.-only queries about effects, Pre.-only queries about preconditions, and Open-loop queries neither of them thus making it an open-loop automated planning method.

4. Conclusion

We propose DKPROMPT that prompts VLMs using action knowledge in PDDL towards task planning and plan monitoring with provably correctness. By doing a set of experiments on robots working on everyday tasks, we demonstrate that DKPROMPT produces higher success rates compared with baselines from the literature.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[2] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. 2

[3] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023. 2

[4] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 1

[5] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023. 1

[6] OpenAI. Gpt-4 technical report, 2023. 2

[7] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. 2

[8] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837, 2022. 1

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 1