Vision-Language Interpreter for Robot Task Planning

Keisuke Shirai[1*], Cristian C. Beltran-Hernandez[2], Masashi Hamaya[2], Atsushi Hashimoto[2], Shohei Tanaka[2], Kento Kawaharazuka[3], Kazutoshi Tanaka[2], Yoshitaka Ushiku[2], Shinsuke Mori[4]

***Abstract*— Large language models (LLMs) are accelerating the development of language-guided robot planners. Meanwhile, symbolic planners offer the advantage of interpretability. This paper proposes a new task that bridges these two trends, namely, *multimodal planning problem specification*. The aim is to generate a problem description (PD), a machine-readable file used by the planners to find a plan. By generating PDs from language instruction and scene observation, we can drive symbolic planners in a language-guided framework. We propose a Vision-Language Interpreter (ViLaIn), a new framework that generates PDs using state-of-the-art LLM and vision-language models. ViLaIn can refine generated PDs via error message feedback from the symbolic planner. Our aim is to answer the question: How accurately can ViLaIn and the symbolic planner generate valid robot plans? To evaluate ViLaIn, we introduce a novel dataset called the problem description generation (ProDG) dataset. The framework is evaluated with four new evaluation metrics. Experimental results show that ViLaIn can generate syntactically correct problems with more than 99% accuracy and valid plans with more than 58% accuracy. Our code and dataset are available at `https://github.com/omron-sinicx/ViLaIn`.**

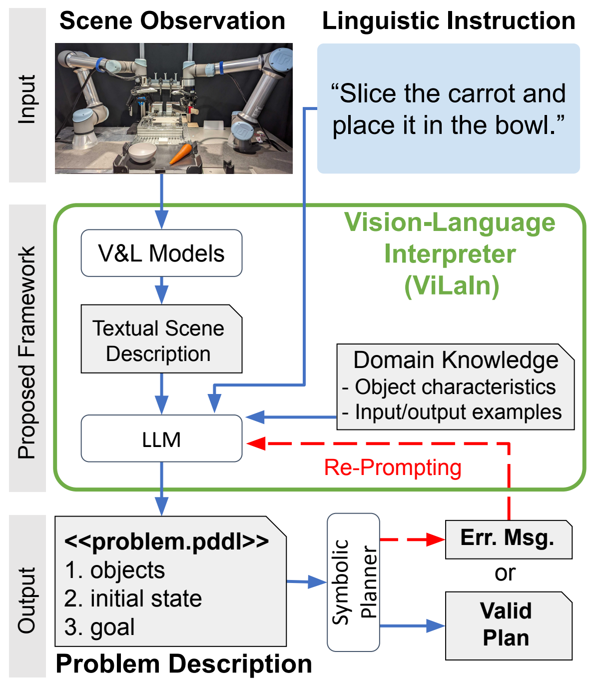

Fig. 1. Overview of our approach. The vision-language interpreter (ViLaIn) generates a problem description from a linguistic instruction and scene observation. The symbolic planner finds an optimal plan from the generated problem description.

I. INTRODUCTION

Natural language is a prospective interface for non-experts to instruct robots intuitively [1]–[3]. Earlier studies have used recurrent neural networks [4], [5] to map abstract linguistic instructions to representations for robots [1], [6], [7]. Here, the linguistic instructions represent desired goal conditions. More recent studies use large language models (LLMs) [8]–[10] to directly generate robot plans from the instructions [11]–[14]. These language-guided planners utilize few-shot prompting to solve tasks without training [15]. The plans are a sequence of discrete symbolic actions (e.g., `pick(a)` and `place(a, b)`) that complete the task. We aim to strengthen the language-guided planners in terms of the improvement of interpretability.[1] Interpretability is essential to gain the trust of the user and provide insights into the robot's decision-making process [16]. For example, the identification of failure causes through interpretation leads to continuous improvement of overall performance.

Robot task planning has traditionally been solved using symbolic planning [17]. Modern symbolic planners use the Planning Domain Definition Language (PDDL) to describe planning problems. In PDDL, a planning problem is defined in two parts: the *domain* that defines the state of variables and actions, and a *problem description* (PD) that defines the objects of interest, their initial state, and the desired goal state [18], [19]. The domain and problem are inputs to the planner to find an optimal plan, a sequence of symbolic actions.

Symbolic planners offer several benefits. The domain and problem descriptions are human-readable, especially when variable names are chosen intuitively. Moreover, the obtained plans are guaranteed to be logically correct. Considering these advantages, combining symbolic planning and language-guided planning is a promising research direction to realize interpretable robots. To that end, we proposed generating the PDs from natural language instructions. Since the linguistic instructions only represent the goal conditions,

[1]Keisuke Shirai and Shinsuke Mori are with Kyoto University, Kyoto 606-8501, Japan (email: [shirai.keisuke.64x@st.kyoto-u.ac.jp,forest@i.kyoto-u.ac.jp])

[2]Cristian C. Beltran-Hernandez, Masashi Hamaya, Atsushi Hashimoto, Shohei Tanaka, Kazutoshi Tanaka, Yoshitaka Ushiku are with the OMRON SINIC X Corporation, Tokyo 113-0033, Japan (email: [cristian.beltran, masashi.hamaya, atsushi.hashimoto, shohei.tanaka, kazutoshi.tanaka, yoshitaka.ushiku]@sinicx.com)

[3]Kento Kawaharazuka is with the University of Tokyo, 73-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan (email: kawaharazuka@jsk.t.u-tokyo.ac.jp)

[*]Work was done while the first author was an intern at OMRON SINIC X Corporation.

[1]We define interpretability as a mechanism to provide insights into the inner workings of the system.

TABLE I

DIFFERENCES BETWEEN PREVIOUS STUDIES AND OURS

Approach	Input other than linguistic instruction	Output
Huang et al. [11]	—	Symbolic action
Raman et al. [12]	—	Symbolic action
Text2Motion [13]	PDDL scene desc.	Symbolic action
SayCan [23]	Image	Pre-defined skill
RT-2 [24]	Image	Low-level action
ProgPrompt [14]	—	Program code
Code as Policies [3]	Image	Program code
LLM+P [25]	Linguistic scene desc.	Problem desc.
ViLaIn (ours)	Image	Problem desc.

additional information about the environment is required to generate the initial state (e.g., an image representing the current environment). We refer to this additional information as *scene observations*.

We tackle the *multimodal planning problem specification* task, a new task for transforming linguistic instructions and scene observations into logically and semantically correct PDs. The PDs have to be executable by the symbolic planners. This paper investigates how accurately we can generate such PDs with a state-of-the-art LLM [9] and vision-language model [20], [21] without additional training. We propose a Vision-Language Interpreter (ViLaIn), a new framework to solve the PD generation task, illustrated in Fig. 1. ViLaIn consists of three modules that generate each part of the PDs. The complete PD is assembled by concatenating these parts. Furthermore, ViLaIn can refine the generated PDs via error feedback from the symbolic planner. The planner uses a pair of the generated PD and the domain description to find a plan. We use the state-of-the-art symbolic planner called Fast Downward [22] throughout this paper.

To evaluate ViLaIn, we introduce a novel dataset called the problem description generation (ProDG) dataset. The ProDG dataset consists of linguistic instructions, scene observations, and domain and problem descriptions. The descriptions are written in PDDL [19]. This dataset covers three domains: cooking as a practical robot domain, and the blocks world and the tower of Hanoi as classical planning domains. We propose four new evaluation metrics to evaluate ViLaIn from multiple perspectives.

The main contributions of this work are three-fold:

- Multimodal planning problem specification, a new task to bridge the language-guided planning and symbolic planners with scene observations.
- Vision-Language Interpreter (ViLaIn), a new framework consisting of a state-of-the-art LLM and vision-language model. ViLaIn can refine erroneous PDs by using error messages from the symbolic planner.
- The problem description generation (ProDG) dataset, a new dataset that covers three domains: the cooking domain, the blocks world, and the tower of Hanoi. The dataset comes with new metrics that evaluate ViLaIn from multiple perspectives.

II. RELATED WORK

This section describes previous work on language-guided planning, symbolic planning, and scene recognition in computer vision. Table I summarizes the difference between several studies mentioned here and ViLaIn.

A. Planning from Natural Language

Task planning from natural language has been actively studied [11], [20], [23]. Converting linguistic instructions into symbolic actions via neural networks is a typical approach [7], [26]. More recent studies [11]–[14] use LLMs and directly generate plans with few-shot prompting [15]. However, these language-guided planners have two issues. First, their systems hide the inner workings by generating plans end-to-end. Second, the obtained plans are not guaranteed to be logically correct. ViLaIn resolves these issues by converting instructions into human-readable PDs and driving symbolic planners to find plans with the generated PDs. A recent study uses LLMs to convert linguistic instructions and images into programs to complete robot tasks [3]. PDs describe tasks more specifically, and their logical correctness is automatically verifiable. In other words, ViLaIn has the potential to deliver validated machine-readable information to other language-guided planners as an auxiliary input.

More recent studies have used LLMs to convert natural language inputs to PDs [25], [27]. However, one study [25] assumes that scene descriptions (the objects and initial state) are provided in natural language, which is not practical for real applications. Another work [27] focuses on only generating the goal specifications. Contrary to these studies, ViLaIn uses images for scene descriptions and generates the whole PDs, including the objects and initial states.

B. Symbolic Planning with PDDL

Symbolic planning (automated planning) has been used to solve robotic tasks [17]. Symbolic planners [22], [28] use domain and problem descriptions to find plans, which are sequences of (symbolic) actions that alter the environment from its initial state to a goal state. The descriptions are written in formal languages, such as PDDL [19] and PDDLStream [29]. Robots execute low-level actions based on the found high-level plans of PDDL [30]–[32]. This framework enables robots to solve various problems but assumes a preparation of corresponding PD for each problem. ViLaIn is designed to collaborate with those PDDL-based planning frameworks by translating linguistic instructions into PDs.

C. Scene Recognition for Planning Problem Specification

The generation of the objects and initial state in PD is related to research in computer vision. This section briefly overviews such previous work.

The object part of PDs lists objects required for the task. This work generates the objects from scene observations. This can be viewed as object detection in computer vision. Classical object detectors [33], [34] have been developed focusing on a fixed number of classes (e.g., person and dog). However, our task handles objects not included in the classes.

Hence, we use an open-vocabulary object detector [20], [35]. These detectors have recently gained attention because they can detect arbitrary objects using text queries.

The *initial state* represents object relationships and their states. Detecting such scene descriptions from images has been addressed on visual relationship detection [36], [37] or scene graph generation [38], [39]. Previous work trained a model with PDDL predicates and demonstrated it in real robot domains [40]. We use a state-of-the-art LLM and vision-language model to generate the initial state.

III. PROBLEM STATEMENT

We focus on multimodal planning problem specification, a new task for bridging language-guided planning and symbolic planning. The input is a quadruple (L, S, D_D, D_K); a linguistic instruction L, a scene observation S, a domain description D_D, and domain knowledge D_K. L is a sequence of words describing the task. S is an RGB image describing the initial state of the environment. D_D defines parts common to all problems: object types (e.g., `location` and `tool`), predicates (e.g., `at` and `clear`), and symbolic actions (e.g., `slice` and `pick`). D_K supports D_D by providing more specific information on each problem, such as object characteristics (e.g., the cutting board is round, the counter is black) and actual input/output examples. Note that the examples in D_K use the object types and predicates defined in D_D.

The output is a PD P consisting of (O, I, G): the objects O, the initial state I, and the goal specification G. O consists of objects required for the task completion (e.g., *carrot* and *knife*). I consists of a set of propositions that represent the initial state of the environment (e.g., `(at carrot counter)`). A proposition is formed by providing a predicate with arguments. For example, providing a predicate `(at ?a1 ?a2)` with (`a1`, `a2`) = (`carrot`, `cutting_board`) forms a proposition `(at carrot cutting_board)` meaning "the carrot is at the cutting board." G consists of a set of propositions that represent the desired goal condition of the environment. For example, `(and (at carrot bowl) (is-sliced carrot))` represents the goal condition that "the carrot should be sliced and should be at the bowl." P and D_D are written in PDDL [19], following previous work [25], [27]. We refer to O, I, or G with PDDL (e.g., the PDDL objects). The goal of this task is obtaining a function $M : (L, S, D_D, D_K) \rightarrow (O, I, G)$. P must be machine-readable and executable by the symbolic planner.

IV. VISION-LANGUAGE INTERPRETER

ViLaIn consists of three modules: the object estimator, the initial state estimator, and the goal estimator. We describe these modules in this section.

A. Object Estimator

The PDDL objects O list objects of interest in the scene observations S. However, the observed objects vary greatly from domain to domain. Further, it must recognize various objects that classical object detectors cannot handle. For this reason, we use Grounding-DINO [20], a state-of-the-art open-vocabulary object detector. Fig. 2 illustrates the estimator. We assume that the list of objects for the task is known. The object list can be used as the text query. However, we found from preliminary experiments that simply using the object list fails to detect several objects. To address this issue, we elaborate the query using the domain knowledge (e.g., "cutting board" → "round cutting board" and "knife" → "kitchen knife").[2] In our setting, these elaborated queries are included in the domain knowledge D_K. The detected objects are converted into a PDDL format by rules.

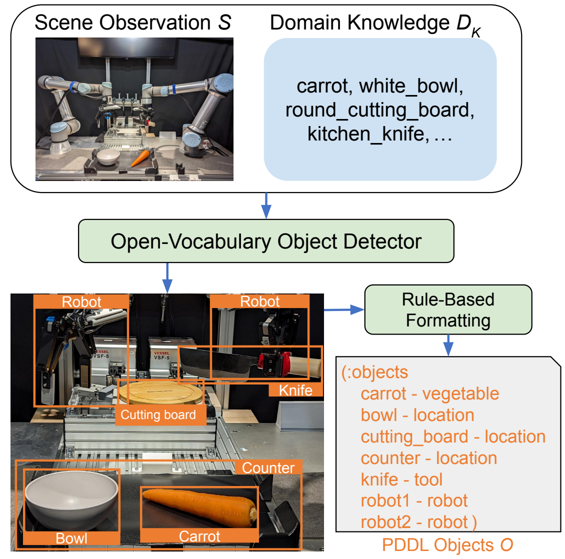

Fig. 2. The open-vocabulary object detector detects objects from the observation. The text query is provided by the domain knowledge. The detected objects are converted into a PDDL format in a rule-based way.

B. Initial State Estimator

The PDDL initial states I must specify the initial state of the environment using propositions. Here, different predicates from D_D should be used for different domains to represent the propositions. In addition, omitting a single proposition could cause an invalid PD by making reaching the goal from the initial state impossible. We implement the initial state estimator with a combination of an LLM and image captioning model. Fig. 3 shows the estimator. We use BLIP-2 [21] as the captioning model and GPT-4 [9] as the LLM. Given the objects' bounding boxes, BLIP-2 generates captions for each object with a prompt of "Q: what does this object describe? A: ." GPT-4 generates the PDDL initial state I from the bounding boxes and captions. GPT-4 uses few-shot prompting and leverages input/output examples in D_D to derive available predicates.

C. Goal Estimator

The PDDL goal specifications G must represent the desired goal conditions specified by the linguistic instructions

[2] In this work, we assume that the domain knowledge is created by humans, and we leave the automatic generation of it to future work.

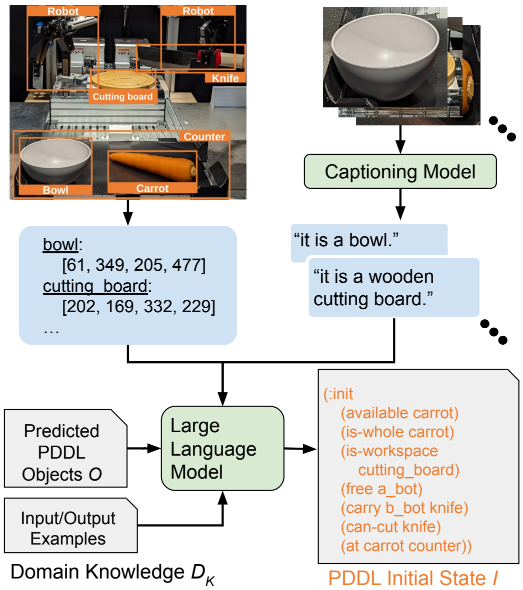

Fig. 3. The captioning model generates captions for each object. The LLM generates the PDDL initial state from the bounding boxes and the captions using few-shot prompting.

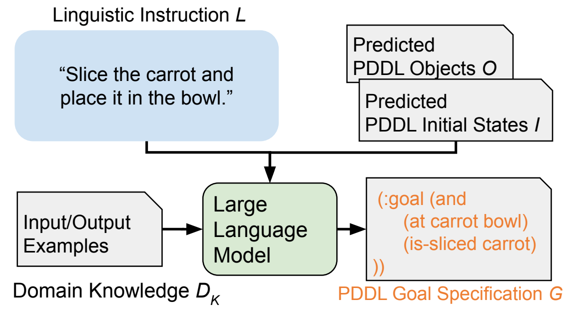

Fig. 4. The LLM directly generates the PDDL goal specification from the instruction and the PDDL objects and initial state using few-shot prompting.

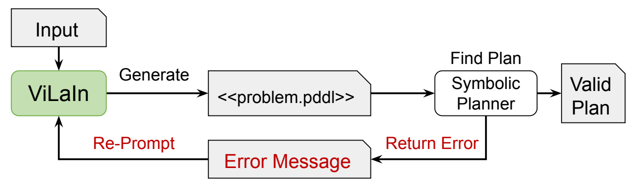

Fig. 5. ViLaIn can refine the generated problem description via an error message from the planner.

L. Generating G requires O to refer to the object list and I to consider the relationships of the objects. We implement the goal estimator with an LLM, following previous work [13], [27]. Fig. 4 shows the estimator. We use GPT-4 to generate G from L, O, and I. Similarly to Section IV-B, GPT-4 uses few-shot prompting with D_K.

D. Corrective Re-Prompting

Generated PDs are used by the planner to find plans. The planning might fail in the following two cases. One is when the PDs are syntactically incorrect. Generating propositions with undefined objects in O or undefined predicates in D_D results in such PDs (e.g., create `(at cucumber counter)`, but `cucumber` is not listed in O). The other is when the generated O is unreachable from the generated I. Contradictory propositions create such a PD (e.g., both of a proposition `(on red_block blue_block)` and the opposite one `(on blue_block red_block)` exist in I). In both cases, the planner stops planning and returns an error message, a clue to refine the erroneous parts. It is ideal if the system automatically refines the PDs via the error messages. ViLaIn has such a mechanism, and we describe it in this section.

When the planning fails, ViLaIn creates a prompt and re-prompts GPT-4 to refine the PD. We refer to this technique as Corrective Re-prompting (CR), following previous work [12]. Fig. 5 shows ViLaIn with CR. The prompt consists of input/output examples in D_K, the current input (L and S), the generated problem P, and the error message.

Chain-of-Thought prompting: We use Chain-of-thought (CoT) prompting [41]–[43] to further strengthen CR. CoT is a technique for solving complex reasoning tasks by LLMs. CoT introduces an intermediate reasoning step before generating the final output. With CoT, GPT-4 generates an explanation of the error message with a prompt template of "What part of the PDDL problem do you think is causing this error?." The generated explanation is then added to the input prompt, and GPT-4 generates the refined problem based on it. CR with CoT can be repeated as often as necessary until the planner returns error messages. In the rest of this paper, ViLaIn generates the PDs using CR with CoT unless otherwise specified. Note that ViLaIn performs CR with CoT only if the planner returns an error message.

V. Dataset

The evaluation of ViLaIn requires a dataset with linguistic instructions, scene observations, and PDDL domains and problems. However, to our knowledge, no such dataset has been proposed. To this end, we created the ProDG dataset. The ProDG dataset consists of three domains: cooking, the blocks world (Blocksworld), and the tower of Hanoi (Hanoi).

Cooking is a simplified task of making a salad. Planning is simpler than the other two domains because it only considers slicing vegetables and placing them in the bowl. Cooking actions are supposed to be performed by two robot arms installed on both sides of the environment. The left and right robot arms are named `a_bot` and `b_bot`, respectively, in O. This domain handles a greater variety of objects than the other domains. G represents the vegetable state and location.

Blocksworld is a classical planning domain [44]. Fewer types of objects than the cooking appear, but a longer horizon planning is required. Seven colored blocks without duplicates are used for each problem. A robot arm does not hold anything at first. G specifies the relationships of the blocks.

Hanoi is a classical planning domain [45]. Similarly to Blocksworld, a longer horizon planning with fewer types of objects than the cooking domain is required. Ten disks with six colors and three pegs are used. Disks of the same

TABLE II

DEFINED OBJECT TYPES, PREDICATES, AND ACTIONS IN THE DOMAIN DESCRIPTIONS

Domain	Object types	Predicates	Actions
Cooking	vegetable, location, tool, robot	available, is-whole, is-sliced, free, carry, can-cut, at, at-workspace	pick, place, slice
Blocksworld	block, robot	on, ontable, clear, handempty, handfull, holding	pick-up, put-down, stack, unstack
Hanoi	disk, peg	clear, on, smaller, move	move

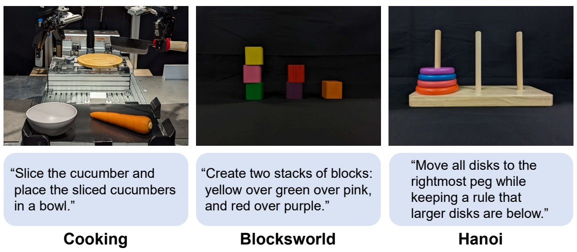

Fig. 6. Examples of scene observations and linguistic instructions.

TABLE III

PERFORMANCE ON THE PRODG DATASET

Domain	R_{syntax}	R_{plan}	R_{part}			R_{all}
			O	I	G	
Cooking	0.99	0.99	1.00	0.93	0.93	0.71
Blocksworld	0.99	0.94	0.98	0.79	0.89	0.36
Hanoi	1.00	0.58	0.89	0.46	0.33	0.12

color are named by the number in order of increasing width (e.g., `blue_disk1` and `blue_disk2`). The three pegs are named by the number from left to right (e.g., `peg1`, `peg2`, and `peg3`). I and G specify the positions of the disks. Completing this task requires correctly recognizing the disk sizes since L only instructs the rule of the task, "larger disks are below," but mentions no concrete objects.

Each domain has one domain description and ten PDs. Table II shows object types, predicates, and actions in the domain descriptions. Each problem has one linguistic instruction and one scene observation. Fig. 6 shows examples of linguistic instructions L and scene observations S. For the Hanoi domain, L is identical through all problems. This aims to investigate whether ViLaIn can generate different G based on O and I. The descriptions for the cooking domain were created from scratch, while those for the Blocksworld and Hanoi domains were created based on the PDDL files in pddlgym [46]. We confirmed that all the created PDs are syntactically correct and have solutions using Fast Downward [22] and VAL, a plan validation software.[3]

A. Evaluation Metrics

In PD generation, previously proposed metrics roughly calculate the planning success rate or are domain-specific ones [25], [27]. It would be ideal to have metrics that evaluate PDs from multiple perspectives regardless of domain. To this end, we introduce a new suite of metrics: R_{syntax} and R_{plan} for logical correctness and R_{part} and R_{all} for semantic correctness. We describe these metrics below.

R_{syntax}: PDs must be syntactically correct. R_{syntax} calculates the ratio of such PDs. A PD is considered to be syntactically correct if VAL returns no warnings and exit codes for a pair of the domain and the generated PD.

R_{plan}: Even if the PDs are syntactically correct, they might not have valid plans due to incorrect objects in O and incorrect or contradictory propositions in I and G. R_{plan} calculates the ratio of the PDs having valid plans. The plans are obtained using Fast Downward [22]. A plan is considered to be valid if VAL returns no error messages.

R_{part} and R_{all}: The above two metrics ignore whether the PDs are written about our intended tasks. For example, the PD might be about an unintended task while it is syntactically correct and has a valid plan. R_{part} evaluates how close the generated problems are to the ground truth ones. R_{part} calculates the recall of the problem parts between the ground truth and generated ones. R_{part} is independently computed for O, I, and G. The recall of object labels is calculated for O, while the recall of propositions is computed for I and G. Unlike R_{part}, R_{all} calculates the ratio of problems containing all the ground truth object labels and propositions. Thus, R_{all} can be viewed as a harder metric than R_{part}.

VI. EXPERIMENTS

We conduct experiments to investigate how accurately ViLaIn can generate PDs on the ProDG dataset. This section first describes the generation settings of ViLaIn and then discusses experimental results.

A. Generation Settings of ViLaIn

GPT-4 used few-shot prompting with three input/output examples in the same domain as the current task. ViLaIn can refine erroneous PDs by CR n times. PDs with corrected grammatical errors can still have semantic errors, causing no valid solutions. In such cases, CR should be performed at least twice. Thus, we set n to two. For evaluation, we generated ten PDs per problem by varying the example combinations. The resulting 100 problems per domain are used to evaluate ViLaIn.

B. Evaluation of Generation Results by ViLaIn

Table III shows the results. The R_{syntax} scores are more than 99% in all the three domains. This means that ViLaIn can generate syntactically correct PDs for these domains utilizing the three input/output examples. The R_{plan} scores indicate that 94% or more PDs have valid plans in the cooking and Blocksworld domains. However, in the Hanoi domain, the R_{plan} score is only 58% due to its challenging

[3]https://github.com/KCL-Planning/VAL

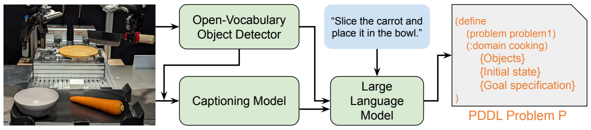

Fig. 7. ViLaIn$_{\text{whole}}$ generates the whole problem description at once.

TABLE IV
GENERATING THE WHOLE PROBLEM DESCRIPTIONS AT ONCE

Domain	R_{syntax}	R_{plan}	R_{all}
Cooking	1.00 (+0.01)	1.00 (+0.01)	0.54 (-0.17)
Blocksworld	0.99 (+0.00)	0.99 (+0.05)	0.13 (-0.23)
Hanoi	1.00 (+0.00)	0.94 (+0.36)	0.21 (+0.09)

setting. We found from the outputs that ViLaIn tends to omit some propositions in this domain, making the PDs invalid.

For R_{part}, the scores on I and G are smaller than those on O. This implies that generating I and G is more challenging than O. We found that mistakenly detected objects cause this. Predicates such as `on` or `at` take two objects as arguments. Propositions created with the predicates and mistakenly detected objects affect other propositions. For example, `(on red_block blue_block)` can be `(on red_block green_block)` `(on green_block blue_block)` with a mistakenly detected `green_block`, making them all incorrect propositions. We consider that generating these incorrect propositions causes such results.

Finally, the R_{all} score is 71% in the cooking domain, 36% in the Blocksworld domain, and 12% in the Hanoi domain. The scores in the cooking and Hanoi domains make sense considering the R_{plan} and R_{part} scores. However, the score is unexpectedly low in the Blocksworld domain. We found that PDs in the Blocksworld domain tend to contain a few incorrect propositions of block relationships. In some cases, the block positioning is mistakenly reversed (e.g., `(on blue_block red_block) (on red_block green_block)` is reversed to `(on green_block red_block) (on red_block blue_block)`). We consider that these lead to the low R_{all} score in this domain.

C. Generating the Whole Problem at Once

ViLaIn generates the parts of PDs using different modules. If a single module can generate the whole problem at once, it greatly simplifies the system. Here, we consider a variant of ViLaIn generating the whole PD at once, as illustrated in Fig. 7. We refer to this model as ViLaIn$_{\text{whole}}$. The generation is performed with few-shot prompting as the original model.

Table IV shows the results with R_{syntax}, R_{plan}, and R_{all}. Values inside parenthesis indicate gains from ViLaIn. In the cooking and Blocksworld domains, ViLaIn$_{\text{whole}}$ slightly improves R_{plan} but worsens R_{all}. This means that using three modules is more effective for these domains. In the Hanoi domain, ViLaIn$_{\text{whole}}$ outperforms ViLaIn in both R_{plan} and R_{all}. When considered with Section VI-B, this means that ViLaIn$_{\text{whole}}$ generates more correct propositions than ViLaIn. Generating the whole PDs makes the distance between tokens of O and I or G closer. We consider that this might work effectively and result in these improvements.

TABLE V
PERFORMANCE WITHOUT CR AND COT

CR configurations		R_{syntax}	R_{plan}	R_{all}
CR (n times)	CoT			
2	✓	0.99	0.99	0.71
1	✓	0.99	0.94	0.68
1		0.97	0.85	0.59
0		0.60	0.18	0.09

D. Generating PDs without CR and CoT

ViLaIn uses corrective re-prompting (CR) and chain-of-thought (CoT) prompting. The CR is performed twice at most as described in Section VI-A. Since all the PDs so far are generated using CR with CoT, the impact of CR on performance is still unknown. Here, we investigate performance without CR and CoT, considering the following configurations: (i) CR with CoT (n = 1 in Section VI-A), (ii) CR without CoT (n = 1), and (iii) without CR (n = 0).

Table V shows the results in the cooking domain. The first line is the same result in Table III. First, performing CR with CoT only once (the first line) slightly drops R_{plan} and R_{all}, meaning that repeating CR is effective. Next, removing CoT (the third line) worsens all the scores. This demonstrates that the introduced intermediate reasoning step by CoT has a large impact on performance. Finally, removing CR (the fourth line) degrades the scores significantly. This model tends to suffer from *hallucinations* [47][4], such as propositions with undefined objects (e.g., `(at cucumber counter)` in I while the cucumber is not defined in O). We found that CR effectively refines these incorrect propositions and makes the PDs consistent.

VII. CONCLUSION

This paper has tackled multimodal planning problem specification, a new task for connecting language-guided planning and symbolic planner. We have proposed Vision-language interpreter (ViLaIn) that generates problem description (PD)s from linguistic instructions and scene observations. A novel dataset called the problem description generation (ProDG) dataset has proposed with new metrics to evaluate ViLaIn. The experimental results show that ViLaIn can generate syntactically correct PDs and more than half of the PDs have valid plans. Interesting future directions include (i) constructing a robotic system with ViLaIn that executes linguistic instructions, (ii) refining PDs via errors from real robots, and (iii) reducing human effort for new tasks.

ACKNOWLEDGMENT

We would like to thank Hirotaka Kameko for his helpful comments. This work was supported by JSPS KAKENHI Grant Number 20H04210 and 21H04910 and JST Moonshot R&D Grant Number JPMJMS2236.

[4]Also referred to as *confabulations*. Generating factually incorrect texts by LLMs is a common problem in natural language processing.

REFERENCES

[1] J. Hatori, Y. Kikuchi, S. Kobayashi, K. Takahashi, Y. Tsuboi, Y. Unno, W. Ko, and J. Tan, "Interactively picking real-world objects with unconstrained spoken language instructions," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3774–3781.

[2] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 25–55, 2020.

[3] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1724–1734.

[6] D. Arumugam, S. Karamcheti, N. Gopalan, L. Wong, and S. Tellex, "Accurately and efficiently interpreting human-robot instructions of varying granularities," in *Proceedings of the 2017 Robotics: Science and Systems (RSS)*, 2017.

[7] C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Fox, "Prospection: Interpretable plans from language by predicting the future," in *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6942–6948.

[8] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[9] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[10] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, *et al.*, "PaLM 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[11] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of the 2022 International Conference on Machine Learning (ICML), vol. 162, 2022, pp. 9118–9147.

[12] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex, "Planning with large language models via corrective re-prompting," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[13] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2Motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023.

[14] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530.

[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the 2020 Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[16] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An approach to evaluating interpretability of machine learning," *arXiv preprint arXiv:1806.00069*, p. 118, 2018.

[17] E. Karpas and D. Magazzeni, "Automated planning for robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 417–439, 2020.

[18] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone, *An introduction to the planning domain definition language*. Springer, 2019, vol. 13.

[19] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[20] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[21] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proceedings of the 2023 International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 202, 2023, pp. 19 730–19 742.

[22] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[23] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[24] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[25] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "LLM+P: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[26] P. Sharma, A. Torralba, and J. Andreas, "Skill induction and planning with latent language," in *Proceedings of the 2022 Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2022, pp. 1713–1726.

[27] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating natural language to planning goals with large-language models," *arXiv preprint arXiv:2302.05128*, 2023.

[28] B. Bonet and H. Geffner, "Planning as heuristic search," *Artificial Intelligence*, vol. 129, no. 1, pp. 5–33, 2001.

[29] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," *Proceedings of the 2020 International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 30, no. 1, pp. 440–448, 2020.

[30] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3801–3808.

[31] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 866–894, 2021.

[32] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," in *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3182–3189.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proceedings of the 2015 Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang, "Open-vocabulary object detection using captions," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 14 393–14 402.

[36] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in *Proceedings of the 2016 European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 852–869.

[37] S. Inayoshi, K. Otani, A. Tejero-de Pablos, and T. Harada, "Bounding-box channels for visual relationship detection," in *Proceedings of the 2020 European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 682–697.

[38] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the 2017 IEEE

Conference on Computer Vision and Pattern Recognition (CVPR)), 2017, pp. 5410–5419.
[39] J. Yang, Y. Z. Ang, Z. Guo, K. Zhou, W. Zhang, and Z. Liu, "Panoptic scene graph generation," in *Proceedings of the 2022 European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 178–196.
[40] T. Migimatsu and J. Bohg, "Grounding predicates through actions," in *Proceedings of the 2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3498–3504.
[41] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 2022 Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837.
[42] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Proceedings of the 2022 Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35. Curran Associates, Inc., 2022, pp. 22 199–22 213.
[43] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, *et al.*, "Least-to-most prompting enables complex reasoning in large language models," in *Proceedings of the 2023 International Conference on Learning Representations (ICLR)*, 2023.
[44] N. Gupta and D. S. Nau, "On the complexity of blocks-world planning," *Artificial intelligence*, vol. 56, no. 2-3, pp. 223–254, 1992.
[45] R. Alford, U. Kuter, and D. S. Nau, "Translating HTNs to PDDL: A small amount of domain knowledge can go a long way." in *Proceedings of the 2009 International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 9, 2009, pp. 1629–1634.
[46] T. Silver and R. Chitnis, "PDDLGym: Gym environments from PDDL problems," in *Proceedings of the 2020 International Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop*, 2020.
[47] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," in *Proceedings of the 2020 Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2020, pp. 1906–1919.