2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

ISR-LLM: Iterative Self-Refined Large Language Model for Long-Horizon Sequential Task Planning

Zhehua Zhou[1], Jiayang Song[1], Kunpeng Yao[2], Zhan Shu[1] and Lei Ma[3,1]

***Abstract*— Motivated by the substantial achievements of Large Language Models (LLMs) in the field of natural language processing, recent research has commenced investigations into the application of LLMs for complex, long-horizon sequential task planning challenges in robotics. LLMs are advantageous in offering the potential to enhance the generalizability as task-agnostic planners and facilitate flexible interaction between human instructors and planning systems. However, task plans generated by LLMs often lack feasibility and correctness. To address this challenge, we introduce ISR-LLM, a novel framework that improves LLM-based planning through an iterative self-refinement process. The framework operates through three sequential steps: *preprocessing*, *planning*, and *iterative self-refinement*. During preprocessing, an LLM translator is employed to convert natural language input into a Planning Domain Definition Language (PDDL) formulation. In the planning phase, an LLM planner formulates an initial plan, which is then assessed and refined in the iterative self-refinement step by a validator. We examine the performance of ISR-LLM across three distinct planning domains. Our experimental results show that ISR-LLM is able to achieve markedly higher success rates in sequential task planning compared to state-of-the-art LLM-based planners. Moreover, it also preserves the broad applicability and generalizability of working with natural language instructions.**

I. Introduction

Large Language Models (LLMs) have recently revolutionized artificial intelligence by demonstrating unprecedented abilities in areas such as natural language processing [1], data analysis [2], code generation [3] and reasoning [4]. Due to their rich internalized knowledge about the world [5], [6], LLMs have also garnered considerable attention within the field of long-horizon sequential task planning [7]. Unlike short-term robotic planning problems, *long-horizon sequential task planning* often involves devising interconnected actions that are spanned over extended timeframes to achieve control objectives. Since the execution of actions at one point in time can greatly impact subsequent actions and outcomes, long-horizon planning is usually considered a more challenging problem due to its inherent intricacy in managing temporal dependencies and combinatorial complexity [8].

The traditional way to address long-horizon sequential task planning typically relies on first establishing a symbolic and logic-based representation of the planning problem [9] and then employing search-based techniques [10], [11] to find a feasible solution. However, this method usually requires the manual specification of symbolic planning domains, which demands a notable degree of expertise in the field. Furthermore, many desirable properties of plans, e.g., user preferences, which can be specified in natural language by individuals without specialized training, may prove intricate or even infeasible to encapsulate within formal logic frameworks. As a result, the adaptability of conventional methods is constrained, limiting their utility in diverse contexts.

To overcome this limitation, recent studies have started exploring the potential of utilizing LLMs as task-agnostic reasoning modules, with the aim of facilitating more generalized and intelligent robotic planning [12], [13]. Leveraging their pre-trained knowledge, these LLM-based planners are able to effectively comprehend both explicit human-generated natural language directives and the inherent constraints interwoven within planning tasks [14]. This greatly reduces the necessity for labor-intensive manual rule encoding and circumvents the need for intricate specification of symbolic planning domains [15]. However, as LLMs are essentially engineered to generate word sequences that align with human-like context, their efficacy and reliability in planning are often not guaranteed [16]. This limitation becomes further pronounced in long-horizon sequential task planning, where complex action dependencies and extended temporal considerations introduce additional difficulties that challenge the planning abilities of LLMs.

Drawing inspiration from recent research that reveals the potential for LLM improvements through self-refinement [17], [18], we propose in this work the Iterative Self-Refined LLM (ISR-LLM) framework that utilizes the power of iterative self-refinement to improve planning outcomes of LLMs. Our framework consists of three steps (see Fig. 1): (1) *Preprocessing*, where an LLM translator is employed to translate the natural language inputs into their respective Planning Domain Definition Language (PDDL) [9] formulations; (2) *Planning*, where an LLM planner takes the translated PDDL problem as input and determines the action sequence to accomplish the long-horizon sequential task planning; (3) *Iterative self-refinement*, where a validator is used to examine the correctness of the generated action plan and provide feedback to the LLM planner. Then, based on the feedback, the LLM planner performs the iterative self-refinement process to find a revised action plan. We consider two different types of validators in our approach: an LLM-based self-validator and an external validator that leverages auxiliary verification tools.

[1]Zhehua Zhou, Jiayang Song and Zhan Shu are with the University of Alberta, Canada. Emails: {zhehua1, jiayan13, zshu1}@ualberta.ca

[2]Kunpeng Yao is with the Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. Email: kunpeng.yao@epfl.ch

[3]Lei Ma is with The University of Tokyo, Japan, and the University of Alberta, Canada. Email: ma.lei@acm.org

The code related to this work is available at `https://github.com/ma-labo/ISR-LLM`.

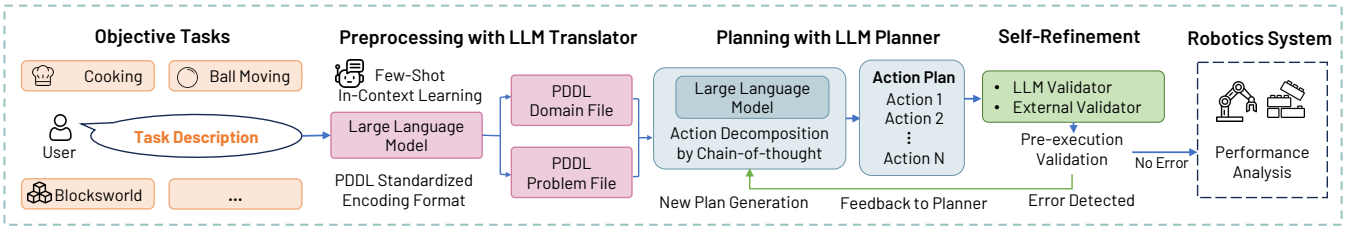


Fig. 1: Our proposed ISR-LLM framework. It consists of three steps: *preprocessing*, *planning*, and *iterative self-refinement*.

The contributions of this work are threefold:

- We present ISR-LLM, a novel framework achieved by integrating a self-refinement mechanism into LLM. This approach addresses long-horizon sequential task planning and offers remarkable advancements in both feasibility and correctness.
- We introduce and evaluate the effectiveness of two types of validators in providing feedback to the LLM planner for executing the iterative self-refinement process.
- We highlight the superiority of our proposed framework in comparison to state-of-the-art methods through comprehensive experiments across three diverse planning domains.

II. Related Work

Long-Horizon Sequential Task Planning In recent robotic studies, PDDL and Answer Set Programming (ASP) [20] are often used as the language for representing long-horizon sequential task planning problems [21]. A prevalent method employed to tackle these planning tasks is to employ a search-based or sampling-based algorithm to find a viable plan [22], [23], [24]. This strategy has been successfully applied across diverse research domains in robotics, e.g., mobile robots [25], autonomous vehicles [26], and robotic manipulators [27]. However, these approaches rely on a predetermined symbolic and logical representation of the planning domain, which usually demands a high level of expert knowledge for formulation. Moreover, due to the inherent abundance of potential action options associated with long-horizon sequential task planning, search-based or sampling-based strategies may encounter impediments in such scenarios.

Task and Motion Planning (TAMP) Another important research problem in robotics is TAMP [28], which combines high-level task planning and low-level robot motion planning as a hierarchical planning framework. The focus of TAMP extends beyond mere task planning to encompass the executability of the determined actions, i.e., the actions must be executable by the robot with a viable motion trajectory that is subject to both robotic and environmental constraints [29], [30], [31]. However, how to accurately ground actions generated by LLMs into feasible robot motions remains a challenging and ongoing area of research [12], [13]. In this work, we mainly focus on exploring the task planning capabilities of LLMs, and reserve the consideration of addressing TAMP problems for future endeavors.

Action Grounding Recent studies have started utilizing LLMs as task-agnostic planners [32], [33], [34], [35]. A multitude of studies have delved into grounding the language commands generated by LLMs to executable robotic actions [12], [13], [36], [15]. For instance, in [12], scores are assigned to potential actions through a value function, and the action with the highest likelihood of success is selected. Although the focus of this work is not the grounding of actions, these studies illustrate the competencies of LLMs in addressing diverse robotic planning tasks.

LLM with PDDL Moreover, LLMs are often combined with PDDL to elevate the performance of LLM-based planners. In [16], a Blocksworld [37] benchmark is proposed to assess the LLM's capability in handling natural language inputs for planning. However, the results reveal a discouraging performance of LLMs in long-horizon task planning. In [38], [39], instead of natural language inputs, planning problems in PDDL syntax are directly presented to LLMs for generating action sequences. While this strategy contributes to enhanced performance, it often demands additional effort and expert knowledge for composing the corresponding PDDL files. In [40], LLM is employed not as a planner but rather as a translator that converts natural language inputs into PDDL problems, which are subsequently solved using classical planners. Similar to our self-refinement concept, [41] collects the error information returned from the execution of the plan for directing the LLM toward correcting erroneous actions. However, such a refinement process occurs subsequent to the action execution phase. Conversely, our approach not only considers the utilization of an external validator to perform a similar self-refinement process but also investigates the potential of LLMs for enabling pre-execution action corrections through self-validation capabilities.

III. Preliminary

Task Planning In this work, we consider the task planning problem $P = \langle S, A, T, s_{\text{init}}, G \rangle$ in a setting with discrete and fully observable states, finite actions, and deterministic transitions. For each state $s \in S$, an action $a \in A$ can be selected from the set of applicable actions $A(s) \subseteq A$ whose preconditions are fulfilled. The transition function $T : S \times A \to S$ determines the next state. $s_{\text{init}} \in S$ represents the initial state and $G \subseteq S$ is a set of goal states. A solution to the planning problem is a sequential action plan $\pi = (a_1, a_2, \ldots, a_n)$ that controls the initial state s_{init} to a goal state. For long-horizon sequential task planning, the number of actions n tends to be relatively large. In this work, our primary focus is the capabilities of LLM in solving the designated task planning problem, i.e., the feasibility and success rate of planning rather than the optimality.

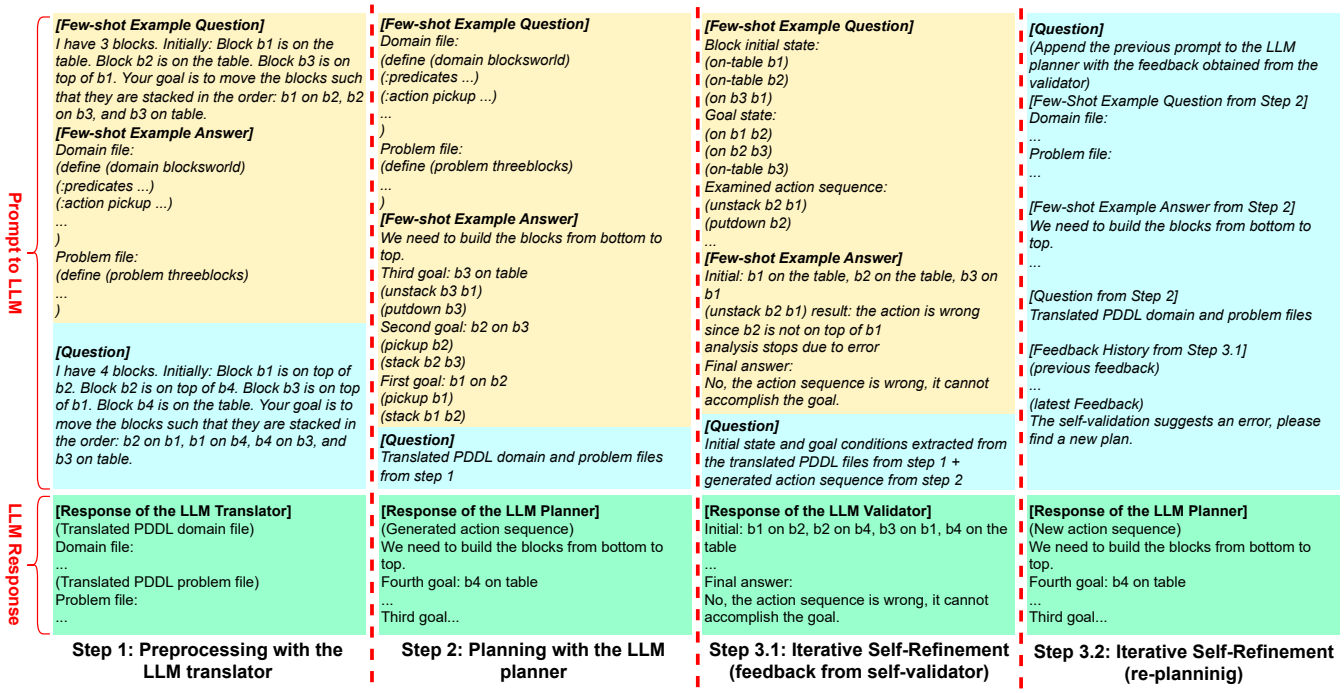


Fig. 2: Examples of the prompts used in ISR-LLM. The prompt provided to the LLM contains two parts: the few-shot examples (yellow) and the actual question (blue). The texts shaded with a green color represent the LLM's responses. Details about all few-shot examples used in this work are given in the preprint version of this paper [19].

PDDL PDDL is a standardized encoding format designed for classical planning problems [42]. A planning problem represented in PDDL syntax consists of two files: a domain file and a problem file. The domain file embodies the foundational rules of the planning domain, i.e., the predicates that represent the state space and the preconditions and effects of all possible actions. The problem file defines the available objects within the planning domain, as well as the initial state and goal conditions. We assume that the natural language input provided to the LLM should include both the initial state and the goal conditions, such that the LLM translator is able to convert it into corresponding PDDL files. For more details about PDDL, the readers can refer to [9].

IV. ISR-LLM

In this section, we introduce the ISR-LLM framework (see Fig. 1), which includes three steps: preprocessing, planning, and iterative self-refinement.

A. Preprocessing with LLM Translator

The LLM translator first converts the given natural language instructions into a PDDL formulation with domain and problem files. The rationale for employing such a translator is grounded in its notable advantages, even though an LLM planner could be designed to operate directly on natural language inputs, as demonstrated in [15]. The adoption of a formal representation, i.e., PDDL, offers twofold benefits to the subsequent validation process of the generated plan. Firstly, it enables the usage of existing PDDL validators, e.g., VAL [43] or PDDL.lj [44], as the external validator. This obviates the necessity of developing a custom validator and thereby saves substantial time and effort. Secondly, rather than relying solely on language cues, this approach enables the LLM-based self-validator to acquire a comprehension akin to a state-machine understanding of the system state. This, in turn, facilitates a more precise evaluation of the correctness of the selected actions.

In order to ensure the structural accuracy of the translated PDDL files, we adopt a technique known as few-shot in-context learning [1]. This technique involves embedding illustrative examples within the prompt, effectively instructing the LLM on how to formulate responses to given queries in a desired manner. Similar to [40], we assume that the domain-specific knowledge pertinent to each considered planning task is available in advance and thus include it within the few-shot examples provided to the LLM translator. An example of the prompt presented to the LLM translator for the Blocksworld planning domain (see Sec. V-A for a detailed explanation about this domain) is shown in Fig. 2.

B. Planning with LLM Planner

Once the natural language input is translated, the LLM planner takes these PDDL files as inputs and determines an action sequence aimed at achieving the given task (see Fig. 1). In addition to few-shot in-context learning, we also integrate the Chain-of-Thought (CoT) technique [45] into the prompts provided to the LLM planner. CoT operates by decomposing the overall problem into intermediate steps, thus enabling the LLM to tackle complex reasoning problems that may not be solvable via standard prompting methods. An illustrative prompt example for the LLM planner is given in Fig. 2.

C. Iterative Self-Refinement Loop with Validator

With the initial action plan obtained from the previous step, we perform an iterative self-refinement process to examine and improve its feasibility and correctness. The central component of the iterative self-refinement loop is the validator, as demonstrated in Fig. 1. Through the examination of the generated action sequence, the validator constructs feedback, pinpointing any actions considered incorrect, and subsequently conveys this information to the LLM planner. Then, based on the feedback, the LLM planner initiates a self-refinement process to rectify the incorrect action and devise a new action plan. Note that while the generated action sequence may contain multiple errors, analyzing actions subsequent to the initial error is often unnecessary since the first error could potentially render the foundation of all ensuing actions fundamentally flawed. Thus, the self-refinement process is executed iteratively within a loop, where in each step, the validator stops at the first identified error. The information concerning this error is then returned, ensuring that each iterative stage is solely focused on rectifying this detected mistake. The iterative self-refinement loop persists until either the validator identifies no errors or a predefined maximum number of iterations is reached. The action sequence, resulting from the iterative self-refinement loop, is then accepted as the final generated action sequence. An example of the prompt provided to the LLM-based self-validator is shown in Fig. 2, where few-shot learning and CoT techniques are also employed.

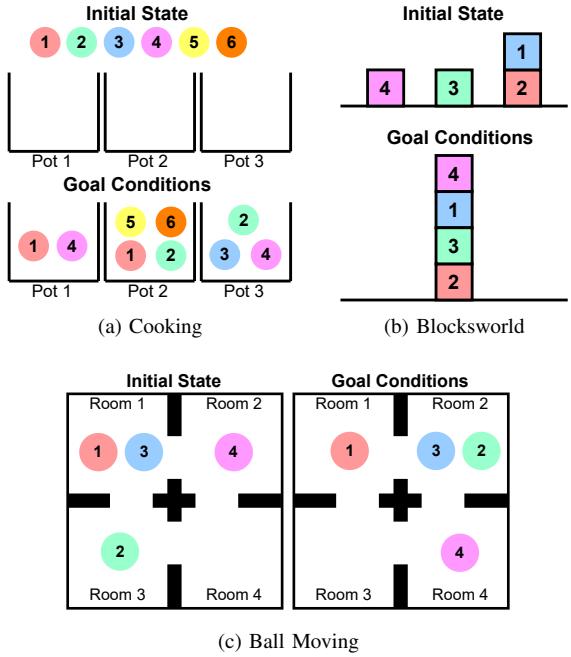
We consider two types of validators: a self-validator, which employs the LLM to assess the correctness of the generated action plan, and an external validator, which leverages external tools for performing the analysis. It is worth mentioning that although the external validator is capable of providing accurate feedback on the feasibility of the generated plan, its implementation often demands a considerable amount of effort and may be unavailable for certain tasks. Conversely, the usage of an LLM as an internal self-validator economizes both time and effort. However, it has the inherent risk of possibly yielding imprecise or even erroneous feedback. The selection of the validator type, therefore, hinges upon the specific evaluation requirements and the context of the validation scenario.

V. Experimental Results

A. Experimental Setup

We utilize the following three planning domains as benchmark problems to evaluate the performance of ISR-LLM. These domains are derived from existing literature and are extensively employed in planning research [40], [39], [16]. More details and examples about each planning domain are presented in the preprint version of this paper [19].

- *Cooking:* There are n pots and a total of 6 different ingredients (see Fig. 3a). The robot's task is to add ingredients to each pot according to a prescribed recipe. Each pot possesses its own randomly generated recipe that includes 2 to 4 different ingredients. However, each ingredient may only be retrieved once by the robot, i.e., once the robot has picked up an ingredient, it must distribute it to all pots that require this ingredient as per their individual recipes.
- *Blocksworld:* There are n blocks, initially randomly placed on a table. The objective of the robot is to assemble these blocks into a stack, adhering to a specific prescribed order (see Fig. 3b). However, the robot can only manipulate one block at a time, i.e., any block that has other blocks situated on top of it is considered fixed.
- *Ball Moving:* There are n balls, initially randomly distributed among 4 rooms (see Fig. 3c). The robot needs to relocate the balls to their predefined goal rooms, with the constraint that it can hold no more than one ball at a time.



(a) Cooking (b) Blocksworld

(c) Ball Moving

Fig. 3: Three planning domains used in this work.

For all three planning domains, we investigate two specific cases with $n = 3$ and $n = 4$, to examine the influence of the number of objects, which is directly correlated with the complexity of the task, on the performance of the proposed ISR-LLM framework. Furthermore, to evaluate the impacts of various LLMs on the planning outcomes, we employ two LLMs, namely GPT3.5 and GPT4, and compare their capabilities in task planning within the ISR-LLM framework.

For each planning task, we evaluate three different methods: (1) *LLM-direct*, which is the baseline approach grounded in [39], [38], [16]. It leverages the LLM to formulate an action plan directly from the given PDDL input. To ensure a fair comparison with ISR-LLM, we utilize the LLM translator to convert natural language inputs into PDDL files in this method. (2) *ISR-LLM-self*, which employs the ISR-LLM framework with an LLM-based self-validator; (3) *ISR-LLM-external*, which incorporates an external validator to generate feedback for ISR-LLM. In order to mitigate the

TABLE I: Success rate of ISR-LLM in different planning domains.

Planning Domain		GPT3.5			GPT4		
		LLM-direct	ISR-LLM-self	ISR-LLM-external	LLM-direct	ISR-LLM-self	ISR-LLM-external
Cooking	$n=3$	47%	67%	100%	100%	100%	100%
	$n=4$	40%	53%	63%	100%	100%	100%
Blocksworld	$n=3$	20%	37%	70%	43%	60%	97%
	$n=4$	10%	17%	53%	40%	60%	80%
Ball Moving	$n=3$	33%	50%	70%	93%	100%	100%
	$n=4$	17%	27%	57%	90%	93%	97%

influence of existing PDDL validators and focus on analyzing the performance of ISR-LLM, we implement our own custom external validators in this work. It evaluates whether the preconditions for each action are satisfied and provides feedback on any errors. More implementation details are available on the Github page of this paper.

We randomly generate 30 unique cases with varying initial states and goal conditions for each planning task. The maximum number of iterations allowed for self-refinement is set at 10. The success rates of task accomplishments for the three aforementioned methods are recorded. All experiments are conducted on a laptop equipped with an Intel® Core™ i7-10870H CPU.

B. Performance of ISR-LLM

The results of the experiments are summarized in Table I. In the cases utilizing GPT3.5, the proposed ISR-LLM framework demonstrates a notable enhancement in success rates across all planning domains when compared to the baseline approach. While the LLM-based self-validator contributes to an approximate 15% increase in performance, the external validator can further amplify the success rate by roughly 40% to 50%.

The success rates are also influenced by task complexity, as indicated by the number of objects. Increases in object numbers correspond to decreased success rates in the Cooking, Blocksworld, and Ball Moving domains for all three approaches. This trend reflects the increased difficulty in rectifying erroneous actions as the planning horizon extends. Moreover, the success rate varies among planning domains. Compared to the Cooking and Ball Moving domains, the Blocksworld domain, which demands more sophisticated logical thinking, demonstrates lower success rates. Nevertheless, the proposed ISR-LLM is still able to improve the planning outcomes within this domain.

It can also be observed that GPT4 greatly outperforms GPT3.5, corroborating the common assertion that GPT4 possesses a markedly superior reasoning capability. The baseline method, i.e., LLM-direct, when coupled with GPT4, is able to achieve a success rate exceeding 90% in the Cooking and the Ball Moving domains, where ISR-LLM also maintains this high-performance level. However, in the more logically complex Blocksworld domain, GPT4 demonstrates diminished performance using the baseline approach. Nevertheless, the employment of ISR-LLM also elevates the success rate for this domain, with the self-validator contributing an increase of about 20%, and the external validator enhancing it by more than 40%. Interestingly, the influence of the number of objects appears to be less pronounced when GPT4 is utilized. This may be attributed to GPT4's enhanced reasoning capabilities, which facilitate more effective logical thinking and thereby mitigate the impact of the number of objects on the results.

TABLE II: Success rate of ISR-LLM with and without the LLM translator in Blocksworld ($n=3$) with GPT3.5.

Method	With LLM Translator	Without LLM Translator
LLM-direct	20%	13%
ISR-LLM-self	36%	16%
ISR-LLM-external	70%	63%

C. Influence of the LLM Translator

We also evaluate the influence of the LLM translator using the Blocksworld domain with $n=3$ and GPT3.5 as an example, as this case demonstrates where the efficacy of ISR-LLM is most obvious. By omitting the LLM translator and directly utilizing natural language input, we compare the success rates of task planning and present the results in Table II. It can be observed that while the LLM translator slightly improves the planning performance of the baseline approach, the self-validator greatly benefits from the translator, showing a 20% increase in the success rate. The reason could be that the translated PDDL files offer a symbolic and logical representation of the planning domain, thereby allowing the LLM to form a more concrete understanding of the system state, as opposed to relying solely on linguistic cues. In contrast, the performance of the external validator remains relatively consistent, irrespective of the presence of the LLM translator. This consistency arises from our custom validator's ability to provide accurate feedback, whether PDDL formulations are employed or not. However, as previously mentioned, introducing translated PDDL files enables the usage of existing PDDL validators, potentially saving substantial time and effort needed for implementing a custom validator.

D. Grounding the Actions

Although it is beyond the scope of this work, we further demonstrate that the generated action plan can be directly grounded into feasible robot actions when paired with a suitable motion planner. This highlights another advantage of

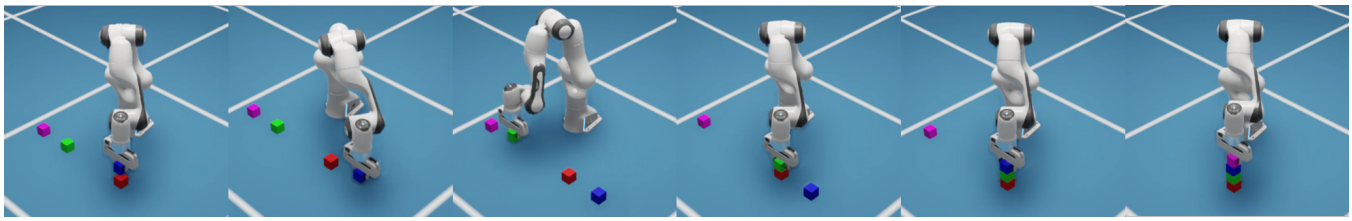


Fig. 4: Grounding of actions in the Blocksworld domain with four blocks. Initially, block b2 (red), b3 (green), b4 (pink) are on the table, and block b1 (blue) is on top of block b2. The goal is to stack the blocks in the given order: b4 on b1, b1 on b3, b3 on b2, and b2 on the table. The detailed execution of actions can be viewed in the supplementary video. The simulation is conducted in NVIDIA Omniverse Isaac Sim [46].

employing the LLM translator within the ISR-LLM framework, as the use of PDDL formulation ensures that each generated action conforms to a predefined definition and structure. Consequently, this simplifies the task of the motion planner in converting the action plan into executable robot movements. Figure 4 illustrates this grounding process, using an example from the Blocksworld domain with four blocks (see also the supplementary video). Here, a pick-and-place controller is employed to execute the four different types of actions, assuming the robot knows the locations of the blocks.

VI. Discussion

Self-Validator and External Validator Generally, the external validator is capable of providing feedback to a degree of precision that identifies the exact action in which an error resides. Conversely, the self-validator usually only provides an overarching estimation regarding the correctness of the entire generated action plan. As a consequence, the external validator often leads to superior performance, as precise feedback can greatly facilitate the correction of erroneous actions. However, this does not guarantee that the LLM can fully comprehend this feedback and rectify the errors, resulting in persistent issues despite the accurate external validator's inputs.

Planning Domains The planning capabilities of LLMs are influenced by the inherent characteristics of the planning domains. As observed from our experimental results, LLMs appear to excel in planning tasks that focus on adhering to specific instructions, such as Cooking, or performing repeated actions with identifiable patterns, e.g., Ball Moving. Conversely, when the planning tasks demand more complex logical thinking, as seen in the Blocksworld domain, their performance tends to diminish. The reason could be that LLMs are essentially trained to generate word sequences that mirror human-like thought processes, which suits tasks requiring instruction or pattern following. However, when critical logical reasoning becomes a vital component of the task, the inherent reasoning abilities of the LLMs become more important.

Limitations One general limitation of the state-of-the-art LLM-based planners - also in our proposed ISR-LLM framework - is that the overall success rate often fails to exceed that of traditional search-based planners. However, as an initial exploratory work, we demonstrate the potential of utilizing LLM as a versatile and task-agnostic planner. This has the possibility to significantly facilitate the deployment of various robotic systems across diverse scenarios and minimize the required effort in planning system design. Moreover, the planning abilities of the ISR-LLM framework may see substantial improvements through refinements in the underlying reasoning capabilities of the LLMs. Another limitation stems from the inherent randomness within LLMs, complicating assurances such as correctness or constraint satisfaction in the generated action plan. While LLMs offer promising capabilities, their application in safety-critical tasks requires careful consideration to ensure reliability.

VII. Conclusion

In this paper, we explore the potential of leveraging LLMs for long-horizon sequential task planning based on natural language input. To improve the correctness of the generated action plan, we introduce the ISR-LLM framework, which employs an iterative self-refinement approach for automatic plan revisions. This framework consists of three steps. First, an LLM translator converts the natural language input into a PDDL formulation, represented by PDDL files. Second, using these translated PDDL files, an LLM planner formulates an initial action plan. Third, an iterative self-refinement loop is initiated, wherein either an LLM-based self-validator or an external validator provides feedback on the correctness of the action plan, allowing the LLM planner to make necessary revisions to the action plan. Through extensive experiments across three diverse planning domains, we demonstrate that ISR-LLM surpasses the performance of existing state-of-the-art LLM-based planners in long-horizon sequential task planning. While maintaining the flexibility and generalizability to work with natural language input, our ISR-LLM framework consistently achieves high success rates in task accomplishments. For future work, we plan to incorporate motion planning within the current ISR-LLM framework, aiming to facilitate reliable and efficient task and motion planning across various robotic application scenarios.

Acknowledgement

This work was supported in part by the Canada First Research Excellence Fund as part of the University of Alberta's Future Energy Systems research initiative, Amii RAP Grant, Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, and No.DGECR-2021-00019), as well as JST-Mirai Program Grant No.JPMJMI20B8, JSPS KAKENHI Grant No.JP21H04877, No.JP23H03372.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] M. Agrawal, S. Hegselmann, H. Lang, Y. Kim, and D. Sontag, "Large language models are zero-shot clinical information extractors," *arXiv preprint arXiv:2205.12689*, 2022.

[3] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Chi Conference on Human Factors in Computing Systems Extended Abstracts*, 2022, pp. 1–7.

[4] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, "Star: Bootstrapping reasoning with reasoning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 476–15 488, 2022.

[5] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *arXiv preprint arXiv:1909.01066*, 2019.

[6] J. Davison, J. Feldman, and A. M. Rush, "Commonsense knowledge mining from pretrained models," in *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, 2019, pp. 1173–1178.

[7] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[8] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 239–252, 2022.

[9] 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.

[10] W. Zhang, *State-space search: Algorithms, complexity, extensions, and applications*. Springer Science & Business Media, 1999.

[11] S. Edelkamp and S. Schrödl, *Heuristic search: theory and applications*. Elsevier, 2011.

[12] 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.

[13] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[14] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022.

[15] 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.

[16] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, "Large language models still can't plan (a benchmark for llms on planning and reasoning about change)," *arXiv preprint arXiv:2206.10498*, 2022.

[17] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang *et al.*, "Self-refine: Iterative refinement with self-feedback," *arXiv preprint arXiv:2303.17651*, 2023.

[18] J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han, "Large language models can self-improve," *arXiv preprint arXiv:2210.11610*, 2022.

[19] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, "Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning," *arXiv preprint arXiv:2308.13724*, 2023.

[20] G. Brewka, T. Eiter, and M. Truszczyński, "Answer set programming at a glance," *Communications of the ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[21] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl-and asp-based systems," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, pp. 363–373, 2019.

[22] J. Levine and D. Humphreys, "Learning action strategies for planning domains using genetic programming," in *Workshops on Applications of Evolutionary Computation*. Springer, 2003, pp. 684–695.

[23] J. Segovia-Aguas, S. Jiménez, and A. Jonsson, "Generalized planning as heuristic search," in *International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 569–577.

[24] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *International Conference on Robotics and Automation*. IEEE, 2010, pp. 2902–2908.

[25] S. Zhang, F. Yang, P. Khandelwal, and P. Stone, "Mobile robot planning using action language with an abstraction hierarchy," in *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, 2015, pp. 502–516.

[26] Y. Ding, X. Zhang, X. Zhan, and S. Zhang, "Task-motion planning for safe and efficient urban driving," in *International Conference on Intelligent Robots and Systems*. IEEE, 2020, pp. 2119–2125.

[27] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[28] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 265–293, 2021.

[29] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *International Joint Conference on Artificial Intelligence*, 2015, pp. 1930–1936.

[30] D. Driess, O. Oguz, and M. Toussaint, "Hierarchical task and motion planning using logic-geometric programming (hlgp)," in *RSS Workshop on Robust Task and Motion Planning*, 2019.

[31] X. Zhang, Y. Zhu, Y. Ding, Y. Jiang, Y. Zhu, P. Stone, and S. Zhang, "Symbolic state space optimization for long horizon mobile manipulation planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 866–872.

[32] P. Sharma, A. Torralba, and J. Andreas, "Skill induction and planning with latent language," *arXiv preprint arXiv:2110.01517*, 2021.

[33] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar *et al.*, "Pre-trained language models for interactive decision-making," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 199–31 212, 2022.

[34] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

[35] 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 *International Conference on Robotics and Automation*. IEEE, 2023, pp. 11 523–11 530.

[36] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," *arXiv preprint arXiv:2303.06247*, 2023.

[37] J. Slaney and S. Thiébaux, "Blocks world revisited," *Artificial Intelligence*, vol. 125, no. 1-2, pp. 119–153, 2001.

[38] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling, "Pddl planning with pretrained large language models," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[39] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," *arXiv preprint arXiv:2305.11014*, 2023.

[40] 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.

[41] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex, "Planning with large language models via corrective re-prompting," *arXiv preprint arXiv:2211.09935*, 2022.

[42] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, "Pddl— the planning domain definition language," *Technical Report*, 1998.

[43] R. Howey, D. Long, and M. Fox, "Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl," in *International Conference on Tools with Artificial Intelligence*. IEEE, 2004, pp. 294–301.

[44] T. Zhi-Xuan, "Pddl. jl: An extensible interpreter and compiler interface for fast and flexible ai planning," Ph.D. dissertation, Massachusetts Institute of Technology, 2022.

[45] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.
[46] NVIDIA, "Nvidia isaac sim," 2021. [Online]. Available: https://developer.nvidia.com/isaac-sim