Boosting Robot Intelligence in Practice: Enhancing Robot Task Planning with Large Language Models

Yisheng Zhang
School of Mechanical Engineering
Shanghai Jiaotong University
Shanghai, China
zys_edward@sjtu.edu.cn

Zhigang Wang
Intel Labs China
Beijing, China
zhi.gang.wang@intel.com

Shengmin Zhang
School of Mechanical Engineering
Shanghai Jiaotong University
Shanghai, China
zhangshengmin@sjtu.edu.cn

Yanlong Peng
School of Mechanical Engineering
Shanghai Jiaotong University
Shanghai, China
me-pengyanlong@sjtu.edu.cn

Ming Chen
School of Mechanical Engineering
Shanghai Jiaotong University
Shanghai, China
mingchen@sjtu.edu.cn

2023 8th International Conference on Robotics and Automation Engineering (ICRAE) | 979-8-3503-2765-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRAE59816.2023.10458574

***Abstract*—Task planning capabilities are crucial for intelligent robots to operate autonomously in the physical world. However, traditional Planning Domain Definition Language (PDDL) based methods often suffer from combinatorial explosion and unsatisfactory planning time. In this paper, we propose enhancing robot task planning with large language models (LLMs) in an innovative way - using LLMs to guide the search process of PDDL planners rather than replacing PDDL planning completely. The LLMs guide the search process of PDDL planners with learned heuristics and provide constraint reasoning to reduce the search space. To address potential pitfalls of LLMs, a verification mechanism is added at the execution stage to validate plan correctness. We evaluated our method on a real scenario, end-of-life vehicle battery disassembly. Experimental results demonstrate that incorporating LLMs into the planning pipeline can significantly improve planning efficiency and scalability while maintaining plan validity. This research provides a promising direction towards integrating language models with classical approaches to boost robot intelligence for practical applications. The proposed framework makes a solid step forward in enhancing the task planning capability of future intelligent robotic systems.**

***Keywords*—robot, task planning, LLM**

I. Introduction

Recently, the revolutionary progress in the field of natural language processing (NLP) dominated by large language models (LLM) has attracted a lot of attention. Many attempts have been made to advanced models to solve general tasks in different fields, which to a certain extent demonstrates the advantages of the increased model scale. However, when dealing with some reasoning problems involving arithmetic or symbolic reasoning, the result is often unsatisfactory if the problem is directly used as input [1].

Studies have confirmed that prompting with expected input-output paradigms enables the model to perform in-context few-shot learning, thereby improving its performance on various tasks [2]. In extensive practice, it has been found that when LLMs are asked to reason in steps, they tend to produce more robust solutions, especially for the aforementioned problems. On this basis, chain-of-thought (CoT) prompting is proposed by Wei et al. (2023) [3] to induce the model to make step-by-step reasoning by inserting intermediate reasoning steps between input-output paradigms. Tree-of-thought (ToT) prompting is improved on CoT by Yao et al. (2023) [4], generating tree-structured thought steps and introducing a self-evaluation mechanism. Through few-shot learning, the intermediate reasoning steps are quantitatively evaluated, and subsequently the most reliable chain of thought is obtained with a search algorithm.

There have been some works attempting to apply LLMs to robot perception, control, and task and motion planning. Defining a set of high-level robot APIs or function library, Vemprala et al. (2023) succeeded in performing robotic manipulation tasks in a self-programmed manner with the help of ChatGPT [5]. Other approaches implement task and motion planning by developing customized LLMs for robotic manipulation, which take multi-modal prompts as input to enable robotic agents with embodied intelligence to understand instructions and perform corresponding tasks [6]–[8]. These models seem to have good generalization ability, but they still cannot overcome the LLM hallucination, and the probability of failure cannot meet actual needs, especially in industrial scenarios.

In our previous works [9], we have proposed an architecture of Neuro-symbolic task and motion planning (TAMP) to tackle

This work was supported by the Ministry of Industry and Information Technology of China for financing this research within the program "2021 High Quality Development Project (TC210H02C)"

the uncertainty issues in the human-robot hybrid disassembly pipeline under the unstructured conditions, aiming to replace workers in repetitive tasks with low-complexity, such as screw removal. Planning Domain Definition Language (PDDL) [10]–[12] is used to formally define the disassembly tasks and primitives for Neuro-symbolic TAMP. To represent system states in detail, neural predicates are introduced to map sensor data to quasi-symbolic states using neural networks. In the planner, the set of neural predicates is represented as a state node, and an action plan is obtained through logical search algorithms such as breadth-first search (BFS).

However, such traditional algorithms often lead to problems such as combinatorial explosion and unsatisfactory planning time. In order to eliminate these shortcomings, this paper made attempts to guide the search process of PDDL planners via LLMs. The challenge is how to make the model understand PDDL and induce it to generate feasible plans while maintaining a limited search space (especially when a large number of predicates and primitives involved). In our proposed reasoning engine, we introduced LLMs to guide the search process of PDDL planners with learned heuristics and perform constraint reasoning. To improve the robustness and reliability of reasoning, we deliberately designed the prompts and developed a three-layer nested search mechanism to address potential pitfalls of LLMs. Through a series of experiments, we compared the performance of different prompting methods and prompt compositions, and discussed how to stably generate executable sequences that are coordinated with the real-time status of the working scene.

II. LLM-BASED ROBOTIC MANIPULATION TASK PLANNING

The Neuro-symbolic artificial intelligence (AI) framework implemented in our work forms a closed-loop control system for robots that cohesively integrates reasoning and decision-making with perception and control by fusing the reasoning ability of the symbolic system and the perception and learning ability of the neural systems. As input to the neural networks, multi-modal sensory information are extracted into neural predicates. To perform the screw removal task, a set of independent action primitives inspired from workers' manual operations were designed in advance. Formal definition of these primitives in PDDL described in terms of relevant parameters, preconditions and expected effects were given based on the aforementioned predicates (see https://sites.google.com/view/robot-llm). When performing a task, the robot initially senses the environment, generates a feasible action plan based on the system state, and then executes it in sequence. The robot continuously checks the current state and will re-plan if it is inconsistent with expectations. In our previous work, a first-in-first-out (FIFO) based BFS algorithm (as shown in Algorithm 1) was used to generate the solution with the fewest operations [13].

Obviously, when the number of predicates and primitives increases, the time complexity of this algorithm will increase significantly, which is not conducive to frequent real-time reasoning. Since our ideal goal is to enable robots to solve problems with complete autonomy, that is, to plan and execute solely through perception of the external environment and self-awareness, we are bound to explore some options to replace the existing task planning module. Recently, We are interested in applying the popular LLMs to task planning for robots, so that the robot can make decisions on its own rather than by human-designed algorithms. Therefore, we developed a task planning method introducing the state-of-the-art language model, GPT-4 [14].

The architecture of the robotic manipulation task planning is as Figure 1 shows, including a reasoning engine and a verification engine that interact with the working scene. As Figure 2(b) shows, we use prompts to generate action plans, including descriptions of predicates and primitives, introduction to work scenarios, definition of planning rules, task instructions, and exemplars containing some positive and negative samples in the reasoning engine. Predicates and primitives serve as key elements of the reasoning engine. The definition of planning rules and task instructions are used to intuitively convey the concept of PDDL to the language model, as well as the task goals and corresponding implementation means. On account of the uncertainty in reality, we introduced a verification engine that interacts with the work scenario in real time. Since our system is derived from PDDL, the verification engine performs a secondary check on the predefined preconditions of each primitive before execution to ensure that the robot only executes when all conditions are met, otherwise the reasoning engine is required to re-plan based on the current situation.

Many studies have demonstrated that LLMs perform poorly on complicated zero-shot reasoning tasks, and even simple few-shot learning is not enough to meet the needs at the application level [2]. Inspired by the good performance of the collaborative mechanism of the proposal module and the evaluation module applied in ToT in solving general problems, we proposed a three-layer framework to build our reasoning engine. Unlike existing methods such as CoT that sample complete chains of thought, we generate a thought tree that maintains a certain size, in which each thought node consists of a coherent language sequence as an intermediate stage in solving the problem (shown in Figure 2(a)). As is illustrated in Figure 3, this structure allows the language model to check the feasibility of each sampled intermediate thought node instantiated in PDDL and subsequently evaluate it, so as to achieve deliberate reasoning by autonomously deciding whether to retain or discard nodes based on the score of the decision obtained by few-shot learning, and consequently generate a reliable complete thought chain to solve the problem. Naturally, proposals judged as "NO" in the self-evaluation mechanism will be rejected, while those judged as "YES" are considered to be retainable, among which those judged as "SURE" have a higher priority than "LIKELY".

Within our proposed framework, a variety of search algorithms can be plugged in, such as the BFS and DFS algorithms proposed in [4]. When the search space is large, DFS can be used to quickly obtain an executable primitive sequence, improving the efficiency of task planning. In order to save

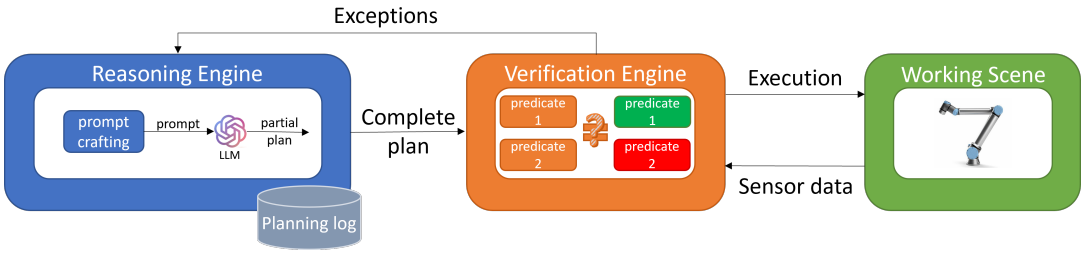

Fig. 1. LLM-based robotic manipulation task planning

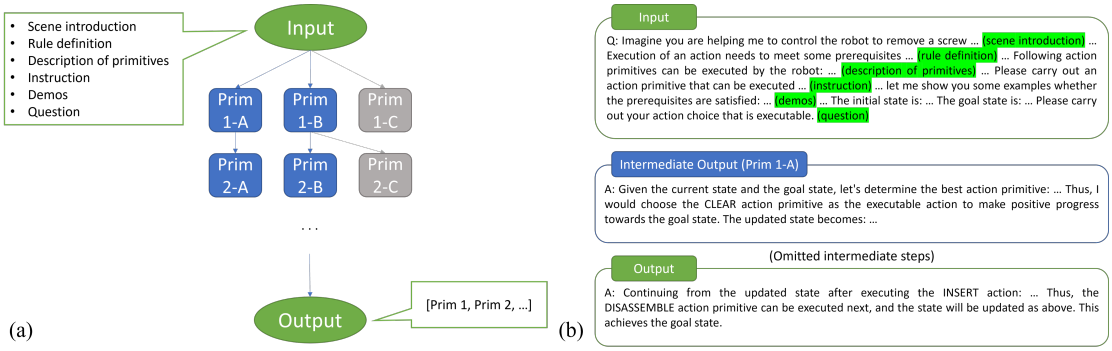

Fig. 2. Prompt-driven search tree framework for stable generation of feasible action plans. (a) demonstrates the overall structure of the thought tree to maintain its limited size, (b) presents details in the prompts of the instantiated thought nodes.

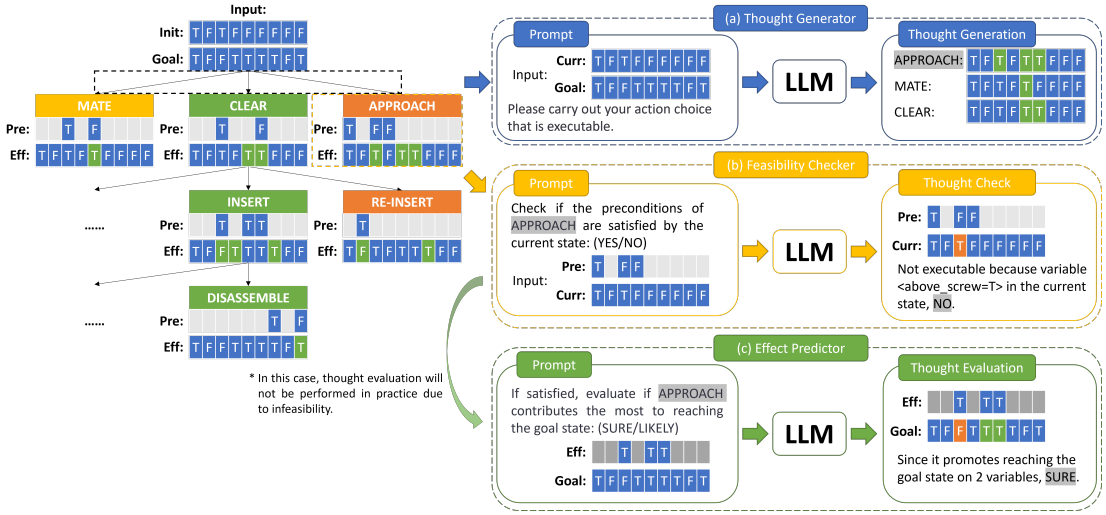

Fig. 3. The mechanism of three-layer prompting autonomously generating and assessing solutions in the reasoning engine. The LLM is prompted for (a) thought generation, (b) thought check, and (c) thought evaluation.

the execution time of the robot, BFS can be applied to find the shortest operation path. Moreover, with the robustness brought by the self-evaluation mechanism, the LLM can also be directly commanded to autonomously generate a complete plan with the shortest steps at once. On this basis, we can implement heuristic search by directly accessing LLM instead of pre-programming or learning, thus making this autonomous system closer to embodied intelligence.

III. Experiments and Results

In order to test the performance of our proposed reasoning engine, we conducted a series of simulation and real-machine experiments combined with Neuroysmbolic TAMP using both GPT-3.5 and GPT-4. We used natural language and key-value pairs composed of variable names and Boolean values respectively to express predicates and perform thought sampling in the thought generator, and compared the reliability of the two descriptions. An ablation study was also conducted on components of the prompt. For the feasibility checker and effect predictor in the inference engine, we also conducted independent simulation experiments to examine their performance. Furthermore, we connected our planner to the original Neuro-symbolic TAMP, conducted experiments in real scenarios, analyzed its comprehensive performance based on the results, and compared it with existing LLM-based planning methods, such as CoT.

A. Comparison of Different Descriptions

To explore how to make LLM better understand the concept of PDDL, we selected two forms: ordinary natural language and concise key-value pairs, to express the predicate and then describe the system state. Given the current and goal states,

Algorithm 1: Neuro-symbolic TAMP for Disassembly Task

```
Input: S_0, S_G, A ;                                   /* Initial State, Target State, Primitive set */
Output: op_list ;                                      /* primitive sequence to accomplish the task */
initial_queue(Q_1);
Q_1.enqueue((S_0, {})) ;      /* Initialize Q1, The initial state of the planning is S_0; {} is the list of disassemble action
  primitives, initial state is empty  */
while Q_1 ≠ null do
    curr_status, curr_op_list = Q_1.pop_front();
    initial_set(tmp_set) ;   /* Temporarily record the current state and the new state after using the operation primitive */
    for a in A do ;                          /* Traverse all of the action primitives definition in the planning problem */
        if curr_status ⊃ a.precondition then
            new_status = apply(curr_status, a);                    /* Apply action primitives a to the current state */
            new_op_list = curr_op_list + a;
            if new_status ⊃ SG then                                                  /* Achieve the goal state */
                return new_op_list
            end
            tmp_set.add((new_status, new_op_list));
        end
    end
    List L = sort_and_filter(tmp_set); /* Sort the tmp set tuples by likelihood and filter out tuples with a low probability */
    for t in L do
        Q_1.enqueue(t);                                  /* Add the tuples into the queue that meet the requirements  */
    end
end
```

TABLE I
FEASIBILITY OF PLANNING UNDER DIFFERENT DESCRIPTION TYPES

Descritpion	Success Rate	
	GPT-3.5	*GPT-4*
Natural Language	38.8%	100.0%
Key-value Pairs	33.8%	95.0%

single-step sampling was performed in the thought generator, and the feasibility of the thought (whether it was executable in the context of PDDL) was treated as the criterion in the zero-shot case. Randomly selected from 8 legal system states as the initial state, 400 sets of samples were collected under each description type, and the results are shown in Table I. It can be found that for each specific model, the feasibility of the plan generated using natural language description is slightly higher than that of key-value pairs without significant difference. In subsequent experiments, standardized key-value pair descriptions were used to conveniently generate prompts.

B. Ablation Study on Components of the Prompt

For logical reasoning methods that rely on LLMs, the design of prompts is crucial. In our planner, prompts are divided into 6 main sections, among which description of primitives, instructions and questions are indispensable. To analyze the role played by the remaining components, we conducted ablation studies in GPT-3.5. In each case, 160 random samples of single-step planning with 3 randomly selected feasible examples in the exemplar component were performed. As shown in Table II, narrative content such as scenarios and rules did not meaningfully promote the generation of feasible solutions. On the contrary, the introduction of exemplars brought a non-negligible improvement reaching 58.1% in performance, sparking our interest in the impact of different numbers of exemplars. It can be observed from Figure 4 that for unchecked single-step sampling, simply increasing the number of examples cannot further improve the success rate of planning. Therefore the reasoning framework

TABLE II
ABLATION STUDY RESULTS WITH DIFFERENCT PROMPT COMPONENTS

Prompt Component			Success Rate
Scene introduction	*Rule definition*	*Exemplars*	
-	-	-	30.6%
✓	-	-	38.1%
-	✓	-	23.1%
-	-	✓	56.9%
✓	✓	-	24.4%
✓	-	✓	50.6%
-	✓	✓	51.9%
✓	✓	✓	58.1%

we proposed including a checking mechanism as well as an evaluation mechanism is necessary.

C. Comparison of Comprehensive Performance Using Different Approaches

Before judging the overall performance of our proposed reasoning engine, simulation tests were conducted on the feasibility checker and the effect predictor respectively. Since they essentially acted as classifiers for the output of the thought generator, each category was given an example as a reference in the experiment. For 720 randomly generated thoughts, the feasibility checker's accuracy was as high as 99.2%. In advance, the net contribution of the primitive's effect to moving closer to the goal state was treated as the criterion for the effect predictor, that is, a positive change of no less than two predicates would be judged as "SURE", otherwise it would be "LIKELY". With the same sample size, the accuracy was only 63.8%, but fortunately, this would not affect the feasibility of the plan.

Subsequently, we connected the reasoning engine to the Neuro-symbolic TAMP proposed in our previous work and conducted experiments in real scenarios to test its comprehensive performance in GPT-3.5. In this experiment, we used a search strategy similar to BFS, but limited the search space. At each level of the search tree, thought generation was stopped when two different feasible solutions were obtained,

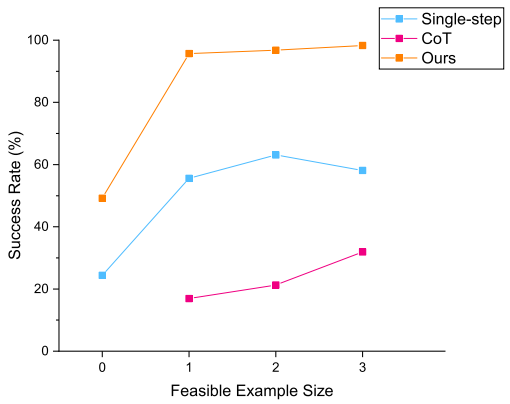

Fig. 4. Performance of different approaches with a various number of exemplars

and if not satisfied, up to five thoughts would be generated (in practice, it turned out to be enough to ensure at least one feasible solution). During the layer-by-layer iteration, no more than two feasible thought nodes were maintained, and the shortest thought chain was finally transformed into the task planning solution and exported to the robot arm for execution. The robust execution of the primitives was ensured through the verification engine proposed that interacted with the working scenario. As shown in Figure 4, our reasoning engine achieved excellent performance even given only one exemplar. In 160 sets of experiments with randomly selected initial states, the success rate of generating a complete feasible action sequence reached 95.7%, which was much higher than unsupervised single-step planning and the existing popular LLM-based prompting method CoT. And in execution, due to the existence of the verification engine, incorrect plans that were inconsistent with reality would be re-planned, ensuring that the success rate of system work reached 100%.

IV. Discussions

We provide a reliable and efficient robot planning execution system by seamlessly integrating LLMs and logical reasoning. We can adjust the number of steps in each generation to adapt to different LLMs so as to meet deployment needs. The existence of the verification engine prevents us from worrying about the influence of LLM hallucination. However, we are currently only applying it to the relatively simple scenario of removing bolts. Due to the limited number of primitives and predicates, we cannot yet make a very accurate efficiency comparison. In the future, we will conduct more extensive research in more complex scenarios.

Acknowledgment

The authors express their sincerest thanks to the Ministry of Industry and Information Technology of China for financing this research within the program ”2021 High Quality Development Project (TC210H02C)”.

References

[1] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P.-S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J.-B. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d’Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. Hechtman, L. Weidinger, I. Gabriel, W. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving, “Scaling language models: Methods, analysis insights from training gopher,” 2022.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. 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. M. 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 34th International Conference on Neural Information Processing Systems*, ser. NIPS’20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[3] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” 2023.

[4] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” 2023.

[5] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, “Chatgpt for robotics: Design principles and model abilities,” Microsoft, Tech. Rep. MSR-TR-2023-8, February 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/

[6] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, “Vima: General robot manipulation with multimodal prompts,” 2023.

[7] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, “Palm-e: An embodied multimodal language model,” 2023.

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” 2023.

[9] Y. Zhang, H. Zhang, Z. Wang, S. Z. Zhang, H. Li, and M. Chen, “Development of an autonomous, explainable, robust robotic system for electric vehicle battery disassembly,” *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 409–414, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260387028

[10] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, and D. Weld, “Pddl - the planning domain definition language,” 08 1998.

[11] 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, dec 2003.

[12] H. L. S. Younes and M. L. Littman, “Ppddl 1 . 0 : An extension to pddl for expressing planning domains with probabilistic effects,” 2004.

[13] H. Zhang, H. Yang, H. Wang, Z. Wang, S. Zhang, and M. Chen, “Autonomous electric vehicle battery disassembly based on neurosymbolic computing,” in *Intelligent Systems and Applications*, K. Arai, Ed. Cham: Springer International Publishing, 2023, pp. 443–457.

[14] OpenAI, “Gpt-4 technical report,” 2023.