Bridging Natural Language Understanding, Symbolic Planning, and Robotic Execution through Hybrid AI Systems

Luis Palacios Medinacelli[1] , **Matteo Morelli**[1] , **Gaël de Chalendar**[1] , **Mykola Liashuha**[1] , **Jaonary Rabarisoa**[1] , **Lucas Labarussiat**[1] , [1] and **Raphaël Lallement** [1]
[1]Université Paris-Saclay, CEA, List. Palaiseau, France
{luis.palacios, matteo.morelli, gael.de-chalendar, mykola.liashuha, jaonary.rabarisoa, lucas.labarussiat, raphael.lallement }@cea.fr

Abstract

The integration of hybrid AI technologies, specifically Symbolic AI and Generative AI, has the potential to create more robust systems capable of addressing problems that neither approach can solve independently. LLMs enable human-like interactions with artificial systems by leveraging vast amounts of encoded ”common sense” knowledge. However, these models often lack certainty, explicit rules, and the ability to incorporate domain-specific knowledge effectively. In this work, we explore and demonstrate the feasibility of combining the reasoning capabilities of Symbolic AI with the natural language understanding and contextual reasoning strengths of LLMs. We apply our approach to a robotic pick-and-place use case, where LLMs interpret user commands expressed in natural language to infer PDDL goals, enabling classical planning and execution by a robot system. By combining these technologies, not only we increase the scope of the system's solving capabilities but also enable controlled behavior of the LLM using domain-specific knowledge. We present a proof of concept (PoC) implementation to concretely demonstrate these claims, bridging existing theoretical discussions with a practical robotic application. Our results highlight how the hybrid approach expands the scope of solvable tasks and ensures greater interpretability and control.

1 Introduction

Recent advancements in Large Language Models (LLMs) have enabled more human-like interactions with AI systems by leveraging extensive ”common sense” knowledge encoded in natural language. However, these models face notable limitations, including a lack of certainty, explicit rule-based reasoning, and integration of domain-specific knowledge. These deficiencies restrict their ability to perform structured and precise tasks required in many practical applications.

Symbolic AI, in contrast, excels in tasks requiring explicit domain knowledge, deterministic rules, and interpretability. While powerful within its scope, Symbolic AI struggles with natural language understanding and lacks the ability to reason flexibly about open-ended problems.

Our work explores the integration of Large Language Models (LLMs) and Symbolic AI for planning in robotics. We present a hybrid system where LLMs interpret natural language (NL) descriptions to infer PDDL goals, enabling classical planning and execution in robotic domains. Our pipeline ensures seamless interaction between natural language, AI planning, and actuation, with feedback loops enhancing transparency and robustness. The approach allows the integration of contextual reasoning and natural language understanding (via LLMs) with the structured reasoning and precision of Symbolic AI (PDDL). This combination enables systems to expand the range of solvable tasks while simultaneously enhancing control and interpretability through explicit symbolic representations. In this paper we highlight the contributions our approach and validate its effectiveness with a proof-of-concept implementation.

Our main contributions are:

- Goal Interpretation: Demonstrate how Large Language Models (LLMs) can flexibly interpret and infer nuanced goals from natural language descriptions, showing their capabilities go beyond simple translation.

- Hybrid AI Workflow: Integrate LLMs with symbolic planners and robotic systems to establish a controlled, interpretable process for goal setting and execution.

- Proof-of-Concept (PoC) Validation: Demonstrate the feasibility of the approach by implementing and end-to-end system and running initial experiments in robotics tasks.

The remainder of this paper is organized as follows: Section 2 describes our approach, our methodology and depicts the system architecture. Section 3 shows a first set of experiments, aimed to showcase the ways the system can be used and how some of the latest open-source LLMs perform when integrated in our pipeline. Section 4 reviews related work, highlighting the theoretical basis for combining Symbolic AI and LLMs, with focus on planning. Section 5 discusses the broader implications of this work and outlines directions for future research.

2 Approach

The approach has been developed as part of a larger project named ”Magicoders”, at our research institution. The project was aimed at exploring the use of generative AI (GenAI) in robotics, in particular LLMs and VLMs. One of the veins of research was the use of GenAI for planning in robotics. A classic approach to planning in symbolic AI is the Planning Domain Definition Language (PDDL) [Aeronautiques *et al.*, 1998]. PDDL version 2.1 [Fox and Long, 2003], which introduced support for time and numeric resources, was used in the Third International Planning Competition (held in 2002). We adopt this version for our approach. PDDL is based on first-order logic (FOL) and enables the description of a domain of interest (e.g., goods distribution, pick-and-place tasks, etc.) in terms of the objects and actions relevant to that domain. A PDDL problem consists of a PDDL domain and an initial state (the initial configuration of objects). Given a goal state (a desired final arrangement), a PDDL solver attempts to find a finite sequence of actions that transforms the initial state into the goal state. If such a sequence exists, it is called a plan and it is a solution for the PDDL problem. An example of some of the PDDL elements can be found in section 2.2.

In this context, as depicted in figure 1, the project considers a robotic arm, and a set of objects (e.g. cubes) to be arranged depending on the user's instructions. The system is made aware of the initial state of the scene, using a VLM, and then awaits a description from the user, of what the goal state should be.

The user expresses then the goal (as speech or text) which should talk about a new arrangement of the objects in the scene. This intention is then passed to a LLM component whose job is to provide an interpretation of the user's intention in terms of the PDDL domain, and express it as a PDDL goal. Here the semantics of the PDDL domain, the objects, the predicates, the possible actions, and the terminology have to be respected to provide a coherent goal. Thus, it is no trivial task.

Take for example a set of cubes placed on a table. If we want to **”make a tower with all the cubes”**, we could interpret this goal as placing the cubes on top of each other. This answer is far from the format a PDDL solver can use, and more importantly there are many uncertainties and ambiguities in the formulation of the goal that can't be deterministicaly harmonized with a PDDL formulation. Yet, a person with knowledge about PDDL and looking at the scene might arrive to the conclusion that the cubes should be placed on top of each other, and then use the objects declared in the PDDL problem formulation along with the ”affordable” actions defined in the PDDL domain, to formulate a valid PDDL goal and feed it to a solver. This complexity is handled in our approach relying on a LLM.

If the goal or the interpretation is flaw, the user can then ”interact” (chat) with the system to steer it and arrive to the desired (or acceptable) result.

Once a valid goal is attained, it is then easily coupled with the PDDL domain and the PDDL formulation of the initial state, to search for a plan. This is done relying on a classic PDDL solver (see section 2.3) and, if a plan is found (it can be the case that even though the goal's syntax is proper, there might not exist a plan to achieve it), it is fed to the execution layer. The execution layer contains a mapping from the possible actions in the PDDL domain, to basic robotic skills. (e.g. pick, place, stack, unstack). Thus in principle any plan found is executable by the robot. Thus, once the PDDL goal is formulated, if a plan is found, then there is a *guarantee* that the current state of the system can be changed into the goal state. This guarantee can not be given by the LLM component alone.

In the following we present the stages of the approach.

2.1 Scene Semantization

Scene semantization is a crucial component in a robotic task planning pipeline. Its role is to transform sensed data coming from the perception systems into a symbolic representation of robot's operational environment. In the workflow of Figure 1, the scene semantization component produces the initial state (init) of PDDL problem model, where object instances conform to object types of the domain model, as well as the list of predicates which are true. In our current implementation, it is made up of two parts. First, a Visual Language Model (VLM) module, which uses a carefully designed prompting strategy to adapt a pre-trained model (specifically InternVL 2.5) to the specific concepts of our domain model, therefore avoiding the need of fine-tuning the VLM on task-specific data. Second, a software module, which processes the scene description produced by the VLM in JSON format and translates it to a syntactically consistent *PDDL init* section of the problem model.

An in-depth description of the scene semantization component is beyond the scope of this paper. In addition, other approaches are possible. The investigation is on-going and will be subject of future publications.

2.2 Goal Interpretation and Formulation

We have chosen a classic a PDDL domain from the 2nd International Planning Competition (2000) [1]. The domain, called ”blocks world” includes the main actions and predicates for a pick-and-place scenario. Specifically the domain considers:

- Objects: The domain considers only 1 type of object:
  - *block*
- Predicates: It considers
  - a binary predicate: *(on ?x - block ?y - block)*,
  - three unary prediactes: *(ontable ?x - block), (clear ?x - block), (holding ?x - block)*
  - and a 0-arity predicate, to establish that the robot arm's is free: *(handempty)*
- Actions: Four actions, along with their preconditions and effects to manipulate the blocks:
  - *pick-up*
  - *put-down*
  - *stack*

[1] https://github.com/potassco/pddl-instances/blob/master/ipc-2000/domains/blocks-strips-typed/domain.pddl

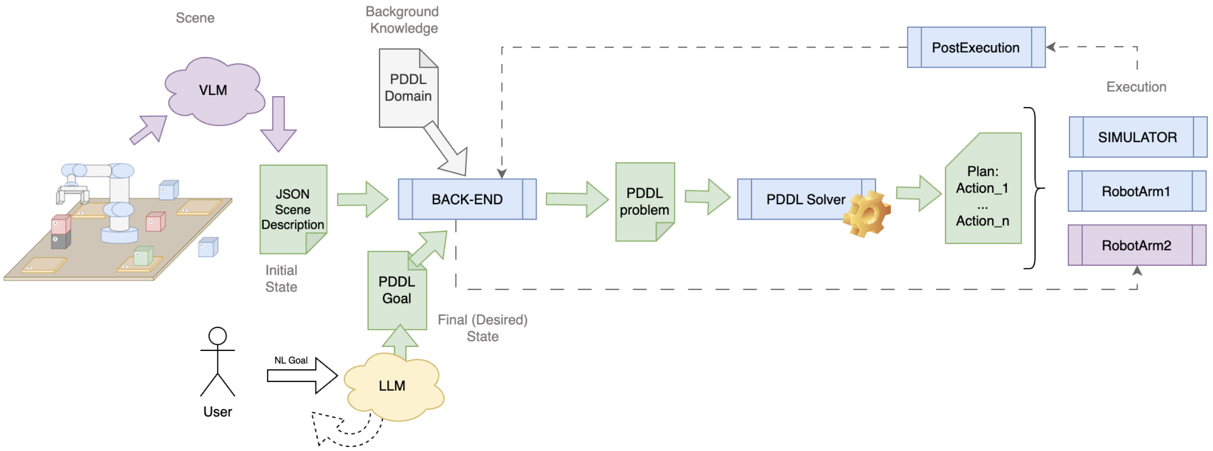
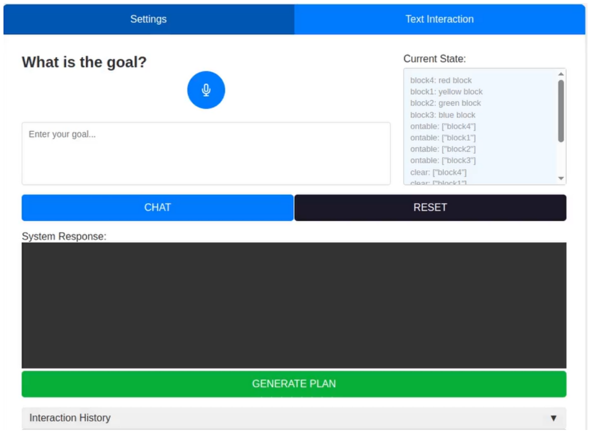


Figure 1: The workflow and components of the approach. From the left to right: A scene composed by a robotic arm and a set of objects, is captured and interpreted by a VLM, providing a semantic description of the scene. This is expressed in terms of the PDDL Domain, and fed to the ”BACK-END” module. This module centralizes the processes and keeps track of the workflow. A user (bottom left) expresses in natural language a rearrangement of the objects in the scene. This intention is interpreted by an LLM, to generate a PDDL goal. The initial state, along with the goal and the PDDL domain are then used to produce a PDDL problem, that can be processed by a solver. The output of the solver is a plan, expressed as a sequence of actions belonging to the domain. This plan is then passed to the execution layer. In the case of the RobotArm2 setting, the backed directly produces actuation instructions, bypassing the PDDL solver. Finally, a post-execution module enables feedback in case the execution of the plan fails.



Current State:

```
block4: red block
block1: yellow block
block2: green block
block3: blue block
ontable: ["block4"]
ontable: ["block1"]
ontable: ["block2"]
ontable: ["block3"]
clear: ["block4"]
clear: ["block1"]
```

Figure 2: The GUI of the PoC.

   - *unstack*

The PDDL domain, represents knowledge and an interpretation on how the actions should be applied along with the conditions that define their affordability. For example, to unstack a block, it *mus be clear* (i.e. no block can be stack on top of it), therefore to move a block in the middle of a pile of blocks, we first have to unstack the upper blocks. This behaviour, which could be crucial for some domains, is encoded in PDDL. The expected results and the intended meaning of domain specific actions are formally and unambiguously established in the PDDL domain. These are the features we want to propagate to the behaviour of the hybrid system (traceability and guarantees about the result). A human, or an ”intelligent robot”, when asked to *clear* a block in the middle of a pile, could just pick the middle cube, along with all the cubes already stuck on top of it and place it on the table. This would result in a goal successfully achieved (i.e. the block is clear), but the implicit understanding that upper cubes should be unstuck first is not necessarily guaranteed.

In our approach we make the following considerations:

- we assume that the PDDL domain is given: this implies the vocabulary, the types of objects, the possible predicates to describe the scene, the actions, their preconditions and effects. All must be known to the system/agent/LLM.
- the goal must talk about the objects already in the scene (no new objects, no inexistent objects) and must be expressed in terms of the PDDL domain.
- the interpretation of the intended meaning of the goal may carry imperfections, and thus the user is allowed to interact with the system (see figure 2), so that the system can *correct itself* to come out with an acceptable version of the goal.
- the user is not expected to type or edit the goal directly, but to interact with the system (via prompting) so that the system itself comes out with the intended goal, and thus the hybrid system behaves as a whole. This is a design choice and it could be adapted depending on the use-case.

2.3 Planning

Planning is done in our approach by a classic PDDL solver. In our case we relied of PROBE[2] as a solver, which is an open source, PDDL2 compliant solver. PROBE can be compiled as a command-line linux-executable, which is used as a

[2] https://github.com/aig-upf/probe

”black-box” whose inputs are a PDDL problem and a PDDL goal. The output of PROBE is a plan, if it exists (and if it can be found within a time limit), or a failure. The failure could be: there is no plan, where the formulation of the goal is trivial or unachievable; or an error message. Otherwise a valid plan is returned, that is, a linear sequence of PDDL actions represented by (usually grounded) predicates. This sequence is then fed to the execution layer, which ”understands” the actions and can execute them.

In this fashion, we ensure that the found plan:

- is expressed in terms of the PDDL domain
- is a valid PDDL.2 plan
- it *guarantees* the transformation of the initial/current state into the final/goal state. This, of course, under the assumption that no external factors affect the states or the execution.

It is to be noted that in our approach we did not target to obtain the best NL2PDDL LLM model, or train a LLM for a specific planning domain. In our work we focus on the interaction of hybrid approaches (LLM+PDDL) and on the means of control and conditioning of the interaction and output of the LLM, brought by the symbolic AI component of the system.

Among the main features symbolic AI contributes to the system we find:

- Guaranteed plans: If a plan is found, then there is a guarantee that the initial state can be transformed into the final state.
- Unambiguity and Explainability: The description of the domain, the scene, the objects and their interactions are well understood and traceable.
- Reusability: Symbolic knowledge, domains, plans, etc. are easily transferable to other systems concerned by the same or similar domains.

2.4 Execution Layer

The execution layer is the part of the robotic system in charge of executing the sequence of actions computed by the task planner to reach the goal state. Our implementation uses a modular, component-based and skill-oriented execution architecture. Skills are software components that organize robotic algorithms’ functionality (motion planning, trajectory following, object manipulation, visual perception, etc.) and expose it to higher-level task and mission specification modules. Skills are of two types, atomic or composite. Atomic skills bind directly to the algorithmic modules, while composite skills are composed of lower-level skills which can be composite and/or atomic. Composite skills are organized into sequences using policies Behavior Trees, although other formalisms like Finite State Machines can also be used.

Each of the actions in the PDDL domain, has a corresponding primitive (or composed) skill in the execution layer. This, seemingly evident, requirement is very important for the development of the approach. The PDDL domain and its terminology provide the basis for expressing the goal, and moreover establish what actions can be used by an actuator. This coherence has to be preserved all along the workflow, settings and interactions of the system. If either the domain or the execution layer were to evolve or be modified independently, the coherence could be lost and the approach would fail. The synchronization, automation, reformulation and aggregations of the robotic skills with the PDDL domain constitute current and future work, to make the approach more robust and adaptable to different scenarios and requirements.

As for the current approach, 3 execution settings have been tested, 2 that consume the plan and sequence of actions generated in the previous stage (a simulator and a robot arm), and 1 end-to-end approach (a VLM bypassing PDDL).

Each one of these settings presents it own challenges, peculiarities and opportunities. Yet, their full details fall out of the scope of this paper, we briefly mention them to provide context and completeness to our report.

3 Experiments

As stated before, this work is both innovative and in current progress. The interdisciplinary nature of our work opens several directions to evaluate and exploit. The current paper focuses on what, to our view, are some of the most relevant features in such a hybrid system. To this end we have established two sequences of tasks, which use natural language terminology and common sense notions to express the goal. This interpretation is collapsed by an LLM into the vocabulary of the PDDL domain and restricted to its semantics.

Success or failure in our case is given by 1) the appropriate interpretation of the scene and the goal, using generative AI, and by 2) the correctness, feasibility and executability of the produced plan.

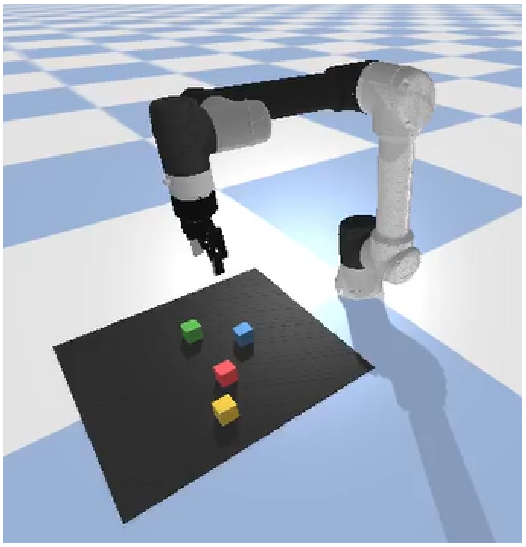


Figure 3: The scene for the experiments.

The scenario of the experiments is illustrated in figure 3 and involves a set of colored cubes and a robotic arm. Intuitively, the user looks at the scene and describes how the cubes should be arranged (as the goal). The system interprets this requirement, produces a plan and this plan is then executed by the robot arm.

3.1 The tasks

The first task tests the ability to re-interpret *loosely defined goals* in terms of the specific domain, and use general features and descriptions (like color and position) to define the goals. Additionally, we want to show that the execution does respect the rules and intended meaning of the PDDL domain.

1. Task 1:
   (a) Make a tower with all cubes.
   (b) Spread all the cubes on the table.
   (c) Make a tower with all cubes except the red one, and where the blue cube is on the bottom.
   (d) Exchange the blue cube by the red one.
2. Task 2:
   (a) Make two towers with 2 cubes each, choose the cube colors yourself.
   (b) Exchange the cubes on top of the towers.

3.2 Results

We tested the system with two different settings (see Figure 4) for the scene description: using a simulator (PyBullet [3]) and using a VLM to describe the scene, and for each setting we have tested 3 LLMs, deployed locally.

The results of the experiments can be seen in figure 4, where for each experiment we test the same sets of tasks (section 3.1) combining: a) a setting for the scene description (VLM or virtual/simulator) and b) a LLM (Ollama, Mistral, DeepSeek), thus the 6 experiments. The results can be found in the columns:

- INIT : whether the description of the initial state was correct.
- GOAL: whether the goal was interpreted as a correct PDDL goal.

Generative AI analysis

As shown in Figure 4, Mistral and OLLama, performed quite well on simple instructions and both succeeded on providing a valid and sound goal for task 1. Interestingly for task 2 they had problems, mainly interpreting the two towers with two cubes each, as a correct PDDL goal. DeepSeek failed the most, but not necessarily because it did not understand the task, or because is not capable, but it tends to provide long answers which are verbose and with exceeding analysis in the output. It was ”avoiding’ to provide a straight answer. Although this might be improved with prompt engineering and fine tuning, this remains out of the scope of this paper and remains as open question for direct further research.

There were some interpretations where the goal was syntactically correct, but not semantically. We find that even humans could interpret these differently. Also the PDDL version and its variants depend on the intended meaning, expected format and PDDL solver. Sometimes the LLM would do a ”too good” job (i.e. including variables and placeholders) that needed to be downsized.

Symbolic AI analysis

Once the goal was syntactically correct it could be the case that it does not express a valid goal, or that it is unattainable. The solver in this case would provide a ”no- plan”, or trivial response. This information can be used to rephrase the goal, change it or find faults.

If the goal does indeed represent the intended meaning, is feasible and well formed, then the solver tries to find a plan. If a plan is found, then *there is a formal guarantee* that the initial state can be transformed into the final state, using the set of actions defined in the domain. That is, if we consider in the world representation **only** the elements of the current state of the scene (initial state) in terms of the domain, the goal will succeed.

This guarantee is important for safety-critical domains, and is a PoC of the controlled behaviour we can impose on LLMs/GenAI using background knowledge. It is important to note here, that this analysis and results apply *regardless of the chosen LLM.* Thus the background knowledge can be ’plugged-in’ into any general-purpose or fine-tuned LLM, while respecting the guarantees.

4 Related Work

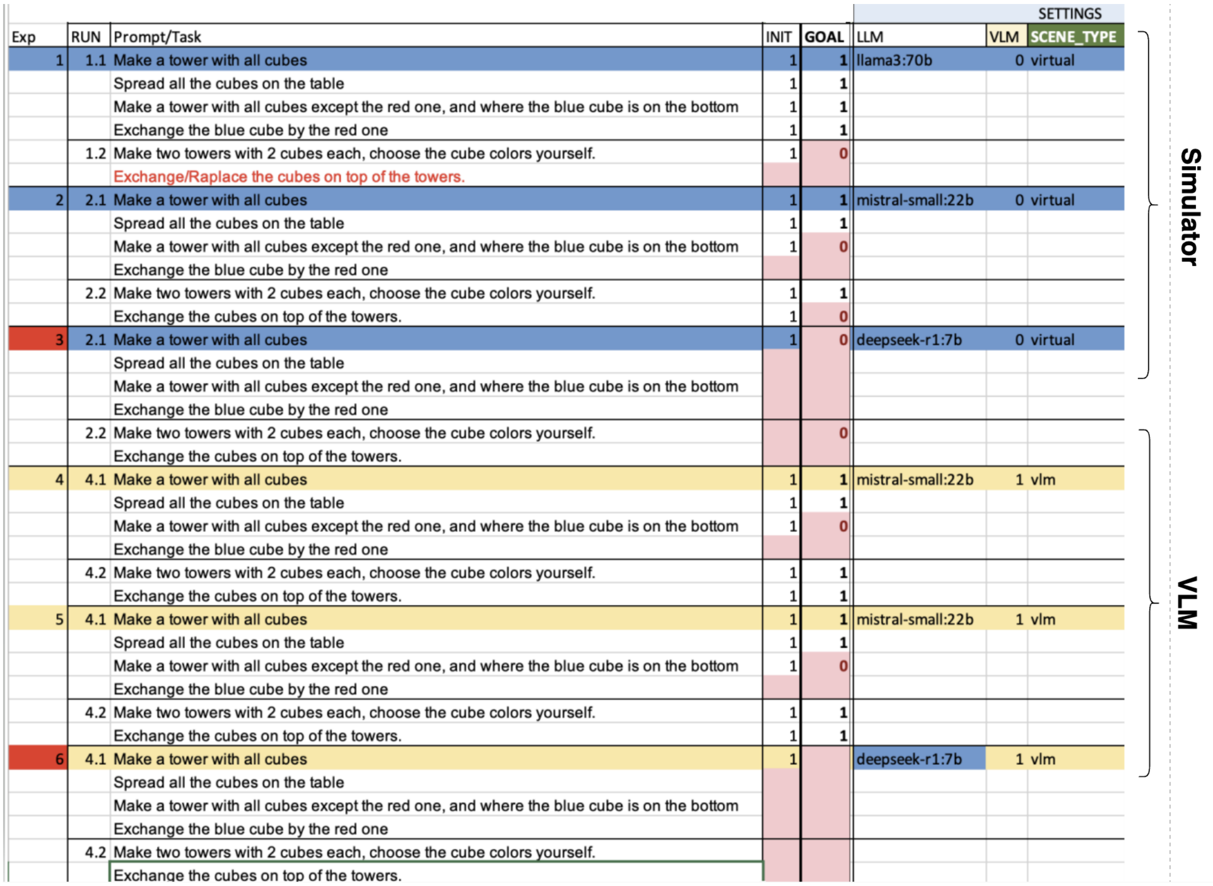
In this section we provide a review of the most relevant research related to planing using PDDL and LLMs.

In [Guan *et al.*, 2023], the authors investigate a two-phase pipeline for planning by engaging LLMs to generate PDDL models from natural language, then having classical solvers plan from these generated models. The authors establish that GPT-4 can make complex domains, containing more than 40 actions, and that in the case where the model is corrected and validated, these models can produce valid solutions to planning problems. Therefore, the authors’ pipeline shows LLMs can significantly reduce the burden of specifying formal domains by hand.

In [Xie *et al.*, 2023] the autors note that while LLMs excel at many natural language processing tasks, they struggle with accurate reasoning and planning. Their work investigates whether LLMs can translate human-specified goals in natural language into structured planning language. Experiments using GPT-3.5 variants show that LLMs are more effective at translation than planning, successfully inferring missing details in under-specified goals using commonsense reasoning. However, they still face challenges with tasks requiring numerical or spatial reasoning and are sensitive to prompt design. Overall, LLMs show promise for goal translation in planning systems, though limitations remain.

In the work of [Liu *et al.*, 2023] the authors argue that even though LLMs have shown impressive zero-shot capabilities in natural language tasks, they still struggle with long-horizon robot planning problems. In contrast, classical planners can efficiently generate optimal plans when provided with structured input. To bridge this gap, their work presents a framework (LLM+P) that combines the language understanding strengths of LLMs with the planning efficiency of classical methods. LLM+P translates natural language problem descriptions into PDDL, uses a classical planner to find a solution, and then converts the resulting plan back into natural

[3] https://pypi.org/project/pybullet/



Simulator
VLM

					SETTINGS		
Exp	RUN	Prompt/Task	INIT	GOAL	LLM	VLM	SCENE_TYPE
1	1.1	Make a tower with all cubes	1	1	llama3:70b	0	virtual
		Spread all the cubes on the table	1	1			
		Make a tower with all cubes except the red one, and where the blue cube is on the bottom	1	1			
		Exchange the blue cube by the red one	1	1			
	1.2	Make two towers with 2 cubes each, choose the cube colors yourself.	1	0			
		Exchange/Raplace the cubes on top of the towers.					
2	2.1	Make a tower with all cubes	1	1	mistral-small:22b	0	virtual
		Spread all the cubes on the table	1	1			
		Make a tower with all cubes except the red one, and where the blue cube is on the bottom	1	0			
		Exchange the blue cube by the red one					
	2.2	Make two towers with 2 cubes each, choose the cube colors yourself.	1	1			
		Exchange the cubes on top of the towers.	1	0			
3	2.1	Make a tower with all cubes	1	0	deepseek-r1:7b	0	virtual
		Spread all the cubes on the table					
		Make a tower with all cubes except the red one, and where the blue cube is on the bottom					
		Exchange the blue cube by the red one					
	2.2	Make two towers with 2 cubes each, choose the cube colors yourself.		0			
		Exchange the cubes on top of the towers.					
4	4.1	Make a tower with all cubes	1	1	mistral-small:22b	1	vlm
		Spread all the cubes on the table	1	1			
		Make a tower with all cubes except the red one, and where the blue cube is on the bottom	1	0			
		Exchange the blue cube by the red one					
	4.2	Make two towers with 2 cubes each, choose the cube colors yourself.	1	1			
		Exchange the cubes on top of the towers.	1	1			
5	4.1	Make a tower with all cubes	1	1	mistral-small:22b	1	vlm
		Spread all the cubes on the table	1	1			
		Make a tower with all cubes except the red one, and where the blue cube is on the bottom	1	0			
		Exchange the blue cube by the red one					
	4.2	Make two towers with 2 cubes each, choose the cube colors yourself.	1	1			
		Exchange the cubes on top of the towers.	1	1			
6	4.1	Make a tower with all cubes	1		deepseek-r1:7b	1	vlm
		Spread all the cubes on the table					
		Make a tower with all cubes except the red one, and where the blue cube is on the bottom					
		Exchange the blue cube by the red one					
	4.2	Make two towers with 2 cubes each, choose the cube colors yourself.					
		Exchange the cubes on top of the towers.					

Figure 4: Experiments

language.

The work in [Oswald *et al.*, 2024] states that Large language models (LLMs) have shown strong capabilities in generating structured outputs from natural language, prompting interest in their use for knowledge engineering tasks such as domain model creation in symbolic AI. Since manually building planning domain models remains a bottleneck in automated planning, their work explores whether LLMs can generate accurate PDDL domain models from simple textual descriptions. A framework is introduced to evaluate these models by comparing sets of generated plans, and an empirical study is conducted across seven LLMs and nine planning domains. Results show that high-parameter LLMs can moderately succeed in generating valid planning domains, suggesting promise for reducing human effort in symbolic problem representation.

As LLMs continue to demonstrate impressive reasoning and decision-making abilities, their potential for enhancing autonomous agent planning has gained significant attention. The survey in [Huang *et al.*, 2024] provides a systematic review of LLM-based agent planning, focusing on recent advancements aimed at improving planning capabilities. In their work a taxonomy is proposed categorizing existing works into five key areas: Task Decomposition, Plan Selection, External Modules, Reflection, and Memory. A thorough analysis is provided for each category, along with a discussion of ongoing challenges in the field. For further details and in-depth discussion, the interested reader is referred to the full survey, which also evaluates the advantages and limitations of current methodologies in LLM-based planning.

Symbolic task planning is a widely used method for enabling robot autonomy, but it struggles [Capitanelli and Mastrogiovanni, 2024] in dynamic, real-world human-robot collaboration scenarios, especially when frequent re-planning or long-term plan validity is required. To solve these challenges, the authors introduce Teriyaki, a framework that bridges symbolic task planning with machine learning approaches. By training LLMs, specifically GPT-3, to function as neurosymbolic planners compatible with PDDL, Teriyaki aims to address the limitations of traditional symbolic planners. Key benefits include improved scalability with increasing domain complexity and the ability to generate plans incrementally, enabling concurrent planning and execution.

In [Silver *et al.*, 2024], they examine how large language models can be generalized in classical planning settings. They propose a hybrid protocol that combain GPT-4 to synthesize domain-specific Python planners from PDDL descriptions and a small number of training tasks. Their approach combines chain-of-thought prompting, natural language strategy design, and iterative self-debugging, resulting in strong generalization across multiple domains. While the LLM operates autonomously, their setup is heavily scaffolded, indicating the importance of feedback and structure in guiding planning capabilities.

In [Valmeekam *et al.*, 2023], the criticisms of LLMs are present in their assessment of LLMs on classical planning tasks. They draw a distinction between both autonomous and heuristics and argue that even if LLMs do not independently create valid plans, the responses can be useful heuristics when incorporated in symbolic planners. In the end, they conclude that hybrid approaches that utilize LLM to help symbolic systems are much better than LLM systems by themselves.

5 Conclusions and Further Work

We have presented our current research and approach, with focus on hybrid plannig with LLMs. Our work, comprised in a larger setting, requires and relies on scene interpretation as its input, and is intended to be used by an actuator in the next step in the workflow. As such, it is an interdisciplinary work (planning, robotics, formal languages, NLP, machine-learning).

We show via our PoC, that hybrid planning enhances classic planning and pure generative-AI based planning with flexibility, robustness, traceability and certain guarantees.

We emphasize that the work is not a LLM benchmarking, but that we rely on generic and open-source LLMs. The choice lies on the fast advancing pace, the training costs and the large customisation available for these models. We look for an approach where a LLM (specific or not) can be ”plugged-in” and where the whole hybrid system will still preserve its properties. Thus, since the tendency is to find each time smaller LLM models, with better and better capabilities, we expect that our approach can only improve. We do not tackle the LLMs improvement at this point, but their control, manipulation and interaction to solve complex tasks in a hybrid setting.

We also consider that in order to keep up with the fast pace, provide relevant contributions and obtain valuable feedback, divulgation of ongoing results and their discussion are essential.

6 Acknowledgments

This work has been partially funded by the ”AI-Augmented Software Engineering for Intelligent Robotics” (ASTIR) project, under the Horizon Europe, European Program for Research and Innovation.

References

[Aeronautiques *et al.*, 1998] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins Sri, Anthony Barrett, Dave Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[Capitanelli and Mastrogiovanni, 2024] Alessio Capitanelli and Fulvio Mastrogiovanni. A framework for neurosymbolic robot action planning using large language models. *Frontiers in Neurorobotics*, 18:1342786, 2024.

[Fox and Long, 2003] Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[Guan *et al.*, 2023] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

[Huang *et al.*, 2024] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

[Liu *et al.*, 2023] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[Oswald *et al.*, 2024] James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large language models as planning domain generators. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 423–431, 2024.

[Silver *et al.*, 2024] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 20256–20264, 2024.

[Valmeekam *et al.*, 2023] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

[Xie *et al.*, 2023] Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.