

# Are General-Purpose LLMs Ready for Planning? A Large-Scale Evaluation in PDDL

Kaustubh Vyas, Damien Graux, Sébastien Montella, Pavlos Vougiouklis & Jeff Z. Pan  
Huawei Technologies Ltd., United Kingdom  
{firstname.lastname}@huawei.com

## Abstract

In recent advancements, large language models (LLMs) have exhibited proficiency in code generation and chain-of-thought reasoning, laying the groundwork for tackling automatic formal planning tasks. This study evaluates the potential of LLMs to understand and generate Planning Domain Definition Language (PDDL), an essential representation in artificial intelligence planning.

We conduct an extensive analysis across 20 distinct models spanning 7 major LLM families, both commercial and open-source. We focus exclusively on general-purpose, off-the-shelf models, excluding recent reasoning-centric models, to avoid confounding from task-specific architectural scaffolding and to evaluate the native planning fluency of widely deployed LLMs. Our comprehensive evaluation sheds light on the zero-shot LLM capabilities of parsing, generating, and reasoning with PDDL. Our findings indicate that while some models demonstrate notable effectiveness in handling PDDL, others pose limitations in more complex scenarios requiring nuanced planning knowledge. These results highlight the promise and current limitations of LLMs in formal planning tasks, offering insights into their application and guiding future efforts in AI-driven planning paradigms.

## 1 Introduction

Automated planning has long been a cornerstone of artificial intelligence, traditionally relying on explicit domain knowledge encoded in formal languages such as PDDL. In recent years, the rapid evolution of large language models (LLMs) has sparked considerable interest in their ability to bridge the gap between natural language descriptions and formal planning representations.

Early studies by [Zuo et al. \(2024\)](#) and [Oswald et al. \(2024\)](#) demonstrated that LLMs are capable of translating natural language descriptions into syntactically valid PDDL representations. However, these pioneering works also revealed significant gaps, as the generated planning domains frequently diverge from gold-standard models, both syntactically and semantically. This observation has spurred further research into the underlying reasoning capabilities of LLMs and their potential role in executing complete planning tasks.

Recent developments have introduced LLMs explicitly optimized for reasoning, such as OpenAI’s o1 model [OpenAI et al. \(2024\)](#) and DeepSeek’s R1 [DeepSeek-AI et al. \(2025\)](#). These models incorporate internal mechanisms or intermediate representations designed to facilitate formal reasoning. However, we intentionally exclude such reasoning-centric models from our evaluation. Their architecture often obscures whether planning outputs emerge from semantic understanding or embedded symbolic heuristics. Additionally, they are not yet broadly accessible for integration in real-world planning pipelines. By focusing on widely available, general-purpose LLMs, we aim to isolate natural language-to-PDDL fluency without reasoning-specific scaffolding.

Advancement in LLMs fuelled recent efforts that looked into how these multi-billion parameter models can be best employed as agents [Huang et al. \(2024\)](#). Building on this

momentum, several strategies have been proposed to map user instructions into PDDL problems [Pallagani et al. \(2023\)](#); [Liu et al. \(2023a\)](#); [Dagan et al. \(2023\)](#); [Gestrin et al. \(2024\)](#); [Zhang et al. \(2024\)](#), without however providing conclusive evidence for the feasibility of the task in the general domain. These studies underscore both the promise and the challenges inherent in leveraging LLMs for complex planning and reasoning tasks, where transforming natural language into an executable agentic workflow remains a non-trivial endeavor.

In this study, we step back to examine the fluency of twenty LLMs from seven major families in the PDDL language, focusing on their ability to parse, generate, and reason with PDDL. Specifically, we leverage the Planetarium benchmark [Zuo et al. \(2024\)](#) alongside the dataset introduced by [Oswald et al. \(2024\)](#) to assess how well these models understand and generate actions, problems, and plans. By analyzing a randomly sampled subset of over 13,000 (NL-instruction, PDDL-problem) pairs, our results show that although some models demonstrate moderate proficiency in handling PDDL, the majority struggle to convert natural language instructions into fully correct PDDL representations. This challenge is especially evident in smaller LLMs, which often fail to produce parsable PDDL.

It is important to note that our evaluation focuses exclusively on zero-shot prompting, as our aim is to test the planning capabilities of off-the-shelf LLMs. Although few-shot prompting approaches might further improve performance, they were not considered in this study to maintain a clear assessment of the models in their unmodified state.

Our contributions are twofold: (1) an extensive evaluation of the zero-shot planning performance of LLMs as reflected in their PDDL handling, and (2) an investigation into the feasibility of employing these models as co-pilots in planning tasks. Importantly, we do not perform a detailed error analysis aimed at improving model outputs. Instead, our focus is to characterize the current capabilities of off-the-shelf LLMs—assessing how well they perform out-of-the-box and whether they are viable for use as planning assistants. Understanding and addressing specific model errors is left to future work.

## 2 Extensive PDDL capability evaluation

From a high-level point of view, PDDL involves three types of elements: the **domains** to represent the possible *actions* available in a certain space, the **problems** which roughly encode the premise and the goal of a *real world* operation to be performed in a defined space (*i.e.* domain) and finally the **plans** that represent the effective set of *actions* to be run to perform the *real world* operation, achieving the *goal* (refer to Appendix A to read more).

Therefore, practically, we stressed the considered LLMs to generate all or part of the aforementioned elements, while maintaining a wide set of evaluation scores across the involved steps to fuel the discussion and draw conclusions.

### 2.1 Action Generation

**Task Signature** = [input: NL instruction, PDDL domain predicates; output: PDDL action]

We rely on the benchmark proposed by [Oswald et al. \(2024\)](#) to evaluate the action generation capabilities: given a seed domain file and the NL description of an action, we let the LLM generate it in proper PDDL syntax. In their article, the authors shared a set of 32 NL-to-Action instructions distributed across 9 popular PDDL domains. We enriched these by generating 4 NL-variations for each NL-to-instruction pair to obtain a dataset of 160  $[(1 + 4) \times 32]$  instructions.

To assess the results, we score along the following dimensions: **Parsable**: Determines if the output conforms to correct PDDL syntax. **Solvable**: Measures how well the action integrates into the target domain (*e.g.* the action may be syntactically correct but involving type mismatches, wrong number of variables for some predicates, ...). **Equivalent**: Syntactically valid PDDL that integrates with the desired domain under the domain equivalence heuristic.

To measure the similarity between the generated action and the gold standard, we calculate the normalised differences in their preconditions and effects, and then subtract this value

from one to derive a similarity score.

$$\text{Similarity} = 1 - \frac{|A_{\text{pre}} \Delta \hat{A}_{\text{pre}}| + |A_{\text{ef}} \Delta \hat{A}_{\text{ef}}|}{|A_{\text{pre}} \cup \hat{A}_{\text{pre}}| + |A_{\text{ef}} \cup \hat{A}_{\text{ef}}|}$$

Where  $A_{\text{pre}}, A_{\text{ef}}$  are preconditions and effects in the gold action and  $\hat{A}_{\text{pre}}, \hat{A}_{\text{ef}}$  are preconditions and effects in the LLM generated action.

## 2.2 Problem Generation

**Task Signature** = [input: NL instruction, PDDL domain; output: PDDL problem]

We choose the Planetarium benchmark to evaluate the problem generation capabilities of models Zuo et al. (2024). The benchmark was primarily selected due its size that enabled a comprehensive evaluation on our side. In particular, we randomly selected 10% of the full dataset, resulting in a test set consisting of 13 203 (NL-instruction, PDDL-problem) pairs. Metrics for this set of experiments are as above (with a slight difference): **Parsable**: generated PDDL adheres to the syntactic rules of the language, **Solvable**: the generated problem can be effectively processed by existing PDDL planners, reflecting its practical utility, **Equivalent**: matches the gold standard in both structure and semantics.

When it comes to measure the similarity between the gold and the generated PDDL problem, we use ChrF Popović (2015) as it is a standard metric to evaluate code generation tasks Evtikhiev et al. (2023). By employing the ChrF metric, we can objectively assess subtle differences between the generated and reference PDDL code, offering a nuanced understanding of each model’s translation accuracy.

## 2.3 Plan Generation

**Task Signature** = [input: PDDL domain, PDDL problem; output: Plan]

Finally, although LLMs are not expected to outperform conventional planners<sup>1</sup>—since their reasoning capabilities rely on intrinsic parametric knowledge rather than explicit logical reasoning<sup>2</sup>—we also aimed to assess their ability to *plan in PDDL* when provided with pairs of domain and problem. For this purpose, we selected domain-problem pairs from the Planetarium benchmark to prompt the models for plan generation. To evaluate generalisation, we categorised these problems based on their level of abstractness, classifying descriptions as either **explicit** or **abstract**. Explicit descriptions are direct propositions found in the PDDL problem (e.g., “block1 is on block2”), whereas abstract descriptions summarise a state (e.g., “all blocks are in a single tower”). Because these descriptions encapsulate both the initial and goal states, there are four possible categories: (1) Abstract initial and goal states, (2) Abstract initial but explicit goal, (3) Explicit initial but abstract goal and (4) Explicit initial and goal states. In total, we selected 40 representatives from each category, yielding 160 pairs. The correctness of the generated plans is then verified using VAL<sup>3</sup>, a tool that assesses whether a plan is compatible with the specified PDDL domain and problem.

## 2.4 Considered LLMs

To review the capabilities of language models to deal with PDDL, we utilised LLMs from several leading organisations, ensuring that both general-purpose and specialist models (i.e. chatting, code generation or instruction-following modes) are considered. Our set of models includes LLMs from

<sup>1</sup>See Frances et al. (2017) for an extensive evaluation of PDDL planners, which also showcases the planner used as reference in our study: BFWS-FF Lipovetzky & Geffner (2017a;b).

<sup>2</sup>Mirzadeh et al. (2024) provides a more nuanced understanding of LLMs’ capabilities and limitations in (mathematical) reasoning.

<sup>3</sup><https://github.com/KCL-Planning/VAL>

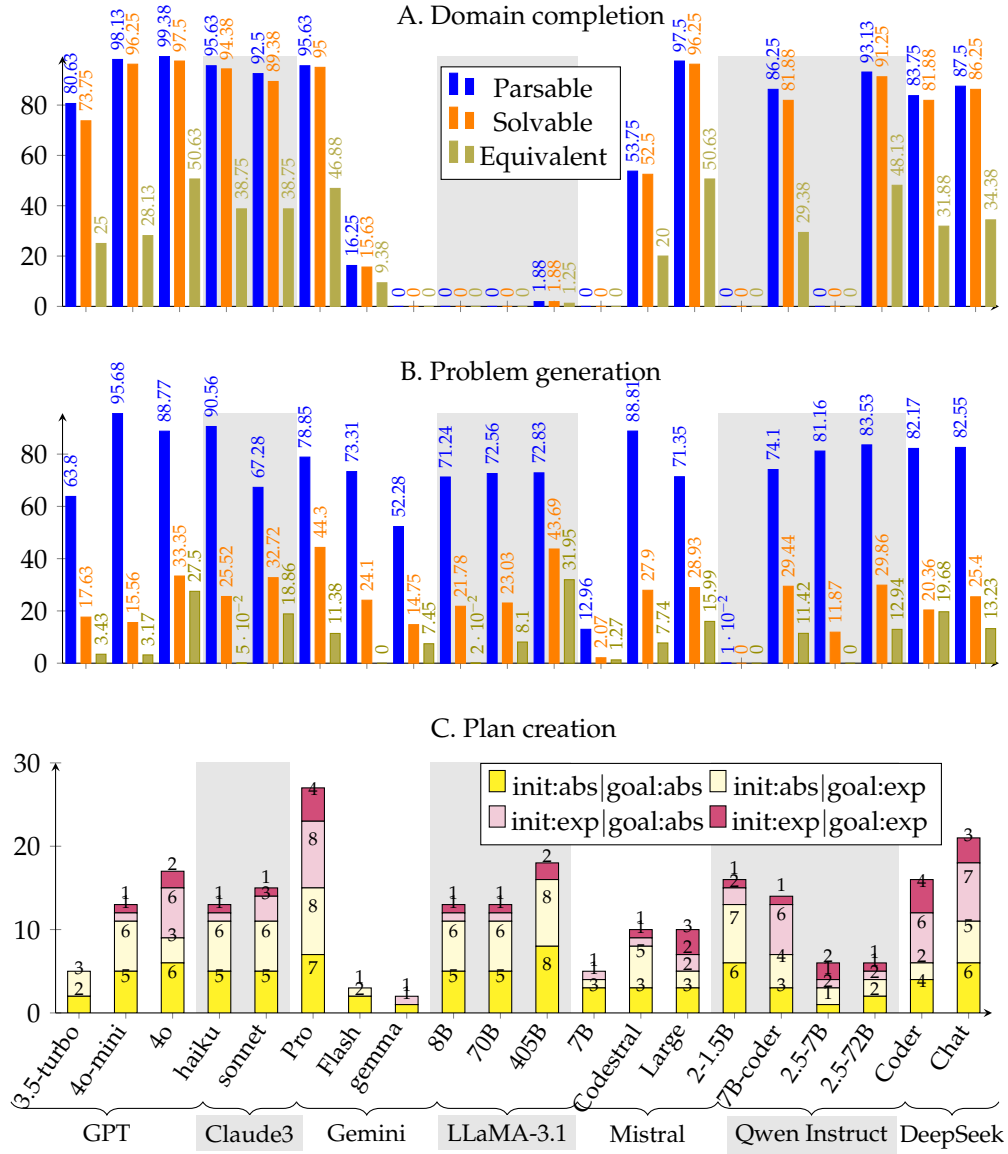


Figure 1: LLM performances across the three benchmarks (higher the better).

- **OpenAI** (GPT-3.5-turbo, GPT-4o-mini, GPT-4o),
- **Anthropic** (Claude-3-Haiku and Claude-3-Sonnet),
- **Google** (Gemini-1.5-Pro, Gemini-1.5-Flash and Gemma-2-9B-it),
- **Meta** (LLaMA-3.1-8B-Instruct, LLaMA-3.1-70B-Instruct, and LLaMA3.1-405B-Instruct),
- **Mistral** (Large2, 7B-Instruct, and Codestral),
- **DeepSeek** (Coder-V2 and Chat-V2),
- **Alibaba** (Qwen2-1.5B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-Coder-7B-Instruct, and Qwen2.5-72B-Instruct).

Overall, this set involves members of 7 distinct providers, including commercial and open LLMs. In addition, this set allows us to compare behaviors and performances across different parameter numbers and specialities.

## 2.5 Results

**Domain (Fig.1A)** We review the performance of 20 distinct LLMs in populating PDDL domain files with new actions based on NL instructions. While most LLMs perform well in generating correct actions, a notable decline in performance is observed in their ability to produce equivalent actions. Among the models, *GPT-4o*, *Qwen2.5-72B-instruct*, and *Mistral-Large2* stand out as the top performers across all three metrics for action generation. In contrast, some models consistently fail to adhere to the required action syntax. Notably, the entire *LLaMA* family performs very poorly, irrespective of the number of parameters. It is worthwhile noting that all the models have their respective *parsable* and *solvable* values very close, in other words the difference between them for a model is of few points. This implies that not only models are able to generate proper PDDL syntax (*parsable*) but they comply with the given domain (*solvable*), such a property could be useful if the models were used as assistants, see Figure 2 and its associated discussion for more details.

**Problem (Fig.1B)** We next assessed the ability of LLMs to generate complete PDDL problems based on NL instructions and a corresponding PDDL domain. While most models can produce syntactically correct and parsable PDDL problems, their performance declines significantly when required to solve the problem for generating a plan or producing problems equivalent to the gold. Interestingly, the *LLaMA* family, which struggled with syntax in action evaluations, demonstrates improved accuracy for this task. Even though *LLaMA-3.1-405B* achieves the highest equivalency rate at about 32%, its performance –and that of all off-the-shelf models– remains inadequate for this task, suggesting that further techniques such as few-shot prompting, fine-tuning, or other enhancements are necessary to assist PDDL problem generation, as explored by Zuo et al. (2024) for instance. Furthermore, models are not consistent through the three metrics: for instance *GPT-4o-mini* which has the best score for *parsable* ends up being in the worst when it comes to *equivalent*, a similar behaviour goes for *Codestral* too.

**Plan (Fig.1C)** Investigating the plan generation, as expected, we find that given a PDDL domain and problem, the models struggle to generate PDDL plans. This was tested on 160 data points, with *Gemini-1.5-pro* performing the best, yet achieving a valid plan in only 16.87% of the cases. In contrast, the BFWS-FF planner succeeded in generating a conclusive plan 86.25% of the time. Additionally, we explored the plan generation relative to the abstractness of the initial state and goal of the PDDL problem. Across all LLMs, we observe that they perform better when the initial state of the PDDL problem is abstract, though, no such pattern is noticed regarding the abstractness of the goal. Once again, the models are not performing similarly across the different tasks. Typically, our smallest model in the mix (*Qwen2-1.5B-Instruct*) which had scores almost all null for the domain and problem tasks, happens to be in the top-5 for the plan generation.

## 2.6 Discussions

Our results indicate that while most LLMs can generate syntactically correct PDDL, they largely lack the capacity to generate effective problems and plans for addressing the input instructions. This behaviour becomes increasingly evident as the complexity of the experimental setup rises.

**Parameter Number** With the exception of the *LLaMa* family, an increased number of parameters in LLMs does not consistently lead to better PDDL fluency. This suggests that current pre- and post-training approaches are not effectively scaling these models to meet the demands of complex PDDL generation, including, but not limited to, long-horizon planning tasks. A clear challenge emerges with models like the *LLaMA* family (see Fig.1), which, while effective in problem generation, struggle significantly with action generation. This issue stems from a syntactic bias: for example, instead of the correct keyword “:precondition”, they generate “:preconditions”, making actions unparsable. Similar patterns are observed in *Gemma-2-9B-it*, *Mistral-7B-Instruct*, or *Qwen2.5-72B-Instruct*, which incorrectly output “(action. . .)” rather than “(:action. . .)”. Similarly, while



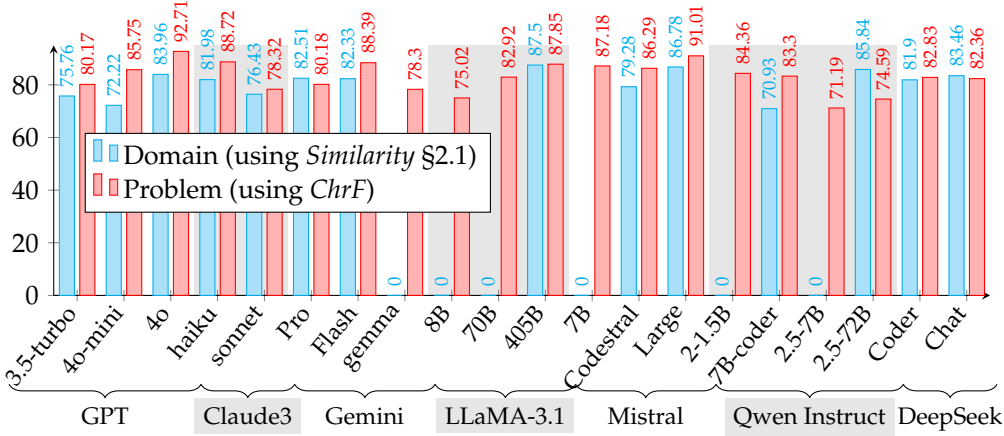


Figure 2: Performances of LLMs as co-pilots: closeness (%) of generations to the “gold”.

generating problems, models often struggle with maintaining correct action sequences in the goal, occasionally placing semantically similar actions in the goal that are not defined in the domain, leading to mismatches.

**LLMs as copilots** However, our findings, in Fig.2, reveal that despite these challenges, LLM-generated PDDL actions and problems demonstrate a high degree of closeness to the gold standard. 8 of the 20 models achieved over 80% similarity for both domain-action generation and problem generation, and showcased near-accurate generation of formal planning structures. This suggests that, while the models (off-the-shelf) may not yet be fully reliable for independent use, they hold strong potential as supportive tools. By generating near-accurate PDDL structures, these models can serve as co-pilots, streamlining the drafting process and allowing experts to focus on refinement and optimization rather than building from the ground up.

The position articulated in Kambhampati et al. (2024) underscores that, although LLMs may not inherently plan effectively, they can nonetheless play a significant supportive role in LLM-modulo planning frameworks. This further reinforces our findings, highlighting the viability of these models as valuable assistants in structured problem-solving tasks. PDDL is actively used in a variety of high-impact disciplines, including robotics, automated logistics, satellite tasking, and mission planning for space exploration systems such as NASA’s MAPGEN Ai-Chang et al. (2004). In these settings, domain experts are often required to manually encode goals, constraints, and available actions into formal PDDL representations—a process that is both time-consuming and error-prone. LLMs can alleviate these burdens by assisting in the initial drafting of domain and problem files, identifying potential syntax issues, and even proposing plausible but incomplete plans that serve as starting points. For instance, in autonomous warehouse operations or planetary rover scheduling, LLMs could support engineers by converting high-level goals into draft PDDL structures, accelerating development cycles.

This co-pilot role aligns with human-in-the-loop paradigms<sup>4</sup>, where automated assistance augments expert reasoning without fully replacing it. Our evaluation suggests that current LLMs, while insufficient as planners, are nonetheless effective collaborators in real-world planning pipelines.

**Horizon Distance** In order to explore how the number of actions (context) in a PDDL problem might affect the generation, in Table 1, we analyse the performance of models with three horizons: **Short** (up to 10 actions), **Medium** (from 11 to 50) and **Long** (more than 50).

<sup>4</sup>See e.g. the recent Magentic-UI initiative by Microsoft adding human feedback on top of their Magentic-One orchestrator Fourney et al. (2024).

Table 1: Horizon scores: (%) for the problem equivalence.

LLM	Short	Medium	Long
<i>GPT-3.5-turbo</i>	20.98	3.09	1.13
<i>GPT-4o-mini</i>	1.13	3.23	3.19
<i>GPT-4o</i>	20.42	28.08	11.32
<i>Claude-3-Haiku</i>	0.36	0.04	0.0
<i>Claude-3-Sonnet</i>	15.52	19.29	6.42
<i>Gemini-1.5-Pro</i>	5.71	11.98	1.51
<i>Gemini-1.5-Flash</i>	0.0	0.0	0.0
<i>Gemma-2-9B-it</i>	4.24	7.71	1.13
<i>LLaMA-3.1-8B-Instruct</i>	0	0.02	0
<i>LLaMA-3.1-70B-Instruct</i>	0	8.58	1.51
<i>LLaMA-3.1-405B-Instruct</i>	32.27	33.29	10.65
<i>Mistral-7B-Instruct</i>	3.15	1.23	1.13
<i>Codestral</i>	5.43	6.72	4.74
<i>Mistral-Large2</i>	11.27	16.03	35.5
<i>Qwen2-1.5B-Instruct</i>	0.0	0.0	0.0
<i>Qwen2.5-Coder-7B-Instruct</i>	27.44	13.15	0.38
<i>Qwen2.5-7B-Instruct</i>	0.0	0.0	0.0
<i>Qwen2.5-72B-Instruct</i>	13.48	13.49	4.45
<i>DeepSeek-Coder-V2</i>	11.83	20.79	2.26
<i>DeepSeek-Chat-V2</i>	6.55	14.05	1.88

While short horizons perform decently as they are straightforward to process, they often lack the necessary context to provide sufficient information for optimal model performance. In contrast, medium horizons offer a richer and more comprehensive context, enabling the model to achieve results that are not only comparable to those with short horizons but often surpass them (Table 1). This suggests that medium horizons strike a balance by providing enough information to enhance generation for more complex queries without overwhelming the models.

The extended context and complexity of PDDL problems exacerbate these issues, as models find it challenging to sustain coherence and alignment across the entire generation process, as scores decrease with long-horizon planning. As described in [Chen et al. \(2024\)](#), long-horizon planning tasks involve “extended sequences of actions” or span “over a prolonged period of time”. Reasoning over longer context windows has been a traditional challenge associated with language models, especially in tasks requiring understanding of complex, long-range dependencies [Li et al. \(2024\)](#). We find that this behaviour exacerbates in the context of generating longer, expressive problems. We believe that problem decomposition strategies, that would seek to split and hierarchically construct an expected problem in a bottom-up fashion could offer promising directions for future work.

### 3 Related Work — PDDL $\cap$ LLM

The generation of PDDL domains and problems has recently garnered significant attention as a means to enhance planning via large language models (LLMs) [Strobel & Kirsch \(2020\)](#); [Silver & Chitnis \(2020\)](#); [Silver et al. \(2022\)](#); [Vyas et al. \(2025\)](#). In parallel, the advent of sophisticated prompting techniques has unlocked new applications for LLMs [Liu et al. \(2023b\)](#); [Graux et al. \(2024\)](#). Nonetheless, while LLMs have demonstrated planning capabilities [Huang et al. \(2024\)](#), they continue to struggle with long-horizon planning, uncertainty in generated plans, and generalisation to unseen domains [Sermanet et al. \(2023\)](#). Consequently, several works have aimed to bridge the gap between the probabilistic nature of LLMs and the deterministic requirements of PDDL-based planners. For instance, [Collins et al.](#)

(2022) compared the out-of-distribution robustness of PDDL-augmented LLMs with human reasoning, highlighting clear limitations in current LLM approaches.

In many settings, LLMs have proven more effective at translating natural language into formal representations rather than performing the planning itself, as noted in works such as Alford et al. (2009); Helmert (2009); Xie et al. (2023). This observation has spurred strategies that decompose the problem into translating user instructions into PDDL problems, solving these problems via formal logic within the PDDL framework, and then translating the resulting plans back into natural language Pallagani et al. (2023); Liu et al. (2023a); Dagan et al. (2023); Silver et al. (2024); Gestrin et al. (2024); Mahdavi et al. (2024); Zhang et al. (2024).

More recent contributions have further refined the dialogue between LLMs and planning. Hao et al. (2023) propose that reasoning with a language model can be reinterpreted as planning with an integrated world model, while Rossetti et al. (2024) explore the learning of general policies for planning directly via GPT models. In addition, benchmark efforts such as PlanBench introduced by Valmeekam et al. Valmeekam et al. (2023) and critical investigations into LLM planning abilities Valmeekam et al. (2023) provide valuable insights into the performance and limitations of current models.

Novel benchmarks such as PlanBench Valmeekam et al. (2023), AutoPlanBench Stein et al. (2024), Planetarium Zuo et al. (2024), and the domain benchmark from Oswald et al. (2024) have been introduced to assess LLMs’ planning capabilities using PDDL. However, to the best of our knowledge, the recent families of foundational models have not yet been extensively benchmarked to reveal their inherent robustness and reliability in handling PDDL generation. In this study, we explore the capacity of these foundational models to generate both PDDL domains and problems, thereby extending prior evaluations and situating our work alongside the latest advances in planning with LLMs.

## 4 Conclusion

In this study, we experimentally reviewed the PDDL capabilities of a large panel of language models: twenty in total, representing multiple dimensions of the current state-of-the-art, while keeping aside the most recent reasoning one in order to focus on general-purpose LLMs. Our evaluations show that (some) LLMs can be used to generate actions to complete PDDL domains, they may also be used to assist in the task of generating PDDL problems from NL instructions. However, as expected, they reveal being poor planners and it is better to rely on “real” planners which have been developed by the community for decades. Surprisingly also, behaviours given a specific model are not uniform across tasks as good performers (even leaders) for a certain PDDL aspect may turn out to be among the worst ones later. Nevertheless, when reviewing how far LLMs are with domain and problem to the gold answers, we witnessed that models could be used as assistants since they often are “not-too-far” from being correct. As a consequence, general-purpose LLMs could then be involved to help experts in their daily tasks.

As previously mentioned, the focus of this study was not to improve LLMs’ capabilities with PDDL formal understanding but rather shedding light on current general capabilities where we showed that these capabilities would be more useful for an expert than for lay-users. However, the obtained results, through our benchmarking experiments, may also benefit research in the domain of formal language model reasoning and may lead researchers to finding new directions. That is why we provide all the results as supplementary material (see Appendix D) for further analysis.

Overall, we hope to pave the road to future efforts in AI-driven planning challenges.



## References

- Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins Sri, Anthony Barrett, Dave Christian-son, et al. Pddl—the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.
- M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.C.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, Kanna Rajan, J. Yglesias, B.G. Chafin, W.C. Dias, and P.F. Maldague. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems*, 19(1):8–12, 2004. doi: 10.1109/MIS.2004.1265878.
- Ronald Alford, Ugur Kuter, and Dana Nau. Translating htms to pddl: A small amount of domain knowledge can go a long way. pp. 1629–1634, 01 2009.
- Xiaoshuai Chen, Wei Chen, Dongmyoung Lee, Yukun Ge, Nicolas Rojas, and Petar Kormushev. A backbone for long-horizon robot task understanding. *arXiv:2408.01334*, 2024.
- Katherine M. Collins, Catherine Wong, Jiahai Feng, Megan Wei, and Joshua B. Tenenbaum. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks, 2022. URL <https://arxiv.org/abs/2205.05718>.
- Gautier Dagan, Frank Keller, and Alex Lascarides. Dynamic planning with a llm, 2023. URL <https://arxiv.org/abs/2308.06391>.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojuan Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qishi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shutong Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Mikhail Evtikhiev, Egor Bogomolov, Yaroslav Sokolov, and Timofey Bryksin. Out of the bleu: how should we assess quality of the code generation models? *Journal of Systems and Software*, 203:111741, 2023.

- Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.
- Guillem Frances, Miquel Ramírez Jávega, Nir Lipovetzky, and Hector Geffner. Purely declarative action descriptions are overrated: Classical planning with simulators. In *26th Int. Joint Conf. on Artificial Intelligence; Aug 19-25; Melbourne, Australia*. p. 4294-301. IJCAI Organization, 2017.
- Elliot Gestrin, Marco Kuhlmann, and Jendrik Seipp. Towards robust LLM-driven planning from minimal text descriptions. In *Workshop on Human-Aware Explainable Planning*, 2024. URL <https://openreview.net/forum?id=NmzHuV101q>.
- Damien Graux, Sébastien Montella, Hajira Jabeen, Claire Gardent, and Jeff Z Pan. [PromptEng] First International Workshop on Prompt Engineering for Pre-Trained Language Models. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 1311–1312, 2024.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *EMNLP*, pp. 8154–8173, Singapore, December 2023. doi: 10.18653/v1/2023.emnlp-main.507.
- Malte Helmert. Concise finite-domain representations for pddl planning tasks. *Artif. Intell.*, 173:503–535, 2009. URL <https://api.semanticscholar.org/CorpusID:9377590>.
- 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.
- Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, . . . , and Anil B Murthy. Position: LLMs can’t plan, but can help planning in LLM-modulo frameworks. In *ICML*, volume 235, pp. 22895–22907. PMLR, 21–27 Jul 2024.
- Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhui Chen. Long-context llms struggle with long in-context learning, 2024.
- Nir Lipovetzky and Hector Geffner. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017a.
- Nir Lipovetzky and Hector Geffner. A polynomial planning algorithm that beats lama and ff. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pp. 195–199, 2017b.
- B. Liu, Yuqian Jiang, Xiaohan Zhang, Qian Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency. *ArXiv*, abs/2304.11477, 2023a. URL <https://api.semanticscholar.org/CorpusID:258298051>.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9), jan 2023b. ISSN 0360-0300. doi: 10.1145/3560815. URL <https://doi.org/10.1145/3560815>.
- Sadegh Mahdavi, Raquel Aoki, Keyi Tang, and Yanshuai Cao. Leveraging environment interaction for automated PDDL generation and planning with large language models, 2024. URL <https://arxiv.org/abs/2407.12979>.
- Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O’Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatabaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiye Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL <https://arxiv.org/abs/2412.16720>.

James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large Language Models as Planning Domain Generators. In *Proceedings of the Int. Conference on Automated Planning and Scheduling*, volume 34, pp. 423–431, 2024.

Vishal Pallagani, Bharath Muppasani, Biplav Srivastava, Francesca Rossi, Lior Horesh, Keerthiram Murugesan, Andrea Loreggia, Francesco Fabiano, Rony Joseph, and Yathin Kethepalli. Plansformer tool: Demonstrating generation of symbolic plans using transformers. In Edith Elkind (ed.), *IJCAI-23*, pp. 7158–7162. IJCAI Organization, 8 2023. doi: 10.24963/ijcai.2023/839.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049.

- Nicholas Rossetti, Massimiliano Tummolo, Alfonso Emilio Gerevini, Luca Putelli, Ivan Serina, Mattia Chiari, and Matteo Olivato. Learning general policies for planning through gpt models. In *ICAPS*, volume 34, pp. 500–508, 2024.
- Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, . . . , and Yuan Cao. Robovqa: Multimodal long-horizon reasoning for robotics. In *arXiv preprint arXiv:2311.00899*, 2023.
- Tom Silver and Rohan Chitnis. Pddl gym: Gym environments from pddl problems. In *Int. Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop*, 2020. URL <https://github.com/tomsilver/pddl gym>.
- Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDL planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL <https://openreview.net/forum?id=1QMMUB4zf1>.
- Tom Silver, Soham Dan, Kavitha Srinivas, Josh Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.
- Katharina Stein, Daniel Fišer, Jörg Hoffmann, and Alexander Koller. Autoplanbench: Automatically generating benchmarks for llm planners from pddl, 2024. URL <https://arxiv.org/abs/2311.09830>.
- Volker Strobel and Alexandra Kirsch. *MyPDDL: Tools for Efficiently Creating PDDL Domains and Problems*, pp. 67–90. Springer International Publishing, 2020. ISBN 9783030385613. URL [http://dx.doi.org/10.1007/978-3-030-38561-3\\_4](http://dx.doi.org/10.1007/978-3-030-38561-3_4).
- Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023.
- Kaustubh Vyas, Damien Graux, Yijun Yang, Sébastien Montella, Chenxin Diao, Wendi Zhou, Pavlos Vougiouklis, Ruofei Lai, Yang Ren, Keshuang Li, and Jeff Z. Pan. From An LLM Swarm To A PDDL-Empowered HIVE: Planning Self-Executed Instructions In A Multi-Modal Jungle. *International Conference on Learning Representations (ICLR)*, 2025.
- Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models, 2023. URL <https://arxiv.org/abs/2302.05128>.
- Li Zhang, Peter Jansen, Tianyi Zhang, Peter Clark, Chris Callison-Burch, and Niket Tandon. PDDLEGO: Iterative planning in textual environments. In Danushka Bollegala and Vered Shwartz (eds.), *13th Joint Conf. on Lexical and Computational Semantics (\*SEM 2024)*, pp. 212–221, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.starsem-1.17. URL <https://aclanthology.org/2024.starsem-1.17>.
- Max Zuo, Francisco Piedrahita Velez, Xiaochen Li, Michael L Littman, and Stephen H Bach. Planetarium: A Rigorous Benchmark for Translating Text to Structured Planning Languages. *arXiv:2407.03321*, 2024.

## A Primer on PDDL

The **Planning Domain Definition Language (PDDL)** is a standard formalism developed for representing planning problems in artificial intelligence. It was first introduced by Drew McDermott and colleagues in 1998 to provide a unified format for comparing AI planners in competitions such as the International Planning Competition (IPC). PDDL is inspired by first-order logic and STRIPS-based action representations.

A planning problem in PDDL typically consists of three key components:

- **Domain:** A domain file declares the types of objects in the world, predicates (relations over those objects), and actions. Each action includes:
  - :parameters – the variables used in the action.
  - :precondition – a logical formula that must hold for the action to be applicable.
  - :effect – the state changes caused by executing the action.
- **Problem:** A problem file defines a specific instance of the domain. It includes:
  - :objects – the objects present in the instance.
  - :init – the initial state described using ground predicates.
  - :goal – the desired state (goal condition) that must be achieved.
- **Plan:** A plan is a sequence of instantiated actions (i.e., actions where parameters are grounded with actual objects) that transitions the system from the initial state to a state that satisfies the goal.

### Domain Example

```
(define (domain transport)
  (:predicates (at ?x ?loc) (connected ?from ?to))
  (:action move
    :parameters (?x ?from ?to)
    :precondition (and (at ?x ?from) (connected ?from ?to))
    :effect (and (not (at ?x ?from)) (at ?x ?to))))
```

### Problem Example

```
(define (problem move-box)
  (:domain transport)
  (:objects box1 roomA roomB)
  (:init (at box1 roomA) (connected roomA roomB))
  (:goal (at box1 roomB)))
```

### Plan Example

```
0: (move box1 roomA roomB)
```

PDDL supports additional features such as typing, negative preconditions, conditional effects, temporal planning, and numeric fluents (in later versions such as PDDL 2.1+). In this work, we focus on classical (non-temporal) PDDL constructs that form the foundation of most LLM-based planning experiments. For further reference, readers may consult the original PDDL specifications and tutorials such as [Aeronautiques et al. \(1998\)](#) or [Alford et al. \(2009\)](#).

## B Examples of 0-shot prompts

In order to give a better idea on the prompts we triggered at the language models, we provide in this Appendix an example for each of the tested task, *i.e.* Action creation, Problem generation and Plan creation. We refer the reader to the main body of the article for more details, see *e.g.* Figure 1 (A, B and C) for details of the results<sup>5</sup>. In particular, here we provide:

- The action generation (put-down) for the BLOCKS domain;
- A problem generation in blocksworld domain (init=holding\_one);
- A plan generation in blocksworld domain (init=on\_table, goal=tower).

<sup>5</sup>Moreover, we will provide –once not *double-blind* anymore– a link toward our Github repository which gathers all the benchmarks (including the prompts) that were used to assess the LLM performances.



**Prompt and result for the action generation (put-down) for the BLOCKS domain**

Given a description of an action in some domain, convert it to Planning Domain Definition Language (PDDL) action. You may only use the allowed predicates for each action.

Allowed Predicates:

- (handempty) : the hand is empty.
- (holding ?x - block): block x is held.
- (clear ?x - block): block x is clear.
- (on ?x - block ?y - block) : block x is on block y.
- (ontable ?x - block): block x is on the table.

Input: The action, "put-down" will have the hand put down a block.

PDDL Action:

```
(:action put-down
  :parameters (?x - block)
  :precondition (holding ?x)
  :effect (and (not (holding ?x)) (clear ?x) (handempty) (ontable ?x)) )
```

**A problem generation in blocksworld domain (init=holding\_one)**

Provide me with the complete, valid problem PDDL file that describes the following planning problem directly without further explanations or texts. The domain for the planning problem is:

```
(define (domain blocksworld) (:requirements :strips) (:predicates (arm-empty) (clear
?x) (holding ?x) (on ?x ?y) (on-table ?x)) (:action pickup :parameters (?ob) :precondition
(and (clear ?ob) (on-table ?ob) (arm-empty))) :effect (and (holding ?ob) (not (clear
?ob)) (not (on-table ?ob)) (not (arm-empty)))) (:action putdown :parameters (?ob)
:precondition (holding ?ob) :effect (and (clear ?ob) (arm-empty) (on-table ?ob) (not
(holding ?ob)))) (:action stack :parameters (?ob ?underob) :precondition (and (clear
?underob) (holding ?ob)) :effect (and (arm-empty) (clear ?ob) (on ?ob ?underob) (not
(clear ?underob)) (not (holding ?ob)))) (:action unstack :parameters (?ob ?underob)
:precondition (and (on ?ob ?underob) (clear ?ob) (arm-empty)) :effect (and (holding
?ob) (clear ?underob) (not (on ?ob ?underob)) (not (clear ?ob)) (not (arm-empty)))) )
```

Planning problem:

You have 17 blocks. You are holding b1. b2 is clear. b2 is on the table. b3 is clear. b3 is on the table. b4 is clear. b4 is on the table. b5 is clear. b5 is on the table. b6 is clear. b6 is on the table. b7 is clear. b7 is on the table. b8 is clear. b8 is on the table. b9 is clear. b9 is on the table. b10 is clear. b10 is on the table. b11 is clear. b11 is on the table. b12 is clear. b12 is on the table. b13 is clear. b13 is on the table. b14 is clear. b14 is on the table. b15 is clear. b15 is on the table. b16 is clear. b16 is on the table. b17 is clear. b17 is on the table.

Your goal is to stack the blocks into a towers of heights 1, 1, 1, 2, 3, 9.

Problem PDDL:

```
(define (problem holding_one_to_tower_1_1_1_2_3_9)
  (:domain blocksworld)
  (:requirements :strips)
  (:objects b1 b10 b11 b12 b13 b14 b15 b16 b17 b2 b3 b4 b5 b6 b7 b8 b9)
  (:init (clear b10) (clear b11) (clear b12) (clear b13) (clear b14) (clear b15) (clear b16)
(clear b17) (clear b2) (clear b3) (clear b4) (clear b5) (clear b6) (clear b7) (clear b8) (clear
b9) (holding b1) (on-table b10) (on-table b11) (on-table b12) (on-table b13) (on-table
b14) (on-table b15) (on-table b16) (on-table b17) (on-table b2) (on-table b3) (on-table
b4) (on-table b5) (on-table b6) (on-table b7) (on-table b8) (on-table b9))
  (:goal (and (arm-empty) (clear b1) (on-table b1) (clear b2) (on-table b2) (clear b3)
(on-table b3) (clear b4) (on b4 b5) (on-table b5) (clear b6) (on b6 b7) (on b7 b8) (on-
table b8) (clear b9) (on b9 b10) (on b10 b11) (on b11 b12) (on b12 b13) (on b13 b14)
(on b14 b15) (on b15 b16) (on b16 b17) (on-table b17))))
```

**Plan generation in blocksworld domain (init=on\_table, goal=tower)**

Given a PDDL domain and a PDDL problem file, come up with the plan associated with the problem. The domain describes the possible actions and their effects, while the problem file details the specific scenario to be solved. Do not generate anything but the correct plan

Domain PDDL:

```
(define (domain blocksworld) (:requirements :strips) (:predicates (arm-empty)
(clear ?x) (holding ?x) (on ?x ?y) (on-table ?x)) (:action pickup :parameters (?ob)
:precondition (and (clear ?ob) (on-table ?ob) (arm-empty)) :effect (and (holding
?ob) (not (clear ?ob)) (not (on-table ?ob)) (not (arm-empty)))) (:action putdown
:parameters (?ob) :precondition (holding ?ob) :effect (and (clear ?ob) (arm-empty)
(on-table ?ob) (not (holding ?ob)))) (:action stack :parameters (?ob ?underob)
:precondition (and (clear ?underob) (holding ?ob)) :effect (and (arm-empty) (clear
?ob) (on ?ob ?underob) (not (clear ?underob)) (not (holding ?ob)))) (:action unstack
:parameters (?ob ?underob) :precondition (and (on ?ob ?underob) (clear ?ob)
(arm-empty)) :effect (and (holding ?ob) (clear ?underob) (not (on ?ob ?underob)) (not
(clear ?ob)) (not (arm-empty)))) )
```

Problem PDDL:

```
(define (problem on_table_to_tower_1.1.1.1.3.13) (:domain
blocksworld) (:requirements :strips) (:objects b1 b10 b11 b12 b13 b14 b15 b16
b17 b18 b19 b2 b20 b3 b4 b5 b6 b7 b8 b9) (:init (arm-empty) (clear b1) (clear b10)
(clear b11) (clear b12) (clear b13) (clear b14) (clear b15) (clear b16) (clear b17) (clear
b18) (clear b19) (clear b2) (clear b20) (clear b3) (clear b4) (clear b5) (clear b6) (clear
b7) (clear b8) (clear b9) (on-table b1) (on-table b10) (on-table b11) (on-table b12)
(on-table b13) (on-table b14) (on-table b15) (on-table b16) (on-table b17) (on-table
b18) (on-table b19) (on-table b2) (on-table b20) (on-table b3) (on-table b4) (on-table
b5) (on-table b6) (on-table b7) (on-table b8) (on-table b9)) (:goal (and (arm-empty)
(clear b1) (on-table b1) (clear b2) (on-table b2) (clear b3) (on-table b3) (clear b4)
(on-table b4) (clear b5) (on b5 b6) (on b6 b7) (on-table b7) (clear b8) (on b8 b9) (on b9
b10) (on b10 b11) (on b11 b12) (on b12 b13) (on b13 b14) (on b14 b15) (on b15 b16)
(on b16 b17) (on b17 b18) (on b18 b19) (on b19 b20) (on-table b20))))
```

Plan:

```
(define (problem on_table_to_tower_1.1.1.1.3.13)
(:domain blocksworld) (:requirements :strips)
(:objects b1 b10 b11 b12 b13 b14 b15 b16 b17 b18 b19 b2 b20 b3 b4 b5 b6 b7 b8 b9)
(:init (arm-empty) (clear b1) (clear b10) (clear b11) (clear b12) (clear b13) (clear b14)
(clear b15) (clear b16) (clear b17) (clear b18) (clear b19) (clear b2) (clear b20) (clear b3)
(clear b4) (clear b5) (clear b6) (clear b7) (clear b8) (clear b9) (on-table b1) (on-table b10)
(on-table b11) (on-table b12) (on-table b13) (on-table b14) (on-table b15) (on-table
b16) (on-table b17) (on-table b18) (on-table b19) (on-table b2) (on-table b20) (on-table
b3) (on-table b4) (on-table b5) (on-table b6) (on-table b7) (on-table b8) (on-table b9))
(:goal (and (arm-empty) (clear b1) (on-table b1) (clear b2) (on-table b2) (clear b3)
(on-table b3) (clear b4) (on-table b4) (clear b5) (on b5 b6) (on b6 b7) (on-table b7)
(clear b8) (on b8 b9) (on b9 b10) (on b10 b11) (on b11 b12) (on b12 b13) (on b13 b14)
(on b14 b15) (on b15 b16) (on b16 b17) (on b17 b18) (on b18 b19) (on b19 b20) (on-table
b20)))) )
```

## C How much did it cost?

Moreover, as highlighted in Table 2, these performances are not correlated with the various LLM costs. Indeed, overall, looking at the aggregated costs for the whole experiments, we cannot say that most expensive models lead to best results, see for instance the cases of Claude-3-Sonnet (\$118) and Qwen2.5-72B-Instruct (\$8.8) which have similar scale of performances. However, we can see that **within** an LLM-family, the more expensive models usually allow for better results, see *e.g.* OpenAI’s models where GPT-4o is strictly better than the other two.

It is important to notice that the costs mentioned in Table 2 were computed at the time of the experiments *i.e.* in November 2024. Therefore, they may not be accurate anymore, nevertheless, we still think that the orders of magnitude can be interesting to compare at-a-glance the ratio performances / costs for the different families and models.

## D Supplementary Material

Associated with this article, we provide a Github archive containing additional material available from <https://github.com/dgraux/PDDL-COLM-2025>. This archive mainly contains the benchmarks we used and the obtained results from the 20 LLMs we reviewed. The hierarchy is as follows:

- README.md
- Domain/
  - | – action\_gen\_raw.json
  - | – action\_gen\_score.csv
- Problem/
  - | – planetarium\_test\_raw.json
  - | – problem\_gen\_score.csv
- Plan/
  - | – plan\_gen\_raw.json
  - | – plan\_gen\_score.csv

Table 2: Costs (US\$) of the 3 experiments, NA when model-API is free.

LLM	Cost
<i>GPT-3.5-turbo</i>	13.28
<i>GPT-4o-mini</i>	5.16
<i>GPT-4o</i>	83.16
<i>Claude-3-Haiku</i>	12.150
<i>Claude-3-Sonnet</i>	118.25
<i>Gemini-1.5-Pro</i>	46.19
<i>Gemini-1.5-Flash</i>	3.28
<i>Gemma-2-9B-it</i>	1.22
<i>LLaMA-3.1-8B-Instruct</i>	0.89
<i>LLaMA-3.1-70B-Instruct</i>	21.81
<i>LLaMA-3.1-405B-Instruct</i>	86.13
<i>Mistral-7B-Instruct</i>	NA
<i>Codestral</i>	6.68
<i>Mistral-Large2</i>	71.51
<i>Qwen2-1.5B-Instruct</i>	0.32
<i>Qwen2.5-Coder-7B-Instruct</i>	NA
<i>Qwen2.5-7B-Instruct</i>	NA
<i>Qwen2.5-72B-Instruct</i>	8.84
<i>DeepSeek-Coder-V2</i>	3.24
<i>DeepSeek-Chat-V2</i>	3.24