Exploring the Limitations of using Large Language Models to Fix Planning Tasks

Alba Gragera[1] and **Alberto Pozanco**

[1]Computer Science and Engineering Department, Universidad Carlos III de Madrid, Spain
agragera@pa.uc3m.es, alberto.pozanco@gmail.com

Abstract

Large language models (LLMs) have revolutionized natural language processing (NLP), enabling human-like text generation, question answering, and translation. Despite claims of emergent reasoning capabilities, it has been demonstrated their lack of planning skills in tasks such as plan generation, plan reuse or replanning. In this work, we present ongoing efforts on exploring the limitations of LLMs in another task requiring reasoning and planning competences: that of assisting humans in the process of fixing planning tasks.

Introduction

In recent years, large language models (LLMs) have significantly advanced natural language processing (NLP) tasks, such as text generation, question answering, and translation. These models have been trained on vast amounts of text data and have shown remarkable abilities to understand and generate natural language. Despite their inherent limitations, i.e., LLMs learn statistical patterns to predict the most probable next word or sentence, some works claim LLMs might exhibit emergent reasoning capabilities (Bubeck et al. 2023; Kojima et al. 2023). However, recent evidence strongly suggests that current LLMs are poor at tasks requiring planning capabilities such as plan generation, plan reuse or replanning (Valmeekam et al. 2022, 2023).

The planning community uses a standard language to specify planning tasks: the Planning Domain Definition Language (PDDL) (McDermott et al. 1998). This language is domain-independent, so that the code of the planners does not have to be changed to solve problems of different domains. PDDL separates the task definition into two parts: domain and problem. The domain describes the object types, predicates and actions that can be used to solve a task, while the problem defines the objects, initial state and goals of the specific task to be solved. Most AI planners assume an accurate task definition specification, focusing on sequencing a set of actions that can achieve the desired goals when applied. However, formalizing planning tasks in PDDL requires a broad knowledge of the domain, the current task, and the formal language. This process can be complex and prone to errors, which may result in incomplete or inaccurate initial state descriptions and action definitions, which can ultimately render unsolvable planning tasks.

Although LLMs have shown limited performance in reasoning (currently they cannot substitute planners to find a valid plan to achieve the goals), their web-scale knowledge and their utility as code assistants (Feng et al. 2020) suggest that they could assist users during the formalization phase.

Motivated by the recent surge in popularity of LLMs, in this work we present ongoing efforts on exploring the limitations of using LLMs to model planning tasks. While previous works have focused on using LLMs to model PDDL tasks from natural language descriptions (Liu et al. 2023), we are interested in determining whether LLMs can assist users to fix unsolvable planning tasks resulting from incorrect task specifications. In particular, we explore how good LLMs are at repairing planning tasks when the prompt is given in PDDL and when it is given in natural language. For both cases, we consider first incomplete initial states from which the goal cannot be reached, and then incomplete domains which lack a necessary action effect to achieve the goals. In all cases, LLMs are used as stand-alone, and we directly assess the correctness of the solutions it generates.

Throughout the remainder of the paper, we demonstrate that although LLMs can in principle facilitate iterative refinement of PDDL models through user interaction, their limited reasoning abilities render them insufficient for identifying meaningful changes to ill-defined planning models that result into solvable planning tasks.

Background

Automated Planning (AP) tasks define problems whose solutions are sequences of actions, called plans, that achieve the problem goals when applied to a specific initial state. We use the first-order (lifted) planning formalism, where a classical planning task is a pair $\Pi = \langle D, I \rangle$, where D is the *planning domain* and I defines a *problem instance*. A planning domain is a tuple $D = \langle \mathcal{H}, \mathcal{P}, \mathcal{A} \rangle$; where $\mathcal{H}$ is a type hierarchy; $\mathcal{P}$ is a set of predicates defined by the predicate name and the types of its arguments; and $\mathcal{A}$ is a set of action schemas. If $p(t) \in \mathcal{P}$ is an n-ary predicate, and $t = t_1, \ldots, t_n$ are either typed constants or typed free variables, then $p(t)$ is an atom. An atom is grounded if its arguments do not contain free variables. Action schemas $a \in \mathcal{A}$ are tuples $a = \langle name(a), par(a), pre(a), add(a), del(a) \rangle$, defining the action name; the action parameters (a finite set of free variables); the precondition ($pre(a)$), a set of literals representing what must be true or false in a state to apply the action; and $add(a)$ and $del(a)$, that represent the changes produced in a state by the application of the action (added and deleted atoms, respectively). A problem instance is a tuple $I = \langle \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{O}$ is a set of typed constants representing problem-specific objects; $\mathcal{I}$ is the set of ground atoms in the initial state; and finally, $\mathcal{G}$ is the set of ground atoms defining the goals.

Related Work

AP assumes well-defined planning tasks that are solvable. However, producing complete model descriptions can be challenging and time-consuming, and there may be scenarios where neither completeness nor correctness of the planning task specification can be ensured (Kambhampati 2007; McCluskey, Vaquero, and Vallati 2017). These issues can result in an incomplete specification of the initial state or actions, rendering the planning task unsolvable. Providing users with comprehensive explanations that (i) outline the reasons why the planning task does not have a solution; and (ii) guide users in how to fix the initial state / domain to turn the task solvable, is a important research challenge for the planning community (Fox, Long, and Magazzeni 2017; Chakraborti, Sreedharan, and Kambhampati 2020). Regarding incomplete initial state specifications, (Sreedharan et al. 2019) propose to assist the user in understanding why a given AP task is infeasible by identifying unreachable subgoals of the problem. Since it is challenging to extract subgoals from a unsolvable problem, the authors derive them from abstract and solvable models of the given task by using planning landmarks (Hoffmann, Porteous, and Sebastia 2004). Given an unsolvable initial state, it is also possible to provide several alternatives that can transform it into a solvable one. This approach is addressed in (Göbelbecker et al. 2010) where, based on counterfactuals theory (Ginsberg 1985), the authors explain why a plan fails and what can be done to prevent it, creating *excuse states* from which the given task is solvable.

Fixing planning tasks in cases where the error lies in the domain model is not trivial due to the number of potential changes to the set of actions (Lin and Bercher 2021). Most works assume guidelines from users, often in the form of a suggested valid plan that helps in modifying the original set of actions (Simpson, Kitchin, and McCluskey 2007; Nguyen, Sreedharan, and Kambhampati 2017; Lin, Grastien, and Bercher 2023). More recently, (Gragera et al. 2023) removed this assumption, being able to repair incomplete domain models without user inputs by compiling the unsolvable task into a new extended task that includes operators to fix any domain action.

As we can see, fixing initial states or domains to turn planning tasks solvable is far from trivial, and is an active research field within the planning community. Now we will try to understand to what extent LLMs can help to alleviate the human effort in fixing planning models.

Experimental Setting

We use CHATGPT at its March 14, 2023 version. We select four domains from the IPC, namely GRIPPER, TRANSPORT, BLOCKSWORLD and BARMAN, with 3, 3, 4 and 12 lifted actions, respectively. All domains contain descriptive action and predicate names that should be easy to interpret by LLMs. We structure the evaluation as follows.

PDDL Prompts. The prompt consists of an lifted domain description, followed by an instance of a planning problem. For each domain, we generate 5 problems of increasing difficulty (Seipp, Torralba, and Hoffmann 2022), and provide CHATGPT with each one, along with the corresponding domain. The prompts are designed to ask CHATGPT to solve two different tasks: fix incomplete initial states, and fix incomplete domains. In the former, we randomly delete one to three facts from the initial state, making sure that deleting these facts results in an unsolvable planning task, and ask CHATGPT to provide an alternative PDDL initial state that will make the task solvable. For example, we omit the *handempty* predicate from the initial state in BLOCKSWORLD. In the latter task, we randomly delete one action effect from one of the actions in the domain, making sure that deleting the effect results in an unsolvable planning task, and ask CHATGPT to provide an alternative PDDL domain that will make the task solvable. For example, we omit the *holding* predicate from the effects of the *pick-up* action in BLOCKSWORLD. When we delete predicates from the initial state, we preserve the correct domain. Likewise, when we delete effects from the actions in the domain, we preserve the original problem.

Natural Language Prompts. The prompt consists of a natural language domain description, followed by a natural language description of an instance of a planning problem. For each domain, we select only one of the 5 tasks we previously defined in PDDL, formulate the domain and problem in natural language, and ask CHATGPT to solve the same two tasks as before: fix incomplete initial states and fix incomplete domains.

We mainly aim to investigate how CHATGPT responds to PDDL prompts, as we believe this is the most realistic and useful setting for planning practitioners. On the input side, modeling and debugging PDDL tasks is typically easier compared to their natural language counterparts due to inherent ambiguity in the latter. More importantly, while checking the validity and solvability of a PDDL task is straightforward (requiring only a call to the planner), doing the same with a task described in natural language is very time consuming, if not impossible. However, one might conjecture that CHATGPT has not seen many examples of planning tasks described in PDDL, and natural language prompts are presumed to be more amenable for CHATGPT, reason why we also formulate the tasks in natural language, as done by previous works (Valmeekam et al. 2022). In this way, we aim to observe if CHATGPT can identify the flaw from the narrative and provide a comprehensive description of the domain without omitting the crucial detail that renders the task unsolvable. We provide quantitative metrics for the PDDL prompts, while we will just report a qualitative analysis in the case of natural language prompts. In the case of PDDL outputs, we use the *seq-opt-lmcut* configuration of Fast-Downward (Helmert 2006) to try to solve the PDDL tasks returned by CHATGPT. We report two metrics: whether the new tasks compile or not, meaning whether they are correctly parsed by the planner; and whether the new tasks have a solution or not, meaning if the provided solution renders the tasks solvable. All domains, problems, prompts and outputs are available at https://github.com/albagragera/LLMs-planning-repair.

PDDL Prompts

In this section we explore the use of LLMs as modeling assistants when the incomplete (unsolvable) task is inputted in PDDL and the output is required to be a new (hopefully solvable) PDDL task.

Incomplete Initial States

The template prompt is shown in Listing 1. We provide the LLM with a lifted definition of a planning task, followed by an explanation about the unsolvability of the task due to missing information in the initial state. Then, we request the LLM to propose a reformulation, also in PDDL, of the initial state that would make the problem solvable. We understand that the most straightforward way to achieve this is to directly include the goals in the initial state. To avoid this type of solution, we introduce in the prompt a final statement to encourage alternative (non-trivial) approaches. We repeat this process 5 times (5 problems) per domain, using the same domain and changing the problem as well as the fact removed from it.

Listing 1: Incomplete initial state user prompt (PDDL).

```
Consider this domain description:
[PDDL DOMAIN DESCRIPTION]

And this problem associated to it:
[INCOMPLETE PDDL PROBLEM DESCRIPTION]

The planning task is unsolvable due to an incomplete
initial state description. Can you provide an
alternative initial state in PDDL from which the goals
are achievable? This state must not include the goals
directly.
```

The results over the 20 planning tasks are shown in Table 1. In all cases the output was a complete PDDL problem or initial state, of which 84% compiled and just 50% made the planning task solvable. CHATGPT manages to fix the initial state in most BLOCKSWORLD and GRIPPER tasks. We conjecture this good performance can be explained by two factors. First, the problem sizes in these domains were small, with initial states described by few predicates. Second, CHATGPT has likely been trained on some BLOCKSWORLD descriptions (as it is the archetype of planning problems), thus being able to retrieve initial state patterns (such as the hand being empty). On the other hand, CHATGPT struggled with TRANSPORT and BARMAN. In BARMAN, which comprises the most complex tasks, CHATGPT consistenlty created predicates that were not defined in the PDDL domain such as *mixed* or *handright*, which prevented the task to compile. An excerpt of one of these outputs is shown in Listing 2. In TRANSPORT domain, CHATGPT was sometimes able to add the missing predicates to the initial state. However, it also deleted information such as the position of packages, thus keeping the planning task unsolvable.

Domain	Compile	Solvable
GRIPPER	4	4
TRANSPORT	4	1
BLOCKS	5	5
BARMAN	3	0
#Total	16	10

Table 1: Results show the number of alternative problems returned by the LLM that compiled and made the task solvable. The scores are over a total of 20 tasks (5 tasks per domain).

Listing 2: Excerpt of a new initial state in BARMAN as generated by CHATGPT.

```
 ....
  ; Additional facts to make the goal achievable
  (contains cocktail1 ingredient1)
  (contains cocktail1 ingredient2)
  (used shaker1)
  (handright right)
  (handleft left)
  (at shaker1 left)
  (at shot1 right)
  (at dispenser1 right)
  (at dispenser2 left)
  (dispensed dispenser1)
  (dispensed dispenser2)
  (mixed shaker1)
  (shaker-level shaker1 l2)
)
```

Incomplete Domains

The template prompt is shown in Listing 3. Following a similar process, we provide the LLM with a lifted definition of a planning task, followed by an explanation about the unsolvability of the task. In this case, we state that the domain is missing some action effects essential to achieve the goal. We requested the LLM to provide a PDDL domain description for the same domain using STRIPS classical planning, but rewritten to make the task solvable. We emphasize the word *same* used in the prompt because in previous tests where we requested alternative domains, CHATGPT generated solutions for completely different domains such as one to buy fruits, which is definitely not what we were looking for. We repeat this process 5 times (5 problems) per domain, using for each problem a different version of the domain, with different effects removed from it.

Listing 3: Incomplete domain user prompt (PDDL).

```
Consider this domain description:
[INCOMPLETE PDDL DOMAIN DESCRIPTION]

And this problem associated to it:
[PDDL PROBLEM DESCRIPTION]

The planning task is unsolvable because some domain
actions are missing effects. Can you  provide the same
domain in PDDL using STRIPS classical planning, but
including new action effects that would make the task
solvable?
```

The results over the 20 planning tasks are shown in Table 2. As expected, CHATGPT struggles in repairing domains more than in repairing initial states, being able to only turn solvable 3 out of the 20 tasks. We conjecture this bad performance can be explained again by two factors. First, repairing PDDL actions is a more challenging task (Lin and Bercher 2021). Second, the number of tokens in the incomplete initial states was lower than the number of tokens in the incomplete domains. CHATGPT only performs acceptably in the GRIPPER domain, which is the smallest among the domains. Common errors that we observed included omitting the definition of types (resulting in compilation errors) or incorrectly deleting other effects necessary for the task to be solvable. We present some notable incorrect fixes proposed by CHATGPT for the BARMAN domain in Listing 4, where it suggests to introduce a new *make-cocktail* action that is syntactically wrong. Effects use object types not defined in the parameters, and include the negation of nested predicates. The original predicate *(contains ?c - container ?b - beverage)* is also reformulated in a semantically incorrect way as *(contains ?c - cocktail ?x - container)*.

Domain	Compile	Solvable
GRIPPER	3	2
TRANSPORT	1	1
BLOCKS	0	0
BARMAN	0	0
#Total	4	3

Table 2: Results of alternative domains returned by the LLM that compiled and made the task solvable. The scores are over a total of 20 tasks (5 tasks per domain).

Listing 4: CHATGPT output that generates a new Make-cocktail action in BARMAN

```
(:action make-cocktail
    :parameters (?c - cocktail ?x - container ?i1 ?i2 -
        ingredient)
    :precondition (and (contains ?c ?x) (holding ?i1 ?h)
         (holding ?i2 ?h))
    :effect (and (contains ?c ?x) (not (contains (make-
        cocktail) ?x)) (not (holding ?i1 ?h)) (not (
        holding ?i2 ?h))))
)
```

Natural Language Prompts

In this section we explore the use of LLMs as modeling assitants when the incomplete (unsolvable) task is inputted in natural language and the output is required to be a new natural language description of the fixed task. First, we use CHATGPT itself to generate a draft of the domain and problem in natural language from the PDDL description, that we manually verify and refine. The prompts to do so are shown in Listings 5 and 6. Listing 7 shows an excerpt of the resulting output generated by CHATGPT for the BLOCKSWORLD domain.

Listing 5: Prompt to generate natural language description of a domain from its PDDL definition.

```
Consider this domain in PDDL:
[PDDL DOMAIN DESCRIPTION]

Give me an informal description of the domain following
this template:
- Summary of the domain
- Available actions
- Restrictions over actions
```

Listing 6: Prompt to generate natural language description of a problem from its PDDL definition.

```
And this problem instance in PDDL, associated to it:
[PDDL PROBLEM DESCRIPTION]

Give me an informal description of the problem following
this template:
- Objects
- Initial state
- Goals
```

Listing 7: BLOCKSWORLD domain description generated by ChatGPT.

```
The "blocks" domain is a classic problem in Artificial
Intelligence where we have a set of blocks that can be
stacked and unstacked to form towers. The objective is
to create a particular configuration of blocks by moving
them around. The blocks can be picked up, put down,
stacked on top of each other, or unstacked from each
other.

Available actions:
The domain provides four actions: pick-up, put-down,
stack, and unstack. The "pick-up" action allows the
agent to pick up a block that is on the table and has
no other block on top of it...
```

Initially, we attempted to introduce the PDDL domain description with missing information, such as the effect of *holding* in the *pick-up* action, to directly obtain an incomplete natural language description. However, we found that in most cases the domain explanation went beyond simply describing the information represented in the given PDDL domain, and it provided a complete and accurate description instead. This is not surprising given that LLMs only provide the most likely text completion for a given prompt based on the training data, so even if the prompt is not entirely accurate, the functionality of actions such as *pick-up* can still be perfectly described by interpreting its meaning. Therefore, we had to manually supervise and remove some information from the generated output. After that, we reintroduce the description into CHATGPT and ask for modifications to the domain in order to make it complete and accurate.

For this analysis, we select just one of the five problems used in the previous evaluation to see if natural descriptions are more suitable for the LLM to fix the planning tasks. Since the generated output is also presented in natural language, it cannot be solved or compiled, so the results will be discussed qualitatively.

Incomplete Initial States

We introduce the generated informal planning task description generated by CHATGPT in a similar prompt that the one used in the PDDL case. The template prompt is shown in Listing 8. We omit essential information from the use case description as follows. For the GRIPPER task, we removed the location of the *ball2* from the initial state. Similarly, in the TRANSPORT task, we deleted paths between cities from the problem. In the BLOCKSWORLD task, we removed the effect *handempty* from the initial state. Finally, in the BARMAN task, we deleted *empty shot1* from the initial state. In total, we repeat this process 4 times, one per domain.

Listing 8: Incomplete initial state user prompt (natural language).

```
Consider this domain description:
[NATURAL LANGUAGE DOMAIN DESCRIPTION]

And this problem associated to it:
[INCOMPLETE NATURAL LANGUAGE PROBLEM DESCRIPTION]

The resulting planning task is unsolvable due to an
incomplete initial state description. Can you provide
an alternative initial state in natural language from
which the goals are achievable? This state must not
include the goals directly. Be as concise as possible.
```

Listing 9: Natural language description for an alternative solvable initial state in GRIPPER generated by CHATGPT.

```
An alternative initial state that makes the goals
achievable could be:

Ball2 is located in "roomb" and all other balls are
located in "rooma".
```

From the 4 planning tasks, CHATGPT manages to provide a solvable initial state for the GRIPPER and BARMAN tasks. Listing 9 shows the reformulation of the state for the GRIPPER task, where we removed the location of *ball2* and it provides an accurate explanation. On the other hand, CHATGPT cannot return valid initial states for the TRANSPORT and BLOCKSWORLD tasks. For example, the BLOCKSWORLD task is unsolvable due to the hand not being empty. However, CHATGPT does not realize this in its output, and just propose another unsolvable initial state where it re-arranges the blocks without emptying the hand.

Incomplete Domains

In this case, we use as prompt the planning tasks in natural language by omitting certain information from the domain descriptions. The template prompt is shown in Listing 10. For the GRIPPER domain, we removed the description of the gripper state as *free* in the *drop* action. Similarly, in the TRANSPORT domain, we deleted the package as *in* the truck in the *pick-up* action. In the BLOCKSWORLD domain, we removed the effect *holding* from the *pick-up* action. Finally, in the BARMAN domain, we deleted the effect *handempty* from the *leave* action. In total, we repeat this process 4 times, one per domain.

Listing 10: Incomplete domain user prompt (natural language).

```
Consider this domain description:
[INCOMPLETE NATURAL LANGUAGE DOMAIN DESCRIPTION]

And this problem associated to it:
[NATURAL LANGUAGE PROBLEM DESCRIPTION]

The resulting planning task is unsolvable because some
domain actions are missing some effects. Can you provide
the same domain description in natural language, but
including new action effects that will make the task
solvable? Be as concise as possible.
```

In all cases, CHATGPT is able to fix the domain by providing complete descriptions of the actions. Just as it occurred with the generation of the natural language description from the PDDL code, this is not entirely unexpected, since the names of the actions and predicates retain their semantic meaning. For example, it is able to fix the *pick-up* action by accurately describing it because pick-up has an English meaning. We hypothesized that removing direct meanings from predicates and actions would prevent the LLM from utilizing common-sense knowledge in generating repairs or explanations. This was demonstrated in (Valmeekam et al. 2023), where they employed a version of the MYSTERY BLOCKSWORLD domain created by Drew McDermott. They altered the names of predicates and actions to unrelated entities, resulting in a decrease in the number of successful plans generated by the LLMs in the MYSTERY BLOCKSWORLD compared to the original BLOCKSWORLD.

Although in this case CHATGPT was able to provide complete descriptions, the explanations provided are not entirely clear and require users to dive into the narrative. Listing 11 shows an example of the modifications proposed for the BARMAN domain. Only one of the changes is correct, the one regarding the *leave* action, which sets the hand as free to make the task solvable. The other modifications are not necessary, since the domain was originally defined in that way.

Listing 11: Natural language description for an alternative domain that makes the problem solvable in BARMAN generated by CHATGPT.

```
To make the planning task solvable, the following
effects need to be added to the actions:

    grasp: the barman is holding the container
    leave: the container is on the table and the hand is
    empty
    fill-shot: the shot glass is filled with the
    ingredient and the dispenser is empty
    refill-shot: the shot glass is refilled with the
    ingredient and the dispenser is not empty
    empty-shot: the shot glass is empty
    clean-shot: the shot glass is clean and not used
    pour-shot-to-clean-shaker: the shaker is filled with
    the shot glass contents and is clean
    pour-shot-to-used-shaker: the shaker is filled with
    the shot glass contents and has been used to make a
    cocktail
    empty-shaker: the shaker is empty, unshaked, and
    shaker-empty-level is set to 1

With these new effects, the planning task becomes
solvable.
```

Conclusions and Future Work

This paper presents an exploratory study on the usefulness of LLMs as assistants to fix ill-defined planning tasks. Our preliminary results show that, far from their performance as code assistants, current LLMs do not provide much assistance to repair planning models regardless of the way in which they are specified. We conjecture this occurs due to two main reasons. First and most important, LLMs cannot plan (Valmeekam et al. 2022, 2023), and fixing planning tasks often require planning competences. Secondly, while LLMs have been trained in vast amounts of code examples, they have not seen many planning tasks, especially described in PDDL. Interestingly, even when the outputs were wrong, CHATGPT did not make many syntactic mistakes, respecting most of the PDDL syntax and even adding comments to its changes. These changes were also supported with some kind of natural language explanations. However, these explanations did not aid in understanding the solution, as in most cases they did not correspond to the model's modifications.

The evaluation we conducted in the paper is limited and preliminar, and we would like to improve it in future work. For example, we did not focus on prompt engineering or interactive refinement of outputs, which has been shown to improve the performance of LLMs on different tasks. We would also like to explore the performance of LLMs in fixing planning tasks if trained with PDDL domains, problems and plans. Finally, we would like to understand the performance of these models in assisting humans with simpler (but still time consuming) tasks such as generating PDDL domains and problems from natural language (Xie et al. 2023).

Acknowledgements

This work has been partially funded by PID2021-127647NB-C21 and PDC2022-133597-C43 projects, MCIN/AEI/10.13039/501100011033/ and by "ERDF A way of making Europe". Also by the Madrid Government under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17) in the context of the V PRICIT (Regional Programme of Research and Technological Innovation).

References

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; and Zhang, Y. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2020. The Emerging Landscape of Explainable Automated Planning & Decision Making. In *Proceedings of IJCAI 2020*, 4803–4811. ijcai.org.

Feng, Z.; Guo, D.; Tang, D.; Duan, N.; Feng, X.; Gong, M.; Shou, L.; Qin, B.; Liu, T.; Jiang, D.; et al. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. *CoRR*, abs/1709.10256.

Ginsberg, M. L. 1985. Counterfactuals. In Joshi, A. K., ed., *Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985*, 80–86. Morgan Kaufmann.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming Up With Good Excuses: What to do When no Plan Can be Found. In *Proceedings of ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 81–88. AAAI.

Gragera, A.; Fuentetaja, R.; Garcia-Olaya, A.; and Fernandez, F. 2023. A Planning Approach to Repair Domains with Incomplete Action Effects. In *Proceedings of ICAPS 2023, Prague, Czech Republic (To Appear)*.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *J. Artif. Intell. Res.*, 22: 215–278.

Kambhampati, S. 2007. Model-lite Planning for the Web Age Masses: The Challenges of Planning with Incomplete and Evolving Domain Models. In *Proceedings of AAAI 2007, July 22-26, 2007, Vancouver, British Columbia, Canada*, 1601–1605. AAAI Press.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2023. Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916.

Lin, S.; and Bercher, P. 2021. Change the World - How Hard Can that Be? On the Computational Complexity of Fixing Planning Models. In *Proceedings of IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 4152–4159. ijcai.org.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *Proceedings of AAAI 2023, Washington, USA*.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. LLM+ P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint arXiv:2304.11477*.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering Knowledge for Automated Planning: Towards a Notion of Quality. In *Proceedings of K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, 14:1–14:8. ACM.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL–the planning domain definition language–version 1.2. Technical report, Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational ....

Nguyen, T.; Sreedharan, S.; and Kambhampati, S. 2017. Robust planning with incomplete domain models. *Artif. Intell.*, 245: 134–161.

Seipp, J.; Torralba, Á.; and Hoffmann, J. 2022. PDDL Generators. https://doi.org/10.5281/zenodo.6382173.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning domain definition using GIPO. *Knowl. Eng. Rev.*, 22(2): 117–134.

Sreedharan, S.; Srivastava, S.; Smith, D. E.; and Kambhampati, S. 2019. Why Can't You Do That HAL? Explaining Unsolvability of Planning Tasks. In *Proceedings of IJCAI 2019, Macao, China, August 10-16, 2019*, 1422–1430. ijcai.org.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). arXiv:2206.10498.

Valmeekam, K.; Sreedharan, S.; Marquez, M.; Olmo, A.; and Kambhampati, S. 2023. On the Planning Abilities of Large Language Models (A Critical Investigation with a Proposed Benchmark). arXiv:2302.06706.

Xie, Y.; Yu, C.; Zhu, T.; Bai, J.; Gong, Z.; and Soh, H. 2023. Translating Natural Language to Planning Goals with Large-Language Models. arXiv:2302.05128.