# PDDL Domain Repair: Fixing Domains with Incomplete Action Effects

**Alba Gragera, Raquel Fuentetaja, Ángel García-Olaya, Fernando Fernández**

Computer Science and Engineering Department, Universidad Carlos III de Madrid, Spain
agragera@pa.uc3m.es, {rfuentet, agolaya, ffernand}@inf.uc3m.es

## Abstract

Automated planning is a problem solving technique for a wide range of scenarios and goals, which typically involves the creation of domain and problem files in formal languages. However, producing complete model descriptions can be challenging and time-consuming, and errors such as incomplete specification of the initial state or the set of actions often result in unsolvable tasks for planners. Explaining the absence of a solution in such cases is essential to support humans in the development of automated planning tasks. In this paper, we present an tool to repair planning models where the effects of some actions are incomplete. The received input is compiled to a new extended planning task, in which actions are permitted to insert possible missing effects. The solution is a plan that achieves the goals of the original problem while also alerting users of the modifications made.

## Introduction

Automated planning tasks are typically defined by a domain description, which specifies all available actions and predicates, and a problem description that contains the initial state and goals. However, modeling a planning task is not always trivial, and there may be scenarios where the completeness of the planning task specification cannot be ensured (Kambhampati 2007; McCluskey, Vaquero, and Vallati 2017). In such cases, an incomplete specification of the initial state or actions can render the planning task unsolvable.

Providing users a comprehensive explanation about the absence of solution and how to solve it is an important challenge for the planning community (Fox, Long, and Magazzeni 2017; Chakraborti, Sreedharan, and Kambhampati 2020). Previous works have considered scenarios where initial states prevent the achievement of goals (Göbelbecker et al. 2010; Sreedharan et al. 2019), providing explanations and alternative initial states that render the task solvable. However, these works assume the proper specification of the domain and do not consider modifications to it. Due to the number of potential changes to the set of actions, repairing faulty domains is not trivial (Lin and Bercher 2021). Previous works assume guidelines from users, often in the form of a suggested valid plan (Nguyen, Sreedharan, and Kambhampati 2017; Lin, Grastien, and Bercher 2023). In contrast, in this work we propose the use of automated planning to repair planning tasks themselves, without additional information from the user. We focus on errors in the domain model that render the planning task unsolvable. Specifically, we consider missing action effects, which can compromise the task's solvability.

We present a tool where users can upload domain and problem files. If the domain is flawed and no solution exists, the interface provides suggestions to repair it. This is achieved by compiling the unsolvable task into a new extended planning task that includes operators to link predicates as new effects of the actions. The resulting plan achieves the original goals while also provides information on how the model was repaired to make the task solvable.

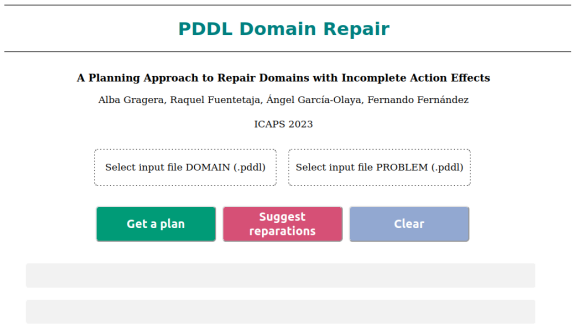


Figure 1: PDDL Domain Repair interface.

## Input

The tool receives domain and problem PDDL files. If the planning task is fully specified and solvable, the 'Get plan' button returns the solution plan. If the provided domain has flaws that make the task unsolvable, clicking 'Suggest reparations' launches the compilation of the task, as explained below. Let us consider a BLOCKSWORLD planning domain that has the following missing effects:

- *holding* effect from the *pick-up* action
- *on* effect from the *stack* action
- *holding* effect from the*unstack* action

We will use this example in the remainder of the paper.

## Compilation to Classical Planning

To identify modifications that make the task solvable, we compile the received domain and problem into a new planning task that includes additional operators to repair any flawed action in the domain. At the operational level, each grounded action is now divided into three stages: (1) the application of the action, (2) any necessary repairs, and (3) the closure of the action.

To manage this process, we reformulate the planning task elements so that action and predicate names become domain objects. In addition, we introduce new predicates to replace the original ones. These predicates are designed to access the elements of the original task and to control the repairing process. The new set of action schemes is explained below. Further details about the compilation can be found in the original paper (Gragera et al. 2023).

**ACTIONS FROM ORIGINAL ACTIONS** Consists of actions generated from the original domain actions, but compiled to match the new object representation.

**FIX ACTIONS** We include repair operators to select a predicate symbol and link it as a new effect of any action. This operator has an associated cost, which will be used as a bias to minimize the number of reparations made in the domain.

After a predicate symbol is linked to an action as a new effect, we include the following two actions to establish whether the effect will be positive or negative.

**ADD-FIX ACTIONS** These operators perform the reparation as a positive effect by matching the predicate symbol with its parameters, and adding it to the state with the appropriate objects.

**DEL-FIX ACTIONS** These operators follow a similar action scheme as the ADD-FIX operators, but they remove atoms from the current state. This simulates a negative effect of the action to which it was linked.

**CLOSE ACTION** The application of a close action concludes the reparation of an action. It deletes the information about the current action and the objects used, and updates the action as already fixed.

**PROBLEM INSTANCE** The original problem instance is also compiled to match the new object representation. The new predicates are instantiated to represent static domain information about the original predicates and actions.

## Output

The solution to this extended planning task consists of a plan that achieves the original goals by repairing the actions of the original domain with additional effects. Figure 2 shows the resulting solution of the extended planning task compiled from the BLOCKSWORLD task input, where the repair actions are highlighted.

To obtain the interface output, we parse the solution plans to show users only the suggested repairs. We filter the FIX actions to display the repaired actions and the predicates used in the repairs. These reparations ensure that the task is solvable. The interface is illustrated in Figure 3.

```
(unstack b4 b3)
(fix____adding_different holding unstack)
(add-fix_____1par holding unstack b4 t_block)
(completed_fixed unstack)
(put-down b4)
(completed_nofixed put-down)
(unstack b3 b2)
(add-fix_____1par holding unstack b3 t_block)
(completed_fixed unstack)
(stack b3 b4)
(fix____adding_different on stack)
(add-fix_____2par_goal on stack b3 b4 t_block t_block)
(completed_fixed stack)
(pick-up b1)
(fix____adding_different holding pick-up)
(add-fix_____1par holding pick-up b1 t_block)
(completed_fixed pick-up)
(stack b1 b3)
(add-fix_____2par_goal on stack b1 b3 t_block t_block)
(completed_fixed stack)
```

Figure 2: Solution plan for the BLOCKSWORLD domain.

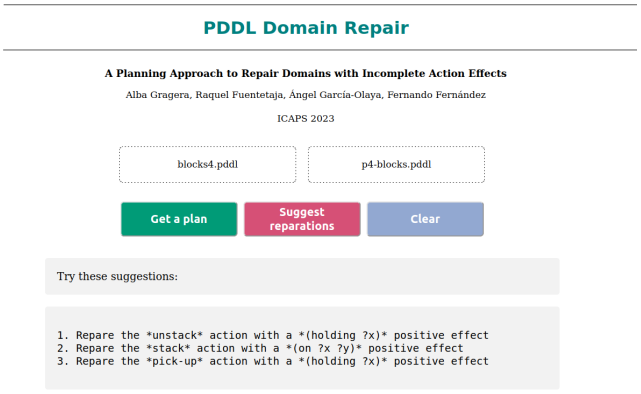


Figure 3: Suggested reparations as shown in the interface.

## Discussion

There are many changes that can be made to a domain to make the planning task solvable. To avoid trivial or unwanted repairs (such as directly adding goals to action effects), we bias the search by assigning different costs to each repair action. We define a metric to minimize such costs, guiding the search towards more desirable plans. However, the lack of information about the number and location of flaws, as well as the user's mental model, can lead to estimated repairs.

One strength of our approach is that it can obtain a fairly accurate reparation without requiring additional information from the user, only a domain and a single problem. However, this may also have the drawback of generating reparations that are over-fitted to the given problem, compromising the ability to generalize. We believe that incorporating multiple problem instances or automatically generating problems to identify possible flaws in the domain pose interesting challenges that can motivate future research.

## 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

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.

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

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)*.

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.

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*.

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.

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

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.