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

Motivation

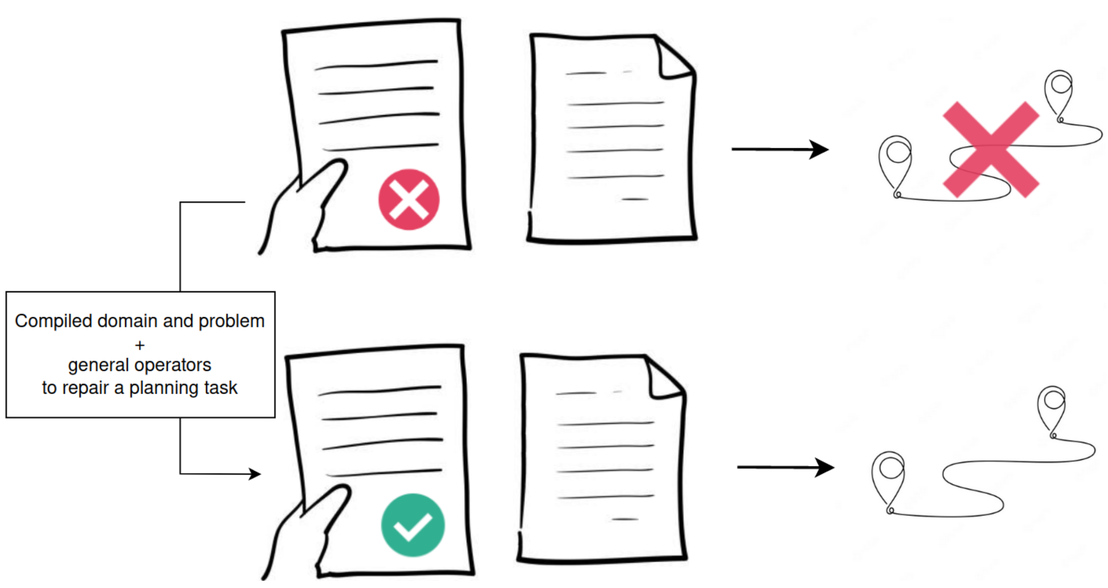
- Automated planning typically assumes an accurate task specification, defined by a domain and a problem description, specified in PDDL
- Modeling a planning task is complex and error-prone, requiring broad knowledge of the domain, the current task, and the formal language
- An incomplete model may render the planning task unsolvable

Explaining the absence of a solution in such cases is essential to support humans in the development of automated planning tasks.

Approach to repair flawed domains

The unsolvable task is compiled into a new planning task where:

- Actions can be repaired to **insert possible missing effects**
- The solution is a plan that achieves the goals of the original problem while at the same time repairs the original actions



Input Example

A Blocksworld planning domain with the following missing effects:

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

```
(:action pick-up
   :parameters (?x - block)
   :precondition (and (clear ?x) (ontable ?x) (handempty))
   :effect  (and (not (ontable ?x))(not (clear ?x))(not (handempty))))
```

Classical Planning Compilation

- New predicates to represent the planning task elements
- Control predicates to manage the repair process
- New actions to modify the original actions: fix, add-fix, del-fix, close
- Internal process to repair an action: open → link new effects → close
- Bias to guide the reparation using action costs

```
(:action fix____adding_different
  :parameters (?p - predicate ?a - action)
  :precondition (and (current-action ?a)
                     (functor ?p)
                     (not (checked ?a))
                     (not (patched ?a))
                     (not (add_eff ?p ?a))
                     (not (del_eff ?p ?a))
                     (open))
  :effect (and (fix ?a ?p)
               (patched ?a)
               (increase (total-cost) X)))
```

```
(:action add-fix_____1par
  :parameters (?p - predicate ?a - action
                     ?o - objectdomain ?v - typedomain)
  :precondition (and (current-action ?a)
                     (fix ?a ?p) (pred_1var ?p ?v)
                     (type ?v ?o) (used ?o)
                     (not (used ?p))
                     (not (del_added ?p ?a))
                     (not (goal_1var ?p ?o))
                     (not (in_state1 ?p ?o)))
  :effect (and (already-fixed)
               (in_state1 ?p ?o)
               (used ?p)))
```

Output Example

The result of the compiled planning task is a plan that includes the reparations made in the domain to achieve the goals.

```
(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)
```

The plan is parsed and shown in the interface as repair suggestions to the original domain...

PDDL Domain Repair

A Planning Approach to Repair Domains with Incomplete Action Effects

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

ICAPS 2023

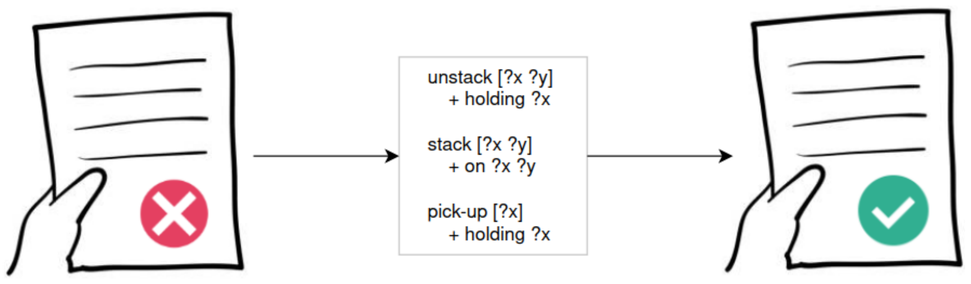
blocks4.pddl p4-blocks.pddl

Get a plan Suggest reparations Clear

Try these suggestions:

```
1. Repare the *unstack* action with a *(holding ?x)* positive effect
2. Repare the *stack* action with a *(on ?x ?y)* positive effect
3. Repare the *pick-up* action with a *(holding ?x)* positive effect
```

... ensuring that they make the task solvable.



Conclusions

- A fairly accurate reparation without requiring additional information from the user, only a domain and a single problem
- The lack of information about the number and location of flaws, as well as the user's mental model, can lead to estimated repairs