Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains

Songtuan Lin Alban Grastien Pascal Bercher

School of Computing
College of Engineering, Computing and Cybernetics
The Australian National University

February 2023

Australian National University

Motivation and Objective

The task of modeling a planning domain is a major obstacle for deploying AI planning techniques more broadly.

- We need tools for modeling assistance!
  - E.g., Planning.Domains, itSIMPLE, plugin(s) for Visual Studio Code, etc.
- We want to repair a flawed planning domain.

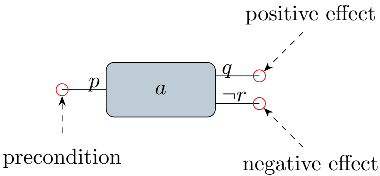
Objective

- **Inputs**: A flawed planning domain.
  A (set of) plan(s) contradicting the flawed domain but demanded to be valid.
- **Output**: A *cardinality-minimal* repair set to the domain that turns each plan into a solution.

Background

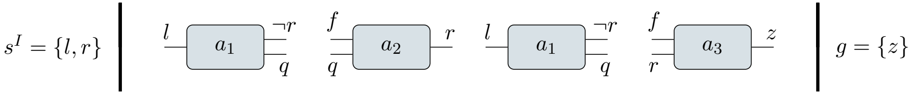
A planning problem is $\Pi = (\mathcal{D}, s^I, g)$ where $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \alpha)$ is the domain of Π.

- $\mathcal{F}$: A finite set of propositions.
- $\mathcal{A}$: A finite set of action names.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$ mapping each action name to its precondition and effects.



- $s^I \in 2^{\mathcal{F}}$: The initial state.
- $g \subseteq \mathcal{F}$: The goal description.

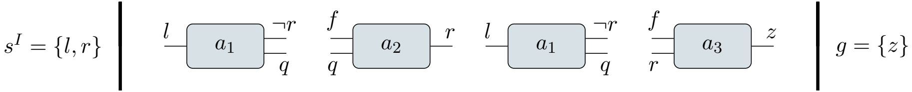
Atomic Repairs



For each action $a \in \mathcal{A}$, we define the atomic repairs:

- $\langle F_a|^p_f \rangle$: Removing the proposition f from a's precondition.
    - E.g., $\langle F_{a_3}|^p_r \rangle$ removes r from a_3's precondition.
- $\langle F_a|^-_f \rangle$: Removing the proposition f from a's negative effects.
    - E.g., $\langle F_{a_1}|^-_r \rangle$ removes r from a_1's negative effects.
- $\langle F_a|^+_f \rangle$: Adding the proposition f to a's positive effects.
    - E.g., $\langle F_{a_2}|^+_f \rangle$ adds f to a_2's positive effects.

Approach



Solution

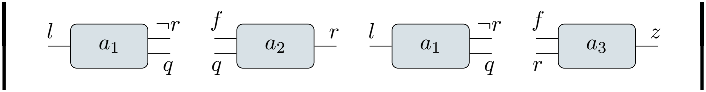
A cardinality-minimal set $\delta^* \subseteq F_\Pi$ with F_Π being the set of all atomic repairs that turns the plan into a solution.

We maintain a set of sets (of repairs) Θ^* which is initially empty.

- On each iteration, we find a set θ of repairs in which at least one *must* be applied for making the plan valid. Then, we add θ into Θ^*.
  - Such a θ is called a *conflict*.
- A *hitting set* δ of Θ^* is a set such that $\delta \cap \theta \neq \emptyset$ holds for every $\theta \in \Theta^*$. δ^* denotes a *minimal* hitting set of Θ^* (minimal number of repairs).

Computing a Conflict

$s^I = \{l, r\}$

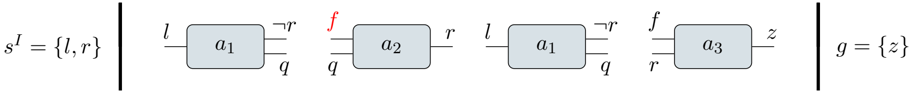


$g = \{z\}$

Given a plan $\langle a_1 \cdots a_n \rangle$, let a_j be the first action in the plan having a precondition f that is unsatisfied.

- $\langle F_{a_j}|^p_f \rangle$ is in the conflict.
- For each $i < j$ in descending order:
  - If a_i deletes f, $\langle F_{a_i}|^-_f \rangle$ is in the conflict, and we can stop the computation.
  - Otherwise, $\langle F_{a_i}|^+_f \rangle$ is in the conflict.
- E.g., $\left\{ \langle F_{a_2}|^p_f \rangle, \langle F_{a_1}|^+_f \rangle \right\}$ is a conflict.

Example



Iteration #1

- $\Theta^* = \emptyset$
- $\delta_1 = \emptyset$
- $\theta_1 = \left\{ \begin{matrix} \langle F_{a_2}|^{p}_{f}\rangle \\ \langle F_{a_1}|^{+}_{f}\rangle \end{matrix} \right\}$
- $\Theta^* = \{\theta_1\}$

Iteration #2

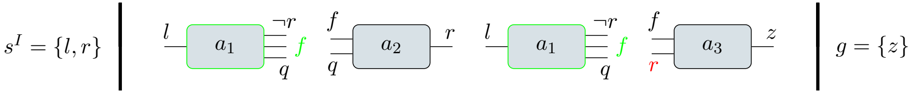
- $\Theta^* = \{\theta_1\}$
- $\delta_2 = \langle F_{a_1}|^{+}_{f}\rangle$
- $\theta_2 = \left\{ \begin{matrix} \langle F_{a_3}|^{p}_{r}\rangle \\ \langle F_{a_1}|^{-}_{r}\rangle \end{matrix} \right\}$
- $\Theta^* = \{\theta_1, \theta_2\}$

Iteration #3

- $\Theta^* = \{\theta_1, \theta_2\}$
- $\delta_3 = \left\{ \begin{matrix} \langle F_{a_1}|^{+}_{f}\rangle \\ \langle F_{a_1}|^{-}_{r}\rangle \end{matrix} \right\}$
- $\Theta^* = \{\theta_1, \theta_2\}$
- Done!

- Each δ_i is a minimal hitting set of Θ^*.
  - $\delta_3 = \delta^*$ is a solution.
- Each θ_i is a conflict.

Example



Iteration #1

- $\Theta^* = \emptyset$
- $\delta_1 = \emptyset$
- $\theta_1 = \begin{Bmatrix} \langle F_{a_2}|_f^p\rangle \\ \langle F_{a_1}|_f^+\rangle \end{Bmatrix}$
- $\Theta^* = \{\theta_1\}$

Iteration #2

- $\Theta^* = \{\theta_1\}$
- $\delta_2 = \langle F_{a_1}|_f^+\rangle$
- $\theta_2 = \begin{Bmatrix} \langle F_{a_3}|_r^p\rangle \\ \langle F_{a_1}|_r^-\rangle \end{Bmatrix}$
- $\Theta^* = \{\theta_1, \theta_2\}$

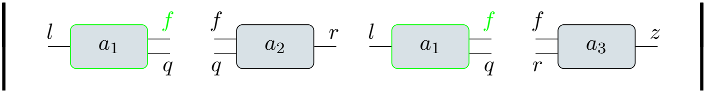
Iteration #3

- $\Theta^* = \{\theta_1, \theta_2\}$
- $\delta_3 = \begin{Bmatrix} \langle F_{a_1}|_f^+\rangle \\ \langle F_{a_1}|_r^-\rangle \end{Bmatrix}$
- $\Theta^* = \{\theta_1, \theta_2\}$
- Done!

- Each δ_i is a minimal hitting set of Θ^*.
  - $\delta_3 = \delta^*$ is a solution.
- Each θ_i is a conflict.

Example

$s^I = \{l, r\}$



$g = \{z\}$

Iteration #1

- $\Theta^* = \emptyset$
- $\delta_1 = \emptyset$
- $\theta_1 = \left\{ \begin{array}{l} \langle F_{a_2}|^p_f \rangle \\ \langle F_{a_1}|^+_f \rangle \end{array} \right\}$
- $\Theta^* = \{\theta_1\}$

Iteration #2

- $\Theta^* = \{\theta_1\}$
- $\delta_2 = \langle F_{a_1}|^+_f \rangle$
- $\theta_2 = \left\{ \begin{array}{l} \langle F_{a_3}|^p_r \rangle \\ \langle F_{a_1}|^-_r \rangle \end{array} \right\}$
- $\Theta^* = \{\theta_1, \theta_2\}$

Iteration #3

- $\Theta^* = \{\theta_1, \theta_2\}$
- $\delta_3 = \left\{ \begin{array}{l} \langle F_{a_1}|^+_f \rangle \\ \langle F_{a_1}|^-_r \rangle \end{array} \right\}$
- $\Theta^* = \{\theta_1, \theta_2\}$
- Done!

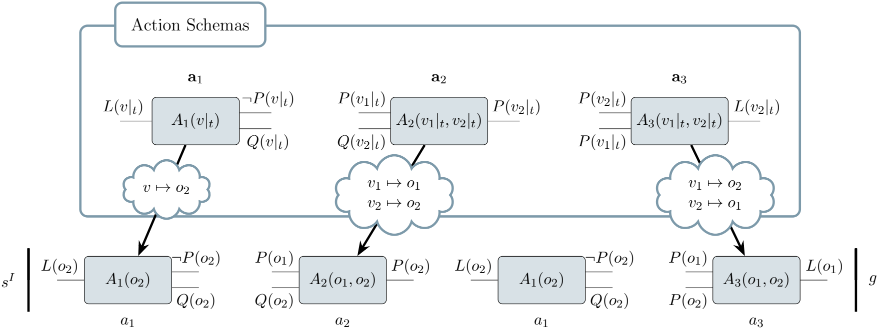
- Each δ_i is a minimal hitting set of Θ^*.
  - $\delta_3 = \delta^*$ is a solution.
- Each θ_i is a conflict.

Lifted Planning Formalism

A *lifted* problem is $\mathbf{\Pi} = (\mathcal{D}, \mathcal{O}, s^I, g)$ with $\mathcal{D} = (\mathcal{P}, \mathcal{A}, \alpha)$:

- $\mathcal{O}$: A set of *objects*.
  - Each object has a *type*.
- $\mathcal{P}$: A set of *predicates*.
  - E.g., $\mathbf{f} = P(v_1|_{t_1}, \cdots, v_n|_{t_n})$ where P is the predicate's name, $v_i \in \mathcal{V}$ is a variable, and t_i is the respective type.
- $\mathcal{A}$: A set of *action schemas*.
  - E.g., $\mathbf{a} = A(v_1|_{t_1}, \cdots, v_n|_{t_n})$ where A is the action's name.
  - α maps each action schema to its precondition and effects, each of which is a set of predicates $\mathbf{f}$.
    - E.g., $\mathbf{f} = P(v_{i_1}|_{t_{i_1}}, \cdots, v_{i_j}|_{t_{i_j}})$ with $i_k \in \{1, \cdots, n\}$ for each $k \in \{1, \cdots, j\}$.
- Substituting each variable with an object of the same type is called grounding.

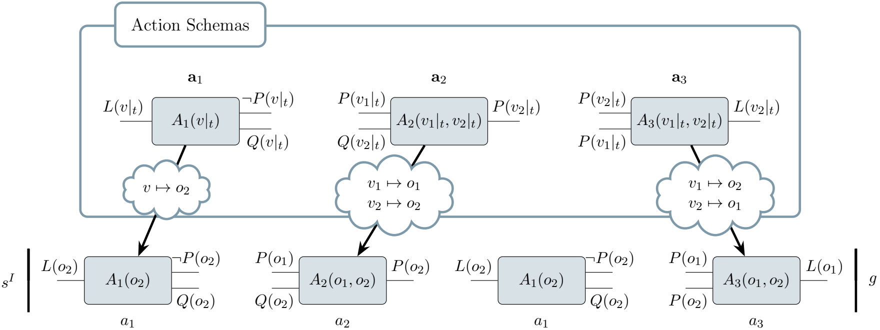
Lifted Domain Repair Problem



For an action schema $\mathbf{a}$, an atomic repair is one of the following:

- $\langle \mathbf{F_a}|_\mathbf{f}^p \rangle$: Removing the predicate $\mathbf{f}$ from $\mathbf{a}$'s precondition.
- $\langle \mathbf{F_a}|_\mathbf{f}^- \rangle$: Removing $\mathbf{f}$ from $\mathbf{a}$'s negative effects.
- $\langle \mathbf{F_a}|_\mathbf{f}^+ \rangle$: Adding $\mathbf{f}$ to $\mathbf{a}$'s positive effects.
    - The parameters of $\mathbf{f}$ must align with those of $\mathbf{a}$.
    - E.g., $\langle \mathbf{F_{a_2}}|_\mathbf{f}^+ \rangle$ with $\mathbf{f} = Q(v_1|_t)$ or $\mathbf{f} = Q(v_2|_t)$

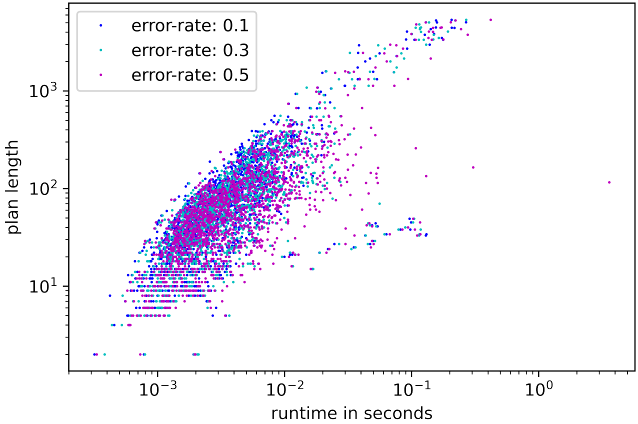
Solving the Lifted Domain Repair Problem



Let a_j be the first action in the plan having a proposition f in its precondition that is unsatisfied, e.g., $f = P(o_1)$ in a_2.

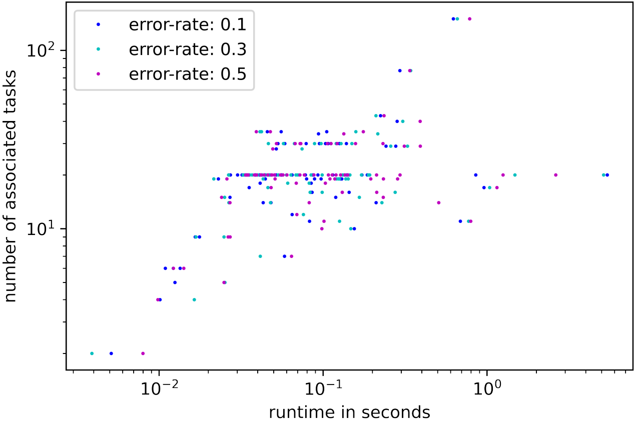
- $\langle \mathbf{F_a}|^p_\mathbf{f} \rangle$ is in the conflict if $\mathbf{a}$ and $\mathbf{f}$ are grounded to a_j and f, respectively. E.g., $\langle \mathbf{F}_{\mathbf{a}_2}|^p_\mathbf{f} \rangle$ with $\mathbf{f} = P(v_1|_t)$.
- $\langle \mathbf{F_a}|^+_\mathbf{f} \rangle$ is in the conflict if $\mathbf{a}$ and $\mathbf{f}$ are grounded to an a_i $(i < j)$ and f, respectively. (Such $\mathbf{a}$ and $\mathbf{f}$ does *not* exist for $\mathbf{a}_1$.)
- Stop at a_i, $i < j$, if there exists $\langle \mathbf{F_a}|^-_f \rangle$ with $\mathbf{a}$ and $\mathbf{f}$ being grounded to a_i and f.

Empirical Evaluation (with Single Plan)



The plot depicts the runtime for repairing domains where one plan is given. (Up to > 1000 plan length.)

Empirical Evaluation (with Multiple Plans)



This plot depicts the runtime for repairing domain where multiple plans are given. (Up to > 100 plans.)

Conclusion

We developed an approach for repairing planning domains:

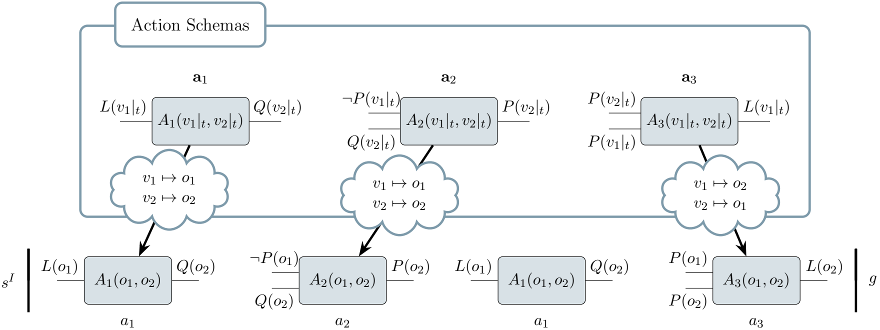
- The approach works for both grounded and lifted planning domains (both shown),
- both with and without negative preconditions (the latter wasn't shown).
- We support repairing a single plan and *sets* of plans.
- **Approach can be used to repair unsolvable problems!**
- Most domains in our benchmark set can be repaired within *one* second.

Future Work

Possible and planned directions of future work are:

- Building an interactive tool for repairing domains. (Plugin for Planning.Domains)
- Extending the approach to support more advanced features, e.g.,
  - negative plans (which plans should be rejected?)
  - change parameters (see first action in lifted example: should we allow to add the effect?)
  - block certain repairs (should we allow adding an effect if it already gets deleted?)
  - and possibly many more!

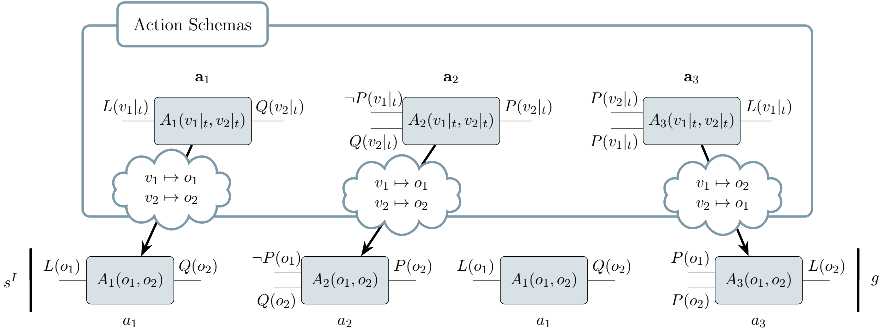
Lifted Domain Repair Problems with Negative Preconditions



For each $\mathbf{a}$, we define the following *additional* atomic repairs:

- $\langle \mathbf{N_a}|^p_{\mathbf{f}} \rangle$: Removing $\mathbf{f}$ from $\mathbf{a}$'s *negative* precondition.
- $\langle \mathbf{N_a}|^-_{\mathbf{f}} \rangle$: *Adding* $\mathbf{f}$ to $\mathbf{a}$'s negative effects.
  - The parameters of $\mathbf{f}$ must align with those of $\mathbf{a}$.
- $\langle \mathbf{N_a}|^+_{\mathbf{f}} \rangle$: *Removing* $\mathbf{f}$ from $\mathbf{a}$'s positive effects.

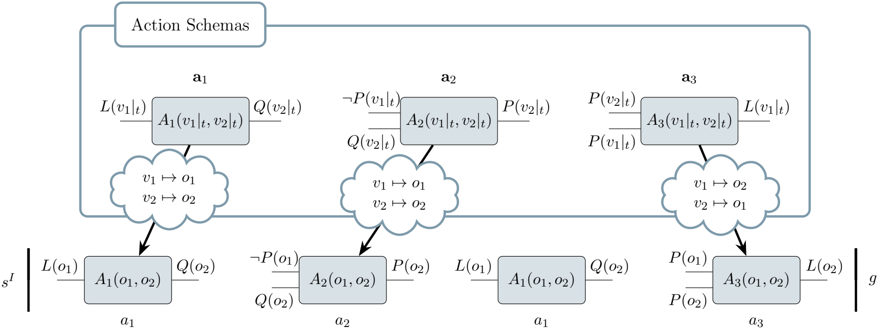
Conditional Conflicts



A repair set is now *not* a hitting set of a conflict set.

- A repair may result in violating some action's precondition.
  - E.g., $\langle \mathbf{F}_{\mathbf{a}_1}|^+_{\mathbf{f}} \rangle$ with $\mathbf{f} = P(v_1|_t)$ will make a_3 applicable and make a_2 *inapplicable*.
- We need to compute *conditional* conflicts (φ, θ).
  - If all repairs in φ are applied to the domain, then at least one repair in θ must be applied.

Computing Conditional Conflicts



We first compute a conflict θ using the same way as before.

- If θ has some repair that undoes a previous applied repair, then we remove it from θ and add it to the condition φ.
- E.g., assuming that $P(v_1|_t)$ is added to $\mathbf{a}_1$'s positive effect.
  - $\theta = \{\langle \mathbf{N}_{\mathbf{a}_2}|^p_{\mathbf{f}}\rangle, \langle \mathbf{N}_{\mathbf{a}_1}|^-_{\mathbf{f}}\rangle\}$ with $\mathbf{f} = P(v_1|_t)$.
  - Removing $\langle \mathbf{N}_{\mathbf{a}_1}|^-_{\mathbf{f}}\rangle$ from θ and adding $\langle \mathbf{F}_{\mathbf{a}_1}|^+_{\mathbf{f}}\rangle$ to φ.
    - $\langle \mathbf{N}_{\mathbf{a}_1}|^-_{\mathbf{f}}\rangle$ undoes $\langle \mathbf{F}_{\mathbf{a}_1}|^+_{\mathbf{f}}\rangle$.