Modeling Assistance for AI Planning: From the Perspective of Model Reconciliation – Dissertation Abstract

Songtuan Lin[1]

Supervisors: Pascal Bercher[1] **& Gregor Behnke**[2] **& Alban Grastien**[1]

[1] **College of Engineering & Computer Science, The Australian National University**

[2] **Institute for Logic, Language and Computation, University of Amsterdam**

[1]**{firstName.lastName@anu.edu.au}** [2]**g.behnke@uva.nl**

Introduction

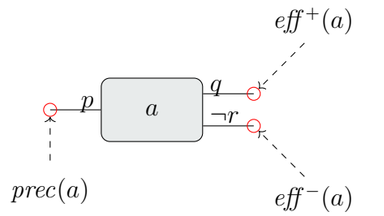
The ultimate goal of the thesis is to provide modeling assistance to human domain modelers. For this purpose, we would like to study two critical questions in the context of both non-hierarchical (classical) planning and hierarchical planning:

Domain Validation: Is a domain correctly modeled?

Domain Repair: How to repair a flawed domain?

Classical Planning Framework

A classical planning problem is a tuple: $\mathcal{P} = (\mathcal{F}, \mathcal{A}, \delta, s_I, g)$:



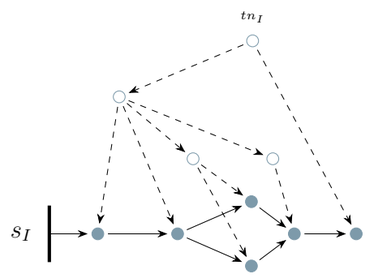
$\delta(a) = (prec(a), eff^+(a), eff^-(a))$

- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of actions
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $s_I \in 2^{\mathcal{F}}$: Initial state
- $g \subseteq \mathcal{F}$: Goal description

An action sequence $\overline{a} = \langle a_1 \cdots a_n \rangle$ (i.e., a plan) is a solution to $\mathcal{P}$ *iff* there exists a state sequence $\pi = \langle s_0 \cdots s_n \rangle$ such that

- $s_0 = s_I$
- $g \subseteq s_n$
- $prec(a_i) \subseteq s_{i-1}$ for each $1 \leq i \leq n$
- $s_i = (s_{i-1} \setminus eff^-(a_i)) \cup eff^+(a_i)$ for each $1 \leq i \leq n$

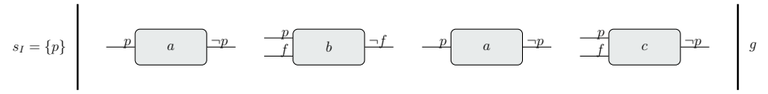
Hierarchical Planning Framework



HTN planning has so-called compound (abstract) tasks on top of actions (primitive tasks), which can be further refined into primitive and compound tasks.
An action sequence is a solution *iff* it is executable in the initial state, **and** it is a refinement of the initial compound task.

Remark: HTN planning subsumes classical planning.

Domain Validation via Plan Validation

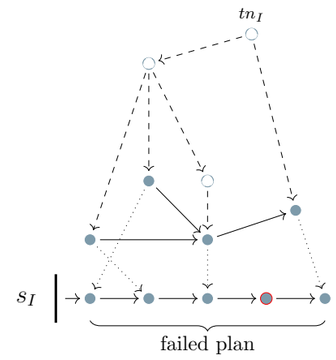


Let a classical planning domain contain the actions a, b, and c whose preconditions and effects are defined as above.

Question: Is this domain correct?

Method: We provide the plan as a test case which is supposed to be a solution to the planning problem.

Answer: NO! Because the plan is not a solution.



Remark: For classical planning, plan validation is trivial. However, plan validation is $\mathbb{NP}$-hard for hierarchical planning.

Contribution: We would like to improve some existing HTN plan validation approaches for the purpose of domain validation.

Domain Repair as Model Reconciliation

Classical Model Reconciliation Setting

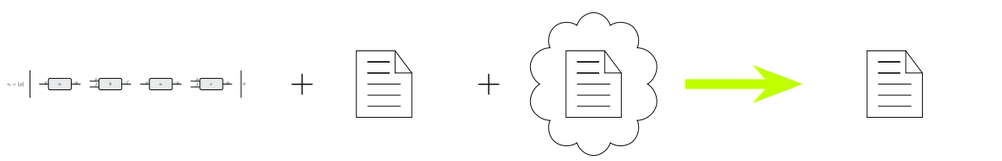
Observed plan π

Human mental model $\mathcal{M}_h^R$:
- π is *not* opt./exe. to $\mathcal{M}_h^R$

Robot's model $\mathcal{M}^R$:
- π is opt./exe. to $\mathcal{M}^R$

Changes to $\mathcal{M}_h^R$:
- π is opt./exe. to $\mathcal{M}^R$
- π is opt./exe. to $\mathcal{M}_h^R$

Model Reconciliation Setting for Domain Repair



Observed plan π

Human modeled model $\mathcal{M}_h^R$:
- π is *not* opt./exe. to $\mathcal{M}_h^R$

Human mental model $\mathcal{M}^R$:
- **Not given**
- π is opt./exe. to $\mathcal{M}^R$

Changes to $\mathcal{M}_h^R$:
- π is opt./exe. to $\mathcal{M}^R$
- π is opt./exe. to $\mathcal{M}_h^R$

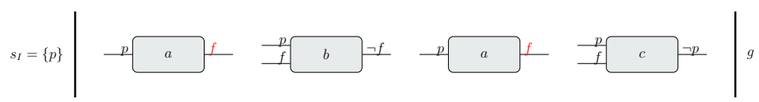
Domain Repair Examples

In non-hierarchical planning, we consider three kinds of repairs:

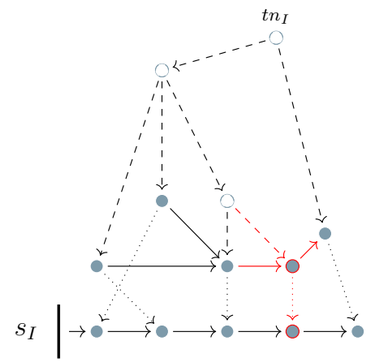
- Adding propositions to actions' positive effects
- Removing propositions from actions' preconditions
- Removing propositions from actions' negative effects

Remark 1: We want to find a **minimal cardinality** set of repairs.

Remark 2: Finding the minimal cardinality set of repairs is $\mathbb{NP}$-hard if adding propositions to actions' positive effects is allowed.



Optimal Repair: Adding f to the positive effects of a



In the context of HTN planning, apart from repairing actions' preconditions and effects, we also consider repairing methods.

Remark 1: Deciding whether a plan can be turned into a solution via repairing an HTN domain is $\mathbb{NP}$-hard.

Remark 2: $\mathbb{NP}$-hardness holds even if we are explicitly given a decomposition hierarchy which is supposed to lead to the plan.

Remark 3: A special case is in $\mathbb{P}$ in which there are **no** duplicated compound tasks in the given decomposition.

Configuration

Input 1: A planning problem

Input 2: A plan that is not a solution to $\mathcal{P}$

Output: A minimal cardinality set of repairs to the planning problem such that the given plan will be a solution.

Remark: The inputs and out specified here are defined for both classical and HTN planning, but what repairs are allowed to use depends on planning frameworks.