Was Fixing this *Really* That Hard? On the Complexity of Correcting HTN Domains

Songtuan Lin & Pascal Bercher

School of Computing
The Australian National University

February 2023

Australian National University

Motivation and Objective

Motivation

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

- Automated assistance for modeling domains is vital!
  - E.g., tools for providing a more user-friendly programming environment.
- We will study the problem of repairing a *flawed* HTN planning domain modeled by a domain modeler.
- Another application of this technology is providing *explanations* for why failed plans are no solutions. (The changes would act as explanation.)

Motivation and Objective

- **Input**: A (flawed) HTN domain, a set of white list plans, and a set of black list plans.
- **Output**: A sequence of changes (repairs) to the domain so that all white list and black list plans will be solutions and non-solutions, respectively.

Objective

We investigate the complexity of repairing a flawed HTN planning domain to turn a set of *white* list plans into *solutions* and a set of *black* list plans into *non-solutions*.

We will identify all hardness sources of the problem.

We do this by gradually factoring each of them out by adding them as additional inputs to the problem.

HTN Planning

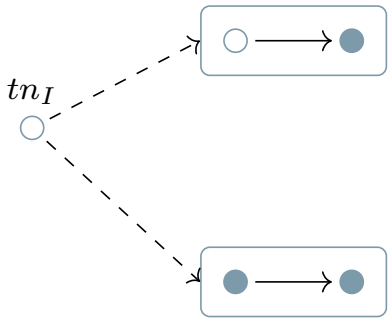
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



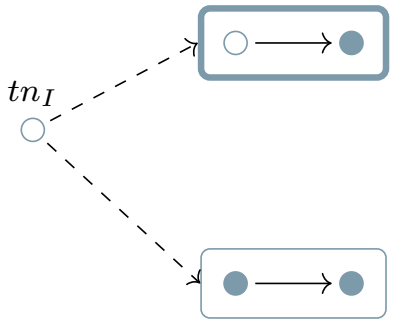
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



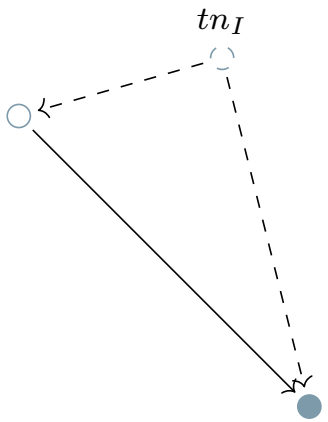
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



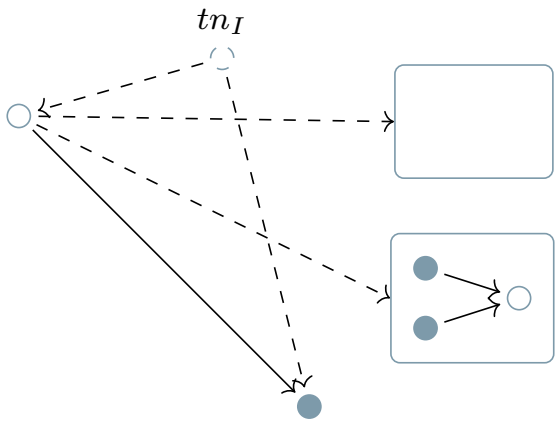
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



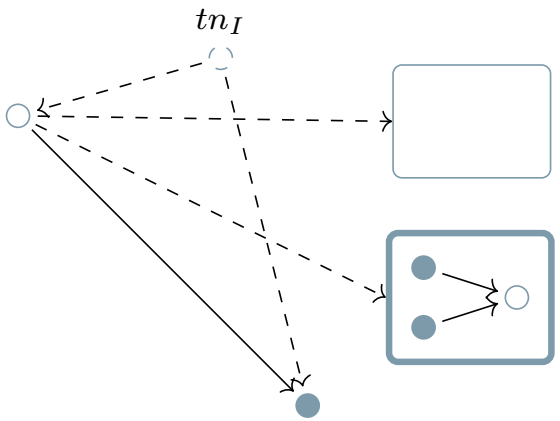
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



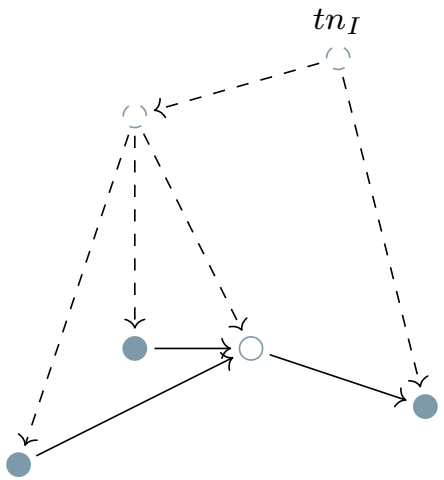
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



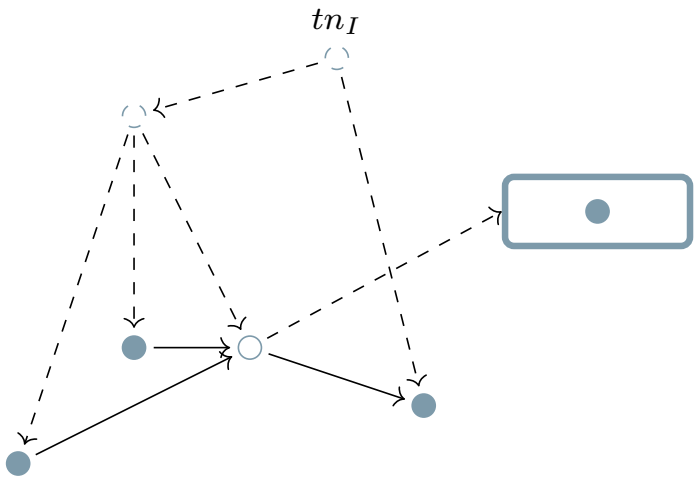
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



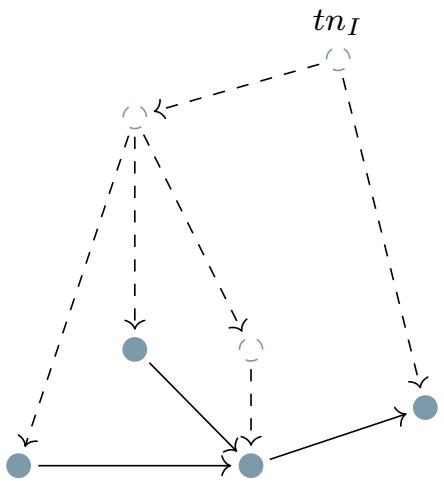
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



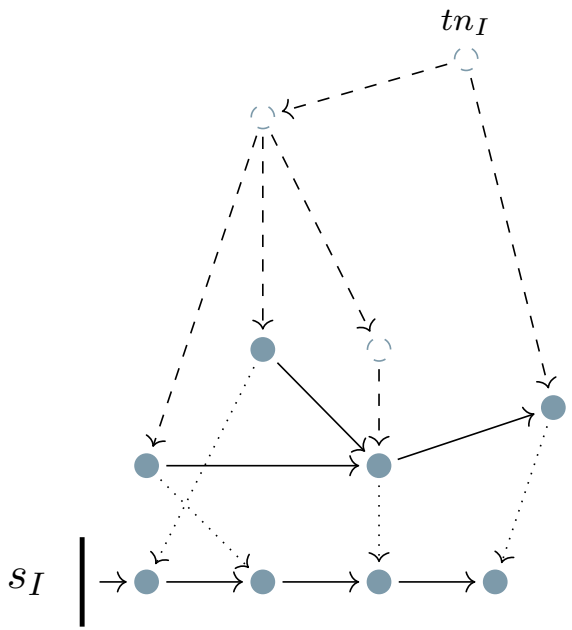
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



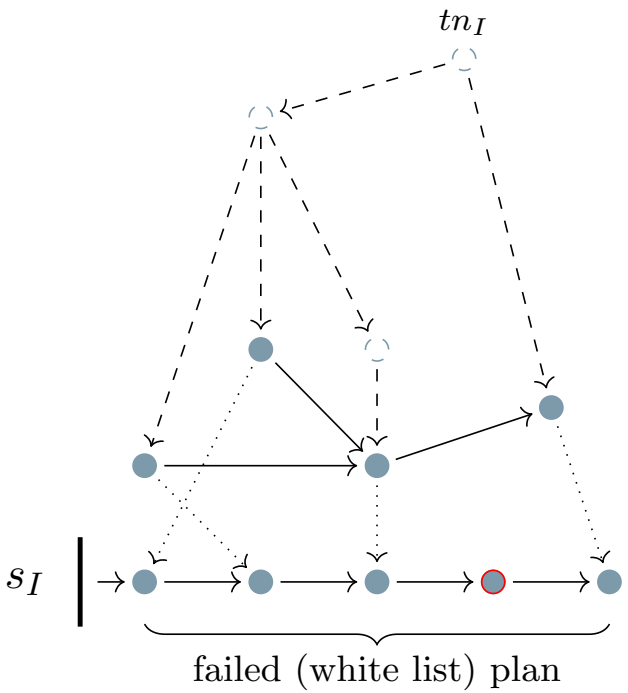
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



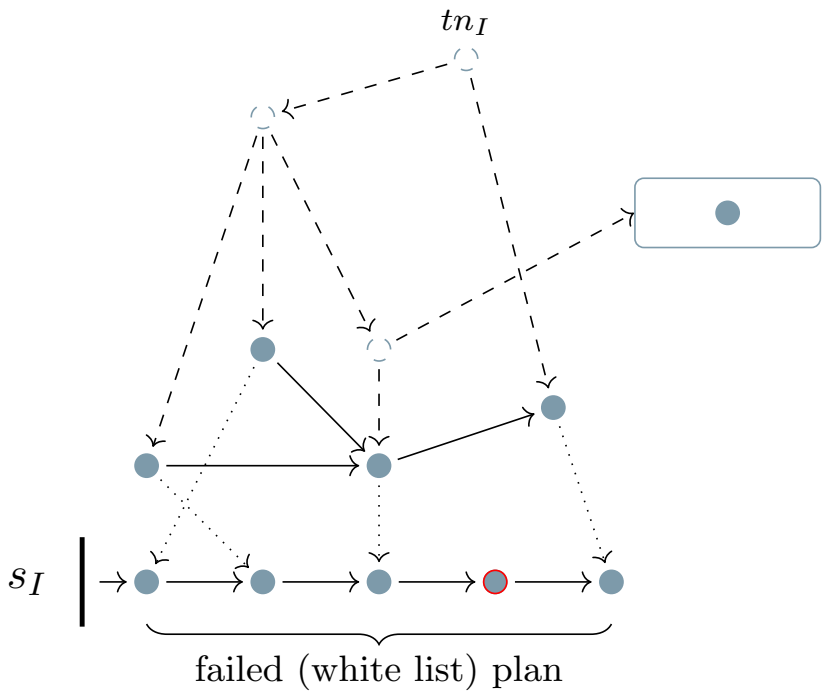
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



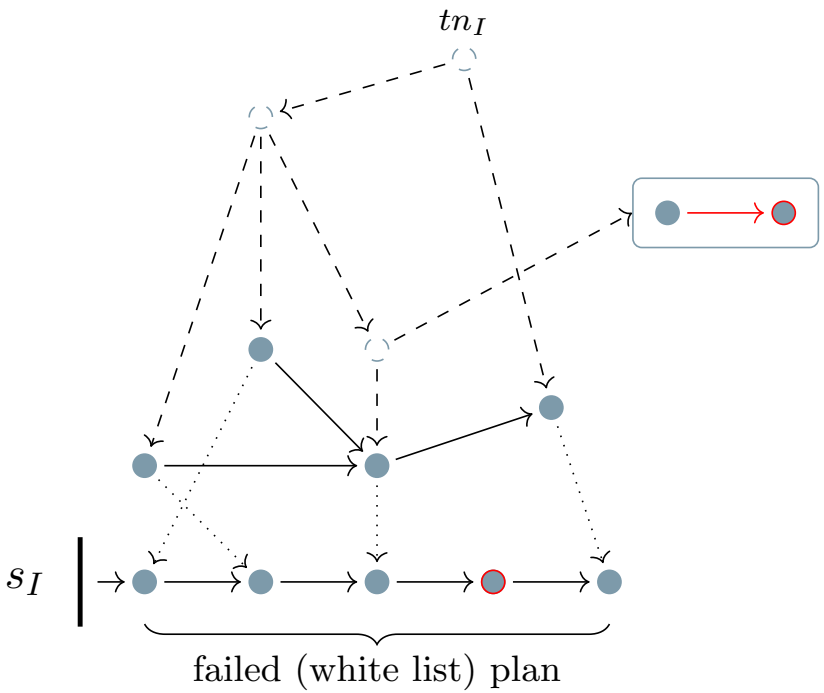
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



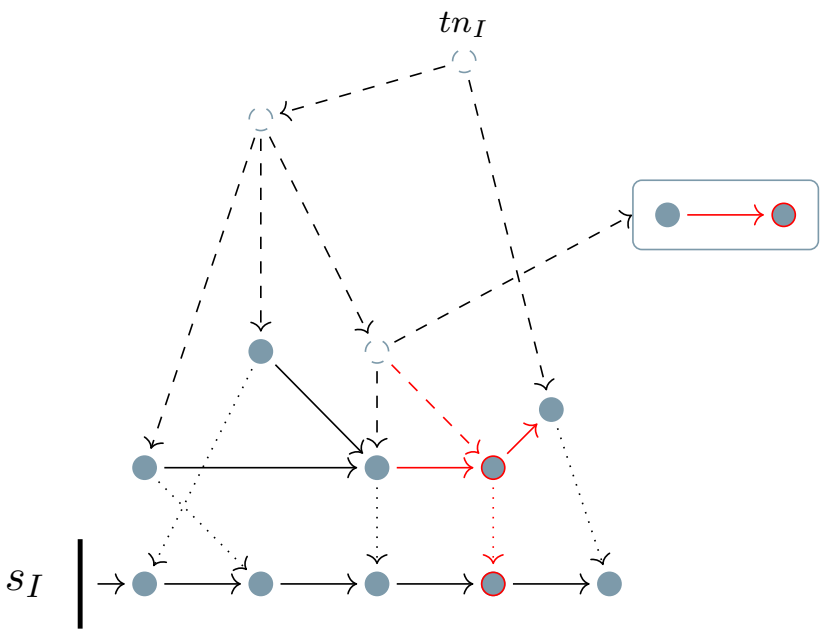
An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

- tn_I: An initial task network.
- s_I: An initial state.

HTN Planning



An HTN domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

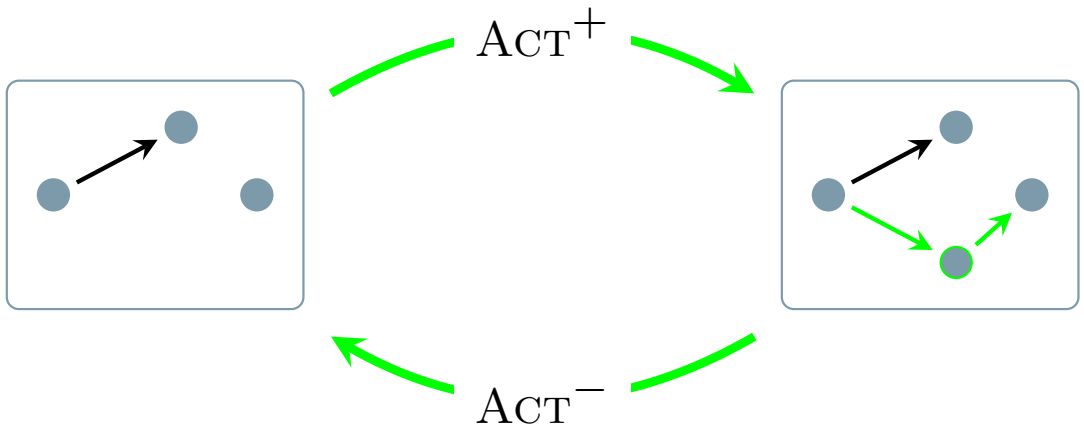
- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

An HTN planning problem $\mathcal{P} = (\mathcal{D}, tn_I, s_I)$:

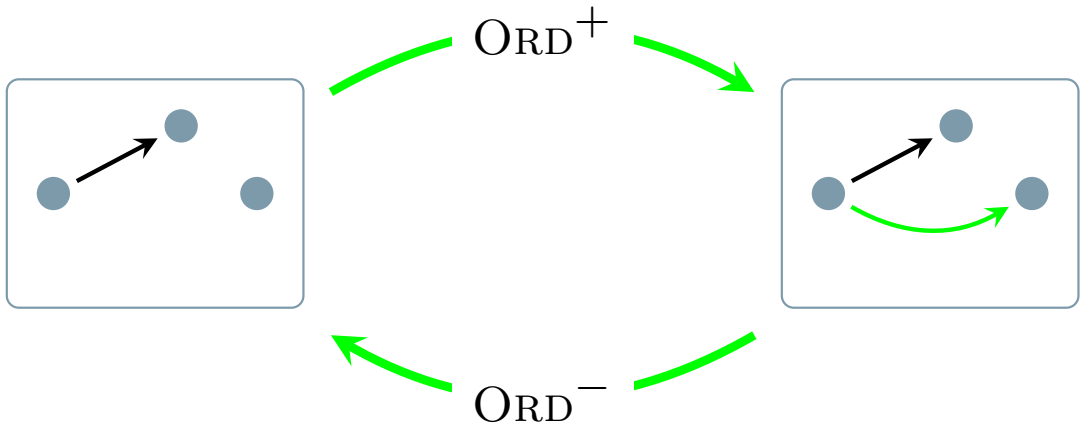
- tn_I: An initial task network.
- s_I: An initial state.

Domain Change Operations

- $\textsc{Act}^+/\textsc{Act}^-$: Adding/removing a primitive task to/from a method together with the respective ordering constraints.



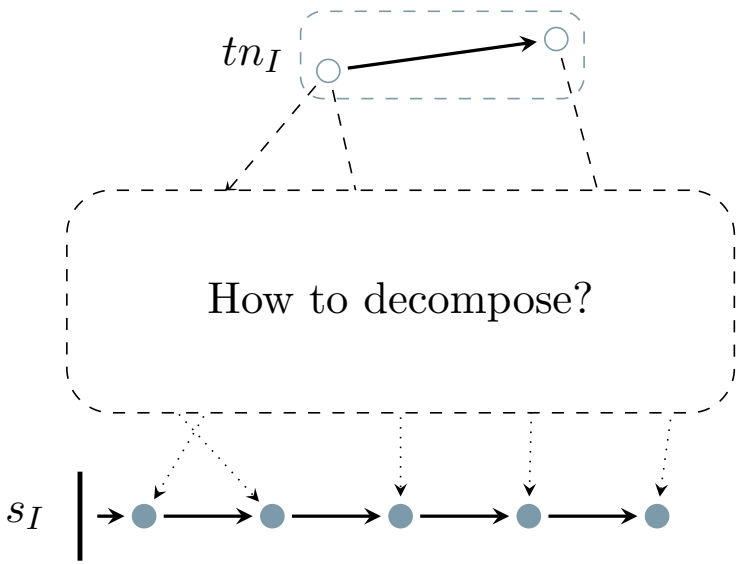
- $\textsc{Ord}^+/\textsc{Ord}^-$: Adding/removing an ordering constraint to/from a method.



Given Only White List Plans: Hardness Source #1

Configuration

- **Inputs**: A set Π of plans.
  An HTN planning problem $\mathcal{P}$.
- **Output**: True if we can change the domain of $\mathcal{P}$ to turn each $\pi \in \Pi$ into a solution, otherwise false.



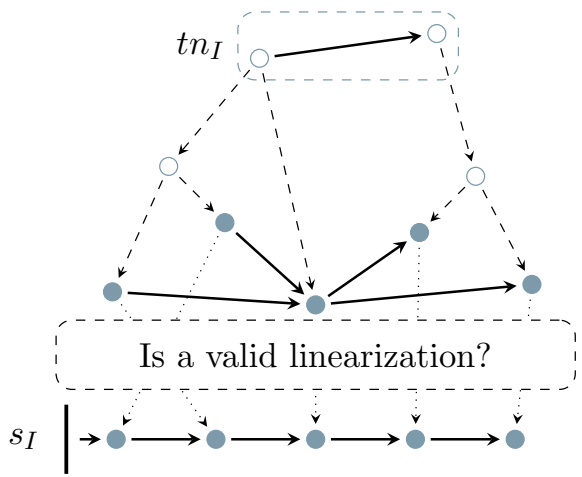
Hardness Source #1: Finding a decomposition hierarchy which leads to a given plan is $\mathbb{NP}$-hard.

- The problem is $\mathbb{NP}$-complete no matter what changes are allowed.

Given Only White List Plans: Hardness Source #2

Configuration

- **Inputs**: A set Π of plans.
  An HTN planning problem $\mathcal{P}$.
  A set Ω of decomposition hierarchies.
- **Output**: True if we can change the domain of $\mathcal{P}$ to turn each plan $\pi \in \Pi$ into a solution, witnessed by a decomposition hierarchy in Ω, otherwise false.



Hardness Source #2: Verifying whether an action sequence is a valid linearization of a primitive task network is $\mathbb{NP}$-complete.

- $\mathbb{NP}$-completeness holds for this configuration no matter what changes are allowed.

Given Only White List Plans: Hardness Source #2

Theorem

Given a *primitive* task network tn and a plan π, deciding whether π is a valid linearization of tn is $\mathbb{NP}$-complete.

(Note that this is closely related to the $\mathbb{NP}$-complete question of determining whether there exists an executable linearization.)

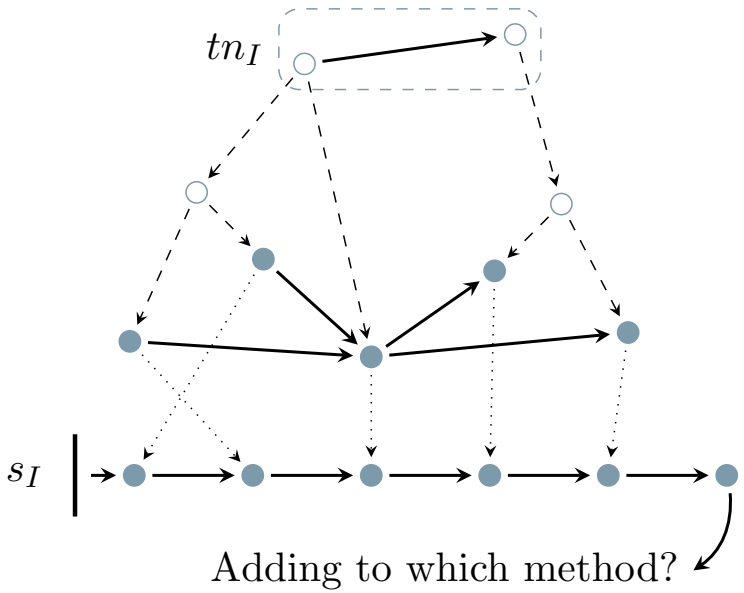
- It serves as a fundamental hardness source for a wide range of problems involving reasoning over partial order. E.g.,
  - for the PlanCompatibility problem,
  - for the PlanVerification problem, and
  - in Partial Order Causal Link (POCL) Planning.

Given Only White List Plans: Hardness Source #3

Configuration

- **Inputs**: A set Π of plans.
  An HTN planning problem $\mathcal{P}$.
  A set Ω of decomposition hierarchies.
  A set Φ of mappings ϱ from the result of a $g \in \Omega$ to the actions in a $\pi \in \Pi$.
- **Output**: True if we can change the domain of $\mathcal{P}$ to turn each plan $\pi \in \Pi$ into a solution, witnessed by a decomposition hierarchy $g \in \Omega$ and a mapping $\varrho \in \Phi$, otherwise false.

Given Only White List Plans: Hardness Source #3



Hardness Source #3: Deciding to which method an action should be added is $\mathbb{NP}$-complete.

- $\mathbb{NP}$-completeness holds for this configuration if $\textsc{Act}^+$ is allowed, otherwise, it is poly-time solvable.

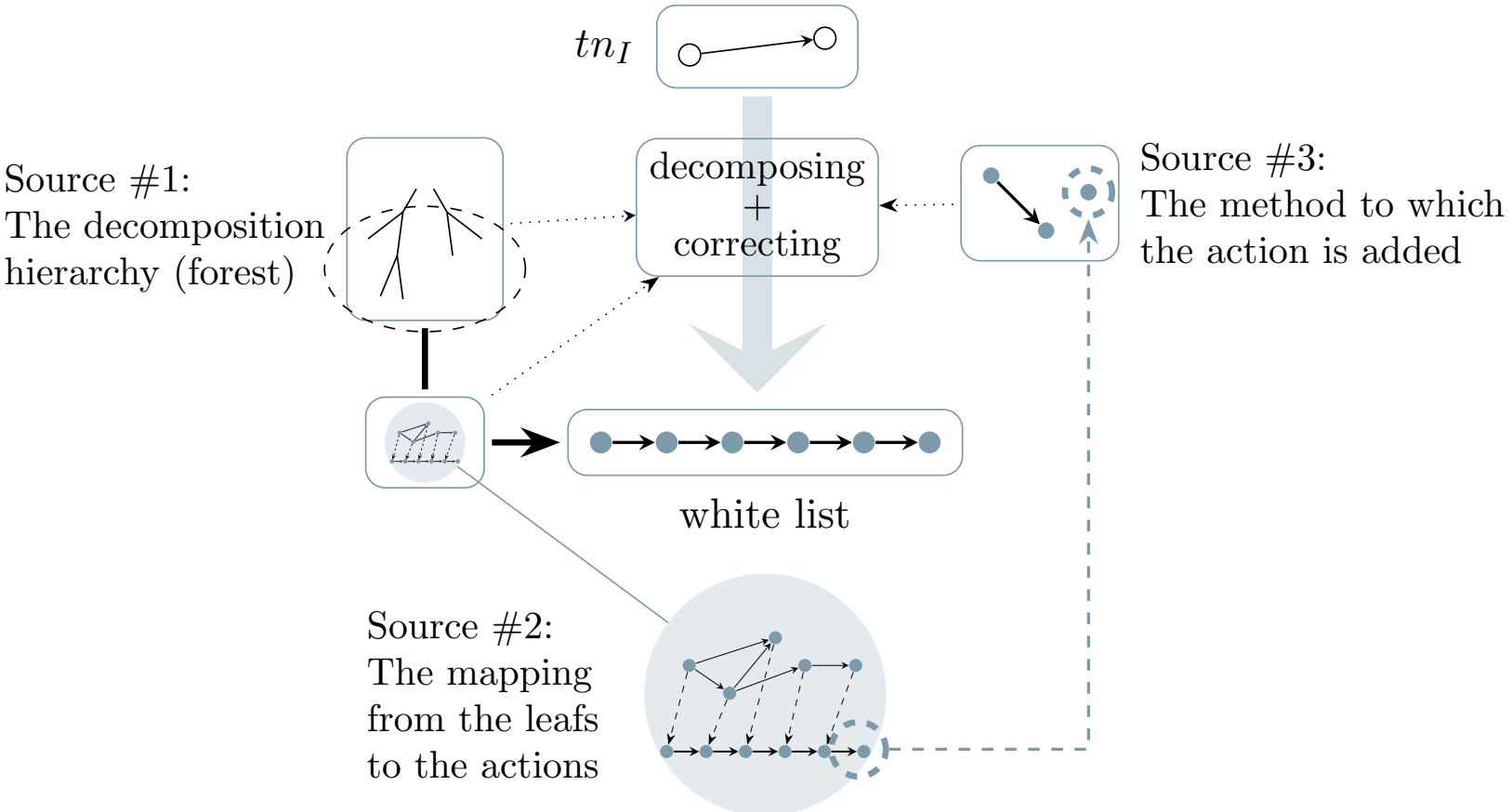
Given White and Black List Plans

Theorem

Given an HTN planning problem $\mathcal{P}$, a set Π^+ of white list plans, and a set Π^- of black list plans, deciding whether each $\pi^+ \in \Pi^+$ can be turned into a solution and each $\pi^- \in \Pi^-$ into a non-solution by changing the domain of $\mathcal{P}$ is $\mathbb{NP}$-hard and is in Σ_p^2.

- Having both white list and black list plans is a generalization of having only white list ones.
  - $\mathbb{NP}$-hardness holds naturally.
- We can assume an oracle machine (a hypothesis machine) deciding whether a plan is a solution to an HTN problem in $\mathcal{O}(1)$ time and develop a non-deterministic algorithm for the problem.
  - This implies Σ_p^2 membership.

Conclusion



- Each source results in $\mathbb{NP}$-hardness.
- When black list plans are also given, the problem is $\mathbb{NP}$-hard but in Σ_p^2.

Future Work – Exploiting the Hardness Sources

Developing practical approaches based on the hardness sources:

- Coping with the hardness source #1.
  - Adapting the existing SAT encoding for searching for a decomposition forest.
- Coping with the hardness source #2.
  - Adapting the SAT encoding for the (sub)graph isomorphism problem.
- Coping with the hardness source #3.
  - We can easily extend the above encoding to incorporate model changes. (E.g., by using a variable x_m^a to indicate whether an action a is added to a method m.)

These ideas base on an exiting HTN plan verifier via SAT. (Though other ideas are possible as well, e.g., by extending an approach that verifies HTN problems via (HTN) planning.)

Future Work – Extension

So far, we only allowed adding/deleting actions and orderings.
Much more is possible!

- Also allow adding/deleting compound tasks.
- Extend the formalism to allow method preconditions (so that we can respect, but also possibly change them)
- Add/delete entire methods.
- Probably several more...