Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Compiling Model Reconciliation Explanation Problems into Stackelberg and FOND Planning Problems

Sarath Sreedharan[1], **Pascal Bercher**[2]
[1]Colorado State University
[2]The Australian National University
ssreedh3@colostate.edu, pascal.bercher@anu.edu.au

Abstract

Despite its popularity, most model reconciliation explanation generation methods rely on blind breadth-first search, and even available heuristics are rudimentary at best. In this paper, we propose two novel approaches to compile the problem of generating bounded model-reconciliation explanations into existing planning formalisms. First, we compile the problem into a Stackelberg planning problem, which is an adversarial problem consisting of a leader and follower agent. Here, the leader agent is responsible for identifying the explanation, while the follower checks the validity of the identified explanation. In the second approach, we see how the same problem can also be converted into a fully observable nondeterministic (FOND) planning problem. Here, the nondeterministic actions are used to generate and test the possibility of a shorter plan. We show the effectiveness of the proposed approaches by comparing them against each other and two existing baselines on standard planning benchmark problems.

Introduction

Model reconciliation explanation remains one of the more popular explanation methods for planning problems (Sreedharan, Chakraborti, and Kambhampati 2021). The method focuses on generating an explanation that addresses the user's misunderstanding about the current planning task. It builds on ideas from cognitive science and psychology (Apperly and Butterfill 2009), and the explanatory method itself has been validated through user studies (Chakraborti et al. 2019; Sreedharan et al. 2024). Despite its popularity, existing methods to generate model reconciliation explanations remain woefully inadequate. The most widely used method remains a search over the space of possible model updates, also referred to as model space search (Sreedharan, Chakraborti, and Kambhampati 2021). Heuristics for the model-space search remain elementary, and most works leverage blind search (Chakraborti et al. 2017). Attempts have been made at leveraging hitting set duality to speed up model reconciliation explanation generation for variations of the problem that focus on resolving inconsistencies between propositional logical knowledge bases (Vasileiou et al. 2022). To the best of our knowledge, there doesn't exist any method that compiles explanation generation into a single problem that an existing solver can directly solve.

Recent work proved that the complexity of the most popular fragment of the model reconciliation explanation in Σ_2^p-complete (Sreedharan, Bercher, and Kambhampati 2022). The results show that the primary computational challenge is to ensure that the identified explanation guarantees the optimality of the plan being explained. In other words, there shouldn't exist a valid plan that is less costly than the target one in the updated model obtained by incorporating the explanation in the human mental model. In this work, we propose two planning compilations that are able to encode the problem in polynomial size. The first one will leverage a Stackelberg planning formulation (Behnke and Steinmetz 2024), which is an adversarial planning formulation for a leader-follower setting. In this case, the leader needs to come up with a plan that can prevent the follower from achieving its goal. In our compilation, the leader is responsible for identifying the explanation, while the follower tries to disprove the optimality of the plan being explained. The next compilation will leverage a fully observable nondeterministic (FOND) planning formulation (Cimatti et al. 2003). Here, the non-determinism is leveraged to iterate over the space of possible cheaper plans, whose validity is tested.

The choice to focus on these two planning formulations presents a classic trade-off within most compilation-based approaches, namely, one of complexity and availability of solvers. While FOND planning is generally understood to be a more general problem than Stackelberg planning complexity-wise (EXPTIME (Rintanen 2004) vs. PSPACE (Behnke and Steinmetz 2024)), there has been much longer research on developing more efficient FOND planners as compared to Stackelberg ones. Thus, a comparison between the two allows us to identify if the effectiveness of modern FOND planners provides them a practical advantage over existing Stackelberg planners.

We will compare the performance of these two compilations, solved using state-of-the-art planners, across a suite of standard planning benchmark problems. We will also compare it against a blind-search-based baseline and the algorithm that leverages hitting set duality. Our preliminary results suggest a clear advantage for Stackelberg planners over other compilation methods. However, we would like to emphasize that the goal of this paper isn't to develop state-of-the-art solvers for the model reconciliation problem, but

to identify novel compilations. As the community makes progress in developing more effective planners, we could expect compilation performance to improve as well.

Background

We start by providing basic definitions for the planning formulations and the model reconciliation problem.

Classical Planning

The plans we will explain will be generated using deterministic planning models described in PDDL syntax with conditional effects (Haslum et al. 2019). Here, a planning model will be described with a tuple of the form $\mathcal{M} = \langle V^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}, I^{\mathcal{M}}, G^{\mathcal{M}} \rangle$. Here, $V^{\mathcal{M}}$ is the set of proposition state variables or fluents that define the state space $S^{\mathcal{M}}$. Each state $s \in S^{\mathcal{M}}$ is represented by a conjunction of literals, such that each fluent appears in each state formula as a positive or negative literal. To simplify the notation and transition function, we will equivalently represent each state as a set of fluents that are true in a given state. $A^{\mathcal{M}}$ is a set of action names; $\delta^{\mathcal{M}} = \langle pre^{\mathcal{M}}, add^{\mathcal{M}}, del^{\mathcal{M}} \rangle$ provides the functions that map a given action name to its precondition, conditional add effects and delete effects, such that

$$pre^{\mathcal{M}} : A \rightarrow \Phi^{V^{\mathcal{M}}}$$
$$add^{\mathcal{M}} : A \rightarrow 2^{\mathbb{C}_{V^{\mathcal{M}}}} \qquad del^{\mathcal{M}} : A \rightarrow 2^{\mathbb{C}_{V^{\mathcal{M}}}},$$

where $\Phi^{V^{\mathcal{M}}}$ is the set of all propositional logical formulas that can be generated using $V^{\mathcal{M}}$, and $\mathbb{C}_{V^{\mathcal{M}}}$ the set of all conditional effects that can be expressed over the same fluent set. Here, each conditional effect is captured as a pair of the form (ϕ, eff), where ϕ is the condition that needs to hold for the effect $\mathit{eff} \subseteq V^{\mathcal{M}}$ to be applied in a state. $I^{\mathcal{M}} \in S^{\mathcal{M}}$ is the initial state from which the agent is trying to achieve the goal; and $G^{\mathcal{M}} \in \Phi^{V^{\mathcal{M}}}$ is the goal specification.

An action $a \in A^{\mathcal{M}}$ is said to be executable in a state s if and only if the corresponding precondition is satisfied in that state. Here, the precondition is said to be satisfied in a state, i.e., $s \models pre^{\mathcal{M}}(a)$, if the truth assignment that satisfies the corresponding conjunctive formula of the state also satisfies the precondition. Our choice to use arbitrary propositional formulas for preconditions allows us to capture negative preconditions, along with disjunctive ones. The result of executing an action in a state is captured by the transition function γ, where $\gamma(s, a, \mathcal{M})$ is given as

$$\begin{cases} (s \setminus \text{del\_set}^{\mathcal{M}}(s,a)) \cup \text{add\_set}^{\mathcal{M}}(s,a), & \text{if } s \models pre^{\mathcal{M}}(a) \\ \textit{undefined} & \text{otherwise.} \end{cases}$$

Here del_set is the set of all conditional delete effects that hold in s, i.e., $\text{del\_set}^{\mathcal{M}}(s,a)) = \bigcup_{(\phi, \mathit{eff}) \in del^{\mathcal{M}}(a), s \models \phi} \mathit{eff}$. Similarly, add_set is defined over all the add effects that hold in s. We will overload the notation γ also to denote the application of a sequence of actions.

A solution to a planning problem is a plan, which is defined as a (possibly empty) sequence of actions whose execution in the initial state results in a state that satisfies the goal specification. More formally, an action sequence $\pi = \langle a_1, ..., a_k \rangle$ is a plan for a model $\mathcal{M}$, if and only if$\gamma(I^{\mathcal{M}}, \pi, \mathcal{M}) \models G^{\mathcal{M}}$. To simplify the proposed compilations and the related discussion, we will focus on planning problems where actions have unit costs. As such, the cost associated with a plan is equal to the number of its actions. A plan is said to be optimal if no plan exists with a smaller cost. We will denote the cost of an optimal plan associated with a model $\mathcal{M}$ as $C^*_{\mathcal{M}}$.

It is worth noting that the original model reconciliation problem formulation (Sreedharan, Chakraborti, and Kambhampati 2021) and further work that established its complexity (Sreedharan, Bercher, and Kambhampati 2022) focused on scenarios where planning models were strictly limited to STRIPS representation (without conditional effects), where the preconditions can only be a conjunction of literals.

Model Reconciliation Explanation

The focus of model reconciliation explanations is to explain a given robot plan π_R, which is optimal for the robot model $\mathcal{M}^R$. The explanation is provided to a user whose understanding of the planning task may differ or be incorrect. Here, the user's understanding of the task will be represented as $\mathcal{M}^H$, and the user will rely on $\mathcal{M}^H$ to evaluate the correctness and optimality of π_R. Given this model discrepancy, the explanation here involves correcting the user's misunderstanding about the planning task. More specifically, the user either doubts the plan's executability (or that it achieves the goal) or thinks that it is suboptimal, i.e., a shorter plan exists. We will represent the explanation that disputes these misconceptions as a set of model updates that are applied to the user model so they can correctly identify that π_R is optimal. Under this explanation scheme, all model updates provided to the user need to align with the robot's true model $\mathcal{M}^R$, as such bringing the user model $\mathcal{M}^H$ closer to $\mathcal{M}^R$. One could, of course, always explain the plan by providing all the differences between $\mathcal{M}^H$and $\mathcal{M}^R$. Instead of doing that, model reconciliation aims at providing only the minimal number of differences between these models that are sufficient to explain the optimality of π_R. In this paper, we will specifically focus on the problem of identifying an explanation of size k for the given human and robot model. Here, the explanation size refers to the number of model updates involved in the explanation. We will denote this problem as $\mathcal{P}^{\mathit{MRE}}_k = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$. The version defined over STRIPS models will be denoted as $\mathcal{P}^{\mathit{MRE\text{-}STRIPS}}_k$.

In this paper, we will focus on scenarios where the models differ only in terms of the initial state (denoted as $\mathcal{P}^{\mathit{MRE\text{-}I}}_k$). We make this choice solely to simplify the discussion, and it doesn't limit the expressivity of the problems it can capture. In fact, one could convert the problem of identifying an explanation for any $\mathcal{P}^{\mathit{MRE\text{-}STRIPS}}_k$ into a problem of identifying an explanation over two PDDL models that differ only in terms of their initial state. This makes our chosen setting a strict generalization of the problem studied by Sreedharan, Bercher, and Kambhampati (2022). Or more formally:

Proposition 1. *For any model reconciliation problem $\mathcal{P}^{\mathit{MRE\text{-}STRIPS}}_k = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$ defined over STRIPS models, there exists a model reconciliation problem of the form*

$\mathcal{P}_k^{\textit{MRE-I}} = \langle \mathcal{M}_I^R, \mathcal{M}_I^H, \pi_R \rangle$, *such that*

- *One could construct* $\mathcal{P}_k^{\textit{MRE-I}}$ *from* $\mathcal{P}_k^{\textit{MRE-STRIPS}}$ *in time polynomial in the combined size of* $\mathcal{M}^H$ *and* $\mathcal{M}^R$,
- *There exists an explanation of length* k *for* $\mathcal{P}_k^{\textit{MRE-I}}$ *if and only if there exists an explanation of length* k *for* $\mathcal{P}_k^{\textit{MRE-STRIPS}}$,
- $\mathcal{M}_I^R$ *and* $\mathcal{M}_I^H$ *only differ in their initial states,*
- *Explanation for* $\mathcal{P}_k^{\textit{MRE-I}}$ *can be uniquely mapped to the explanation for* $\mathcal{P}_k^{\textit{MRE-STRIPS}}$.

Proof Sketch. This follows from the fact that the presence of any model component, whether it be precondition, effects, or goal, can be represented by some meta-fluents being present in the initial state. Here, the model component can be made conditioned on this new meta fluent. For preconditions and goals, this can be achieved through implication, and for effects by making it a conditional one, where the condition is the new fluent being true. This new fluent will be part of the initial state of the variable when the model component is present and will be missing in the model that doesn't contain the model component. This mirrors results from the construction of universal PDDL domain models (Haslum and Corrêa 2024). Thus, we can convert any two arbitrary STRIPS models that share fluent and action labels into two models that differ only in initial states. Additionally, we can construct them in polynomial time, since we only need to iterate over the original model components once. □

To formalize the notion of model updates, we will introduce a set of model parameters that can be used to represent the possible set of models that could result from applying possible model updates. We will represent the model parameters by a set of the form: $\mathbb{F}^V = \{\textit{init-has-}f \mid f \in F\}$.

Unlike previous works in model reconciliation (Sreedharan, Chakraborti, and Kambhampati 2021), our model parameters only need to represent the initial state because that is the only part of the model that will be updated in this case. Each model in the space of possible models ($\mathbb{M}_A^V$) can be represented uniquely by a subset of the model parameters $\mathbb{F}^V$. We will use a function $\Gamma : \mathbb{M}_A^V \rightarrow 2^{\mathbb{F}^V}$ to convert a model to its parametric form, where for a given model $\mathcal{M} = \langle V^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}, I^{\mathcal{M}}, G^{\mathcal{M}} \rangle$, the parametric form is given as $\Gamma(\mathcal{M}) = \{\textit{init-has-}f \mid f \in I^{\mathcal{M}}\}$.

We will use the notation Γ^{-1} to represent a function that converts model parameters back into a full model. A model update is given by a pair of the form $\mathcal{E} = (\mathcal{E}^+, \mathcal{E}^-)$, where $\mathcal{E}^+ \in 2^{\mathbb{F}^V}$ are the set of fluents that will be set to true at the initial state and $\mathcal{E}^- 2^{\mathbb{F}^V}$, are the fluents that will be set false. The application of a model update $\mathcal{E}$ on the model $\mathcal{M}$, is given as $\mathcal{M} + \mathcal{E} = \Gamma^{-1}((\Gamma(\mathcal{M}) \setminus \mathcal{E}^{-1}) \cup \mathcal{E}^+)$.

A valid explanation here corresponds to a model update whose application in $\mathcal{M}^H$ results in a model where the plan π_R is optimal. Additionally, we require that the model updates align with the true robot model, i.e., $\mathcal{E}^+ \subseteq \Gamma(\mathcal{M}^R)$ and $\mathcal{E}^- \cap \Gamma(\mathcal{M}^R) = \emptyset$. As discussed, the goal here is to find an explanation of size k ($\mathcal{P}_k^{\textit{MRE-I}}$), where the size of an explanation $\mathcal{E}$ is given as $|\mathcal{E}^+| + |\mathcal{E}^-|$. We will denote the decision problem corresponding to $\mathcal{P}_k^{\textit{MRE-I}}$ as **MRE-I-**k. Specifically, **MRE-I-**k corresponds to if there exists a valid explanation of size k for a given model reconciliation problem $\mathcal{P}_k^{\textit{MRE-I}}$.

Theorem 1. ***MRE-I-***k *is* Σ_2^p*-complete.*

Proof Sketch. The hardness of the problem is trivial to establish and follows directly from Proposition 1. Indeed, the model reconciliation problem defined over STRIPS models has been proven to be Σ_2^p-complete (Sreedharan, Bercher, and Kambhampati 2022). The proposition shows that we can polynomially reduce the decision version of a $\mathcal{P}_k^{\textit{MRE-STRIPS}}$ to **MRE-I-**k. For the membership, we can follow a strategy similar to that of Sreedharan, Bercher, and Kambhampati (2022) and convert it into a bounded version of a quantified boolean formula called $QSAT_2$. Our compilation will also take the form of

$$\exists X, Z\ \phi_1(X) \wedge \neg(\exists Y\ \phi_2(X, Y)) \wedge \phi_3(X, Z),$$

where X corresponds to the propositions denoting the possible model updates and Z to π_R actions. ϕ_1 captures the requirement that any model updates need to align with $\mathcal{M}^R$. ϕ_2 checks if Y is a plan shorter than π_R that satisfies the goal in the model obtained by including X in $\mathcal{M}^H$, and ϕ_3 checks if π_R is still in the updated model $\mathcal{M}^H$. Note that the encoding for ϕ_1 remains the same as the one used by Sreedharan, Bercher, and Kambhampati (2022). Now, the only difference in ϕ_2 and ϕ_3 is the fact that they need to support disjunctive preconditions and conditional effects. It has been shown that you can still obtain a polynomial SAT encoding for planning problems with disjunctive preconditions and conditional effects (Rintanen 2021). Note that $QSAT2$ is Σ_2^p-complete, hence showing the membership of **MRE-I-**k and thus proving Σ_2^p completeness. □

Stackelberg Planning

In this paper, the first method to identifying a solution to $\mathcal{P}_k^{\textit{MRE-I}}$ will be to convert it into a Stackelberg planning problem (Speicher et al. 2018). A Stackelberg planning problem is a multi-agent planning problem consisting of a leader agent and a follower agent. Here, a planning problem is formally defined by a tuple of the form $\mathcal{M}_{\mathcal{LF}} = \langle V_{\mathcal{LF}}^{\mathcal{M}}, A_{\mathcal{L}}^{\mathcal{M}}, A_{\mathcal{F}}^{\mathcal{M}}, \delta_{\mathcal{LF}}^{\mathcal{M}}, I_{\mathcal{LF}}^{\mathcal{M}}, G_{\mathcal{F}}^{\mathcal{M}} \rangle$, where most of the symbols follow the same convention as before. The subscript $\mathcal{L}$ is used to denote the model components that are exclusive to the leader agent, while $\mathcal{F}$ is used to denote the ones for the follower agent. For example, $A_{\mathcal{L}}^{\mathcal{M}}$ represents the leader actions and $A_{\mathcal{F}}^{\mathcal{M}}$ are the follower actions. Note how the initial state $I_{\mathcal{LF}}^{\mathcal{M}}$ is shared by both agents, but the goal $G_{\mathcal{F}}^{\mathcal{M}}$ belongs only to the follower. The transition function γ is similar to the one defined for classical planning. For a given Stackelberg planning problem, a leader plan $\pi^{\mathcal{L}}$ is an action sequence that is executable from the initial state. Given a leader plan $\pi^{\mathcal{L}}$, the follower's response takes the form of an optimal solution to the classical planning problem $\mathcal{M}_{\mathcal{F}} = \langle V^{\mathcal{M}}, A_{\mathcal{F}}^{\mathcal{M}}, \delta_{\mathcal{LF}}^{\mathcal{M}}, \gamma(I^{\mathcal{M}}, \pi^{\mathcal{L}}, \mathcal{M}_{\mathcal{LF}}), G_{\mathcal{F}}^{\mathcal{L}} \rangle$. Note how the follower starts acting only once the leader finishes executing its actions. In this paper, we will only consider STACKELSAT solutions (Behnke and Steinmetz 2024), where the objective is to find a leader plan $\pi^{\mathcal{L}}$ that renders the follower problem ($\mathcal{M}'$) unsolvable.

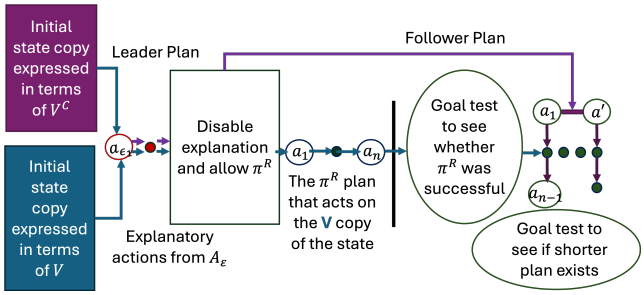

Figure 1: A diagrammatic overview of the Stackelberg compilation, capturing the leader and follower actions and how they influence the two copies constituting the overall compiled planning problem state.

FOND Planning

In addition to a compilation into a Stackelberg planning problem, this paper will also consider compiling the model-reconciliation problem into a fully observable nondeterministic (FOND) planning problem. The problem definition in this case is quite similar to the classical planning problem, and the model is again defined by a tuple of the form $\mathcal{M}_{\mathcal{ND}} = \langle V^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}_{\mathcal{ND}}, I^{\mathcal{M}}, G^{\mathcal{M}} \rangle$. Here, all model components are the same except the action definition, which is given as $\delta^{\mathcal{M}}_{\mathcal{ND}} = \langle pre^{\mathcal{M}}, eff^{\mathcal{M}}_{\mathcal{ND}} \rangle$. In this case, $eff^{\mathcal{M}}_{\mathcal{ND}}$ returns the set of potential mutually exclusive effects that could occur as a result of executing an action. Formally, the function is given as: $eff^{\mathcal{M}}_{\mathcal{ND}} : A^{\mathcal{M}} \rightarrow 2^{\mathbb{E}^{\mathcal{M}}_V}$, where $\mathbb{E}^{\mathcal{M}}_V$ is the set of all add and delete effect pairs that could be defined over the fluent set $V^{\mathcal{M}}$. Here, the transition function $\gamma(s, a, \mathcal{M})$ returns a set of states, where each state in the set corresponds to the application of one of the potential effect pairs in the set $eff^{\mathcal{M}}_{\mathcal{ND}}(a)$. A solution for a FOND problem takes the form of a policy. A policy π is a function that maps states to actions, i.e., $\pi : S^{\mathcal{M}} \rightarrow A^{\mathcal{M}}$. We assume that a policy maps each state to a set of executable actions. A trace associated with a policy π and model $\mathcal{M}$ is a sequence of alternating states and actions of the form $\tau = \langle I^{\mathcal{M}}, \pi(I^{\mathcal{M}}), s_1, \pi(s_1), ... \rangle$, such that it starts with the initial state, every action in the sequence must correspond to the action returned by π for the preceding state, and for any state s_i which is not at the beginning of the sequence, $s_i \in \gamma(s_{i-1}, \pi(s_{i-1}), \mathcal{M})$ holds, where s_{i-1} is the preceding state. A trace is said to be complete if it is either infinitely long or ends in a goal state. Here, we are interested in identifying strong policies (Cimatti et al. 2003). A policy is said to be a strong one if every complete state ends in a goal state and no state is repeated in the sequence.

Compilation to Stackelberg Planning

We compile the problem of identifying a solution to a model reconciliation problem $\mathcal{P}^{MRE\text{-}I}_k = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, with $\mathcal{M}^R = \langle V, A, \delta, I^R, G \rangle$, and $\mathcal{M}^H = \langle V, A, \delta, I^H, G \rangle$, into that of finding a STACKELSAT solution for a Stackelberg planning problem of the form $\mathcal{M}^{MRE}_{\mathcal{LF}} = \langle V^{MRE}_{\mathcal{LF}}, A^{MRE}_{\mathcal{L}}, A^{MRE}_{\mathcal{F}}, \delta^{MRE}_{\mathcal{LF}}, I^{MRE}_{\mathcal{LF}}, G^{MRE}_{\mathcal{F}} \rangle$.

Our goal here is to convert the problem of identifying a solution to $\mathcal{P}^{MRE\text{-}I}_k$ into that of identifying a STACKELSAT solution for a Stackelberg planning problem. This planning problem will make use of two copies of the original fluent set. As discussed, in a STACKELSAT solution, the follower cannot identify a plan for the corresponding $\mathcal{M}_{\mathcal{F}}$ planning problem that results from executing the leader's plan. In our compilation, the problem uses two copies of the original fluent set V. The leader agent will start from an initial state that contains copies of the human initial state over the two sets of fluents. Here, the leader agent will have access to two sets of actions: a set of explanatory actions that will help determine the model-reconciliation explanation to be considered and the actions that are part of π_R. As part of the leader plan, the leader agent is expected to first determine the set of explanatory actions, thus determining the explanation and the new human initial state (this change will be reflected in both copies of the initial state). Following the explanatory actions, the leader plan is supposed to execute the actions in π_R in sequence. The π_R execution will only update the part of the state expressed under one copy of fluents. Now, the only way the follower agent can achieve its goal is by establishing that the model updates identified by the explanatory actions in the leader plan don't constitute a valid solution to the original model reconciliation problem $\mathcal{P}^{MRE\text{-}I}_k$. It can do it in two ways. Either by establishing the fact that the π_R execution carried out by the leader agent failed (thus showing the invalidity of π_R in the new model) or by showing that there exists a plan shorter than π_R that can achieve the goal. The first case is captured by a follower goal action that will check whether π_R was correctly executed in the leader plan by checking the copy of fluents that was updated by π_R actions in the leader plan. The follower also has access to actions that can update the second copy of the fluents. These actions will also start executing from the new initial state determined by the explanatory actions that are part of the leader plan. The follower is also constrained to, at most, use $|\pi_R| - 1$ sequences of these actions. The follower has a second goal action that can check if the goal was reached at the end of such a sequence of actions. Figure 1 presents a diagrammatic representation of the compilation.

For the new compiled model, the new fluent set $V^{MRE}_{\mathcal{LF}}$ is given as: $V^{MRE}_{\mathcal{LF}} = V \cup V^C \cup V^k \cup V^{\pi_R}_{\mathcal{L}} \cup V^{|\pi^R|-1}_{\mathcal{F}} \cup \{v_{\mathcal{E}}, v_g\}$, where $V^C = v^c | v \in V$ is the copy of the original fluents V, V^k are fluents used to track the size of possible explanations (where $|V^k| = k$), $V^{\pi_R}_{\mathcal{L}}$ contains predicates that will enforce the execution of the robot plan, $V^{|\pi_R|-1}_{\mathcal{F}}$ is the budget fluent set used to ensure that the plans considered by the follower are shorter than π_R, $v_{\mathcal{E}}$ is a fluent used to ensure that explanatory actions can only be performed by the leader before executing π_R actions, and finally v_g is the goal fluent.

The leader actions $A^{MRE}_{\mathcal{L}}$ are given as $A^{MRE}_{\mathcal{L}} = A_{\mathcal{E}} \cup A^{\pi_R}_{\mathcal{L}} \cup \{a^{flip}_{\mathcal{L}}\}$. Here, $A_{\mathcal{E}}$ is the set of explanatory actions. $A_{\mathcal{E}}$ is further defined as: $A_{\mathcal{E}} = A^{R \setminus H}_{\mathcal{E}} \cup A^{H \setminus R}_{\mathcal{E}}$, where $A^{R \setminus H}_{\mathcal{E}}$ contains k actions for each element in the set $I^R \setminus I^H$ and $A^{H \setminus R}_{\mathcal{E}}$ contains k actions for each element in $I^H \setminus I^R$. Now, action $a^{R \setminus H}_v \in A^{R \setminus H}_{\mathcal{E}}$ corresponding to a fluent $v \in I^R \setminus I^H$ and a

budget fluent $v_i^k \in V^k$, is defined as:

$$pre^{MRE}_{\mathcal{LF}}(a_v^{R\setminus H}) = v_{\mathcal{E}} \wedge v_i^k$$
$$add^{MRE}_{\mathcal{LF}}(a_v^{R\setminus H}) = \{(\top, \{v\}), (\top, \{v^C\})\},$$
$$del^{MRE}_{\mathcal{LF}}(a_v^{R\setminus H}) = \{(\top, \{v_i^k\})\}$$

where v^C is the corresponding copy of v in V^C. As such, the action will add both copies of the corresponding fluent into the state. Note that a conditional effect of the form $(\top, \mathit{eff})$ corresponds to one that is guaranteed to be applied in all states where the action can be executed. The actions $A_{\mathcal{E}}^{H\setminus R}$ are defined similarly, but instead of adding the fluents, they will delete them. Now, all the explanatory actions include the precondition $v_{\mathcal{E}}$ and a budget fluent. The fact that all explanatory actions will delete the corresponding budget fluents will ensure that only k explanatory actions are used in a leader plan since only k budget fluents will be present in the initial state, and no action can add any budget fluents. The action $a_{\mathcal{L}}^{flip}$ will delete the $v_{\mathcal{E}}$ fluent, which will allow the leader agent to execute actions from $A_{\mathcal{L}}^{\pi_R}$. The set $A_{\mathcal{L}}^{\pi_R}$ contains an action for each action in the plan π_R. Let $a \in A$ be the action that appears in the index i in the plan π_R, then $A_{\mathcal{L}}^{\pi_R}$ will contain an action a^i, which is defined as

$$pre^{MRE}_{\mathcal{LF}}(a^i) = pre^{\mathcal{M}^R}(a) \wedge \neg v_{\mathcal{E}} \wedge v^i$$
$$add^{MRE}_{\mathcal{LF}}(a^i) = add^{\mathcal{M}^R}(a) \cup \{(\top, v^{i+1})\}$$
$$del^{MRE}_{\mathcal{LF}}(a^i) = del^{\mathcal{M}^R}(a) \cup \{(\top, v^i)\},$$

where v^i and v^{i+1} are part of $V_{\mathcal{L}}^{\pi_R}$, and are used to track the ordering between the actions of π_R.

On the other hand, the follower actions are given as follows:$A_{\mathcal{F}}^{MRE} = A_{\mathcal{F}}^C \cup \{a_g^1, a_g^2\}$, where $A_{\mathcal{F}}^C$ are the copies of the actions that the follower will use and $\{a_g^1, a_g^2\}$ are two goal actions. The set $A_{\mathcal{F}}^C$ contains $|\pi^R| - 1$ copies of each action in the set A. Here, each component of the actions is expressed in terms of the fluents in V^C and requires a corresponding budget fluent from the set $V_{\mathcal{F}}^{|\pi^R|-1}$. To simplify the notations, we will overload the notation $V^C(\cdot)$ to stand for a function that converts a propositional formula expressed in terms of fluents from V to one where each fluent is replaced by the corresponding fluent from V^C. Under this notation scheme, the action definition of the i^{th} copy of action $a \in A$, denoted as $a_{\mathcal{F}}^i$, is provided as

$$pre^{MRE}_{\mathcal{LF}}(a_{\mathcal{F}}^i) = V^C(pre^{\mathcal{M}^R}(a)) \wedge v_{\mathcal{F}}^i$$
$$add^{MRE}_{\mathcal{LF}}(a_{\mathcal{F}}^i) = V^C(add^{\mathcal{M}^R}(a^i))$$
$$del^{MRE}_{\mathcal{LF}}(a_{\mathcal{F}}^i) = V^C(del^{\mathcal{M}^R}(a^i)) \cup \{(\top, v_{\mathcal{F}}^i)\},$$

where the fluent $v_{\mathcal{F}}^i \in V_{\mathcal{F}}^{|\pi_R|-1}$ are the budget fluents.

As discussed, the follower can achieve its goal (denoted by the goal fluent v_g) in two ways, denoted by the two goal actions. Firstly, the goal is said to be achieved if the follower couldn't successfully execute π_R. This is denoted by the goal action a_g^1, whose action definitions are given as:

$$pre^{MRE}_{\mathcal{LF}}(a_g^1) = \neg v^{|\pi_R|+1} \vee \neg G$$
$$add^{MRE}_{\mathcal{LF}}(a_g^1) = \{(\top, v_g)\} \quad del^{MRE}_{\mathcal{LF}}(a_g^1) = \emptyset,$$

where $\neg v^{|\pi_R|+1}$ means the leader couldn't complete the execution of the π_R sequence, and $\neg G$ means even if it did, the execution didn't result in the original goal. The follower can also achieve its goals if it can use the actions from $A_{\mathcal{F}}^C$ to achieve the V^C copy of G under $|\pi_R| - 1$ steps, i.e.,

$$pre^{MRE}_{\mathcal{LF}}(a_g^2) = V^C(G)$$
$$add^{MRE}_{\mathcal{LF}}(a_g^2) = \{(\top, v_g)\} \quad del^{MRE}_{\mathcal{LF}}(a_g^2) = \emptyset.$$

Now, we finally come to the goal and the initial state. As discussed, the goal here merely corresponds to v_g (i.e., $G_{\mathcal{F}}^{MRE}$). As for the initial state, it will contain fluents from both V and V^C corresponding to fluents that were present in the original human initial state. It will also contain $v_{\mathcal{E}}$ to support the execution of explanatory actions, $v^1 \in V_{\mathcal{L}}^{\pi_R}$ to ensure the leader agent starts execution from the first index of the π_R and the budget fluents from $V_{\mathcal{F}}^{|\pi^R|-1}$. Or, more formally, $I_{\mathcal{LF}}^{MRE} = I^{\mathcal{H}} \cup V^C(I^{\mathcal{H}}) \cup \{v_{\mathcal{E}}, v^1\} \cup V_{\mathcal{F}}^{|\pi^R|-1}$.

This brings us to the theoretical properties of the compilation. The first property relates soundness, i.e., one can extract a valid explanation for $\mathcal{P}_k^{MRE\text{-}I}$ from a STACKELSAT solution for the compiled model $\mathcal{M}_{\mathcal{LF}}^{MRE}$.

Proposition 2. *For a given model reconciliation problem $\mathcal{P}_k^{MRE\text{-}I} = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, finding a STACKELSAT solution for the compiled model $\mathcal{M}_{\mathcal{LF}}^{MRE}$ is a sound procedure for identifying a valid solution for $\mathcal{P}_k^{MRE\text{-}I}$. Specifically, the explanatory actions used in the leader plan will correspond to a valid explanation for $\mathcal{P}_k^{MRE\text{-}I}$.*

The proof for this property follows from the construction, since the follower is trying to invalidate that the model updates identified by the leader constitute a valid explanation. Hence, the existence of STACKELSAT solution must identify a valid explanation. A more detailed proof sketch for the property is given in the supplementary file[1].

Next, we can also see that the compilation is complete, i.e., there must exist a STACKELSAT solution for the corresponding compiled model for every valid explanation

Proposition 3. *For a given model reconciliation problem $\mathcal{P}_k^{MRE\text{-}I} = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, finding a STACKELSAT solution for the compiled model $\mathcal{M}_{\mathcal{LF}}^{MRE}$ is a complete procedure for identifying a solution for $\mathcal{P}_k^{MRE\text{-}I}$.*

Completeness follows from the fact that one could map a valid explanation to a leader plan for a STACKELSAT plan (proof sketch provided in the supplementary file). Finally, we can see that the size of the compilation will be polynomial in the size of the original explanation problem.

Proposition 4. *For a given model reconciliation problem $\mathcal{P}_k^{MRE\text{-}I} = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, the corresponding Stackelberg planning problem $\mathcal{M}_{\mathcal{LF}}^{MRE}$ takes a form such that it can be created in polynomial time and the total number of fluents and action labels in the new model ($|V_{\mathcal{LF}}^{MRE}| + |A_{\mathcal{L}}^{MRE}|$) will be $\mathcal{O}(|V| * |\pi_R| + |A| * |\pi_R|)$.*

The bound comes directly from the definition of different components. The number of fluents is proportional to $|V| * |\pi_R|$. The primary increase in actions occurs in the follower set, where it uses $|\pi_R| - 1$ copies of each action.

[1]Published at https://doi.org/10.5281/zenodo.19143080.

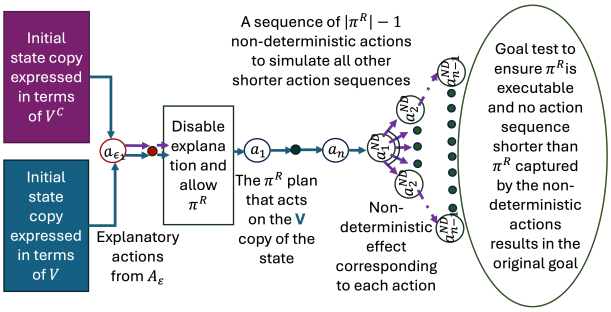

Figure 2: A diagrammatic overview of the FOND compilation. Here, the non-deterministic actions are used to represent all possible subsequences of length shorter than π_R.

FOND Compilation

The FOND compilation will follow a pattern similar to the Stackelberg one. We will again force the explanatory actions to occur first, followed by the execution of π_R, which will then be followed by an attempt to identify a plan shorter than π_R in the updated initial state. The big difference here is that, instead of the adversarial follower, a set of non-deterministic actions will be used to test all action sequences of length shorter than π_R. Here, the action will have an effect corresponding to the effects of each potentially applicable action. At the end of these action sequences, a goal check action is used to check if the original π_R execution was performed successfully and if the execution of the action sequence shorter than π_R didn't result in the original goal. Here, an MCE is said to be identified if the planner can find a strong policy, i.e., a policy in which every complete trace ends in a successful goal check. Here, each trace involves a set of explanatory actions, the execution of π_R, followed by the execution of an action sequence shorter than π_R. Figure 2 presents a diagrammatic representation of the compilation.

The FOND problem here will be defined by a tuple of the form: $\mathcal{M}^{MRE}_{\mathcal{ND}} = \langle V^{MRE}_{\mathcal{ND}}, A^{MRE}_{\mathcal{ND}}, \delta^{MRE}_{\mathcal{ND}}, I^{MRE}_{\mathcal{ND}}, G^{MRE}_{\mathcal{ND}} \rangle$. Given the similarity to the previous Stackelberg compilation, we will reuse components from the previous section. Here, the fluent set is given as: $V^{MRE}_{\mathcal{ND}} = V^{MRE}_{\mathcal{LF}} \cup \{v_\emptyset\}$. In other words, we use the same set of fluents as before, but now introduce a new failure fluent $v_\emptyset$. Moving on to the actions, we again see similarities, and the new action set is given as: $A^{MRE}_{\mathcal{ND}} = A^{MRE}_{\mathcal{L}} \cup A_{\mathcal{ND}} \cup \{a_g\}$. Here, $A^{MRE}_{\mathcal{L}}$ corresponds to the leader action as defined in the Stackelberg compilation. $A_{\mathcal{ND}}$ is a set of actions with non-deterministic effects and a_g is a goal action. The definition of the leader action stays the same as what was described earlier.

In the set $A_{\mathcal{ND}}$, will have $|\pi_R|-1$ actions. Each $a \in A_{\mathcal{ND}}$ will include a budget precondition of the form $v^i_{\mathcal{F}}$, where i corresponds to the time step associated with the action and $v^i_{\mathcal{F}} \in V^{|\pi^R|-1}_{\mathcal{F}}$. They will have a common delete effect, removing the budget fluent. Now, each action $a \in A_{\mathcal{ND}}$ will contain one non-deterministic effect corresponding to each possible action in A, i.e., where the i^{th} effect corresponds to the application of some action $a_i \in A$. More formally, let $(\mathit{eff}^+_i, \mathit{eff}^-_i) \in \mathit{eff}^{MRE}_{\mathcal{ND}}$, then each effect set is given as

$$\mathit{eff}^+_i = \{(\neg V^C(pre(a_i))\}, v_\emptyset) \cup \{(V^C(pre(a_i) \wedge \phi), e) \mid (\phi, e) \in add(a_i)\},$$

where the conditional effect $(\neg V^C(pre(a_i)), v_\emptyset)$ sets the failure fluent true if the precondition of the original action isn't met. The original effects of the action are only applied if the action precondition is met. Also, all the action components are mapped over to fluents from the set V^C. As such, its application only depends on and affects the part of the state defined over those fluent copies. The delete effect part of the non-deterministic effect, i.e., eff^-_i, is defined similarly over the delete effects of the original actions. However, there is no extra effect for the failure fluent.

The action definition for a_g is given as

$$pre^{MRE}_{\mathcal{ND}}(a_g) = v^{|\pi_R|+1} \wedge G \wedge (v_\emptyset \vee \neg V^C(G))$$
$$\mathit{eff}^{MRE}_{\mathcal{ND}}(a_g) = \{(\{(\top, v_g)\}, \{\})\}.$$

The goal and initial state here stays the same as the Stackelberg compilation, i.e., $I^{MRE}_{\mathcal{ND}} = I^{MRE}_{\mathcal{LF}}$ and $G^{MRE}_{\mathcal{ND}} = G^{MRE}_{\mathcal{LF}}$.

Let's now look at the properties of the compilation. Again, the first property we can assert is soundness.

Proposition 5. *For a given model reconciliation problem $\mathcal{P}^{MRE\text{-}I}_k = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, finding a strong policy for the compiled FOND problem $\mathcal{M}^{MRE}_{\mathcal{ND}}$, is a sound procedure for identifying a solution for the given $\mathcal{P}^{MRE\text{-}I}_k$.*

Soundness again follows from the construction. The non-deterministic effects ensure that every possible shorter action sequence is considered. The compilation only allows a strong solution if none of those sequences leads to a goal state. A more detailed proof sketch is given in the supplementary file. We can also see that the FOND compilation also leads to a complete procedure to identify explanations.

Proposition 6. *For a given model reconciliation problem $\mathcal{P}^{MRE\text{-}I}_k = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, finding a strong policy for the compiled FOND problem $\mathcal{M}^{MRE}_{\mathcal{ND}}$ is a complete procedure for identifying a solution for a given $\mathcal{P}^{MRE\text{-}I}_k$.*

This property can also be established by showing that every valid explanation can be mapped over to a strong policy (a detailed sketch is provided in the supplementary file). The compilation is again polynomial in the input size.

Proposition 7. *For a given model reconciliation problem $\mathcal{P}^{MRE\text{-}I}_k = \langle \mathcal{M}^R, \mathcal{M}^H, \pi_R \rangle$, the corresponding FOND planning problem $\mathcal{M}^{MRE}_{\mathcal{ND}}$, takes a form such that the number of fluents is $O*(|V|*|\pi_R|)$ and the number of action $(|A^{MRE}_{\mathcal{ND}}|)$ is $\mathcal{O}(|V| + 2*|\pi_R|)$*

Here, the numbers come from the definition of each model component. As with the Stackelberg compilation, the growth in the number of actions and fluent remains linear.

Related Work

Model reconciliation as a method for explaining plan optimality was first introduced by Chakraborti et al. (2017). Since its inception, the basic framework has been extended multiple times to support explanations in the presence of

multi-agents (Sreedharan, Chakraborti, and Kambhampati 2018) and stochastic transitions (Sreedharan et al. 2019). However, most of these works still leveraged blind search over a space of potential models to identify the target explanations. One could view model reconciliation as being part of a larger body of work that focuses on reasoning about possible model updates that could result in models that satisfy some targeted policy. While some works in this space, like goal recognition design (Keren, Gal, and Karpas 2014), have relied on similar search over the space of models, others, for modeling support, compiled the problem of making input plans executable, have been compiled into finding a minimal hitting set (Lin, Grastien, and Bercher 2023), or solutions to planning problems (Gragera et al. 2025). We refer readers to a recent survey on model repair (Bercher, Sreedharan, and Vallati 2025) for a more comprehensive list of repair problems and solution strategies.

In terms of works that have tried to formalize model reconciliation, the initial set of work was focused on formalizing reconciliation explanation between two logical knowledge bases. Here, the explanation takes the form of a set of formulas, which needs to be incorporated into the knowledge base corresponding to human knowledge, such that it entails some target fact (Vasileiou, Yeoh, and Son 2019). While initial works focused on propositional logic, this was later expanded to also support hybrid planning problems that are encoded using Satisfiability Modulo Theories (SMT) (Vasileiou et al. 2022). As discussed, later work in this direction focused on speeding up the approach for propositional logic formulation by exploiting hitting set solver-based methods. Model reconciliation for logic programs has also been compiled into Answer Set Programming problems (Son et al. 2021).

Stackelberg planning was initially proposed by Speicher et al. (2018) as a means to capture adversarial problems that arise in domains like cybersecurity (Speicher et al. 2019). Since the introduction of this novel notion of planning problems, there have been attempts at developing more effective solvers (Torralba et al. 2021; Sauer et al. 2023). In this paper, we tested our compilation using a solver that uses symbolic search (Torralba et al. 2017) to identify the Pareto frontier associated with the problem. The other planning formalism we leverage within this paper is that of fully observable non-deterministic (FOND) planning problems. Introduced by Cimatti et al. (2003), FOND is a much better-established subfield of planning when compared to the relatively recent Stackelberg planning. While Initial efforts in this direction focused on leveraging model-checkers (Clarke et al. 1996), for our evaluation, we made use of a state-of-the-art FOND planner called PR2 (Muise, McIlraith, and Beck 2024) that makes use of a compilation into classical planning.

In terms of compiling one problem to another, an important aspect to consider is that of computational complexity. For model reconciliation, the problem of identifying the explanation of size k has been shown to be Σ_2^P-Complete for problems defined over STRIPS syntax (Sreedharan, Bercher, and Kambhampati 2022). Similarly, recent works have shown the complexity of identifying STACKELSAT solution over fixed plan sizes to be also Σ_2^P-Complete (Behnke and Steinmetz 2024). Unfortunately, both of these papers consider planning representation schemes that are much less expressive than the one considered in this paper. As such, it is unclear how easily the results carry over into our setting. Since the models we use are much more general than the ones considered by the earlier papers, the complexity of finding model reconciliation explanation in our settings is Sigma-2-P-hard, and membership still has to be established. The complexity of identifying strong solutions for FOND has been shown to be EXPTIME (Rintanen 2004). A related problem worth considering here is that of verifying the optimality of a given plan, which can be thought of as a subprocedure to identifying a model reconciliation explanation. This problem has been shown to be coNP-complete (Lin et al. 2024).

Empirical Evaluation

Our goal with the empirical evaluation was to compare the effectiveness of our proposed compilations against the traditional model space search technique (Chakraborti et al. 2017) (which is a BFS), and the hitting set algorithm (Vasileiou, Previti, and Yeoh 2021) (here referred to as VPY after the author names). All tests were performed over four standard IPC benchmark domains, namely, `Blocksworld (BW)`, `Driverlog (DL)`, `Rover (R)`, and `Zenotravel (ZT)`. We selected five problem instances from each domain of increasing sizes from previous IPCs. For each problem instance, we created five $\mathcal{P}_k^{MRE\text{-}I}$ problems by randomly adding or deleting five fluents from the initial state. The original instance was part of the robot model, while the updated instance was part of the human model. The average optimal plan length across the domains varied from 7.4 in Zenotravel to 14.6 in Rover. The longest optimal plan we considered had a makespan of 22, again in the Rover domain. In terms of the average difference in optimal plan length in the two models, it again ranged from 5.72 in Zenotravel to 10.36 in Blocksworld. We then ran all five methods over each problem with a 30-minute timeout.

For Stackelberg planning, we used a symbolic search-based planner (Torralba et al. 2021), for FOND planning, the PR2 planner (Muise, McIlraith, and Beck 2024), and finally, the model-space search internally used FastDownward planner (Helmert 2006), run under a configuration that uses A* search with lmcut heuristic (Helmert and Domshlak 2009). For VPY, we used the implementation provided by the authors [2]. All experiments were run on an AlmaLinux 8.10 desktop with 62GB Ram and 20 CPU cores (i7-12700K). Our source code, benchmarks, and results are publicly available (Sreedharan and Bercher 2026), and the latest version of our code can be found in our GitHub repo[3].

Table 1 presents the overall results, including the ratio of MRP problems solved by each method, along with the time taken by each method to solve the problems. Each row corresponds to five MRP problems formed from the original IPC

[2]https://github.com/vstylianos/PDDL2CNF-MRP

[3]https://github.com/sarathsreedharan/MCE-STACK-FOND-COMPILATION

Problem Instance				Stackelberg Planning		FOND		VPY		BFS	
Domain	Inst	# of Actions	Min expl length	Fraction Solved	Avg Time (secs)	Fraction Solved	Avg Time (secs)	Fraction Solved	Avg Time (secs)	Fraction Solved	Avg Time (secs)
BW	p1	40	3.60 ± 1.34	5/5	1.52 ± 0.07	5/5	215.01 ± 453.85	5/5	$\mathbf{1.37 \pm 0.05}$	5/5	10.86 ± 5.58
	p2	60	2.80 ± 0.84	5/5	5.60 ± 0.07	3/5	55.51 ± 55.09	5/5	$\mathbf{2.11 \pm 0.03}$	5/5	9.30 ± 4.68
	p3	60	3.00 ± 0.71	5/5	6.18 ± 0.04	0/5	N/A	5/5	$\mathbf{2.76 \pm 0.10}$	5/5	10.87 ± 5.49
	p4	84	3.00 ± 1.00	5/5	6.64 ± 0.55	2/5	770.89 ± 714.10	5/5	$\mathbf{3.35 \pm 0.15}$	5/5	16.24 ± 16.11
	p5	84	2.80 ± 1.10	5/5	6.09 ± 0.35	0/5	N/A	5/5	$\mathbf{3.08 \pm 0.21}$	5/5	16.16 ± 10.94
DL	p1	200	1.60 ± 0.55	5/5	4.08 ± 0.11	3/5	6.80 ± 2.29	5/5	5.25 ± 0.35	5/5	$\mathbf{3.19 \pm 2.04}$
	p2	324	2.40 ± 0.89	5/5	12.34 ± 0.06	0/5	N/A	5/5	19.43 ± 1.80	5/5	$\mathbf{7.81 \pm 5.92}$
	p3	462	1.60 ± 0.89	5/5	$\mathbf{19.48 \pm 0.15}$	0/5	N/A	5/5	454.71 ± 71.61	5/5	50.35 ± 98.84
	p4	480	1.60 ± 1.34	5/5	16.93 ± 0.21	0/5	N/A	5/5	127.44 ± 28.42	5/5	$\mathbf{4.50 \pm 4.52}$
	p5	574	1.40 ± 1.14	5/5	15.18 ± 0.34	0/5	N/A	5/5	33.08 ± 2.11	5/5	$\mathbf{4.21 \pm 3.19}$
R	p1	277	1.20 ± 1.30	5/5	$\mathbf{0.99 \pm 0.02}$	0/5	N/A	5/5	10.95 ± 0.81	5/5	3.07 ± 3.69
	p2	309	1.00 ± 0.71	5/5	$\mathbf{0.85 \pm 0.03}$	0/5	N/A	5/5	10.31 ± 0.50	5/5	2.11 ± 2.41
	p3	636	2.00 ± 0.71	5/5	$\mathbf{3.33 \pm 0.19}$	0/5	N/A	5/5	51.80 ± 8.67	5/5	5.31 ± 3.27
	p4	892	1.40 ± 0.89	5/5	28.45 ± 1.83	0/5	N/A	0/5	N/A	5/5	$\mathbf{11.99 \pm 6.41}$
	p5	892	1.40 ± 1.14	5/5	29.64 ± 2.20	0/5	N/A	0/5	N/A	5/5	$\mathbf{18.78 \pm 29.09}$
ZT	p1	3498	1.40 ± 0.55	5/5	0.18 ± 0.02	5/5	$\mathbf{0.16 \pm 0.02}$	5/5	129.50 ± 7.91	5/5	5.25 ± 2.69
	p2	7020	1.80 ± 1.10	5/5	56.48 ± 3.40	0/5	N/A	0/5	N/A	5/5	$\mathbf{17.00 \pm 15.42}$
	p3	7032	2.00 ± 0.71	5/5	235.17 ± 14.02	0/5	N/A	0/5	N/A	5/5	$\mathbf{16.53 \pm 12.39}$
	p4	12496	1.80 ± 1.10	5/5	326.96 ± 0.24	0/5	N/A	0/5	N/A	5/5	$\mathbf{37.04 \pm 28.27}$
	p5	12512	2.00 ± 1.22	5/5	327.63 ± 0.46	0/5	N/A	0/5	N/A	5/5	$\mathbf{50.45 \pm 26.50}$

Table 1: Details of the empirical evaluation run on all the problem instances across the two compilations, the model-space search, and the VPY algorithm. All average times are listed in seconds along with their standard deviations. The best times are shown in bold. Domain names are abbreviated: BW=Blocksworld, DL=Driverlog, R=Rover, and ZT=Zenotravel.

problem instance. The times are only listed for the portion of the problems it was able to solve, and we list the average time along with the standard deviation. The times listed do not include the time taken to create the compiled problem instances. On average, creating Stackelberg problems took 0.03353 seconds, while FOND took 0.03456 seconds.

First off, we see that among the compilation methods, Stackelberg yields the most consistent performance across the domains tested. For one, apart from the BFS, it is the only method that gives us perfect coverage across all domains we tested. Except for a single problem instance, the Stackelberg compilation did better than FOND in all other instances. This is expected since FOND is a much costlier compilation than the Stackelberg one. After all, in the former, the non-deterministic actions allow for a possible transition for each executable action in the current state.

Coming to the baselines, VPY only did better than Stackelberg on blocksworld problems, but with a very small difference in time taken. Finally, for blind search, we see some interesting results. It outperforms the Stackelberg compilation in three domains at least for a few problems. We believe blind search is surprisingly effective for the considered problems because they have a relatively short minimal explanation length. Table 1 presents the average and standard deviation of the minimal explanation length for each set of model reconciliation problems. A short explanation length means that a blind BFS search would still quickly identify a valid explanation without too many node expansions. It is worth noting that for the Blocksworld domain with longer explanations, the blind search did worse than Stackelberg and VPY.

As discussed, when reviewing the current results, it is worth keeping in mind that the effectiveness of the planning compilation depends on the planners' performance. As planner performance improves, we also expect the overall compilation performance to improve.

Conclusion and Discussion

The paper presents two novel compilations for solving model reconciliation problems. We provide an extensive set of evaluations to test the effectiveness of the proposed compilation. Our preliminary results point toward the Stackelberg compilation being much more effective. We plan to test further compilations. Especially since Σ_2^P is (presumably) much closer to NP than PSPACE (or even EXPTIME) (Arora and Barak 2009), compilations to problems in lower complexity classes might also be fruitful. A natural candidate is QBF formulae with only two quantifiers (Stockmeyer 1976), a formalism also exploited for the membership-/hardness-proofs. Another natural candidate might be a compilation into SAT. Although this is NP-complete and thus (presumably) below Σ_2^P, planning in length-bounded episodes (as done for solving planning problems (Kautz and Selman 1992; Robinson et al. 2008)) might be a promising candidate as well. We plan to investigate CEGAR-based methods (cf. (Seipp and Helmert 2018)) to develop alternative solvers.

Acknowledgements

Sarath Sreedharan's research is supported in part by grant NSF 2303019. Pascal Bercher is the recipient of an Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA), project number DE240101245, funded by the Australian Government. The authors would like to acknowledge the constructive feedback of all our reviewers. We would also like to thank Dr. Stylianos Loukas Vasileiou for his help in setting up the VPY baseline.

References

Apperly, I. A.; and Butterfill, S. A. 2009. Do humans have two systems to track beliefs and belief-like states? *Psychological Review*, 116(4): 953.

Arora, S.; and Barak, B. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.

Behnke, G.; and Steinmetz, M. 2024. On the Computational Complexity of Stackelberg Planning and Meta-Operator Verification. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling, ICAPS 2024*, 20–24. AAAI Press.

Bercher, P.; Sreedharan, S.; and Vallati, M. 2025. A Survey on Model Repair in AI Planning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025*, 10371–10380. ijcai.org.

Chakraborti, T.; Sreedharan, S.; Grover, S.; and Kambhampati, S. 2019. Plan explanations as model reconciliation–an empirical study. In *14th ACM/IEEE International Conference on Human-Robot Interaction, HRI 2019*, 258–266. IEEE.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 156–163. ijcai.org.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2): 35–84.

Clarke, E.; McMillan, K.; Campos, S.; and Hartonas-Garmhausen, V. 1996. Symbolic model checking. In *International conference on computer aided verification*, 419–422. Springer.

Gragera, A.; Fuentetaja, R.; García-Olaya, A.; and Fernández, F. 2025. On the Gains from Using Action Observations in Domain Repair. In *Proceedings of the 35th International Conference on Automated Planning and Scheduling, ICAPS 2025*, 343–347. AAAI Press.

Haslum, P.; and Corrêa, A. B. 2024. The Universal PDDL Domain. *arXiv preprint arXiv:2411.08040*.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? 162–169.

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In Neumann, B., ed., *10th European Conference on Artificial Intelligence, ECAI 1992*, 359–363. John Wiley and Sons.

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal Recognition Design. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling, ICAPS 2014*, 154–162. AAAI Press.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, 12022–12031. AAAI Press.

Lin, S.; Olz, C.; Helmert, M.; and Bercher, P. 2024. On the Computational Complexity of Plan Verification,(Bounded) Plan-Optimality Verification, and Bounded Plan Existence. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence, AAAI 2024*, 20203–20211. AAAI Press.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2024. PRP Rebooted: Advancing the State of the Art in FOND Planning. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence, AAAI 2024*, 20212–20221. AAAI Press.

Rintanen, J. 2004. Complexity of Planning with Partial Observability. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS 2004*, 345–354. AAAI Press.

Rintanen, J. 2021. Planning and SAT. In *Handbook of Satisfiability*, 765–789. IOS Press.

Robinson, N.; Gretton, C.; Pham, D. N.; and Sattar, A. 2008. A Compact and Efficient SAT Encoding for Planning. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling, ICAPS 2008*, 296–303. AAAI Press.

Sauer, P.; Steinmetz, M.; Künnemann, R.; and Hoffmann, J. 2023. Lifted Stackelberg Planning. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling, ICAPS 2023*, 370–374. AAAI Press.

Seipp, J.; and Helmert, M. 2018. Counterexample-guided Cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research*, 62: 535–577.

Son, T. C.; Nguyen, V.; Vasileiou, S. L.; and Yeoh, W. 2021. Model reconciliation in logic programs. In *European Conference on Logics in Artificial Intelligence*, 393–406. Springer.

Speicher, P.; Steinmetz, M.; Backes, M.; Hoffmann, J.; and Künnemann, R. 2018. Stackelberg Planning: Towards Effective Leader-Follower State Space Search. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 6286–6293. AAAI Press.

Speicher, P.; Steinmetz, M.; Hoffmann, J.; Backes, M.; and Künnemann, R. 2019. Towards automated network mitigation analysis. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019*, 1971–1978. ACM.

Sreedharan, S.; and Bercher, P. 2026. Experimental Results for the ICAPS 2026 Paper "Compiling Model Reconciliation Explanation Problems into Stackelberg and FOND Planning Problems". Doi: 10.5281/zenodo.19143616.

Sreedharan, S.; Bercher, P.; and Kambhampati, S. 2022. On the Computational Complexity of Model Reconciliations. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022*, 4657–4664. ijcai.org.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2018. Handling Model Uncertainty and Multiplicity in Explanations via Model Reconciliation. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling, ICAPS 2018*, 518–526. AAAI Press.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of explanations as model reconciliation. *Artificial Intelligence*, 301: 103558.

Sreedharan, S.; Chakraborti, T.; Muise, C.; and Kambhampati, S. 2024. Planning with mental models - Balancing explanations and explicability. *Artificial Intelligence*, 335: 104181.

Sreedharan, S.; Hernandez, A. O.; Mishra, A. P.; and Kambhampati, S. 2019. Model-Free Model Reconciliation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 587–594. ijcai.org.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1): 1–22.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence*, 242: 52–79.

Torralba, Á.; Speicher, P.; Künnemann, R.; Steinmetz, M.; and Hoffmann, J. 2021. Faster Stackelberg Planning via Symbolic Search and Information Sharing. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence, AAAI 2021*, 11998–12006. AAAI Press.

Vasileiou, S. L.; Previti, A.; and Yeoh, W. 2021. On exploiting hitting sets for model reconciliation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2021*, 6514–6521. AAAI Press.

Vasileiou, S. L.; Yeoh, W.; and Son, T. C. 2019. A preliminary logic-based approach for explanation generation. In *ICAPS Workshop on XAIP*, 132–140.

Vasileiou, S. L.; Yeoh, W.; Son, T. C.; Kumar, A.; Cashmore, M.; and Magazzeni, D. 2022. A logic-based explanation generation framework for classical and hybrid planning problems. *Journal of Artificial Intelligence Research*, 73: 1473–1534.