On Exploiting Hitting Sets for Model Reconciliation

Stylianos Loukas Vasileiou,[1] **Alessandro Previti,** [2] **William Yeoh** [1]
[1] Washington University in St. Louis
[2] Ericsson Research
{v.stylianos, wyeoh}@wustl.edu, alessandro.previti@ericsson.com

Abstract

In human-aware planning, a planning agent may need to provide an explanation to a human user on why its plan is optimal. A popular approach to do this is called *model reconciliation*, where the agent tries to reconcile the differences in its model and the human's model such that the plan is also optimal in the human's model. In this paper, we present a logic-based framework for model reconciliation that extends beyond the realm of planning. More specifically, given a knowledge base KB_1 entailing a formula φ and a second knowledge base KB_2 not entailing it, model reconciliation seeks an explanation, in the form of a cardinality-minimal subset of KB_1, whose integration into KB_2 makes the entailment possible. Our approach, based on ideas originating in the context of analysis of inconsistencies, exploits the existing hitting set duality between *minimal correction sets* (MCSes) and *minimal unsatisfiable sets* (MUSes) in order to identify an appropriate explanation. However, differently from those works targeting inconsistent formulas, which assume a single knowledge base, MCSes and MUSes are computed over two distinct knowledge bases. We conclude our paper with an empirical evaluation of the newly introduced approach on planning instances, where we show how it outperforms an existing state-of-the-art solver, and generic non-planning instances from recent SAT competitions, for which no other solver exists.

Introduction

With increasing proliferation and integration of AI systems in our daily life, there is a surge of interest in *explainable AI*, which includes the development of AI systems whose actions can be easily understood by humans. Driven by this goal, *machine learning* (ML) researchers have begun to classify commonly used ML algorithms according to different dimensions of explainability (Guidotti et al. 2018); improved the explainability of existing ML algorithms (Alvarez Melis and Jaakkola 2018; Petkovic et al. 2018); as well as proposed new ML algorithms that trade off accuracy for increasing explainability (Dong et al. 2017; Gilpin et al. 2018).

In contrast, researchers in the *automated planning* community have mostly taken a complementary approach. While there is some work on adapting planning algorithms to find easily explainable plans[1] (i.e., plans that are easily understood and accepted by a human user) (Zhang et al. 2017), most work has focused on the *explanation generation problem* (i.e., the problem of identifying explanations of plans found by planning agents that when presented to users, will allow them to understand and accept the proposed plan) (Kambhampati 1990; Langley 2016). Within this context, there is a popular theme that has recently emerged called *model reconciliation* (Chakraborti et al. 2017). Researchers in this area have looked at how an agent can explain its decisions to a human user who might have a different understanding of the same planning problem. These explanations bring the human's model closer to the agent's model by transferring the minimum number of updates from the agent's model to the human's model. However, a common thread across most of these works is that they, not surprisingly, employ mostly automated planning approaches.

In this paper, we approach the model reconciliation problem from a different perspective – one based on *knowledge representation and reasoning* (KR). We propose a novel general logic-based framework for model reconciliation, where given a knowledge base KB_a (of an agent) that entails a formula φ and a knowledge base KB_h (of a human user) that does not entail φ, the goal is to identify a subset of KB_a such that when it used to update KB_h, then KB_h entails φ. More specifically, we present a novel algorithm that exploits the hitting set duality of *minimal correction sets* (MCSes) and *minimal unsatisfiable sets* (MUSes) for computing minimum explanations with respect to two knowledge bases. MUSes and MCSes have also been studied by Reiter (1987) under the name of conflicts and diagnoses, respectively. de Kleer, Mackworth, and Reiter (1992) have related MUSes (conflicts) and MCSes (diagnoses) to prime implicates and prime implicants. Further, we implement the proposed algorithm in propositional logic and evaluate its performance against the current state-of-the-art (Chakraborti et al. 2017) on classical planning problems as well as present results on some general instances from recent SAT competitions. Although presented in the context of propositional logic, the algorithm can be applied to any type of constraint system for which the satisfiability of subsets can be decided. Our empirical results demonstrate that our approach signif-

[1] Also called *explicable* plans in the planning literature.

icantly outperforms the current state of the art when the explanations are long or when the difference between the agent's and human's models is large, and that it is efficient and feasible for problems beyond planning.

Preliminaries

Classical Planning

A *classical planning* problem is a tuple $\Pi := \langle D, I, G\rangle$, which consists of the domain $D = \langle F, A\rangle$ – where F is a finite set of fluents representing the world states ($s \in F$) and A a set of actions – and the initial and goal states $I, G \subseteq F$. An action a is a tuple $\langle pre_a, eff_a^{\pm}\rangle$, where pre_a are the preconditions of a – conditions that must hold for the action to be applied; and $eff_a^{\pm}$ are the addition ($+$) and deletion ($-$) effects of a – conditions that must hold after the action is applied. The solution to a planning problem Π is a plan $\pi = \langle a_1, \ldots, a_n\rangle$ such that $\delta_\Pi(I, \pi) = G$, where $\delta_\Pi(\cdot)$ is the transition function of problem Π. The cost of a plan π is given by $C(\pi, \Pi) = |\pi|$. Finally, a cost-minimal plan $\pi^* = \operatorname{argmin}_{\pi \in \{\pi' | \delta_\Pi(I,\pi')=G\}} C(\pi, \Pi)$ is called the optimal plan.

Explainable AI Planning

Explainable AI Planning (XAIP), as introduced by Chakraborti, Sreedharan, and Kambhampati (2019), couples the model of the human user and the planning agent's own model into its deliberative process. Therefore, when there exist differences between those two models such that the agent's optimal plan diverges from the user's expectations, the agent attempts a *model reconciliation* process. In this process, the agent provides an explanation that can be used to update the user's model such that the agent's plan is also optimal in the updated user's model.

More formally, a *Model Reconciliation Problem* (MRP) (Chakraborti et al. 2017) is defined by the tuple $\Psi = \langle \varphi, \pi\rangle$, where $\varphi = \langle M^a, M_h^a\rangle$ is a tuple of the agent's model $M^a = \langle D^a, I^a, G^a\rangle$ and the agent's approximation of the user's model $M_h^a = \langle D_h^a, I_h^a, G_h^a\rangle$, and π is the optimal plan in M^a. A solution to an MRP is an explanation ϵ such that when it is used to update the user's model M_h^a to $\widehat{M}_h^{a,\epsilon}$, the plan π is optimal in both the agent's model M^a and the updated user's model $\widehat{M}_h^{a,\epsilon}$. The goal is to find a shortest explanation.

Propositional Logic

In this section, we provide the basic definitions used throughout the paper. Additional standard definitions are assumed (Biere et al. 2009). Although the new algorithm can be applied to any constraint system for which the satisfiability of constraint subsets can be checked, the focus of this paper is on propositional logic.

A formula in *conjunctive normal form* (CNF) is a conjunction of clauses, where each clause is a disjunction of literals. A literal is either a Boolean variable or its negation. For convenience, and when it is clear from the context, we might refer to formulas as sets of clauses and clauses as sets for literals. An *interpretation* $I : V \to \{0, 1\}$ is a mapping from the set of variables V to $\{0, 1\}$. A formula is *satisfiable* if there exists an interpretation that satisfies it. A satisfying interpretation is referred to as a *model*. A formula is *unsatisfiable* or *inconsistent* when no model exists. In what follows, we assume the knowledge base KB and all the formulas are always expressed in CNF. This is not a restrictive requirement, since any propositional formula can be transformed into a CNF. Moreover, unless stated otherwise, KB will be assumed to be consistent.

Definition 1 (Entailment)**.** *A formula φ is logically entailed by KB, denoted by $KB \models \varphi$, if every model of KB is also a model of φ. $KB \models \varphi$ iff $KB \wedge \neg\varphi$ is unsatisfiable.*

Definition 2 (Minimal Unsatisfiable Set (MUS))**.** *Given an inconsistent KB, a subset $\mathcal{M} \subseteq KB$ is an MUS if $\mathcal{M}$ is unsatisfiable and $\forall \mathcal{M}' \subset \mathcal{M}$, $\mathcal{M}'$ is satisfiable.*

By definition, every unsatisfiable KB contains at least one MUS.

Definition 3 (Minimal Correction Set (MCS))**.** *Given an inconsistent KB, a subset $\mathcal{C}$ of KB is an MCS if $KB \setminus \mathcal{C}$ is satisfiable and $\forall \mathcal{C}' \subset \mathcal{C}$ we have that $KB \setminus \mathcal{C}'$ is unsatisfiable.*

Definition 4. *A set of clauses P is a partial MUS of an inconsistent KB if it exists at least one MUS $M \subseteq KB$ such that $P \subseteq M$.*

Partial MUSes appear when in a inconsistent KB a subset of clauses is set as hard. MUSes and MCSes are related by the concept of *minimal hitting set*.

Definition 5 (Hitting Set)**.** *Given a collection Γ of sets from a universe U, a hitting set for Γ is a set $H \subseteq U$ such that $\forall S \in \Gamma$, $H \cap S \neq \emptyset$.*

A hitting set is *minimal* if none of its subsets is a hitting set. The relationship between MUSes and MCSes is discussed by Liffiton and Sakallah (2008) and Liffiton et al. (2016) and it was firstly presented by Reiter (1987), where MUSes and MCSes are referred to as (minimal) conflicts and diagnoses, respectively.

Proposition 1. *A subset $\mathcal{M}$ ($\mathcal{C}$) of an inconsistent KB is an MUS (MCS) iff it is a minimal hitting set of the collection of all MCSes (MUSes) of KB.*

It follows from the above proposition that a cardinality minimal MUS (MCS) is a minimal hitting set. *Cardinality minimal MUS* are referred to as SMUS, whereas a *cardinality minimal MCS* corresponds to the complement of a MaxSAT solution (Li and Manya 2009). We also refer to a cardinality minimal set as a minimum or smallest set.

Lemma 1. *Given a subset $\mathcal{H}$ of all the MCSes of KB, a hitting set is an SMUS if:*

1. *It is a minimum hitting set h of $\mathcal{H}$, and*
2. *The subformula induced by h is inconsistent.*

See the work by Ignatiev et al. (2015) for a proof. Proposition 1 and Lemma 1 naturally extend to the case of partial MUS. Note that when some clauses are set as hard in an inconsistent knowledge base, the set of all MCSes is a subset of the one where all the clauses in the knowledge base are set as soft. In this case, every minimal hitting set on the set of all MCSes is a partial MUS.

Definition 6 (Support)**.** *Given a KB s.t. $KB \models \varphi$, a support for φ is a subset $\epsilon \subseteq KB$ such that $\epsilon \models \varphi$ and $\forall \epsilon' \subset \epsilon$ we have $\epsilon' \not\models \varphi$.*

In what follows, given a formula F, we will write F^* with $* \in \{s, h\}$ to denote a set of clauses that will be treated as *soft* and *hard*, respectively. Intuitively, the hard clauses are those clauses that will not be removed by the minimization procedure.

MUSes and supports are related by the following:

Proposition 2. *A consistent set of clauses ϵ is a support of φ ($\epsilon \models \varphi$) iff ϵ is a partial MUS of $\epsilon \wedge \neg\varphi$.*

In what follows, we provide a definition of a *logic-based* model reconciliation problem (Vasileiou, Yeoh, and Son 2019):

Definition 7 (Model Reconciliation)**.** *Given two knowledge bases KB_a and KB_h of the agent providing an explanation and the human receiving the explanation, respectively, such that $KB_a \models \varphi$ and $KB_h \not\models \varphi$, the goal of model reconciliation is to find a support[2] $\epsilon \subseteq KB_a \wedge KB_h$ such that $KB_h \wedge \epsilon \models \varphi$.*

We refer to the set of clauses $\epsilon \setminus KB_h$ as the *update* of the knowledge base KB_h. In this paper, we focus on the more specific task of computing a support ϵ such that $\epsilon \setminus KB_h$ is an update of minimum size.

Definition 8 (Partial Support)**.** *A subset ϵ_p is a partial support if there exists at least one support ϵ such that $\epsilon_p \subseteq \epsilon$.*

Given a formula $\epsilon \wedge \neg\varphi$, ϵ_p is thus a subset of the partial MUS ϵ. An update is a partial support.

Encoding Classical Planning Problems as SAT: A classical planning problem can be encoded as a SAT problem (Kautz and Selman 1992; Kautz, McAllester, and Selman 1996). The basic idea is the following: Given a planning problem P, find a solution for P of length n by creating a propositional formula that represents the initial state, goal state, and the action dynamics for n time steps. This is referred to as the *bounded planning problem* (P, n), and we define the formula for (P, n) such that: *Any* model of the formula represents a solution to (P, n) and if (P, n) has a solution, then the formula is satisfiable.

We encode (P, n) as a formula φ involving one variable for each action $a \in A$ at each timestep $0 \leq i < n$ and one variable for each fluent $f \in F$ at each timestep $0 \leq i \leq n$. We denote the variable representing action a in timestep i using subscript a_i, and similarly for facts. The formula φ is constructed such that $\langle a_0, a_1, \ldots, a_{n-1} \rangle$ is a solution for (P, n) if and only if φ can be satisfied in a way that makes the fluents $a_0, a_1, \ldots, a_{n-1}$ true. Finally, we can *extract* a plan by finding a model that satisfies φ (i.e., for all time steps $0 \leq i < n$, there will be exactly one action a such that $a_i =$ *True*). This could be easily done by using a satisfiability algorithm, such as the well-known DPLL algorithm (Davis et al. 1962).

[2]Note that we use the term "support" to mean "explanation" in the traditional sense in this context.

Algorithm 1: Basic algorithm for computing the smallest support (one KB)

Input: KB, φ
Result: A minimum size support ϵ

1 $\mathcal{H} \leftarrow \emptyset$
2 **while** *true* **do**
  `// Compute a minimum hitting set`
3 $seed \leftarrow minHS(\mathcal{H})$
4 $\epsilon \leftarrow \{c_i \mid i \in seed\}$
5 **if** *not* $SAT(\epsilon \wedge \neg\varphi)$ **then**
    `// minimum size support`
6   **return** ϵ
7 **else**
8 $\mathcal{C} \leftarrow getMCS(seed, KB^s \wedge \neg\varphi^h)$
9 $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{C}\}$

Computing Explanations

Ignatiev et al. (2015) introduced an algorithm for computing a smallest MUS of an inconsistent knowledge base KB. Building on that approach, we introduce a new algorithm that computes a smallest support ϵ for a formula φ that needs to be explained. The idea is to reduce the problem of computing a support of minimum size to the one of computing an SMUS over an inconsistent formula. Notice that by definition, we have that $KB \models \varphi$ iff $KB \wedge \neg\varphi$ is unsatisfiable. Moreover, in Proposition 2, we have already stated the relation between a support and an MUS. This suggests that, in order to extract a support, we just need to run an MUS solver over the formula $KB_h^s \wedge \neg\varphi^h$,[3] and then remove $\neg\varphi$ from the returned MUS. The duality relating MUSes and MCSes is a key aspect for the computation of an SMUS. In the next section, we will show how this duality can be exploited for the task of model reconciliation. We first start by reporting the algorithm for computing a smallest support for the case of a single knowledge base, and then we will illustrate how this approach can be revised for the case of two knowledge bases. Algorithm 1 is based on the algorithm for computing a smallest MUS originally presented by Ignatiev et al. (2015). $\mathcal{H}$ is a collection of sets, where each set corresponds to an MCS on KB. At the beginning, it is initialized with the empty set (line 1). Each MCS in $\mathcal{H}$ is represented as the set of the indexes of the clauses in it. $\mathcal{H}$ stores the MCSes computed so far. At each step, a minimum hitting set on $\mathcal{H}$ is computed (line 3). On line 4, the formula induced by the computed minimum hitting set is stored in ϵ. Then, the formula $\epsilon \wedge \neg\varphi$ is tested for satisfiability (line 5). If $\epsilon \wedge \neg\varphi$ is unsatisfiable, then ϵ is a support of minimum size. The algorithm return ϵ and the procedure ends. If instead $\epsilon \wedge \neg\varphi$ is satisfiable, then it means that $\epsilon \not\models \varphi$ and the algorithm continues at line 8. The computation of an MCS of this kind can be performed via standard MCS procedures (Marques-Silva

[3]Recall that KB_h^s denote that the clauses in the human's KB_h are treated as *soft* while φ^h denote that the clauses in the formula φ that needs to be explained are treated as *hard*. Soft clauses may be removed by the MUS solver while hard clauses will not.

Algorithm 2: Model Reconciliation Algorithm

Input: KB_a, KB_h, φ
Result: An explanation ϵ such that $\epsilon \setminus KB_h$ is of minimum size

1 $\mathcal{R} \leftarrow \emptyset$
2 $KB_a^h \leftarrow KB_a \cap KB_h$ // hard clauses
3 $KB_a^s \leftarrow KB_a \setminus KB_a^h$ // soft clauses
4 **if** *not* $SAT(KB_h \wedge KB_a)$ **then**
5 $\quad \mathcal{C} \leftarrow getMCS((KB_h \setminus KB_a)^s \wedge KB_a^h)$
6 $\quad KB_h = KB_h \setminus \mathcal{C}$
7 **while** *true* **do**
8 $\quad seed \leftarrow minHS(\mathcal{R})$
9 $\quad \epsilon_p \leftarrow \{c_i \mid i \in seed\}$ // A partial support ϵ_p induced by the seed
10 $\quad$ **if** *not* $SAT(KB_h \wedge \epsilon_p \wedge \neg\varphi)$ **then**
11 $\quad\quad \epsilon \leftarrow getMUS(KB_h^s \wedge \epsilon_p^h \wedge \neg\varphi^h) \setminus \neg\varphi$
12 $\quad\quad$ **return** ϵ
13 $\quad$ **else**
14 $\quad\quad \mathcal{C} \leftarrow getMCS(seed, KB_h^h \wedge KB_a^s \wedge \neg\varphi^h)$
15 $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{C}\}$

et al. 2013), using the set of clauses indexed by the *seed* as the starting formula to extend. Since the clauses in φ are set to hard (line 8), the returned MCS $\mathcal{C}$ is guaranteed to be contained in KB. Due to the hitting set duality relation, we will also have $\epsilon \subseteq KB$. Notice that the procedure $getMCS$ always reports a new MCS because, by construction, we have $seed \subseteq KB \setminus \mathcal{C}$. In fact, the $seed$ contains at least one clause for each previously computed MCS and thus $seed \cap \mathcal{C} = \emptyset$ (i.e., at least one clause for each previously computed MCS is not in $\mathcal{C}$).

Model Reconciliation

In the previous section, we presented an approach for the computation of the smallest support for the case of a single knowledge base. Here, we show how this method can be further extended for the case of two knowledge bases, a task that we defined as *model reconciliation*. In what follows, we assume that one is the knowledge base of an agent KB_a and the other is the one of a human KB_h. The task we are targeting is to find a support $\epsilon \subseteq KB_a \wedge KB_h$ such that $KB_h \wedge \epsilon \models \varphi$ and $\epsilon \setminus KB_h$ is of minimum size. $\mathcal{R}$ is the formula we use to store the MCSes, which acts as a mediator between KB_a and KB_h. Intuitively, the idea is to compute MCSes over KB_a, add them to $\mathcal{R}$, and then test the minimum hitting sets over KB_h.

Algorithm 2 summarizes the main steps of this new approach. At the beginning of the algorithm, $\mathcal{R}$ is empty (line 1). Lines 2 and 3 are used to specify which clauses of KB_a will be treated as hard and soft, respectively. We then check if $KB_h \wedge KB_a$ is inconsistent (line 4). This is important in order to avoid the possibility of finding subsets ϵ that explains why $KB_h \wedge KB_a$ is inconsistent instead of the target support. In case $KB_h \wedge KB_a$ is inconsistent, we preprocess KB_h by removing from $KB_h \setminus KB_a$ a minimal set of clauses causing the conflict (i.e., an MCS) (lines 5-6). The reconciliation procedure starts on line 7. The algorithm proceeds iteratively by computing a minimum hitting set on $\mathcal{R}$ and then testing for satisfiability the induced subformula ϵ_p. ϵ_p is an under approximation of the final partial support. The test checks whether updating KB_h with ϵ_p is sufficient for entailing φ. If the formula $KB_h \wedge \epsilon_p \wedge \neg\varphi$ is unsatisfiable, then an MUS containing a subset of KB_h, ϵ_p, and $\neg\varphi$ is returned, and the set of clauses $\neg\varphi$ is removed from it. The result, from Proposition 2, is an $\epsilon = M \wedge \epsilon_p$, with $M \subseteq KB_h$, such that $\epsilon \models \varphi$. Otherwise, the algorithm continues on line 14, where a new MCS is computed and added to $\mathcal{R}$. The algorithm is complete in the sense that eventually a support ϵ such that $\epsilon \setminus KB_h$ is of minimum size will be returned. This can be easily verified by observing that every time $KB_h \wedge \epsilon_p \wedge \neg\varphi$ is satisfiable, a new MCS is computed. Eventually, all the MCSes will be computed and, from Propositions 1 and 2, it follows that a minimum hitting set on the collection of all MCSes corresponds to the smallest update. Deciding whether there exists a support of size less or equal to k is Σ_2^p-complete and extracting a smallest support is in $FP^{\Sigma_2^p}$. This follows directly from the complexity of deciding and computing an SMUS on which Algorithm 2 is based (Ignatiev et al. 2015).

Finally, notice that in some settings (e.g., planning), desired supports are required to be a subset of KB_a. In this case, KB_h is replaced with KB_a^h on line 10, KB_h^s is replaced with KB_a^h on line 11, and KB_h^h is replaced with KB_a^h on line 14. Notice that, in this setting, the updated $KB_h \wedge \epsilon$ might be inconsistent. However, in our approach, this case is naturally resolved by the preprocessing step at line 6. In some settings, the update $\epsilon \setminus KB_h$ is expected to be a subset of a support $\epsilon' \models \varphi$ s.t. $\epsilon' \subseteq KB_a$. In this case the MCS returned at line 14 should be further shrinked in order to be an MCS of the sole $KB_a^s \wedge \neg\varphi^h$. Table 1 shows an example trace of Algorithm 2.

	$KB_a = (a \overset{C_1}{\vee} b) \wedge (\neg b \overset{C_2}{\vee} c) \wedge \neg \overset{C_3}{c} \wedge (\neg b \overset{C_4}{\vee} d) \wedge \neg \overset{C_5}{d}$ $KB_h = \neg \overset{D_1}{c} \wedge \overset{D_2}{f}$	} We have that $KB_a \models a$ and $KB_h \not\models a$
1	$\mathcal{R} \leftarrow \emptyset$	
2	$KB_a^h \leftarrow KB_a \cap KB_h = \{C_3\}$	
3	$KB_a^s \leftarrow KB_a \setminus KB_a^h = \{C_1, C_2, C_4, C_5\}$	
4	$seed \leftarrow \emptyset$	# $minHS(\mathcal{R})$
5	$\neg c \wedge f \wedge \emptyset \not\models a$	# $SAT(KB_h \wedge \epsilon_p \wedge \neg\varphi)$
6	$\mathcal{C} \leftarrow \{C_2, C_4\}$	# MCS computed on $KB_h \wedge KB_a^s \wedge \neg a$ starting with the seed $seed$
7	$\mathcal{R} \leftarrow \{\{C_2, C_4\}\}$	
8	$seed \leftarrow \{C_2\}$	# $minHS(\mathcal{R})$
9	$\neg c \wedge f \wedge (\neg b \vee c) \not\models a$	# $SAT(KB_h \wedge \epsilon_p \wedge \neg\varphi)$
10	$\mathcal{C} \leftarrow \{C_1\}$	# MCS computed on $KB_h \wedge KB_a^s \wedge \neg a$ starting with the seed $seed$
11	$\mathcal{R} \leftarrow \{\{C_2, C_4\}, \{C_1\}\}$	
12	$seed \leftarrow \{C_1, C_2\}$	# $minHS(\mathcal{R})$
13	$\neg c \wedge f \wedge (\neg b \vee c) \wedge (a \vee b) \models a$	# $SAT(KB_h \wedge \epsilon_p \wedge \neg\varphi)$
14	$Return\ \{C_1, C_2, D_1\}$	# MUS computed on $D_1 \wedge D_2 \wedge C_1 \wedge C_2 \wedge \neg a \setminus \neg a$

Table 1: Example of Algorithm 2

Experimental Evaluations

We now present our experimental evaluation of Algorithm 2 for computing explanations on classical planning problems from the International Planning Competition as well as on some general problems from the SAT competition.[4]

Setup and Prototype Implementation: We ran our experiments on a MacBook Pro machine comprising of an Intel Core i7 2.6GHz processor with 16GB of memory. The time limit was set to 1500s. Our implementation of Algorithm 2 is written in Python and integrates calls to SAT, MCS/MUS, and minimal hitting set oracles through the PySAT toolkit (Ignatiev, Morgado, and Marques-Silva 2018).

Classical Planning Instances

Explanations in Planning: Following the MRP literature (Chakraborti et al. 2017), we focus on the problem of explaining the *optimality* of a plan to a human user. Recall that, in classical planning problems with uniform action costs, a plan π^* is optimal if no shorter plan exists. Couched in terms of propositional logic, we say that a plan π^* of length n is optimal if, given a knowledge base KB encoding the specific planning problem, no shorter plan exists in *all* models of KB. Particularly, π^* is optimal with respect to a knowledge base KB, if $KB \models \bigwedge_{t=0}^{n-1} \neg g_t$, where g_t is the fact corresponding to the goal of the planning problem at time step t. However, in order to show that a plan π^* is optimal, we would first need to show that π^* is a feasible plan (i.e., the plan is sound and can be executed to achieve the goal). More specifically, we say that a plan π^* is feasible with respect to a knowledge base KB if π^* and g_n are true in at least one model of KB, where g_n is the fact corresponding to the goal of the planning problem at the final time step n. Therefore, when combined with the fact that no plan of lengths 1 to $n-1$ exists, then that plan π^* must be an optimal plan.

Note that Algorithm 2 proposed in the previous section is agnostic to the underlying planning domain. However, it can be used to find explanations for MRP in explainable planning as follows: For checking if a plan is feasible, we impose the literals in π^* and g_n as assumptions in the SAT solver, and check if a model that satisfies them exists. If not, we add the missing actions as part of the explanation to KB_h. Finally, to find an explanation for the optimality of the plan, we use $\varphi = \bigwedge_{t=0}^{n-1} \neg g_t$ as the query and run Algorithm 2 with the changes to lines 10, 11, and 14 as described in the last paragraph in the "Model Reconciliation" subsection. Then, the algorithm will return the smallest explanation that explains that the plan π^* is both feasible and optimal.

Experimental Scenarios: We used the actual IPC instances as the model of the agent (i.e., KB_a) and tweaked that model by randomly removing parts (preconditions and effects) of the action model, and assigned it to be the model of the human user (i.e., KB_h). We used our own implementation of the encoding by Kautz, McAllester, and Selman (1996) as the SAT encoding of the knowledge bases. We further make two important assumptions: (1) First, we assume that KB_a has the *correct* and *complete* encoding of the planning problem and only KB_h can contain errors or omissions. (2) We assume that π^* is a feasible plan with respect to KB_a and that $KB_a \models \bigwedge_{t=0}^{n-1} \neg g_t$. These assumptions are reasonable, especially when viewed from the perspective of the explaining agent, where explanations are based on the view (or model) of the specific problem (Miller 2018).

We empirically evaluated Algorithm 2, referred to as ALG2, to find minimum explanations against the current planning-based state-of-the-art algorithm by Chakraborti et al. (2017), referred to as CSZK, which runs an A* search algorithm over the explanation search space and prioritizes actions that are in the plan being explained before others.[5] We consider eight different ways to tweak the user's model, resulting in the following eight scenarios:

- **Scenario 1:** We removed one random precondition from

[4]The code repository is: https://github.com/vstylianos/aaai21.

[5]We used the implementation of the authors, which is publicly available at: https://github.com/TathagataChakraborti/mmp.

Prob.		Scenario 1			Scenario 2			Scenario 3			Scenario 4			Scenario 5			Scenario 6		Scenario 7		Scenario 8	
		$\|\epsilon\|$	CSZK	ALG2	$\|\epsilon\|$	CSZK	ALG2	$\|\epsilon\|$	CSZK	ALG2	$\|\epsilon\|$	CSZK	ALG2	$\|\epsilon\|$	CSZK	ALG2	$\|\epsilon\|$	ALG2	$\|\epsilon\|$	ALG2	$\|\epsilon\|$	ALG2
BLOCKS-WORLD	1	2	0.6	**0.05s**	1	0.2s	**0.05s**	4	12.0s	**0.1s**	8	t/o	**0.2s**	5	129.5s	**0.2s**	2	**0.05s**	6	**0.3s**	10	**0.5s**
	2	1	0.4s	**0.2s**	1	**0.2s**	**0.2s**	4	4.5s	**1.2s**	6	t/o	**14.5s**	5	158.0s	**108.0s**	4	**0.2s**	4	**10.0s**	11	**118.0**
	3	3	1.5s	**0.5s**	3	0.5s	**0.3s**	6	11.0s	**1.0s**	7	t/o	**4.5s**	8	256.0s	**5.0s**	1	**0.1s**	9	**5.0s**	12	**10.5s**
	4	2	**1.0s**	4.0s	2	**0.2s**	1.0s	5	**9.5s**	125.0s	6	t/o	**136.5s**	7	273.5s	**145.0s**	12	**0.5s**	8	**150.0s**	–	t/o
LOG-ISTICS	1	2	1.0s	**0.3s**	1	**0.2s**	0.5s	4	47.0s	**0.5s**	6	t/o	**0.5s**	5	255.0s	**0.2s**	3	**0.1s**	7	**0.5s**	8	**1.0s**
	2	2	**1.0s**	**1.0s**	2	**0.2s**	0.8s	2	1.0s	**0.6s**	5	t/o	**1.0s**	6	276.0s	**0.3s**	4	**0.1s**	6	**0.4s**	7	**1.0s**
	3	2	2.5s	**0.5s**	3	1.0s	**0.5s**	3	5.5s	**1.0s**	6	t/o	**1.5s**	6	271.0s	**0.2s**	6	**0.2s**	6	**0.2s**	6	**1.0s**
	4	4	18.0s	**1.0s**	2	**0.5s**	**0.5s**	6	117.0s	**2.0s**	7	t/o	**2.5s**	7	283.0s	**0.5s**	7	**0.5s**	7	**0.5s**	7	**1.0s**
ROVER	1	1	1.0s	**0.2s**	2	**0.5s**	**0.5s**	4	125.0s	**0.8s**	7	t/o	**1.0s**	7	300.0s	**2.0s**	3	**0.5s**	8	**2.5s**	8	**3.0s**
	2	3	1.0s	**0.5s**	1	**0.5s**	**0.5s**	2	1.0s	**0.5s**	7	t/o	**2.0s**	6	311.0s	**3.0s**	5	**0.5s**	8	**4.0s**	9	**4.0s**
	3	1	**0.5s**	**0.5s**	2	1.0s	**0.5s**	3	55.0s	**1.0s**	6	t/o	**3.0s**	6	330.0s	**5.5s**	6	**0.5s**	7	**6.0s**	8	**6.0s**
	4	1	**0.5s**	**0.5s**	3	3.0s	**1.0s**	6	141.0s	**3.5s**	8	t/o	**4.0s**	7	356.0s	**5.0s**	8	**0.5s**	8	**5.0s**	8	**5.0s**

Table 2: PDDL Problem Instances

Prob.		Scenario 9		Scenario 10		Scenario 11		Scenario 12	
		$\|\epsilon\|$	ALG2	$\|\epsilon\|$	ALG2	$\|\epsilon\|$	ALG2	$\|\epsilon\|$	ALG2
SAT GRID	1	76	3.0s	152	5.0s	231	7.0s	298	8.5s
	2	15	0.2s	17	0.2s	19	0.3s	20	0.4s
	3	177	28.0s	354	35.0s	531	56.0s	708	92.0s
BN RATIO	1	14	0.2s	17	0.2s	19	0.3s	22	0.5s
	2	55	1.0s	58	2.0s	59	2.0s	75	6.0s
	3	13	6.0s	26	18.0s	42	47.0s	51	83.0s
ACE	1	13	2.5s	20	12.0s	34	46.0s	45	130.0s
	2	3	1.5s	6	9.0s	14	55.0s	16	118.0s
	3	6	8.5s	6	37.0s	6	131.0s	8	356.0s
BMC	1	18	2.0s	46	12.0s	69	57.0s	95	193.0s
	2	6	8.0s	12	43.0s	22	398.0s	–	t/o
	3	9	15.0s	9	45.0s	20	192.0s	–	t/o

Table 3: SAT Competition Problem Instances

every action in the user's model.

- **Scenario 2:** We removed one random effect from every action in the user's model.
- **Scenario 3:** We removed one random precondition and one random effect from every action in the user's model.
- **Scenario 4:** We removed multiple random preconditions and effects from every action in the user's model.
- **Scenario 5:** We removed all preconditions from every action in the user's model.
- **Scenario 6:** We removed multiple random predicates from the initial state in the user's model.
- **Scenario 7:** We removed all effects from every action in the user's model.
- **Scenario 8:** We removed all actions in the user's model.

Table 2 tabulates the cardinality of the explanations $|\epsilon|$ as well as the runtimes of CSZK and ALG2. We did not report runtimes of CSZK for Scenario 6 as the available implementation was not designed to handle that scenario. We also omit the runtimes of CSZK for Scenarios 7 and 8 as the implementation contained errors on those instances that we were not able to repair, even though it theoretically could handle such scenarios. In general, ALG2 is faster than CSZK in the majority of scenarios and domains; their runtimes for most problems in Scenarios 1 and 2 are similar, but ALG2 is faster than CSZK in Scenarios 3 to 5 by up to two orders of magnitude. In general, the runtime of both algorithms increases as the difference between the models of the agent and user increases since both algorithms search over the explanation search space, which increases as the number of differences between the two models increases. However, ALG2 is faster because it is able to use highly-optimized solvers to find MCSes, MUSes, and hitting sets.

General Instances

In order to test the generality and robustness of our proposed approach, we conducted experiments on a number of problem instances taken from recent SAT competitions. Similarly to the planning experiments, we used the actual instances as the model of the agent (i.e, KB_a), and tweaked that model and assigned it to be the model of the user (i.e., KB_h). The query that we used for each instance was a conjunction of backbone literals,[6] which we computed using the *minibones* algorithm proposed by Janota, Lynce, and Marques-Silva (2015).

We consider four different ways to tweak the user's model, resulting in the following four scenarios:

- **Scenario 9:** We randomly removed 10% of the clauses and removed 20% of literals from 10% of the total clauses in the user's model.
- **Scenario 10:** We randomly removed 20% of the clauses and removed 20% of literals from 20% of the total clauses in the user's model.
- **Scenario 11:** We randomly removed 30% of the clauses and removed 20% of literals from 30% of the total clauses in the user's model.
- **Scenario 12:** We randomly removed 40% of the clauses and removed 20% of literals from 40% of the total clauses in the user's model.

Table 3 presents the results, where we report the explanation cardinality $|\epsilon|$ and runtimes of ALG2. We did not compare with any other system since, to the best of our knowledge, no such system exists. In general, ALG2, given the set time limit, managed to compute an explanation for all instances. In general, the runtimes of ALG2 increase as the size of the knowledge bases and $|\epsilon|$ increase.

Moreover, similar to classical planning, we observe that the runtimes of ALG2 increase as the difference between the two models increase (i.e., from Scenario 9 to Scenario 12).

[6]The backbone literals of a propositional KB are the set of literals entailed by the KB.

Related Work and Discussions

Our work sits in the intersection of knowledge representation and planning – our techniques are inspired by findings from the knowledge representation community, especially on MCS, MUS, and their duality; and our application problem of model reconciliation was introduced by the planning community in the context of explainable planning. Therefore, we will situate our work in the context of these two communities and relate to existing work in those two areas.

Related Work from Knowledge Representation: The definition of the logic-based MRP we are tackling in this paper (see Definition 7) has been introduced in a previous work (Vasileiou, Yeoh, and Son 2019). Building on those theoretical foundations, in this paper, we consider the development of an efficient algorithm for computing minimal explanations, which further extends the logic-based MRP to account for knowledge bases that may become inconsistent after receiving an explanation.

The algorithm presented in this paper is inspired by a procedure for computing an SMUS of an inconsistent formula, originally presented by Ignatiev et al. (2015). The method is also related to other similar approaches for enumerating MUSes and MCSes. Moreover, our approach is similar in spirit to the HS-tree presented by Reiter (1987). Although the original purpose was to enumerate diagnoses, Reiter's procedure can be easily adapted to enumerate MUSes (called conflicts in that paper) as already noted by Previti and Marques-Silva (2013). However, the computation of an SMUS might require more substantial modifications. Procedures like the one presented by Reiter, which target MCSes (diagnoses) instead of MUSes (conflicts), can be seen as the dual version of our algorithm. In particular, the algorithm MaxHS (Davies and Bacchus 2011) applies the same idea of iteratively computing and testing a minimum hitting set for the computation of a MaxSAT solution (the complement of the smallest MCSes).

On a different note, the notion of supports or explanations proposed in this paper and how they are used to update the knowledge base of the human might appear similar to the notion of belief revision (Gärdenfors et al. 1995). However, there are some subtle differences. Belief revision usually refers to a single agent revising its belief after receiving a new piece of information that is in conflict with its current beliefs. As such, there is a temporal dimension in belief revision and a requirement that it should maintain as much as possible the belief of the agent, per AGM postulates (Alchourrón and Makinson 1985; Gärdenfors 1986). On the other hand, our notion of explanation is done with respect to *two knowledge bases* and there is no such requirement on maintaining as much as possible the belief of the human. Instead, our requirement is on minimizing the size of the explanation. Additionally, our notion of explanation might appear similar to the notion of a diagnosis (e.g., (Reiter 1987)). Diagnosis focuses on identifying the reason for the inconsistency of a theory (i.e., KB) whereas an explanation aims at identifying the support for a formula. The difference lies in that a diagnosis is made with respect to the same theory (i.e., KB_a) while explanation is sought for the second theory (i.e., KB_h). For a further exposition on the relationship between our approach and previous works we refer the reader to (Vasileiou, Yeoh, and Son 2020).

Related Work from Planning: Our work was motivated by the work of Chakraborti et al. (2017), who introduced the model reconciliation problem that we are tackling in this paper. Hence, both approached share similarities, such as the types of explanations that can be found. For instance, the cardinality-minimal support in Definition 5 is equivalent to *minimally complete explanations* (MCEs) (the shortest explanation). Our objective of finding a minimal explanation (actually, support) can be viewed as similar to the *minimally monotonic explanations* (MMEs) (the shortest explanation such that no further model updates invalidate it). Similarly, *model patch explanations* (MPEs) (includes all the model updates) are trivial explanations and are equivalent to our definition that KB_a itself serves as an explanation for KB_h. Nonetheless, we would like to point out a fundamental difference between the two approaches: Our approach is based on KR (i.e., propositional logic in this paper), while theirs is based on automated planning and heuristic search techniques. Additionally, the explanation space search in our approach is done in the grounded representation of the planning domain, whereas theirs in the lifted representation. As a consequence, our approach can structurally isolate the reasons for a particular behavior. For example, an explanation from our approach could explain why a state (frame axioms) or an action (action dynamics) was true (or not) at a particular time step during the execution of the plan.

Conclusions and Future Work

In this paper, we presented a new logic-based approach that exploits the notion of hitting sets for model reconciliation problems. The approach builds on top of previous techniques for computing SMUSes and it shares with them the advantage of being constraint agnostic. Experiments showed that our algorithm outperforms state-of-the-art alternatives in the context of planning and it is able to solve representative SAT instances. Due to its logic-based nature, the approach has the additional advantage of being able to deal with problems coming from different settings, as far as the problem can be encoded into a constraint system. Here, we showed a propositional encoding for the planning case.

The algorithm we presented returns a smallest explanation ϵ such that when it is used to update the model of the user to KB_h, the updated model $KB_h \wedge \epsilon$ entails a formula φ that needs to be explained. However, other kind of preferred explanations could be considered by defining a cost function over them. Future research will investigate alternative preferred explanations. This should be possible by simply applying weights to the elements in $\mathcal{R}$ and return a hitting set that minimizes the cost. Another line of research is to include all the optimization techniques that have already been successfully applied for the extraction of an SMUS and consider the possibility of approximate solutions. Finally, we also plan to evaluate our approach with other encodings for other problems (e.g., SMT encodings for hybrid planning problems (Cashmore et al. 2016)).

Acknowledgments

This research is partially supported by NSF grant 1812619. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

Alchourrón, C. E.; and Makinson, D. 1985. On the logic of theory change: Safe contraction. *Studia logica* 44(4): 405–422.

Alvarez Melis, D.; and Jaakkola, T. 2018. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*, 7775–7784.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A Compilation of the Full PDDL+ Language into SMT. In *ICAPS*, 79–87.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2019. Balancing Explicability and Explanations in Human-Aware Planning. In *IJCAI*, 1335–1343.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*, 156–163.

Davies, J.; and Bacchus, F. 2011. Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. In *CP*, 225–239.

Davis, M.; Logemann, G.; Donald; and Loveland. 1962. A Machine Program for Theorem Proving. *Communications of the ACM* 5(7): 394–397.

de Kleer, J.; Mackworth, A. K.; and Reiter, R. 1992. Characterizing Diagnoses and Systems. *Artificial Intelligence* 56(2-3): 197–222.

Dong, Y.; Su, H.; Zhu, J.; and Zhang, B. 2017. Improving interpretability of deep neural networks with semantic information. In *CVPR*, 4306–4314.

Gärdenfors, P. 1986. Belief revisions and the Ramsey test for conditionals. *The Philosophical Review* 95(1): 81–93.

Gärdenfors, P.; Rott, H.; Gabbay, D.; Hogger, C.; and Robinson, J. 1995. Belief Revision. *Computational Complexity* 63(6).

Gilpin, L. H.; Bau, D.; Yuan, B. Z.; Bajwa, A.; Specter, M.; and Kagal, L. 2018. Explaining Explanations: An Overview of Interpretability of Machine Learning. In *DSAA*, 80–89.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A Survey of Methods for Explaining Black Box Models. *ACM Computing Survey* 51(5): 1–42.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Ignatiev, A.; Previti, A.; Liffiton, M. H.; and Marques-Silva, J. 2015. Smallest MUS Extraction with Minimal Hitting Set Dualization. In *CP*, 173–182.

Janota, M.; Lynce, I.; and Marques-Silva, J. 2015. Algorithms for computing backbones of propositional formulae. *AI Communications* 28(2): 161–177.

Kambhampati, S. 1990. A classification of plan modification strategies based on coverage and information requirements. *AAAI Spring Symposium Series* .

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. In *KR*, 374–384.

Kautz, H.; and Selman, B. 1992. Planning as Satisfiability. In *ECAI*, 359–363.

Langley, P. 2016. Explainable agency in human-robot interaction. *AAAI Fall Symposium Series* .

Li, C. M.; and Manya, F. 2009. MaxSAT, Hard and Soft Constraints. *Handbook of Satisfiability* 185: 613–631.

Liffiton, M. H.; Previti, A.; Malik, A.; and Marques-Silva, J. 2016. Fast, flexible MUS enumeration. *Constraints* 21(2): 223–250.

Liffiton, M. H.; and Sakallah, K. A. 2008. Algorithms for computing minimal unsatisfiable subsets of constraints. *Journal of Automated Reasoning* 40(1): 1–33.

Marques-Silva, J.; Heras, F.; Janota, M.; Previti, A.; and Belov, A. 2013. On Computing Minimal Correction Subsets. In *IJCAI*, 615–622.

Miller, T. 2018. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267: 1–38.

Petkovic, D.; Altman, R.; Wong, M.; and Vigil, A. 2018. Improving the explainability of random forest classifier–user centered approach. *Pacific Symposium on Biocomputing* 23: 204–215.

Previti, A.; and Marques-Silva, J. 2013. Partial MUS Enumeration. In *AAAI*, 818–825.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1): 57–95.

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.; and Son, T. C. 2020. On the Relationship Between KR Approaches for Explainable Planning. In *ICAPS Workshop on XAIP*.

Zhang, Y.; Sreedharan, S.; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan explicability and predictability for robot task planning. In *ICRA*, 1313–1320.