Journal of Artificial Intelligence Research 73 (2022) 1473-1534 Submitted 10/2021; published 04/2022

A Logic-Based Explanation Generation Framework for Classical and Hybrid Planning Problems

Stylianos Loukas Vasileiou V.STYLIANOS@WUSTL.EDU
William Yeoh WYEOH@WUSTL.EDU
Washington University in St. Louis
Saint Louis, MO 63130, United States

Tran Cao Son TSON@CS.NMSU.EDU
New Mexico State University
Las Cruces, NM 88003, United States

Ashwin Kumar ASHWINKUMAR@WUSTL.EDU
Washington University in St. Louis
Saint Louis, MO 63130, United States

Michael Cashmore MICHAEL.CASHMORE@STRATH.AC.UK
University of Strathclyde
Glasgow G1 1XH, United Kingdom

Daniele Magazzeni DANIELE.MAGAZZENI@KCL.AC.UK
King's College London
London WC2B 4BG, United Kingdom

Abstract

In human-aware planning systems, a planning agent might need to explain its plan to a human user when that plan appears to be non-feasible or sub-optimal. A popular approach, called *model reconciliation*, has been proposed as a way to bring the model of the human user closer to the agent's model. To do so, the agent provides an explanation that can be used to update the model of human such that the agent's plan is feasible or optimal to the human user. Existing approaches to solve this problem have been based on *automated planning* methods and have been limited to *classical planning* problems only.

In this paper, we approach the model reconciliation problem from a different perspective, that of *knowledge representation and reasoning*, and demonstrate that our approach can be applied not only to classical planning problems but also *hybrid systems planning* problems with durative actions and events/processes. In particular, we propose a logic-based framework for explanation generation, where given a knowledge base KB_a (of an agent) and a knowledge base KB_h (of a human user), each encoding their knowledge of a planning problem, and that KB_a entails a query q (e.g., that a proposed plan of the agent is valid), the goal is to identify an explanation $\epsilon \subseteq KB_a$ such that when it is used to update KB_h, then the updated KB_h also entails q. More specifically, we make the following contributions in this paper: (1) We formally define the notion of logic-based explanations in the context of model reconciliation problems; (2) We introduce a number of cost functions that can be used to reflect preferences between explanations; (3) We present algorithms to compute explanations for both classical planning and hybrid systems planning problems; and (4) We empirically evaluate their performance on such problems. Our empirical results demonstrate that, on classical planning problems, our approach is faster than the state of the art when the explanations are long or when the size of the knowledge base is small

(e.g., the plans to be explained are short). They also demonstrate that our approach is efficient for hybrid systems planning problems.

Finally, we evaluate the real-world efficacy of explanations generated by our algorithms through a controlled human user study, where we develop a proof-of-concept visualization system and use it as a medium for explanation communication.

1. Introduction

From its inception, *Explainable AI Planning* (XAIP) (Fox, Long, & Magazzeni, 2017; Kambhampati, 2019) has garnered increasing interest due to its role in designing explainable systems that bridge the gap between theoretical and algorithmic planning literature and real-world applications (Sreedharan, Chakraborti, & Kambhampati, 2020). The primary motivation of XAIP systems has been revolved around creating well integrated pipelines that can generate explanations for a given planning problem, such as explaining the optimality of a given plan. An ideal XAIP pipeline typically consists of two main components: (i) *Explanation generation*; and (ii) *Explanation communication*.

When designing XAIP systems, one of the main considerations, particularly in the explanation generation component, is taking into account the persona of the explainee (Langley, 2019). Although there can be a variety of different personalities,[1] the personality of the *end user*, that is, the person interacting/collaborating with the system in the form of a user,[2] has gained a lot of focus. This is an important and challenging personality to consider as it is widely accepted that human users often come with their own preconceived notions and/or expectations of the system (Carroll & Olson, 1988) and, as such, human users might evaluate plans on their own models, which may disagree with the system's outcome or quality. One of the recurring themes in this context is the *model reconciliation problem* (MRP) (Chakraborti, Sreedharan, Zhang, & Kambhampati, 2017) – a paradigm that utilizes a popular theory in human psychology, called the *theory of mind* (Premack & Woodruff, 1978) and allows an agent (the explainer) to consider the "mental model" of the human user (the explainee) in its explanation generation process.[3] These explanations, also referred to as *model-based explanations*, are centered on explaining a plan to a user by transferring a minimum number of updates from the agent's model to the user's model, i.e., they bring the model of the user closer to the agent's model (Chakraborti et al., 2017; Sreedharan, Chakraborti, & Kambhampati, 2018). However, a common thread across most works in the MRP literature is that they, not surprisingly, employ mostly automated planning approaches. Further, to the best of our knowledge, they have been applied to classical planning problems only thus far.

To that extent, in this paper we are mainly interested in the *explanation generation* component of XAIP, specifically through the lens of model reconciliation, where we approach it from a different perspective – one based on *knowledge representation and reasoning* (KR). In particular, we propose a logic-based framework for explanation generation, where given

1. The current norm in the XAIP literature considers the following three personas: *End user*, *domain designer*, and *algorithm designer* (Sreedharan et al., 2020).
2. The users can have varying levels of knowledge and expertise, from dilettantes to cognoscenti.
3. In this context, a mental model is just the user's version of the problem that the agent possess, and interestingly, it can be expressed as a graph, a planning model, or even a logical knowledge base. Note that the notion of MRP is agnostic to the actual representation.

a knowledge base KB_a (of an agent) and a knowledge base KB_h (of a human user), each encoding their knowledge of a planning problem, and that KB_a entails a query q (e.g., that a proposed plan of the agent is valid or that the proposed plan is optimal), the goal is to identify an explanation $\epsilon \subseteq KB_a$ such that when it is used to update KB_h, then the updated KB_h also entails q. We then demonstrate that our approach can be applied not only to classical planning problems but also hybrid systems planning problems with durative actions, processes, and events. More specifically,

- We formally define the notion of logic-based explanations in the context of model reconciliation problems.
- We introduce a number of cost functions that can be used to reflect preferences between explanations.
- We present algorithms to compute explanations for both classical and hybrid systems planning problems.
- We empirically evaluate their performance against the current state of the art (Chakraborti et al., 2017) on classical planning problems as well as provide results on hybrid systems planning problems. Our empirical results demonstrate that, on classical planning problems, our approach is faster than the state of the art when the explanations are long or when the size of the knowledge base is small (e.g., the plans to be explained are short). They also demonstrate that our approach is efficient for hybrid systems planning problems.

In summary, our proposed framework advances the state of the art in model reconciliation approaches for explanation generation within XAIP along two key dimensions: (1) It improves the scalability for some types of classical planning problems; and (2) It generalizes the model reconciliation approach such that it can be applied to other types of planning problems beyond classical planning.

At the other end of the spectrum, explanation generation frameworks aimed at human users should be able to effectively communicate explanations to them, ideally, in a manner that minimizes their cognitive effort and maximizes their cognitive effect. To address this important and challenging task, we consider a *proof-of-concept* for communicating explanations to human users, where we use visualizations and text as a medium for presenting explanations. Our empirical results on a controlled human user study support our expectation that explanations in the form of model reconciliation constitute an effective way for explaining plans to human users for problems beyond classical planning.

The paper is organized as follows. In the next section, we elucidate the role of the theory of mind and its usefulness in explanations as model reconciliation, as well as why we characterize the model reconciliation problem through the lens of logic. We then provide the necessary background knowledge in Section 3. In Section 4, we describe our logic-based framework as well as define two types of explanations – plan validity and plan optimality explanations. We provide a working example that highlights concepts used in our approach in Section 5. In Section 6, we describe algorithms for computing explanations in planning problems, and experimentally evaluate them on a set of classical and hybrid systems planning problems in Section 7. Then, in Section 8, we shift our focus to the communicability of explanations, where we describe and evaluate a proof-of-concept for presenting explanations to human users. In Section 9, as our explanation generation framework builds upon

various logic-based techniques, we give a detailed exposition on the relationship between such techniques and their applicability to XAIP, as well as discuss related work from the planning literature. We finally conclude the paper in Section 10.

2. Explanations as Model Reconciliation

The *theory of mind* (ToM) (Premack & Woodruff, 1978) is an important theory about the operations of the human mind and behavior in social and collaborative (or even adversarial) scenarios. In a nutshell, ToM is the ability to attribute mental models to others while recognizing that these models may differ from one's own. These mental models, which comprise mental states such as beliefs, knowledge, intentions, etc. (in other words, a full range of goal and epistemic states), allow one to infer future mental states (i.e., the behavior) of others. However, social interactions can be quite convoluted, and misinterpretations may even yield frantic results. Nonetheless, being able to attribute mental models to other people, for example, ideas about what other people are thinking or know about certain situations, would make social interactions placid and seamless, at least to some reasonable extent. For instance, building shared plans or goals between two people requires the very essence of ToM. Both parties must recognize the intentions of one another and subsequently work out how to mesh their actions with each other in order to achieve a common goal. However, note that in order to verbalize and intentionally communicate any differences in mental states (e.g., differences between actions), such as to provide *explanations* intending to update the receivers knowledge, it is normally assumed that the parties involved in the interaction share some common language and vocabulary (i.e., their mental models are expressed in common terms). ToM, therefore, is viewed as a vital socio-cognitive skill, inherent in the human nature, that we tend to highly use in an intuitive and natural way when interacting with other people. For a comprehensive description on the evolution and significance of ToM, we refer the interested reader to the work by Baron-Cohen (1999).

The *model reconciliation problem* (MRP) (Chakraborti et al., 2017) has gained a lot of success due to the fact that it is rooted in the understanding of the importance of ToM. To be more precise, in the context of planning and MRP, a mental model consists simply of a PDDL expression that characterizes a planning problem (i.e., the model comprise all the fluents, predicates, objects, and actions that are allowed to be used in the particular problem). Important to note here are the assumptions that the agent possesses the human user's model a-priori,[4] the agent's model is correct and complete, and only the human user's model may contain flaws or missing information.[5] In a typical MRP scenario, explanation generation is requested when a plan that is optimal (e.g., a shortest plan) in the agent's model is inexplicable (e.g., infeasible or suboptimal) in the human user's model, because the human user is, say, missing some preconditions from some actions in their model that are necessary for the optimal solution of the planning problem. Then, the agent, by taking into account its own model as well as the human user's model, attempts to "reconcile" their differences by providing information from its own model (e.g., the missing preconditions)

4. However, there has been some interest in relaxing this assumption (Sreedharan, Hernandez, Mishra, & Kambhampati, 2019).
5. By correct and complete model, we mean that the agent believes that its model represents the objective and absolute truth about the specific planning problem.

such that when this information is used by the human user to update their model (i.e., by adding the preconditions to the respective actions in their model), they can compute the optimal plan and, hence, understand its optimality.

As we can see, a key point to note in MRP is that the agent recognizes that the human user may have their own model of the planning problem, and that if there exists a discrepancy between their models such that the agent's plan is inexplicable to the human user, explanations will be couched in terms of model differences. Therefore, explanations as model reconciliation have the potential to play a significant role in explanation generation settings, mostly because of their natural consideration of how humans interact in social settings such as those that require intensionally communicating information with one another.

Unsurprisingly, researchers have empirically demonstrated explanations in the form of model reconciliation constitute a natural and effective way of explaining classical planning problems to human users (Chakraborti, Sreedharan, Grover, & Kambhampati, 2019b; Zahedi, Olmo, Chakraborti, Sreedharan, & Kambhampati, 2019). Specifically, they showed, using map visualizations of a planning problem, that human users not only understand explanations in the form of model reconciliation, but also believe that such explanations are necessary to explain (classical planning) plans. This empirical algorithm-agnostic assessment provides some supporting evidence for the real-world applicability of our proposed explanation generation framework for classical planning problems. Nevertheless, the applicability of explanations as model reconciliation for hybrid systems planning problems remains suspect, to the best of our knowledge. As such, in Section 8, we investigate, through a user study, to what extend explanations as model reconciliation are effective for hybrid systems planning problems.

2.1 A Logical Approach to the Model Reconciliation Problem

Having painted a small picture about the usefulness of explanations in MRP scenarios, our interest in this paper lies in characterizing MRP from the lens of a logic, where we specifically lay the theoretical and algorithmic foundations for a logic-based explanation generation framework. Succinctly, the mental models in our approach are in essence knowledge bases consisting of formulae expressed in some type of logic and fully describe a planning problem. For example, it is well known that a classical planning problem can be encoded as a propositional satisfiability instance (SAT) consisting of formulae that represent the initial state, goal state, and the action dynamics for n time steps, where n is an upper bound on the horizon of the problem, and is typically the length of the plan that can be found in the knowledge base (Kautz & Selman, 1992). In a similar fashion, a hybrid system planning problem can be expressed in first-order logic interpreted in the quantifier-free linear real arithmetic theory (Cashmore, Fox, Long, & Magazzeni, 2016). Nevertheless, we wish to emphasize that the logical nature of a logic-based framework offers a number of attractive features that, to our understanding, are desirable, if not necessary, in systems incorporating explanation generation capabilities. Some of important features worth mentioning are: (i) *Expressivity*; and (ii) *Traceability*.

First, by *expressivity*, we refer to the expressive power of logical languages, that is, their ability to describe various phenomena in a principled and axiomatic way, and their ability

to distinguish between certain structures defined in them. For example, the propositional logical language is defined over a finite set of propositions $P = \{p_1, \ldots, p_n\}$, and the structures are defined as the truth assignment models $M = \{\mu_1, \ldots, \mu_k\}$ of P.[6] In the case of a classical planning problem Π, each p_i describes a state, an action, how to transition between states, and so on, up to a fixed horizon n. Additionally, every μ_j consists of Boolean truth values for each p_i (i.e., μ_j describes which states and actions are true (or false) at a particular horizon during the execution of Π). Then, a knowledge base comprising these propositions can be used to explain certain phenomena that happened during the execution of Π.

Next, by *traceability*, we allude to the fact that given a logical description of a problem, we can easily trace the reasons for particular "behavior." For instance, given a knowledge base KB encoding a planning problem Π, a valid plan π of Π is entailed by KB and, as such, the reasons for its validity can be traced using deductive inference algorithms. These reasons, expressed in the type of logic used to encode Π, can be used to explain the validity of π. It is important to mention here that, even though such explanations are expressed in logical formulae, we do not aim to communicate them to human users in their pure form, but rather express them in an easily human-understandable format, such as natural language and visualizations.

Note that our proposed approach can be used as a standalone method for solving planning problems and generating explanations for them. However, it can be integrated with state-of-the-art search-based planners. For example, a search-based planner (e.g., FastDownward (Helmert, 2006)) can be first used to find a plan of length n for a planning problem Π. Then, Π can be encoded into a knowledge base KB up to horizon n, which can be mapped into our framework and used for explanation generation concerning valid (or optimal) plans of length at most n. Such an integration may be useful in real-world applications, where running time efficiency may be of critical importance.

3. Preliminaries

In this section, we provide a brief introduction to logic, satisfiability, classical planning, and hybrid systems planning. We then describe how to cast classical and hybrid systems planning problems as SAT and SMT instances, respectively. Finally, we discuss the notion of explainable AI planning.

3.1 Logic

A *logic* L is a tuple $\langle KB_L, BS_L, ACC_L \rangle$, where KB_L is the set of well-formed knowledge bases (or theories) of L – each being a set of formulae. BS_L is the set of possible belief sets; each element of BS_L is a set of syntactic elements representing the beliefs L may adopt. $ACC_L : KB_L \rightarrow 2^{BS_L}$ describes the *"semantics"* of L by assigning to each element of KB_L a set of acceptable sets of beliefs. For each $KB \in KB_L$ and $B \in ACC_L(KB)$, we say that B is a *model* of KB. A logic is monotonic if $KB \subseteq KB'$ implies $ACC_L(KB') \subseteq ACC_L(KB)$.

6. Note that a propositional logical language offers maximal expressivity with respect to the truth assignments on a finite set P, as for each μ_j there exists a unique formula satisfied by μ_j and falsified by μ_l (for $l \neq j$).

Example 1 *Assume that L refers to the propositional logic over an alphabet P. Then, KB_L is the set of propositional theories over P, $BS_L = 2^P$, and ACC_L maps each theory KB into the set of its models in the usual sense.*

Definition 1 (Skeptical Entailment) *A formula φ in the logic L is* skeptically entailed *by KB, denoted by $KB \models^s_L \varphi$, if $ACC_L(KB) \neq \emptyset$ and $\varphi \in B$ for every $B \in ACC_L(KB)$.*

Definition 2 (Credulous Entailment) *A formula φ in the logic L is* credulously entailed *by KB, denoted by $KB \models^c_L \varphi$, if $ACC_L(KB) \neq \emptyset$ and $\varphi \in B$ for some $B \in ACC_L(KB)$.*

Definition 3 (Consistent Knowledge Base) *A KB is* consistent *iff $ACC_L(KB) \neq \emptyset$ or, equivalently, iff KB does not skeptically entail false.*

For our later use, we will assume that a negation operator $\neg$ over formulae exists. Additionally, φ and $\neg\varphi$ are contradictory with each other in the sense that, for any KB and $B \in ACC_L(KB)$, if $\varphi \in B$, then $\neg\varphi \notin B$; and if $\neg\varphi \in B$, then $\varphi \notin B$. Therefore, if $\{\varphi, \neg\varphi\} \subseteq KB$, then KB is inconsistent, i.e., $ACC_L(KB) = \emptyset$. $\epsilon \subseteq KB$ is called a *sub-theory* of KB. A theory KB *subsumes* a theory KB', denoted by $KB \lhd KB'$, if $ACC_L(KB) \subset ACC_L(KB')$.

3.1.1 Boolean Satisfiability

Boolean Satisfiability (SAT) (Cook, 1971) is the problem of finding an assignment of truth values to variables in order to make a set of propositional formulae true. The problem can be stated as follows: Given a Boolean expression ψ with variables $V = \{v_1, \ldots, v_n\}$, find an assignment to the variables V that satisfies ψ or prove that one does not exist. For example,

$$\psi = (v_1 \vee v_2) \wedge (\neg v_2 \vee v_3) \wedge \neg v_1 \tag{1}$$

is satisfiable with respect to the truth assignment $M = \{v_1 = F, v_2 = T, v_3 = T\}$.

3.1.2 Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT) (Barrett & Tinelli, 2018) is the problem of deciding the satisfiability of a first-order formula expressed in a given theory. The problem can be stated as follows: Given a first order formula ψ with variables $V = \{v_1, \ldots, v_n\}$ and a set of constraints over those variables, find an assignment to the variables V that satisfies ψ or prove that one does not exist. In contrast to the SAT problem, the variables are not restricted to Boolean values, but depend upon a theory, and the constraints are expressed with respect to a background logic. The theory and logic are critical elements of an SMT problem. Theories exist for Boolean propositions, bit-vectors, arrays, integers, reals, and so on. For example, an SMT problem in the quantifier-free linear real arithmetic theory is:

$$\psi = (v_1 + 3 \leq 2v_2) \vee (v_3 + 4 \geq 2) \vee (v_1 + v_2 + v_3 \geq 1), \tag{2}$$

which is satisfiable with respect to the assignment $M = \{v_1 = 1, v_2 = 1, v_3 = 1\}$.

3.2 Classical Planning

A classical planning problem, typically represented in PDDL (Ghallab, Howe, Knoblock, McDermott, Ram, Veloso, Weld, & Wilkins, 1998), 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 $\langle pre_a, \mathit{eff}_a \rangle$, where pre_a are the preconditions of a – conditions that must hold for the action to be applied; and $\mathit{eff}_a = \langle \mathit{eff}_a^+, \mathit{eff}_a^- \rangle$ are the addition (eff_a^+) and deletion (eff_a^-) effects of a – conditions that must hold after the action is applied. More formally, using $\delta_\Pi : 2^F \times A \to 2^F$ to denote the transition function of problem Π, if $s \not\models pre_a$, then $\delta_\Pi(s, a) \models \bot$; otherwise, $\delta_\Pi(s, a) \models s \cup \mathit{eff}_a^+ \setminus \mathit{eff}_a^-$. The solution to a planning problem Π is a plan $\pi = \langle a_1, \ldots, a_n \rangle$ such that $\delta_\Pi(I, \pi) \models G$, where $\delta_\Pi(s, \pi) = \delta_\Pi(\delta_\Pi(s, a_1), \pi')$ with $\pi' = \langle a_2, \ldots, a_n \rangle$.

The cost of a plan π is given by $C(\pi, \Pi) = |\pi|$. Finally, the cost-minimal plan $\pi^* = \text{argmin}_{\pi \in \{\pi' | \delta_\Pi(I, \pi') \models G\}} C(\pi, \Pi)$ is called the optimal plan.

3.2.1 Encoding Classical Planning Problems as Boolean Satisfiability

A classical planning problem can be encoded as a SAT problem (Kautz & Selman, 1992; Kautz, McAllester, & 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 $t \in \{0, \ldots, n-1\}$ and one variable for each fluent $f \in F$ at each timestep $t \in \{0, \ldots, n\}$. We denote the variable representing action a in timestep t using subscript a_t, 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. The formula Φ is a conjunction of the following formulae:

- **Initial state:** Let F and I be the sets of fluents and initial states, respectively, in the planning problem:

$$\bigwedge_{f \in I} f_0 \wedge \bigwedge_{f \in F \setminus \{I\}} \neg f_0 \tag{3}$$

- **Goal state:** Let G be the set of goal states:

$$\bigwedge_{f \in G} f_n \tag{4}$$

- **Action scheme:** Formulae enforcing the preconditions and effects of each action a at time step t:

$$a_t \Rightarrow \bigwedge_{f \in pre_a} f_t \tag{5}$$

$$a_t \Rightarrow \bigwedge_{f \in eff_a^+} f_{t+1} \tag{6}$$

$$a_t \Rightarrow \bigwedge_{f \in eff_a^-} \neg f_{t+1} \tag{7}$$

- **Explanatory frame axioms:** Formulae enforcing that facts do not change between subsequent time steps t and $t+1$ unless they are effects of actions that are executed at time step t:

$$\neg f_t \wedge f_{t+1} \Rightarrow \bigvee \{a_t \mid f \in eff_a^+\} \tag{8}$$

$$f_t \wedge \neg f_{t+1} \Rightarrow \bigvee \{a_t \mid f \in eff_a^-\} \tag{9}$$

- **Action exclusion axioms:** Formulae enforcing that only one action can occur at each time step t:

$$\bigwedge_{a \in A} \bigwedge_{a' \in A | a \neq a'} (\neg a_t \vee \neg a'_t) \tag{10}$$

where A is the set of actions in the planning problem.

Finally, we can *extract* a plan by finding an assignment of truth values that satisfies Φ (i.e., for all time steps $t = 0, \ldots, n-1$, there will be exactly one action a such that $a_t = \mathit{True}$). This could be easily done by using a satisfiability algorithm, such as the well-known DPLL algorithm (Davis, Logemann, Donald, & Loveland, 1962).

It is worth mentioning that planning as SAT has gathered a lot of traction, as there is a significant number of works which have been devoted to formalizing and improving the encodings of planning problems using propositional logic (Robinson, Gretton, Pham, & Sattar, 2009; Domshlak, Hoffmann, & Sabharwal, 2009; Cashmore, Fox, & Giunchiglia, 2012).

3.3 Hybrid Systems Planning

A hybrid system planning problem, hereinafter simply *hybrid planning*, typically represented in PDDL+ (Fox & Long, 2006), is a tuple $\Pi+ = \langle P, V, A, Ps, E, I, G \rangle$, in which P is a set of propositions; V is a vector of real variables (fluents); A is a set of durative and instantaneous actions; Ps is a set of processes; E is a set of events; and I and G are the initial and goal states, respectively. A durative action $a \in A$ is defined by a tuple $\langle pre_a, eff_a, dur_a \rangle$, where pre_a is the precondition, eff_a is the effect, and dur_a is a duration constraint – a conjunction of numeric constraints corresponding to the duration of action a.

In contrast to classical planning, the precondition $pre_a = \langle pre_{\vdash a}, pre_{\leftrightarrow a}, pre_{\dashv a} \rangle$ of a durative action a consists of three disjoint subsets, where each subset represents conditions that must hold at the start of the action, throughout its execution, and at the end of the action, respectively. In turn, the effect $eff_a = \langle eff^{\pm}_{\vdash a}, eff^{num}_{\vdash a}, eff^{\pm}_{\dashv a}, eff^{num}_{\dashv a}, eff_{\leftrightarrow a} \rangle$ of an action a consists of five disjoint subsets, where $eff^{\pm}_{\vdash a}$ is the set of instantaneous effects of adding/removing propositions at the start of the action, $eff^{num}_{\vdash a}$ is the set of instantaneous numeric effects at the start of the action, $eff^{\pm}_{\dashv a}$ is the set of instantaneous effects of adding/removing propositions at the end of the action, $eff^{num}_{\dashv a}$ is the set of instantaneous numeric effects

at the end of the action, and $\mathit{eff}_{\leftrightarrow a}$ is a conjunction of numeric effects which are applied continuously while the action is executing. Note that the values of instantaneous effects can be exploited to support other actions only after a small amount of time ϵ, which is referred to as epsilon separation (Fox & Long, 2003).

Each process $ps \in PS$, defined by a tuple $\langle \mathit{pre}_{ps}, \mathit{eff}_{ps} \rangle$, is similar to a durative action, except that it does not have a set duration but is instead active when their preconditions are satisfied (without any epsilon separation) and inactive when their preconditions are not satisfied. Consequently, unlike durative actions, processes do not have durative constraints. In addition, a process's precondition consists of a single condition, whereas its effect consists of a single continuous numeric effect. Each event $e \in E$, defined by a tuple $\langle \mathit{pre}_e, \mathit{eff}_e \rangle$, is analogous to an instantaneous action and, thus, also does not have a duration constraint. Further, an event comprises of a single triggering precondition and an instantaneous effect. Note that events can make immediate use of effects (without any epsilon separation). If the effect of an action, process, or other event make true the condition of an event, then it occurs immediately and simultaneously with that effect. In general, processes and events are used to model exogenous events in the world (Gerevini, Saetti, & Serina, 2006). Therefore, they are not under the direct control of the planner and are triggered immediately when their preconditions are satisfied (see Bogomolov, Magazzeni, Podelski, and Wehrle (2014) for more details).

Moreover, the cost of a PDDL+ plan depends upon a specified plan metric. Plan metrics assert, for the benefit of the planner, how a plan will be evaluated for a particular problem.[7] For instance, the same initial and goal states might yield entirely different optimal plans given different plan metrics. Examples of plan metrics include the makespan of the plan (i.e., the sum of the duration of each action in the plan), optimizing a specific quantity in the domain, etc.

Finally, it is important to mention that, to the best of our knowledge, there does not exist any general optimal PDDL+ planners. As a matter of fact, it has been shown that even finding the existence of PDDL+ plans is undecidable (Helmert, 2002). The reason is that PDDL+ domains have an established relationship with the *reachability* problem for hybrid automata (Fox & Long, 2006), where it is known to be undecidable (Henzinger, 2000). In the context of planning, the reachability problem corresponds to the problem of solving plan existence. Nonetheless, there exist PDDL+ fragments where plan existence is decidable (Fox & Long, 2006).

3.3.1 Encoding Hybrid Planning Problems as Satisfiability Modulo Theory

A hybrid planning problem $\Pi+$ can be encoded as an SMT formula (Cashmore et al., 2016) with bound n in the theory of quantifier-free (non-linear) real arithmetic with n copies of the set of variables x, where x is called a *happening*. A happening encodes the change in the state at a particular time point due to effects of actions, processes, or events happening at that time point:

$$x = \left\langle t, \hat{E}, \hat{Ps}, \hat{P}, \hat{V}, A, P^+, V^+, \mathit{flow}_V, \mathit{dur}_{Ps} \right\rangle \tag{11}$$

7. Metrics are specified in the problem description, allowing a planner to easily explore the effect of different metrics in the construction of solutions to problems for the same domain.

where t is the current time point, $\hat{E} = \{E_0, \ldots, E_B\}$ is the chain of events triggered at t, B is a bound on the length of the causal chain of events at each time point, $\hat{Ps} = \{Ps_0, \ldots, Ps_B\}$ is the chain of active processes at t, $\hat{P} = \{P_0, \ldots, P_B\}$ is the causal change in the propositional state variables at t, $\hat{V} = \{V_0, \ldots, V_B\}$ is the causal change in the real state variables at t, A is a set of durative and instantaneous actions, P^+ and V^+ are the values of the propositional and real state variables, respectively, at time $t + \epsilon$, $flow_V = \{flow_v \mid v \in V\}$ is a numerical expression that represents the change in value of v from a time point to the next, and $dur_{Ps} = \{dur_{ps} \mid ps \in Ps\}$ is the remaining duration of each process ps. Note that the dur_{Ps} variable enforces the duration constraint of a durative action, not that of a process, as we highlight in the next paragraph.

Following the same manner as in the encoding of classical planning, the SMT formula is comprised of a conjunction of formulae that represent the dynamics of the given Π+ problem. Then, a plan for Π+ with length n would correspond to the action variables with true assignments in any proof of the SMT formula of Π+. In order to encode a durative action in SMT, we split it into two instantaneous actions representing the start and end of the action respectively, and a process representing the action's durative portion. The start and end actions are constructed in a straightforward way, with the addition of a new effect, whose only purpose is to activate the process. The effect of the process is the continuous effect of the durative action, plus a continuous decrement of some timer variable. The instant end action uses that timer variable and the durative action's duration inequality in its precondition. This ensures that the start and end of the action are the correct distance apart in the timeline. It is only this kind of process, representing a durative action, that has a duration associated with it.

Below, we describe the SMT formulae that characterize a Π+ problem (for a more thorough description, we refer the reader to Cashmore, Magazzeni, and Zehtabi (2020)). Formulae (12) to (25) encode the constraints for each happening $x_0, \ldots, x_n$, and formulae (26) to (36) are additional constraints needed in the SMT formula Π+:

- **Proposition and real variable support:** Formulae ensuring that the values of propositions and real variables remain consistent from $P_0 \cup V_0$ to $P_B \cup V_B$:

$$\bigwedge_{i=0}^{B-1} \bigwedge_{p \in P} p_{i+1} \rightarrow (p_i \bigvee_{e \mid p \in eff_e^+} e_i) \tag{12}$$

$$\bigwedge_{i=0}^{B-1} \bigwedge_{p \in P} \neg p_{i+1} \rightarrow (\neg p_i \bigvee_{e \mid p \in eff_e^-} e_i) \tag{13}$$

$$\bigwedge_{i=0}^{B-1} \bigwedge_{v \in V} (\bigwedge_{e \mid v \in eff_e^{num}} \neg v_i) \rightarrow (v_{i+1} = v_i) \tag{14}$$

- **Event preconditions and effects:** Formulae enforcing that an event is triggered if and only if its preconditions hold, and that if an event is triggered, its effects are present in the next time step:

$$\bigwedge_{i=0}^{B-1} \bigwedge_{e \in E} e_i \rightarrow (pre_e)_i \tag{15}$$

$$\bigwedge_{i=0}^{B-1} \bigwedge_{e \in E} e_i \rightarrow (\mathit{eff}_e)_{i+1} \tag{16}$$

- **Action preconditions and effects:** Formulae enforcing that an action's preconditions must hold in $P_B \cup V_B$ and their effects are enforced in $P^+ \cup V^+$:

$$\bigwedge_{a \in A} a \rightarrow (\mathit{pre}_a)_B \tag{17}$$

$$\bigwedge_{a \in A} a \rightarrow (\mathit{eff}_a)^+ \tag{18}$$

- **Support across epsilon separation:** Formulae ensuring that the values of propositions and real variables remain consistent from $P_B \cup V_B$ to $P^+ \cup V^+$:

$$\bigwedge_{p \in P} p^+ \rightarrow (p_B \bigvee_{a | p \in \mathit{eff}_a^+} a) \tag{19}$$

$$\bigwedge_{p \in P} \neg p^+ \rightarrow (\neg p_B \bigvee_{a | p \in \mathit{eff}_a^-} a) \tag{20}$$

$$\bigwedge_{v \in V} (\bigwedge_{a | v \in \mathit{eff}_a^{num}} \neg a) \rightarrow (v^+ = v_B) \tag{21}$$

- **Process triggering:** Formulae enforcing that a process is active if and only if its preconditions are satisfied in each set $P_0 \cup V_0$ to $P_B \cup V_B$, and ensuring that a process cannot finish outside of a happening:

$$\bigwedge_{i=0}^{B} \bigwedge_{ps \in Ps} ps_i \leftrightarrow (\mathit{pre}_{ps})_i \tag{22}$$

$$\bigwedge_{ps \in Ps} \mathit{dur}_{ps} \geq 0 \tag{23}$$

$$\bigwedge_{ps \in Ps} ps_B \leftrightarrow (\mathit{dur}_{ps} > 0) \tag{24}$$

- **Action exclusion axioms:** Formulae enforcing that only one action can occur at each time step:

$$\bigwedge_{a \in A} \bigwedge_{a' \in A | a \neq a'} (\neg a \vee \neg a') \tag{25}$$

- **Instance description:** Formulae enforcing that the initial state holds in the first happening and the goal holds in the final happening:

$$I_0 \tag{26}$$

$$G_n \tag{27}$$

$$t_0 = 0 \tag{28}$$

$$\bigwedge_{i=1}^{n} t_i \geq t_{i-1} + \epsilon \tag{29}$$

- **Proposition support:** Formulae ensuring that the discrete state variables P do not change between happenings:

$$\bigwedge_{i=1}^{n} \bigwedge_{p \in P} (p_0)_i \rightarrow (p^+)_{i-1} \tag{30}$$

$$\bigwedge_{i=1}^{n} \bigwedge_{p \in P} \neg(p_0)_i \rightarrow \neg(p^+)_{i-1} \tag{31}$$

- **Invariants:** Formulae ensuring that the continuous numeric change between happenings is valid:

$$\bigwedge_{i=1}^{n} \bigwedge_{ps \in Ps} (ps_B)_{i-1} \rightarrow ((dur_{ps})_i) = (dur_{ps})_{i-1} + t_i - t_{i+1}) \tag{32}$$

$$\bigwedge_{i=0}^{n-1} \bigwedge_{ps \in Ps} (ps_B)_i \leftrightarrow (pre_{\leftrightarrow ps})_i \tag{33}$$

$$\bigwedge_{i=0}^{n-1} \bigwedge_{e \in E} \neg(pre_{\leftrightarrow e})_i \tag{34}$$

- **Continuous change on real variables:** Formulae enforcing the continuous change on real variables:

$$\bigwedge_{i=0}^{n-1} \bigwedge_{v \in V} (flow_v)_i = \int_{t_i}^{t_{i+1}} \bigcup_{ps \in Ps} (\mathit{eff}^{num}_{\leftrightarrow ps}[v])_i \; dt \tag{35}$$

$$\bigwedge_{i=1}^{n} \bigwedge_{v \in V} ((v_0)_i = (v^+)_{i-1} + (flow_v)_{i-1}) \tag{36}$$

As mentioned at the end of Section 3.3, there does not exist any optimal PDDL+ planners. Consequently, SMT solvers fall within this category as well. For example, in a temporal setting, a lower number of happenings does not mean a higher-quality plan. It could be that by adding happenings, a plan of shorter duration, or with better cost, can be found. As an example, consider a domain with a car, actions to increase and decrease acceleration by one step, and a goal to move the car a given distance. The optimal plan in terms of duration will be to accelerate as many times as possible until the half-way point, and then to decelerate until the car stops at the specified distance. A plan with the fewest steps/happening only accelerates and decelerates once each, and takes much longer. Therefore, a solution found by an SMT solver is not guaranteed to be optimal with respect to time.

3.4 Explainable AI Planning

In *Explainable AI Planning* (XAIP) (Kambhampati, 2019), the (planning) agent must have knowledge of the human model in order to be able to contemplate the goals of humans as well as foresee how its plan will be perceived by them. This is of the highest importance

in the context of explainable planning since an explanation of a plan cannot be *one-sided* (i.e., it must incorporate the human's beliefs of the planner). In a plan generation process, a planner performs argumentation over a set of different models (Chakraborti, Kambhampati, Scheutz, & Zhang, 2017). These models are usually the model of the agent incorporating the planner, the model of the human in the loop, the model that the agent thinks the human has, the model that the human thinks the agent has, and the agent's approximation of the latter. Therefore, the necessity for plan explanations arises when the model of the agent and the model the human diverge so that the optimal plan in the agent's model is inexplicable to the human.

During a collaborative activity, an explainable planning agent (Fox et al., 2017) must be able to account for such model differences and maintain an explanatory dialogue with the human so that both of them agree on the same plan. This forms the nucleus of explanation generation of an explainable planning agent, and is referred to as *model reconciliation* (Chakraborti et al., 2017).

Human-aware planning, as introduced by Chakraborti, Sreedharan, and Kambhampati (2019c), 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 human's expectations, the agent attempts a *model reconciliation* process. In this process, the agent provides an explanation that can be used to update the human's model such that the agent's plan is also optimal in the updated human's model.

More formally, a *Model Reconciliation Problem* (MRP) is defined by the tuple $\Psi = \langle \Phi, \pi \rangle$, where $\Phi = \langle M^R, M_H^R \rangle$ is a tuple of the agent's model $M^R = \langle D^R, I^R, G^R \rangle$ and the agent's approximation of the human's model $M_H^R = \langle D_H^R, I_H^R, G_H^R \rangle$, and π is the optimal plan in M^R. A solution to an MRP is an explanation ϵ such that when it is used to update the human's model M_H^R to $\widehat{M}_H^{R,\epsilon}$, the plan π is optimal in both the agent's model M^R and the updated human model $\widehat{M}_H^{R,\epsilon}$. The goal is to find a cost-minimal explanation, where the cost of an explanation is defined as the length of the explanation by Chakraborti et al. (2017); We later propose several other possible cost explanation functions in Section 4.1. Additionally, while the model reconciliation problem is only defined for optimal plans (Chakraborti et al., 2017), it can be generalized for any arbitrary plan of the agent, which we do so in Section 4.2.

4. Explanation Generation Framework

In this section, we introduce the notion of an explanation in the following setting, where, for brevity, we use the term $\models_L^x$ for $x \in \{s, c\}$ to refer to skeptical (s) or credulous (c) entailment:

> **Explanation Generation Problem:** Given two knowledge bases KB_a and KB_h and a formula φ in a logic L, where $KB_a \models_L^x \varphi$ and $KB_h \not\models_L^x \varphi$, the goal is to identify an explanation (i.e., a set of formulae) $\epsilon \subseteq KB_a$ such that when it is used to *update* KB_h to $\widehat{KB}_h^{\epsilon}$, the updated $\widehat{KB}_h^{\epsilon} \models_L^x \varphi$.

When updating a knowledge base KB with an explanation ϵ, the updated knowledge base $KB \cup \epsilon$ may be inconsistent as there may be contradictory formulae in KB and ϵ.

As such, to make the knowledge base consistent again, one needs to remove this set of contradictory formulae $\gamma \subseteq KB$ from KB. More formally:

Definition 4 (Knowledge Base Update) *Given a knowledge base KB and an explanation ϵ, the updated knowledge base is $\widehat{KB}^{\epsilon} = KB \cup \epsilon \setminus \gamma$, where $\gamma \subseteq KB \setminus \epsilon$ is a set of formulae that must be removed from KB such that the updated $\widehat{KB}^{\epsilon}$ is consistent.*[8]

We now define the notion of a *support* of a formula w.r.t. a knowledge base KB before defining the notion of *explanations*.

Definition 5 (Support) *Given a knowledge base KB and a formula φ in a logic L, where $KB \models^x_L \varphi$, $\epsilon \subseteq KB$ is a* support *of φ w.r.t. KB if $\epsilon \models^x_L \varphi$. Assume that ϵ is a support of φ w.r.t. KB. We say that $\epsilon \subseteq KB$ is a* $\subseteq$-minimal support *of φ if no proper sub-theory of ϵ is a support of φ. Furthermore, ϵ is a* $\lhd$-general support *of φ if there is no support ϵ' of φ w.r.t. KB such that ϵ subsumes ϵ'.*

Definition 6 (Explanation) *Given two knowledge bases KB_a and KB_h and a formula φ in a logic L, where $KB_a \models^x_L \varphi$ and $KB_h \not\models^x_L \varphi$, an* explanation for φ from KB_a for KB_h *is a support ϵ w.r.t. KB_a for φ such that the updated knowledge base $\widehat{KB}^{\epsilon}_h \models^x_L \varphi$, where $\widehat{KB}^{\epsilon}_h$ is updated according to Definition 4.*

Example 2 *Consider propositional logic theories over the set of propositions $\{a, b, c\}$ with the usual definition of models, satisfaction, etc. Assume $KB_a = \{a, b, a \rightarrow c, a \wedge b \rightarrow c\}$ and $KB_{h_1} = \{a\}$. We have that $\epsilon_1 = \{a, a \rightarrow c\}$ and $\epsilon_2 = \{a, b, a \wedge b \rightarrow c\}$ are two $\subseteq$-minimal supports of c w.r.t. KB_a. Only ϵ_1 is a $\lhd$-general support of c w.r.t. KB_a since $\epsilon_2 \lhd \epsilon_1$. Both ϵ_1 and ϵ_2 can serve as explanations for c from KB_a for KB_{h_1}. Of course, KB_a is itself an explanation for c from KB_a for KB_{h_1}.*

Now consider $KB_{h_2} = \{a, \neg b\}$. In this case, both ϵ_1 and ϵ_2 are possible explanations for c from KB_a for KB_{h_2}, but if ϵ_2 is chosen, then $\neg b$ will need to be removed from KB_{h_2} so that it is consistent according to Definition 4.

4.1 Preferred Explanations

When considering explanatory systems, a natural question that potentially arises would be: *Are all explanations equal*? For example, one would want to differentiate between *trivial* and *non-trivial* explanations. While it might be acceptable in some cases, trivial explanations,[9] which are akin to a parent providing the explanation "because I said so" when asked "why?" by their child, are not preferred in most cases.

Besides computing an explanation ϵ, the agent also needs to present that explanation to the user or, in other words, describe the content of the explanation ϵ to the user. Given knowledge bases KB_a and KB_h and a formula φ, there might be several explanations for φ from KB_a for KB_h. Therefore, an agent might prefer an explanation that requires the

8. Intuitively, one should prefer the set of formula γ that is removed to be as small as possible, though we chose to not require such a restriction here.
9. There might be cases where we need to explain an assumption or a fact that is missing from a KB, and therefore, trivial explanations will be succinct and acceptable.

least amount of effort[10] in presenting explanation ϵ to the human. One way to characterize the effort of the agent when presenting an explanation is to associate a cost to the elements of explanation ϵ. For example, one might prefer a subset-minimal explanation or a shortest length explanation over others. Next, we quantify the cost of an explanation, which is then used in to define a general preference relation over explanations.

We assume a cost function $\mathcal{C}_L$ that maps knowledge bases and sets of explanations to non-negative real values:

$$\mathcal{C}_L : KB_L \times \Omega \rightarrow \mathcal{R}^{\geq 0} \tag{37}$$

where Ω is the set of explanations and $\mathcal{R}^{\geq 0}$ denotes the set of non-negative real numbers. Intuitively, this function can be used to characterize different complexity measurements of an explanation. A cost function $\mathcal{C}_L$ is *monotonic* if for any two explanations $\epsilon_1 \subseteq \epsilon_2$, $\mathcal{C}_L(KB, \epsilon_1) \leq \mathcal{C}_L(KB, \epsilon_2)$. $\mathcal{C}_L$ induces a preference relation $\prec_{KB}$ over explanations as follows.

Definition 7 (Preferred Explanation) *Given a cost function $\mathcal{C}_L$, a knowledge base KB_h, and two explanations ϵ_1 and ϵ_2 for KB_h, explanation ϵ_1 is* preferred *over explanation ϵ_2 w.r.t. KB_h (denoted by $\epsilon_1 \preceq_{KB_h} \epsilon_2$) iff*

$$\mathcal{C}_L(KB_h, \epsilon_1) \leq \mathcal{C}_L(KB_h, \epsilon_2) \tag{38}$$

and ϵ_1 is strictly preferred *over ϵ_2 w.r.t. KB_h (denoted by $\epsilon_1 \prec_{KB_h} \epsilon_2$) if*

$$\mathcal{C}_L(KB_h, \epsilon_1) < \mathcal{C}_L(KB_h, \epsilon_2) \tag{39}$$

This allows us to compare explanations as follows.

Definition 8 (Most Preferred Explanation) *Given a cost function $\mathcal{C}_L$ and a knowledge base KB_h, an explanation ϵ is a* most preferred explanation *w.r.t. KB_h if there exists no other explanation ϵ' such that $\epsilon' \prec_{KB_h} \epsilon$.*

There are several natural monotonic cost functions. For example:

- $C^1_L(KB_h, \epsilon) = |\epsilon|$, the cardinality of ϵ, indicates the number of formulae that need to be explained;
- $C^2_L(KB_h, \epsilon) = |\epsilon \setminus KB_h|$, the cardinality of $\epsilon \setminus KB_h$, indicates the number of *new* formulae that need to be explained;
- $C^3_L(KB_h, \epsilon) = length(\epsilon)$ indicates the number of literals in ϵ that need to be explained.

4.2 Explanations in Planning Problems

As discussed in the preliminaries section, classical and hybrid planning problems can be encoded using the logic-based SAT and SMT problems, respectively. As such, our logic-based notions of explanations proposed in the previous section can be applied to explainable

10. By "effort," we could use either the effort needed by the robot to present the explanation, the effort needed by the human to understand the explanation, or a combination of both. For example, the length of the explanation can be used to represent both the effort needed by the robot to explain as well as the effort needed by the human to understand.

planning, particularly the model reconciliation problem, in the context of explaining classical and hybrid planning problems. Nonetheless, recall that a model reconciliation problem is strictly defined for explaining optimal plans (see Section 3.4). We can, however, relax this definition and generalize it for arbitrary, valid plans. The reasoning behind this relaxation is that, even if optimality cannot be guaranteed, the user may have doubts about the validity of a plan (i.e., whether the plan is sound and can be executed to achieve the goal). Therefore, valid plan explanations are crucial for engendering trust in the user. Note that, in the case of hybrid planning problems, we restrict ourselves to only explaining valid plans, as guaranteeing optimality is often infeasible for such tasks.

Hereinafter, we focus on the following two problems: (1) Explaining the *validity* of a plan to the user, and (2) Explaining the *optimality* of a plan to the user, where we define them using logical notations. From now on, we use KB_a and KB_h to denote the knowledge bases encoding the planning problem of the planning agent and the human user, respectively.

4.2.1 Plan Validity

Assume π is a valid plan with respect to KB_a but not KB_h. In other words, it is not possible to execute π to achieve the goal with respect to KB_h. For example, an action in the plan cannot be executed because its precondition is not satisfied, an action in the plan does not exist, or the goal is not reached after the last action in the plan is executed. From the perspective of logic, a plan is valid if there exists at least one model in KB_h in which the plan can be executed and the goal is reached:

Definition 9 (Plan Validity) *Given a planning problem Π, a plan π of Π, where α_t is an action of the plan at time step t, and a knowledge base KB_h encoding Π, π is a* valid *plan in KB_h if $KB_h \models^c_L \pi \wedge g_n$, where g_n is the fact corresponding to the goal of the planning problem at time step n.*

4.2.2 Plan Optimality

Assume that π^* is an optimal plan in a model of KB_a. To explain the optimality of π^* to KB_h, we need to prove that no shorter (optimal) plan exists in KB_h. Thus, we need to prove that no shorter plan exists in *all* models of KB_h. This can be easily done by using the notion of skeptical entailment.

Definition 10 (Plan Optimality) *Given a planning problem Π, a plan π of Π with length n, and a knowledge base KB_h encoding Π, the plan π is* optimal *in KB_h if and only if $KB_h \models^c_L \pi \wedge g_n$ and $KB_h \models^s_L \phi$, where $\phi = \bigwedge_{t=0}^{n-1} \neg g_t$ and g_t is the fact corresponding to the goal of the planning problem at time step t.*

In essence, the query ϕ in the above definition is that no plan of lengths 1 to $n-1$ exists. Therefore, when combined with the fact that a plan π of length n that achieves the goal state exists, then that plan must be an optimal plan.

Note that the above definition applies only to classical planning problems and not hybrid planning problems. The reason is because the cost of a hybrid plan depends on a user-specified plan metric, and this cost is not explicitly encoded by SMT encodings of hybrid plans. Nonetheless, we do not view this as a significant loss since finding optimal hybrid plans is often highly intractable.

5. Working Example

For the purpose of clarity and ease of understanding, we have constructed a simplified, classical planning version of the Generator domain[11] as our working example, and use this to demonstrate different concepts explained throughout this paper. We intentionally kept the example very simple so that the explanations are succinct and are easy to present and understand. We refer the interested reader to the appendix where we present an example on the hybrid planning version of the Generator domain.

The domain and problem files are shown in Listings 1 and 2, respectively. The domain consists of two actions gen_on and gen_off. Action gen_on consists of one precondition ($= fuel\_full$), one addition effect ($= gen\_running$), and one deletion effect ($= \neg fuel\_full$). Action gen_off has one precondition ($= gen\_running$) and one addition effect ($= gen\_ran$). In this particular problem, the initial state asserts that the fuel is full ($= fuel\_full$) and the goal state is that the generator has been ran ($= gen\_ran$). The optimal plan for this problem is thus to first execute action gen_on such that $gen\_running$ is true, and then action gen_off such that gen_ran is true. In other words, $\pi^* = [gen\_on_0, gen\_off_1]$.

Listing 1: Domain File of Simple Generator

```
(define (domain Simple-Generator)
(:requirements :strips)
(:predicates (fuel_full)
             (gen_ran)
             (gen_running))
(:action gen_on
    :precondition (and (fuel_full))
    :effect (and (not (fuel_full))
            (gen_running)))

(:action gen_off
    :precondition (and (gen_running))
    :effect (and (gen_ran))))
```

Listing 2: Problem File of Simple Generator

```
(define (problem Sample_Problem)
(:domain Simple-Generator)
(:init (fuel_full))
(:goal (and (gen_ran))))
```

11. The domain is originally defined for hybrid planning problems and it can be found here: https://github.com/KCL-Planning/SMTPlan/tree/master/benchmarks

Given the domain and problem specifications, the knowledge base of the agent KB_a, encoded in propositional logic in the fashion of Kautz et al. (1996), consists of the following set of formulae:

- **Initial states:**

$$fuel\_full_0 \tag{40}$$
$$\neg gen\_ran_0 \tag{41}$$
$$\neg gen\_running_0 \tag{42}$$

- **Goal state:**

$$gen\_ran_2 \tag{43}$$

- **Action** gen_on **preconditions and effects:**

$$gen\_on_i \rightarrow fuel\_full_i \tag{44}$$
$$gen\_on_i \rightarrow \neg fuel\_full_{i+1} \tag{45}$$
$$gen\_on_i \rightarrow gen\_running_{i+1} \tag{46}$$
$$fuel\_full_i \wedge \neg fuel\_full_{i+1} \rightarrow gen\_on_i \tag{47}$$
$$\neg gen\_running_i \wedge gen\_running_{i+1} \rightarrow gen\_on_i \tag{48}$$

- **Action** gen_off **preconditions and effects:**

$$gen\_off_i \rightarrow gen\_running_i \tag{49}$$
$$gen\_off_i \rightarrow gen\_ran_{i+1} \tag{50}$$
$$\neg gen\_ran_i \wedge gen\_ran_{i+1} \rightarrow gen\_off_i \tag{51}$$

- **Action exclusions:**

$$\neg gen\_on_i \vee \neg gen\_off_i \tag{52}$$

for $i = \{0, 1\}$.

Now, let's assume the following knowledge base KB_h for a human user:

- **Initial states:**

$$fuel\_full_0 \tag{53}$$
$$\neg gen\_ran_0 \tag{54}$$
$$\neg gen\_running_0 \tag{55}$$

- **Goal state:**

$$gen\_ran_2 \tag{56}$$

- **Action** *gen_on* **preconditions and effects:**

$$gen\_on_i \rightarrow fuel\_full_i \tag{57}$$
$$gen\_on_i \rightarrow \neg fuel\_full_{i+1} \tag{58}$$
$$gen\_on_i \rightarrow gen\_running_{i+1} \tag{59}$$
$$fuel\_full_i \wedge \neg fuel\_full_{i+1} \rightarrow gen\_on_i \tag{60}$$
$$\neg gen\_running_i \wedge gen\_running_{i+1} \rightarrow gen\_on_i \tag{61}$$

for $i = \{0, 1\}$. That is, the human user is missing the set of formulae that represent action gen_off (Lines 49-51) and the action exclusion of the two actions (Line 52), from her knowledge base.

Due to the omission of that set of formulae, the human user is unaware of the action gen_off and the plan $\pi^* = [gen\_on_0, gen\_off_1]$ is thus not only suboptimal but is also invalid to the human user (specifically, with respect to the user's knowledge base). Therefore, the goal is to explain the validity and/or optimality of π^* to the human user by reconciling the two knowledge bases. Couched in terms of propositional logic, we have to find an explanation ϵ such that $\widehat{KB}_h^{\epsilon} \models_L^c \pi^*$ when explaining the validity of the plan (see Definition 9) and/or $\widehat{KB}_h^{\epsilon} \models_L^s (\neg gen\_ran_0 \wedge \neg gen\_ran_1)$ when explaining the optimality of the plan (see Definition 10).

Now, the set of formulae that yield an explanation for both plan validity and optimality is the set of formulae consisting of action gen_off and the action exclusion axiom of actions *gen_on* and gen_off, i.e.,

$$\epsilon = [gen\_off_i \rightarrow gen\_running_i, gen\_off_i \rightarrow gen\_ran_{i+1}, \tag{62}$$
$$\neg gen\_ran_i \wedge gen\_ran_{i+1} \rightarrow gen\_off_i, \neg gen\_on_i \vee \neg gen\_off_i] \tag{63}$$

for $i = \{0, 1\}$.

Then, after updating KB_h with ϵ using Definition 4 to get the updated $\widehat{KB}_h^{\epsilon} = (KB_h \cup \epsilon) \setminus \emptyset$, the plan π^* is valid (i.e., $\widehat{KB}_h^{\epsilon} \models_L^c \pi^*$) and optimal (i.e., $\widehat{KB}_h^{\epsilon} \models_L^s (\neg gen\_ran_0 \wedge \neg gen\_ran_1)$) to the human user.

6. Computing Explanations

Since finding optimal plans for hybrid planning problems is often infeasible, we focus on describing algorithms that compute explanations for explaining the validity and optimality of classical plans to users in this section. Nonetheless, these algorithms can be trivially adapted to compute explanations for explaining only the validity (but not optimality) of both classical and hybrid plans, which we will describe in Section 6.2.

Our core algorithmic engine is Algorithm 1, which is a general search algorithm that searches through the space of explanations in a best-first manner to find one that is optimal with respect to a given cost function. It takes as inputs two knowledge bases KB_a and KB_h of a logic L, two formulae φ_s and φ_c, and a cost function $\mathcal{C}_L$. The algorithm will output an explanation ϵ such that when it is used to update KB_h to $\widehat{KB}_h^{\epsilon}$ according to Definition 4, the resulting updated knowledge base will credulously entail φ_c (i.e., $\widehat{KB}_h^{\epsilon} \models_L^c \varphi_c$) and skeptically entail φ_s (i.e., $\widehat{KB}_h^{\epsilon} \models_L^s \varphi_s$).

The algorithm makes three assumptions: (1) First, it assumes that KB_a and KB_h encode the version of the same planning problem of the planning agent and human user, respectively. (2) It assumes that KB_a is *correct* and *complete*,[12] and only KB_h can contain errors or omissions. (3) It assumes that $KB_a \models^c_L \varphi_c$ and $KB_a \models^s_L \varphi_s$. The first assumption is reasonable and follows closely the definition of a model reconciliation planning problem (see Section 2 and 3.4). The last two assumptions stem from the fact that the explaining agent bases its explanations on the view (or model) of the specific problem (Miller, 2018). Therefore, the agent should believe that its model KB_a is correct and complete, and that its model correctly and appropriately entails the queries φ_c and φ_s. Together, these three assumptions imply that each erroneous formula in KB_h will have a corresponding correct formula in KB_a, or, more formally:

Definition 11 (Corresponding Formula) *Given two knowledge bases KB_a and KB_h, where each knowledge base encodes the same planning problem Π, a formula φ_h in KB_h has a corresponding formula φ_a in KB_a if both formulae characterize the same action (or state) axiom of Π.*

This is an important property that our algorithm, which we will explain later, will exploit to improve efficiency.

Finally, our algorithm also relies on the existence of an algorithm for checking credulous and skeptical entailment between knowledge bases and formulae (Line 1). For example, one can use the DPLL algorithm for the SAT encoding for classical planning. If KB_h already credulously entail φ_c and skeptically entail φ_s, then there is no need to compute an explanation (Lines 1-2). Otherwise, it goes into a search for an explanation, which is described below.

The key data structures in the algorithm is a priority queue q, initialized to only include the empty set, of potential explanations ordered by their costs (Line 4) and a set *checked* of invalid explanations that have been considered thus far (Line 5). The algorithm repeatedly loops the following steps:

- Move the explanation ϵ with the smallest cost from the priority queue q to *checked* (Lines 7-8).
- Create a copy of KB_h updated with ϵ according to Definition 4 (Line 9).
- Check if the copy $\widehat{KB}^{\epsilon}_h$ credulously entails φ_c and skeptically entails φ_s. If so, return the explanation ϵ (Lines 10-11).
- If not, extend the explanation by 1 (with each formula from $KB_a \setminus KB_h$) and insert the extended explanations into the priority queue q (Lines 12-16). Only the formulae in $KB_a \setminus (KB_h \cap KB_a)$ are considered since formulae that are already in KB_h will not help in the entailment process.

This search process continues until an explanation is found. It is impossible to exhaust all potential explanations and not find a valid explanation. The reason is that, in the worst case, the entire KB_a will serve as an explanation since KB_a credulously and skeptically entail φ_c and φ_s, respectively.

12. By *complete* we mean that the KB encodes the the full planning problem as specified in the PDDL domain.

Algorithm 1: `most-preferred`$(L, KB_a, KB_h, \varphi_s, \varphi_c, \mathcal{C}_L)$

Input: Logic L, KBs KB_a and KB_h, formulae φ_s and φ_c, cost function $\mathcal{C}_L$
Output: A most-preferred explanation w.r.t. $\mathcal{C}_L$ from KB_a to KB_h to skeptically entail φ_s and credulously entail φ_c

```
1  if KB_h ⊨^c_L φ_c and KB_h ⊨^s_L φ_s then
2      return ∅
3  else
4      q = ∅                                   // priority queue of potential explanations
5      checked = ∅                             // a set of sets of elements in KB_a considered
6      repeat
7          ϵ = dequeue(q)
8          insert ϵ into checked
9          KB̂^ϵ_h = KB_h updated with ϵ according to Definition 4
10         if KB̂^ϵ_h ⊨^c_L φ_c and KB̂^ϵ_h ⊨^s_L φ_s then
11             return ϵ
12         else
13             for a ∈ KB_a \ KB_h do
14                 if ϵ ∪ {a} ∉ checked then
15                     v = C_L(KB_h, ϵ ∪ {a})
16                     q = enqueue(ϵ ∪ {a})                    // use v as key
17     until q is empty
```

Note that this algorithm is defined in terms of logic and is agnostic to the underlying planning application domain. However, it can be used to find explanations for the MRP problem in explainable planning by setting $\varphi_c = \pi^* \wedge g_n$ to the optimal plan π^* of length n that needs to be explained and that the goal state is achieved at time step n; and by setting $\varphi_s = \bigwedge_{t=0}^{n-1} \neg g_t$ to the negation of the goal being reached at all time steps before time step n. Then, the algorithm will return a most-preferred explanation, with respect to $\mathcal{C}_L$, that explains that the plan π^* is both valid and optimal.

Constructing the Search Space: Finally, we would like to emphasize on a strategy that can be employed in Algorithm 1 to speed up its search procedure. Notice that the formulae in an encoded knowledge base are often repeated across all time steps (see Section 3.2.1). For instance, consider the example in Section 5. The preconditon ($= fuel\_full$) of the action gen_on in KB_a is repeated across time steps $i = 0, 1$, e.g., $\{gen\_on_0 \rightarrow fuel\_full_0, gen\_on_1 \rightarrow fuel\_full_1\} \in KB_a$. It is straightforward to see that if the knowledge base is encoded at a larger horizon, the search space of the algorithm (i.e., $KB_a \setminus KB_h$, Line 13) will increase in size analogously. However, as the same phenomena should hold across all time steps in a knowledge base encoding a given planning problem, a reasonable and intuitive strategy would be to construct the search space by aggregating the formulae with respect to the time steps, akin to the lifted representation of action dynamics in a PDDL domain. Specifically, the search space can now consist of *macro-formulae*,

e.g., $Pre(gen\_on, fuel\_full) = \{gen\_on_0 \rightarrow fuel\_full_0, gen\_on_1 \rightarrow fuel\_full_1\}$, instead of formulae representing the same action dynamic at each time step.[13]

6.1 Pre-Processing Approximation Algorithm

While Algorithm 1 can compute a most-preferred explanation, it is straightforward to see that the complexity of the problem is at least NP-hard since finding a most-preferred explanation is a combinatorial problem. As the intended use of the algorithm is to provide an explanation that a particular optimal plan π^* is both valid and optimal, we now exploit this assumption and introduce a pre-processing algorithm that can be used to modify KB_h. At a high level, this algorithmic approach can be thought of as "reforming" the knowledge base of the human user in order to make the agent's plan valid.

Before describing the algorithm, we make an observation that there exists only a single model in a knowledge base that is encoding classical and hybrid planning problems that is consistent with a plan π^* of the problem.

We formalize this as Proposition 1 below.

Proposition 1 *Given a plan π^* and a knowledge base KB_a encoding the classical or hybrid planning problem, there exists only a* single *model in KB_a that is consistent with π^*, i.e., $|ACC_L(KB_a \wedge \pi^*)| = 1$.*

Proof. First, observe that both classical and hybrid planning problems are deterministic planning problems without parallel action execution.

Consequently, for classical planning problems, the transition between states is encoded by the frame axioms, which enforce that a state literal becomes True (resp. False) if and only if it was an addition (resp. deletion) effect of an action. The same reasoning can be extended to hybrid planning problems, where state transitions are described by happenings, which encode the causal chain of events, processes and instantaneous actions. Further, due to the action exclusions axioms, only one action can be True at each time step, and hence only the action literals supported by π^* can yield an assignment of True. Therefore, it follows logically that there exists **only one** model consistent with the plan π^*, i.e., $|ACC_L(KB_a \wedge \pi^*)| = 1$. □

Using Proposition 1, one can then generalize that the formulae in KB_h that are false according to this model must be erroneous with respect to KB_a. A trivial approach would be to replace these formulae with their corresponding (correct) formulae from KB_a (see Definition 11) before running Algorithm 1 with this modified KB_h instead of the original KB_h as the input. The (correct) formulae from KB_a as well as the output from Algorithm 1 would then serve as the explanation to KB_h.

However, it is important to note that not all formulae need to be corrected in order for KB_h to credulously entail an optimal plan π^* of length n that reaches the goal at time step n and skeptically entail that the goal cannot be reached before time step n. For example, there may be actions that are not used in the optimal plan with wrong preconditions or effects that need not be corrected. Therefore, we only use a *partial model* – we only extract the

13. This simple idea can be used during the encoding of a planning problem into a logical knowledge base, i.e., by creating a hash table mapping macro-formulae to the associated (repeated) formulae.

Function 1: `extract-partial-model`(L, KB_a, π^*)

Input: Logic L, KB_a, and optimal plan π^*
Output: Partial model from KB_a w.r.t. π^*

18 $\mu = \emptyset$
19 $M =$ `get-SAT-model`(KB_a)
20 $\Lambda =$ `extract-relevant-literals`(KB_a, π^*)
21 **for** $(l, t) \in M$ **do**
22 **if** $l \in \Lambda$ **then**
23 $\mu = \mu \cup \{(l, t)\}$
24 **return** μ

Algorithm 2: `pre-processing`(L, KB_a, KB_h, π^*)

Input: Logic L, KBs KB_a and KB_h, and optimal plan π^*
Output: *Approximated* explanations from KB_a to KB_h

25 $\epsilon = \emptyset$
26 $M =$ `extract-partial-model`(L, KB_a, π^*)
27 **for** $k_h \in KB_h$ **do**
28 **if** $\neg$`evaluate`(L, M, k_h) **then**
29 $\epsilon = \epsilon \cup$ corresponding formula from KB_a
30 **return** ϵ

truth value assignments for literals that are *directly needed for the optimal plan* π^*, i.e., the literals corresponding to the initial and goal states, to the states that are in the precondition of any action in the plan as well as the literals corresponding to all the actions in KB_a. We call this the *relevant literals* with respect to π^*. States that are not preconditions of actions in the plan are not extracted. Function 1 describes the pseudocode of this procedure. First, it initializes the partial model μ as an empty set (Line 18) and extracts the satisfying model M from KB_a that credulously entails the optimal plan π^* of length n and that the goal state is reached only at time step n and not before, where each element of M is a tuple consisting of a literal and its respective truth value (Line 19). Then, it extracts the relevant literals Λ with respect to π^* (Line 20) before it loops through all elements (l, t) in M and adds them to partial model μ if the literal l is a relevant literal (Lines 21-23). Finally, the partial model μ is returned after the loop (Line 24).

It is straightforward to see how this can help speed up the search in Algorithm 1. The pre-processing step will increase the number of formulae that are in both KBs. Consequently, since the search space of Algorithm 1 is the power set of formulae in KB_a that are not in KB_h (see Line 13), our pre-processing step will reduce the search space and runtime of the algorithm. Finally, as long as atomic explanations are formulae in KB_a, the pre-processing step will provide *approximated* explanations, that is, the formulae that are replaced approximate the formulae that are needed for the entailment of the optimal plan. As this method may yield superfluous information, in so far as the explanations may contain formulae not needed for the entailment of the plan, we frame this method as an approximation technique.

Algorithm 2 describes the pseudocode for this pre-processing algorithm.

First, it initializes its set of explanations ϵ as an empty set (Line 25) and extracts a partial model of KB_a that credulously entails the optimal plan π^* of length n and that the goal state is reached only at time step n and not before (Line 26). Then, the algorithm loops through all formulae of KB_h (Line 27) and checks if each formula evaluates to false according to the partial model (Line 28). If it is false, then that formula is replaced with the corresponding formula from KB_a and it is added to the explanation set ϵ (Line 29). The set of *approximated* explanations is returned after the whole loop (Line 30). To illustrate the utility of this procedure, consider the following example.

Example 3 *Assume a version of the Generator domain that consists of two actions* $gen\_on = \{$*precondition:* $fuel\_full$*, effect:* $gen\_running\}$ *and* $gen\_on\_alt = \{$*precondition:* $fuel\_mid$*, effect:* $gen\_running\}$ *with initial and goal states* $fuel\_full$ *and* $gen\_running$*, respectively, and a plan* $\pi^* = [gen\_on]$*. Also, assume that the human user is not aware that action* gen_on *has effect* $gen\_running$*. Then, the knowledge bases encoding the models of the agent and the human are respectively:*

$$KB_a = [fuel\_full_0, \neg fuel\_mid_0, \neg gen\_running_0, gen\_running_1, \tag{64}$$
$$gen\_on_0 \rightarrow fuel\_full_0, gen\_on_0 \rightarrow gen\_running_1, \tag{65}$$
$$gen\_on\_alt_0 \rightarrow fuel\_mid_0, gen\_on\_alt_0 \rightarrow gen\_running_1, \tag{66}$$
$$\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0 \vee gen\_on\_alt_0, \tag{67}$$
$$\neg gen\_on_0 \vee \neg gen\_on\_alt_0] \tag{68}$$

$$KB_h = [fuel\_full_0, \neg fuel\_mid_0, \neg gen\_running_0, gen\_running_1, \tag{69}$$
$$gen\_on_0 \rightarrow fuel\_full_0, gen\_on\_alt_0 \rightarrow fuel\_mid_0, \tag{70}$$
$$gen\_on\_alt_0 \rightarrow gen\_running_1, \tag{71}$$
$$\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on\_alt_0, \neg gen\_on_0 \vee \neg gen\_on\_alt_0] \tag{72}$$

Now, the partial model we extract from KB_a *with respect to* π^* *is:*

$$M = \{fuel\_full_0 = T, fuel\_mid_0 = F, gen\_running_0 = F, gen\_on_0 = T, \tag{73}$$
$$gen\_on\_alt_0 = F, gen\_running_1 = T\} \tag{74}$$

Then, we can see that according to M*, the formula* $\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on\_alt_0$ *from* KB_h *evaluates to false. As such, it would be replaced by the corresponding formula from* KB_a*, namely* $\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0 \vee gen\_on\_alt_0$*.*

Algorithm 3 describes the complete algorithm that uses Algorithm 2 as a pre-processing step. After running the pre-processing algorithm and getting the preliminary set of *approximated* explanations ϵ (Line 32), it creates a copy $\widehat{KB}_h$ with formulae corresponding to ϵ replaced with ϵ (Line 33). Then, it runs Algorithm 1 to find the remaining set of explanations ϵ' (Line 34) and returns the union of those both sets (Line 35). The key observation here is that the input to Algorithm 1 is $\widehat{KB}_h$ and not KB_h. Since $\widehat{KB}_h$ is more similar to

Algorithm 3: `approximate`$(L, KB_a, KB_h, \pi^*, \mathcal{C}_L)$

Input: Logic L, KBs KB_a and KB_h, plan π^* with length n, cost-function $\mathcal{C}_L$
Output: Explanation from KB_a to KB_h to credulously entail π^* and the goal state is reached at time step n, and skeptically entail that the goal state is not reachable before time step n

31 $\phi = \bigwedge_{t=0}^{n-1} \neg g_t$
32 $\epsilon =$ `pre-processing`(L, KB_a, KB_h, π^*)
33 $\widehat{KB}_h = KB_h$ with formulae corresponding to ϵ (if any) replaced with ϵ
34 $\epsilon' =$ `most-preferred`$(L, KB_a, \widehat{KB}_h, \phi, \pi^* \wedge g_n, \mathcal{C}_L)$
35 **return** $\epsilon \cup \epsilon'$

KB_a, the search space of Algorithm 1 will be smaller and it is thus more efficient. It is important to emphasize again that this an *approximation* technique for finding most-preferred explanations of minimal cardinality.

6.2 Modifications for Plan Validity Explanations

Recall that the algorithms described above compute explanations for explaining both the validity and optimality of classical plans to users. However, they can be easily adapted to compute explanations for explaining only the validity (but not optimality) of both classical and hybrid plans. Specifically, the changes are the following:

- For Algorithm 1, one needs to only omit φ_s from the pseudocode. Specifically, φ_s need not be passed in as an argument, and the checks on Lines 1 and 10 need to be changed to only check for the credulous entailment of φ_c.
- For Algorithm 2, no changes are necessary, except that the input plan π^* corresponds to the plan whose validity needs to be explained. It does not have to be an optimal plan.
- For Algorithm 3, similar to the case above, the input plan π^* corresponds to the plan whose validity needs to be explained. Additionally, ϕ need not be passed in as an argument on Line 34 and, consequently, Line 31 can be omitted.

7. Experimental Evaluation

We now describe the empirical evaluations of our algorithms for finding explanations on classical and hybrid planning problems, encoded as SAT and SMT problems, respectively.

Setup and Prototype Implementation: We ran our experiments on a Macbook Pro comprising an Intel Core i7 2.6GHz processor with 16GB of memory. We implemented our algorithms in Python and integrated the well known *z3* solver (De Moura & Bjørner, 2008) for satisfiability and entailment checking, which was accessed through the PyZ3 framework.[14] The knowledge bases representing the planning problems were each encoded up to the time step that the optimal (or valid) plan was found. To encode the knowledge bases for classical planning problems, we used our own implementation of the encoding by Kautz et al. (1996), whereas for hybrid planning problems we used the encoding provided in SMTPlan (Cashmore et al., 2016). Further, note that each knowledge base was encoded

14. https://github.com/Z3Prover/z3.

as a hash table (see Footnote 13). The time limit for all experiments was set to 1500s. We have also made our source code available in a publicly-accessible repository.[15]

Our empirical evaluations were tailored around the following three questions:

- **Question 1:** What is the advantage of using the pre-processing method described in Section 6.1?
- **Question 2:** What is the efficacy of our algorithms on classical planning problems?
- **Question 3:** What is the efficacy of our algorithms on hybrid planning problems?

7.1 Question 1: Advantage of Pre-Processing Approach

With this question, we wanted to examine if there is any advantage in using the pre-processing approach introduced in Section 6.1 for finding most-preferred explanations. To do this, we evaluated Algorithm 1 (referred to as ALG1), which does not use the pre-processing method, and Algorithm 3 (referred to as ALG3), which does use the method, to find most-preferred explanations for plan validity and optimality on classical planning benchmarks from the International Planning Competition (IPC).[16] We used the explanation length $|\epsilon|$ as the cost function of the algorithms and incorporated the knowledge base update on Line 9 in Algorithm 1 by using a simple linear search algorithm (See Appendix C for more details).

We used the actual IPC instances as the model of the agent (i.e., KB_a), and tweaked that model and assigned it to be the model of the human user (i.e., KB_h). In order to make a more comprehensive analysis, we considered five different ways to tweak the models, resulting in the following five scenarios.

- **Scenario 1:** We removed one random precondition from every action in the human's model.
- **Scenario 2:** We removed one random effect from every action in the human's model.
- **Scenario 3:** We removed one random precondition and one random effect from every action in the human's model.
- **Scenario 4:** We removed (on average) fifteen random preconditions and effects from every action in the human's model.
- **Scenario 5:** We removed (on average) ten random predicates from the initial states in the human's model.

Table 1 tabulates the length of the explanations $|\epsilon|$ and the runtimes of the two algorithms as well as a third algorithm called CSZK, which we will describe in the next section. We will focus on the comparisons between ALG1 and ALG3 in this section.

Notice that ALG1 and ALG3 yielded comparable runtimes in most instances of Scenarios 1 and 2. This is due to the fact that the tweaked KB_h produced from those scenarios is consistent with all candidate explanations from $KB_a \setminus KB_h$ (i.e., the search space of potential explanations). In other words, there is no need to restore the consistency of KB_h during an update with a candidate explanation. In that case, ALG1 and ALG3 are virtu-

15. https://github.com/YODA-Lab/Explanation-Generation-for-Planning-Problems.
16. https://github.com/potassco/pddl-instances.

Prob.		Scenario 1				Scenario 2				Scenario 3				Scenario 4				Scenario 5			
		$\|\epsilon\|$	CSZK	ALG1	ALG3	$\|\epsilon\|$	CSZK	ALG1	ALG3	$\|\epsilon\|$	CSZK	ALG1	ALG3	$\|\epsilon\|$	CSZK	ALG1	ALG3	$\|\epsilon\|$	CSZK	ALG1	ALG3
Blocks-world	4	1	**0.5s**	2.5s	3.0s	2	**0.5s**	1.0s	0.7s	3	2.0s	3.0s	**1.5s**	3	32.0s	29.0s	**16.0s**	2	–	1.5s	**0.7s**
	5	2	**2.5s**	8.0s	8.5s	3	**2.0s**	4.0s	2.5s	5	17.0s	6.0s	**6.0s**	7	–	–	**194.0s**	4	–	4.0s	**2.0s**
	6	1	**1.0s**	25.0s	25.0s	2	**0.5s**	5.5s	5.5s	3	**3.0s**	6.0s	6.0s	4	213.0s	–	**120.0s**	5	–	8.0s	**5.0s**
	8	3	**62.0s**	296.5s	297.0s	3	**1.0s**	30.0s	29.5s	6	869.0s	101.0s	**30.0s**	7	–	–	**203.0s**	5	–	40.0s	**27.0s**
Elev-ator	1	1	0.5s	**0.1s**	**0.1s**	2	1.0s	1.5.s	**0.1s**	2	0.5s	1.0s	**0.1s**	2	1.0s	5.0s	**0.1s**	1	–	**0.1s**	**0.1s**
	10	2	1.5s	0.8s	**0.7s**	2	**0.5s**	1.0s	**0.5s**	3	3.0s	2.5s	**0.4s**	4	57.0s	30.0s	**2.5s**	6	–	5.0s	**0.2s**
	15	2	**1.5s**	2.5s	3.0s	2	**1.0s**	15.0s	13.0s	3	3.0s	10.0s	**2.0s**	4	57.0s	61.0s	**10.0s**	6	–	14.0s	**1.2s**
	19	2	**2.0s**	7.5s	8.0s	2	**0.5s**	25.0s	25.0s	3	**2.5s**	50.0s	10.0s	4	49.0s	123.0s	**20.0s**	14	–	40.0s	**5.0s**
Rover	1	1	**0.5s**	6.5s	6.0s	2	**0.5s**	10.0s	7.0s	4	33.0s	10.0s	**5.0s**	6	–	29.0s	**5.0s**	4	–	**1.5**	**1.5s**
	2	1	**1.0s**	4.0s	4.0s	1	**0.5s**	12.0s	4.0s	4	39.0s	6.0s	**4.5s**	6	–	125.0s	**4.5s**	4	–	2.0s	**1.3s**
	3	1	**0.5s**	7.5s	7.0s	2	**0.5s**	22.0s	7.5s	4	35.0s	16.0s	**7.0s**	6	–	173.0s	**10.0s**	6	–	2.0s	**1.5s**
	4	1	**0.5s**	4.0s	4.0s	1	**0.5s**	4.0s	4.0s	2	**1.5s**	4.5s	4.5s	4	–	33.0s	**4.5s**	10	–	15.0s	**5.5s**
Gripper	1	1	**0.5s**	2.0s	1.5s	2	**0.3s**	3.0s	3.0s	3	**1.5s**	41.0s	40.0s	5	70.0s	201.0s	**45.0s**	4	–	3.0s	**2.0s**
	2	1	**0.5s**	5.0s	5.0s	2	**0.8s**	19.0s	7.0s	3	**2.0s**	122.0s	45.0s	5	73.0s	349.0s	**49.0s**	5	–	15.0s	**6.0s**
	3	1	**0.7s**	5.5s	5.0s	2	**1.0s**	22.0s	7.0s	3	**2.5s**	46.0s	45.0s	5	163.0s	555.0s	**60.0s**	8	–	20.0s	**15.0s**
	4	1	**1.5s**	37.5s	38.0s	2	**3.0s**	224.0s	50.0s	3	**5.0s**	149.0s	50.0s	5	–	700.0s	**80.0s**	11	–	66.0	**28.0s**
Hanoi	1	1	**0.3s**	0.5s	0.5s	1	0.2s	**0.1s**	**0.1s**	1	0.3s	0.5s	**0.1s**	1	0.2s	**0.1s**	**0.1s**	4	–	**0.1s**	**0.1s**
	2	1	**0.3s**	0.5s	0.5s	1	**0.3s**	2.0s	2.0s	1	**0.3s**	5.5s	1.5s	5	7.0s	13.0s	**3.0s**	6	–	2.5s	**0.3s**
	3	1	**0.3s**	3.5s	3.0s	1	**0.4s**	7.0s	4.0s	2	**0.4s**	11.5s	1.5s	6	**10.0s**	31.0s	14.0s	8	–	6.0s	**1.5s**
	4	1	**0.3s**	19.5s	20.0s	1	**0.3s**	21.0s	20.0s	2	**0.4s**	44.0s	15.0s	6	**10.0s**	230.0s	30.0s	10	–	30.0s	**10.0s**
TPP	1	1	0.7s	**0.5s**	**0.5s**	2	0.2s	**0.1s**	**0.1s**	6	23.0s	30.0s	**3.5s**	11	–	100.0s	**20.0s**	2	–	**0.1s**	**0.1s**
	2	1	0.7s	**0.5s**	**0.5s**	2	**0.3s**	3.0s	2.0s	6	24.0s	45.0s	**5.5s**	11	–	404.0s	**27.0s**	4	–	3.0s	**0.3s**
	3	1	**0.7s**	3.5s	3.0s	2	**0.4s**	11.0s	4.0s	6	25.5s	95.0s	**7.0s**	11	–	1200.s	**70.0s**	6	–	9.0s	**1.0s**
	4	1	**0.7s**	21.0s	20.0s	2	**0.3s**	71.0s	20.0s	6	25.0s	124.0s	**15.0s**	11	–	–	**111.0s**	8	–	22.0s	**8.0s**
Driver Log	1	1	**1.0s**	2.0s	2.0s	4	2.5s	2.0s	**1.5s**	2	**2.5s**	11.0s	4.0s	5	–	66.0s	**6.0s**	3	–	5.0s	**2.0s**
	2	2	24.0s	14.0s	**13.0s**	5	**5.0s**	7.5s	7.0s	4	–	239.0s	**63.0s**	7	–	–	**90.0s**	5	–	14.0s	**5.0s**
	3	2	**5.0s**	10.0s	10.0s	3	**0.5s**	11.0s	11.0s	5	–	481.0s	**70.0s**	6	–	–	**120.0s**	7	–	19.0s	**9.5s**
	4	2	**7.0s**	11.0s	11.5s	5	**4.0s**	13.0s	12.5s	5	–	355.0s	**55.0s**	9	–	–	**237.0s**	9	–	41.0s	**22.5s**
Logi-stics	1	2	**1.5s**	2.0s	2.0s	3	**1.0s**	4.0s	2.5s	4	30.0s	15.0s	**5.0s**	4	168.0s	250.0s	**12.5s**	2	–	**1.5s**	**1.5s**
	2	2	**2.0s**	3.0s	**2.0s**	3	**1.5s**	2.0s	2.0s	4	29.0s	**5.5s**	**5.5s**	4	169.5s	240.5s	**13.0s**	3	–	3.5s	**1.5s**
	3	2	**2.0s**	2.5s	2.5s	3	**1.0s**	2.5s	2.0s	4	31.0s	**5.5s**	**5.5s**	4	167.0s	400.0s	**13.0s**	4	–	5.0s	**2.0s**
	4	2	**2.0s**	**2.5s**	**2.0s**	3	**0.9s**	4.0s	3.0s	4	30.0s	28.0s	**6.0s**	4	168.5s	103.0s	**12.5s**	5	–	12.0s	**2.0s**
Zeno Travel	1	1	**0.5s**	0.8s	0.7s	1	**0.5s**	**0.5s**	**0.5s**	1	**0.5s**	1.0s	**0.5s**	1	**0.3s**	3.0s	0.4s	3	–	**0.5s**	**0.5s**
	2	4	10.0s	**5.0s**	**5.0s**	1	0.8s	**0.5s**	**0.5s**	4	40.0s	12.0s	**5.5s**	5	–	60.0s	**10.0s**	5	–	4.0s	**2.0s**
	3	3	8.5s	5.0s	**4.5s**	3	1.0s	**0.5s**	**0.5s**	5	30.5s	45.0s	**5.0s**	6	–	100.0s	**20.0s**	7	–	13.0s	**3.0s**
	4	3	10.0s	**5.0s**	**5.0s**	3	1.5s	1.0s	**0.7s**	5	31.0s	50.0s	**6.0s**	6	–	633.0s	**50.0s**	9	–	25.0s	**4.5s**
Storage	1	1	0.5s	0.5s	**0.3s**	1	**0.3s**	1.0s	0.5s	2	0.5s	0.5s	**0.3s**	3	647.0s	450.0s	**10.0s**	2	–	0.5s	**0.2s**
	2	1	0.5s	0.5s	**0.3s**	1	0.5s	0.5s	**0.4s**	3	**4.0s**	14.0s	5.0s	5	–	400.0s	**11.5s**	4	–	3.0s	**0.6s**
	3	1	0.7s	0.5s	**0.3s**	1	**0.5s**	0.5s	0.6s	3	**5.0s**	25.0s	**5.0s**	4	–	504.0s	**21.0s**	6	–	22.0s	**1.5s**
	4	5	**11.0s**	12.0s	12.0s	4	**4.0s**	22.0s	10.0s	6	**41.0s**	222.0s	51.0s	4	–	823.0s	**46.0s**	8	–	41.0s	**5.0s**

Table 1: Evaluation of ALG1, ALG3 and CSZK on Varying PDDL Domain and Scenario.

ally similar, as they both follow the same general search procedure.[17] Nevertheless, the difference in runtime becomes quite substantial in Scenarios 3, 4, and 5, highlighting the strength of the pre-processing approach. In fact, these are scenarios in which the tweaked KB_h is either inconsistent (i.e., there is no valid plan in KB_h) or becomes inconsistent with potential explanations from KB_a. There are two main reasons as to why ALG3 outperformed ALG1 in those scenarios. First, the search space of ALG3 may be smaller than that of ALG1 as ALG3 employs the pre-processing technique before forming the search space of potential explanations. Furthermore, ALG1 may need to restore the consistency of KB_h multiple times throughout its execution, in the worst case with each potential explanation, which consequently may increase the total runtime of the algorithm. Note that ALG1 performs an uninformed search over what formulae to remove when updating KB_h with each

17. Recall that the main advantage of ALG3 is the ability to identify (and replace) formulae in KB_h that evaluate to false with respect to KB_a's partial model.

Optimal Plan Length	Explanation Length $\|\epsilon\|$ 2 CSZK	2 ALG3	4 CSZK	4 ALG3	6 CSZK	6 ALG3	8 CSZK	8 ALG3	10 CSZK	10 ALG3
6	**0.5s**	1.0s	2.0s	**0.9s**	9.5s	**0.8s**	300.0s	**1.0s**	500s	**1.0s**
10	**0.5s**	3.0s	**2.5s**	**2.5s**	9.5s	**3.0s**	300.0s	**3.5s**	500s	**2.5s**
12	**0.5s**	4.5s	**2.0s**	**5.0s**	9.0s	**6.5s**	305.0s	**4.5s**	505s	**7.0s**
16	**1.0s**	28.0s	**2.0s**	27.0s	**10.0s**	26.0s	309.0s	**27.0s**	502s	**31.0s**
26	**1.0s**	70.0s	**8.0s**	75.0s	**20.0s**	73.0s	312.5s	**80.0s**	–	**85.0s**

Table 2: Varying Explanation and Plan Lengths for the Blocksworld PDDL Domain.

Prob.	# of Invalid Shorter-than-Optimal Plans 2 $\|\epsilon\|$	2 CSZK	2 ALG3	4 $\|\epsilon\|$	4 CSZK	4 ALG3	6 $\|\epsilon\|$	6 CSZK	6 ALG3	8 $\|\epsilon\|$	8 CSZK	8 ALG3	10 $\|\epsilon\|$	10 CSZK	10 ALG3
1	2	2.5s	**0.2s**	4	50.0s	**1.0s**	6	1231.0s	**11.0s**	8	–	**104.0s**	10	–	**840.5s**
2	2	3.0s	**0.5s**	4	57.0s	**1.0s**	6	1225.0s	**12.5s**	8	–	**105.5s**	10	–	**842.0s**
3	2	3.0s	**2.0s**	4	62.0s	**2.5s**	6	1240.0s	**11.0s**	8	–	**107.0s**	10	–	**845.0s**
4	2	**2.5s**	3.0s	4	60.0s	**1.5s**	6	1235.0s	**13.0s**	8	–	**111.0s**	10	–	**850.0s**

Table 3: Varying Invalid Shorter-than-Optimal Plans.

potential explanation (see Appendix C). In contrast, the pre-processing technique of ALG3 guarantees that KB_h will be consistent with all potential explanations by performing an informed search over what formulae to remove (i.e., it removes only the necessary set of formulae that evaluate to false with respect to the partial model of KB_a). In conclusion, these results shows that there is a clear advantage in employing the pre-processing approach for finding most-preferred explanations within our framework, especially for problems where the consistency of KB_h needs to be restored during an update.

7.2 Question 2: Efficacy on Classical Planning Problems

This question concerns how well our algorithms perform on classical planning problems. To address this, we compared our algorithms against the current planning-based state-of-the-art algorithm by Chakraborti et al. (2017), referred to as CSZK – the initials of the last names of the authors.[18] In what follows, we will only discuss the results of our best performing algorithm ALG3 and CSZK.

Table 1 tabulates the length of the explanations $|\epsilon|$ as well as the runtimes of the algorithms. In general, CSZK outperformed ALG3 in a majority of cases, except for Scenarios 3 and 4 in all domains. These cases also happen to be the cases where the explanation length $|\epsilon|$ is larger. We did not report runtimes of CSZK for Scenario 5 as the available implementation could not handle that scenario.

To verify if such correlations exist, we conducted an additional experiment where we varied the explanation length $|\epsilon|$ as well as the optimal plan length in the Blocksworld domain. Table 2 tabulates the results. It shows a clear trend that the runtimes of CSZK increases as the explanation lengths increase. The reason is that CSZK needs to search over a larger search space as the explanation length increases. As such, its runtime also increases.

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

In contrast, the runtimes of ALG3 remain relatively unchanged with varying explanation lengths. The reason is that the runtimes of ALG3 are dominated by the size of the encoded knowledge bases, which are independent of the explanation lengths.

The results also show that the runtimes of CSZK remain relatively unchanged with varying optimal plan lengths. The reason is that the runtimes of CSZK are dominated by its search for explanations. CSZK runs an A* search over the explanation search space and as long as the explanation length remains unchanged, the runtime complexity of the search, which is exponential in the explanation length, remains relatively unchanged as well. However, the runtimes of ALG3 increase as the optimal plan lengths increase. We attribute this to the following two reasons. First, longer plans means that there are more combinations of actions to consider in the explanation search space, consequently increasing the runtime of the algorithm. Additionally, the size of the knowledge bases increases as the plan length increases. Thus, there is an increasing number of formulae, which likely results in an increase in the runtime of the underlying SAT solver.

Upon closer inspection of the instances generated the experiments thus far, we observed that in almost all of these instances, the shortest plan in the human's model is at least as long as the optimal plan in the agent's model. Therefore, the experiments thus far were strongly biased in favor of explanations for *plan validity* instead of *plan optimality*. We thus conducted an additional experiment where we varied the number of invalid shorter-than-optimal plans in the human's model. These plans are invalid because they are comprised of some actions with wrong or missing preconditions and/or effects and these actions enable the goal state to be reached earlier than using a plan that is optimal in the agent's model. Table 3 tabulates the results. As expected, the results show that the runtime of both CSZK and ALG3 increases as the number of invalid plans increases. However, interestingly, the runtime of CSZK grows faster than that of ALG3, and ALG3 is up to 2 orders of magnitude faster than CSZK (when there are six invalid plans).

All of these observations result in the following three conclusions that highlight the different situations for when one should use one algorithm over the other: (1) ALG3 outperforms CSZK when explanations are long, and vice versa when explanations are short; (2) ALG3 outperforms CSZK when optimal plans are short, and vice versa when the optimal plans are long; and (3) ALG3 outperforms CSZK when there are many invalid shorter-than-optimal plans.

7.3 Question 3: Efficacy on Hybrid Planning Problems

In this final question, we wanted to investigate the generality of our approach on problems beyond classical planning, particularly on hybrid planning problems. As such, we now provide results on a number of PDDL+ problems.

Recall that finding optimal hybrid plans is often infeasible, and as such, there is no SAT/SMT planner that can prove optimality for general PDDL+ problems (see Section 3.3.1). Therefore, it is not feasible to find optimal plans to explain in our experiments. However, given that optimality cannot be guaranteed, a user may have doubts about the validity of a plan (i.e., whether the plan is sound and can be executed to achieve the goal). Thus, we limited ourselves to experiments for *plan validity* only.

Prob.		Scenario 1		Scenario 2		Scenario 3		Scenario 4		Scenario 5		Scenario 6		Scenario 7		Scenario 8	
		$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3	$\|\epsilon\|$	ALG3
LINEAR GENER.	1	0	0.1s	2	0.1s	2	0.2s	1	0.1s	2	0.1s	1	0.1s	3	0.2s	0	–
	3	0	0.1s	2	0.2s	2	0.8s	1	0.2s	2	0.2s	1	0.1s	3	0.8s	0	–
	5	0	0.2s	2	0.2s	2	2.0s	1	0.4s	2	0.4s	1	0.3s	3	3.0s	0	–
	7	0	0.3s	2	0.5s	2	4.0s	1	1.0s	2	0.6s	1	0.5s	3	14.0s	0	–
TORI-CELLI	1	1	0.2s	2	0.3s	2	0.4s	3	0.6s	4	0.2s	2	0.2s	4	0.2s	0	–
	2	1	0.4s	2	1.3s	2	2.0s	3	1.1s	5	0.9s	2	0.4s	4	0.7s	0	–
	3	1	0.5s	2	5.0s	2	11.0s	3	2.8s	7	3.6s	2	0.5s	4	2.0s	0	–
	4	1	1.0s	2	16.0s	2	38.0s	3	5.8s	5	1.1s	2	1.0s	4	4.0s	0	–
GENER. EVENTS	1	1	0.2s	2	0.2s	3	2.5s	3	0.2s	2	0.2s	1	0.1s	4	0.2s	1	0.2s
	2	1	0.3s	2	0.5s	3	5.0s	3	0.5s	3	0.2s	2	0.7s	5	0.7s	2	0.4s
	3	2	0.8s	2	1.3s	3	10.0s	3	1.5s	4	0.7s	2	0.6s	5	2.0s	2	1.5s
	4	1	1.3s	2	2.0s	3	26.0s	3	2.5s	6	0.9s	1	1.0s	4	4.5s	3	4.5s
CAR NO DRAG	1	2	0.2s	1	0.3s	3	0.3s	3	0.3s	2	0.2s	1	0.4s	3	0.4s	2	0.5s
	2	2	0.2s	1	0.2s	2	0.4s	3	0.3s	3	0.3s	2	0.5s	3	0.3s	2	0.4s
	3	2	0.3s	1	0.3s	2	0.2s	3	0.4s	1	0.1s	2	0.3s	3	0.4s	1	0.6s
	4	2	0.2s	1	0.2s	3	0.3s	3	0.3s	2	0.2s	1	0.2s	4	0.3s	1	1.0s
NONLIN. GENER	1	1	0.2s	2	0.1s	2	0.3s	1	0.2s	2	0.1s	1	0.1s	3	0.2s	0	–
	2	1	0.2s	2	0.2s	1	0.5s	1	0.4s	2	0.2s	1	0.2s	3	0.7s	0	–
	3	1	0.3s	2	0.2s	2	1.0s	1	0.3s	2	0.3s	1	0.4s	3	2.0s	0	–
	4	1	1.2s	2	0.5s	2	5.0s	1	0.5s	2	0.5s	1	0.5s	3	5.5s	0	–
SOLAR ROVER	1	1	0.5s	1	0.4s	2	0.4s	3	0.7s	2	0.2s	1	0.1s	3	0.2s	1	0.4s
	2	1	0.4s	1	0.5s	1	0.6s	2	1.4s	4	0.2s	1	0.2s	3	0.5s	1	0.4s
	3	1	0.8s	1	0.9s	2	1.0s	2	2.0s	6	0.3s	1	0.4s	3	5.0s	1	0.6s
	4	1	1.5s	1	2.0s	1	2.0s	3	3.5s	8	0.4s	1	0.5s	5	10.0s	1	1.5.s
POWERED DESCENT	1	1	0.2s	1	0.4s	2	0.4s	3	1.0s	2	0.3s	1	0.2s	3	0.8s	1	0.6s
	2	1	0.3s	1	0.6s	1	0.5s	2	2.5s	4	0.4s	1	0.1s	3	1.5s	1	0.7s
	3	1	0.3s	1	1.0s	2	1.5s	2	3.0s	6	0.3s	1	0.3s	3	3.0s	1	1.0s
	4	1	0.7s	1	3.0s	1	3.5s	3	5.5s	8	0.4s	1	0.2s	4	8.0s	1	3.0s
NONLIN SOLAR ROVER	1	1	0.5s	2	0.4s	1	0.6s	3	1.0s	2	0.3s	1	0.3s	3	0.8s	1	0.5s
	2	2	1.0s	2	1.0s	2	1.5s	4	3.5s	4	0.4s	2	0.4s	5	1.5s	2	1.5s
	3	2	4.0s	3	4.0s	2	2.5s	4	4.0s	6	0.3s	2	0.4s	5	3.0s	2	1.0s
	4	3	5.0s	4	4.0s	1	2.0s	6	7.5s	8	0.4s	1	0.3s	4	6.0s	2	1.0s

Table 4: Evaluation of ALG3 on Varying PDDL+ Domain and Scenario.

As Algorithm 3, referred to as ALG3, was designed to find explanations that explain both plan validity and optimality, we tweaked it to only check for credulous entailment since that is sufficient for finding explanations for plan validity (see Section 6.2). We did not compare it with any other algorithm since to the best of our knowledge existing explanation generation algorithms are limited to classical planning problems.

Similar to our first experiment for classical planning, we used the actual domain instances as the model of the agent and tweaked it for the model of the human user. We considered the same five scenarios earlier as well as the following three additional scenarios:

- **Scenario 6:** We changed the duration of all the durative actions in the human's model.
- **Scenario 7:** We removed (on average) five random preconditions and effects as well as changed the duration of all durative actions in the human's model.
- **Scenario 8:** We removed (on average) two events and processes in the human's model.

Table 4 tabulates the results. We did not report runtimes for the LINEAR, TORICELLI, and NON-LINEAR GENERATOR domains for Scenario 8 as these domains do not contain

events and/or processes. In general, ALG3 was able to maintain small runtimes of less than 1s in the majority of instances. The reason is that the size of the encoded knowledge bases are relatively small because SMTPlan uses the iterative encoding facility of the $z3$ solver. Specifically, the encoding of each layer consists of the following steps: Adding the new variables and constraints for the next happening, adding the goal constraints to the new constraint set, pushing the constraint set onto the stack, solving, and popping the goal constraint set off the stack. As such, at each step in the iterative deepening with $z3$, only the latest layer needs to be encoded.

In conclusion, these results demonstrate that our approach can be generalized beyond classical planning to hybrid planning, improving the applicability of explainable planning approaches.

8. Proof-of-Concept: Communicating Explanations to Human Users

As noted in Section 1, explanation generation frameworks, especially those targeting human users, should be able to effectively communicate explanations to them, ideally, in a manner that minimizes their cognitive effort and maximizes their cognitive effect. In consequence, this opens up a plethora of research questions one should examine carefully. Among them, two questions, which we posit are of fundamental consideration, are the following: (Q1) *What form should the explanations take?*, and (Q2) *What communication medium should be used to deliver them effectively?*

(Q1) Explanation Form: In general, an explanation can take several forms (Hempel & Oppenheim, 1948; Hempel, 1965; Miller, 2018). Among the most common and intuitive forms of explanation are: *Causal explanation*, an explanation in which the explanandum is derived with deductive arguments (e.g., "All plans reaching state S solve the problem; this plan reached state S; therefore, this plan solves the problem"); *Statistical explanation*, an explanation subsuming the explanandum under a generalization in order to give it inductive support (e.g., "Most invalid plans have at least one action with an unsatisfied precondition; this plan is invalid; therefore, this plan has at least one action with an unsatisfied precondition"); and *Social explanation*, an explanation that models the expectations of the explainee (e.g., "This plan is invalid because you are missing this precondition"). As one can see, the latter form refers to, in essence, the model reconciliation problem that we are tackling in this paper. In other words, explanations in the form of model reconciliation are social explanations because they explicitly capture the effects of the explainee's expectations in the explanation generation process (of the agent).[19]

Within the context of explainable AI planning, explanations as model reconciliation can be further categorized according to two properties: *Selective*, in being able to select explanations from a pool of competing hypotheses; and *Contrastive*, in being able to contrast and differentiate properties of two competing hypotheses (see work by Sreedharan et al. (2020) for more.) Nevertheless, irrespective of the underlying properties, explanations in the form of model reconciliation express discrepancies between the action models of the agent and the human user, or involve differences in the initial and/or goal state assumptions of

19. As we highlight in Section 2, these explanations received a lot of traction because they are consistent with the Theory of Mind, a normative theory aimed at describing the operations of the human mind and behavior in social and collaborative settings.

the planning problems of the agent and the human user. These discrepancies fall into one of the following two categories:

- **Action-space Information:** Given a planning problem Π, and an associated domain D containing actions A, the action-space information corresponds to information about each action in D and their corresponding conditions.
- **State-space Information:** Given a planning problem Π, a plan π, and a sequence of states S involved in the execution of π, the state-space information corresponds to information about the predicates in each state in S.

As one may need to convey explanations that contain action-space and state-space information when explaining a plan, an ideal system for presenting explanations should be able to convey both types of information to human users.

(Q2) Communication Medium: Communicating information to human users can be typically achieved through four mediums: Verbal, non-verbal, text-based, and visual. One of the most common computer interfaces to communicate information is the graphical user interface, where information is presented graphically through a combination of visualizations and text (Mandel, 2002). From the perspective of communicating explanations, although text can be a natural representation for an explanation, when presented alone, it can require increased cognitive effort and possibly increase the likelihood of misunderstanding the task especially for novice users. In contrast, a well-established educational principle, called the *multimedia learning principle*, posits that humans learn better from visuals and words, than from words alone (Mayer, 1997). For example, Clark and Mayer (2016) showed that accompanying text-based instructions with visuals improved students' performance on a test by a median amount of 89%. Interestingly, students got around 65% of answers correct after seeing a combination of text and visuals, compared to less than 40% of answers correct after reading a text comprised of words alone. Similar results have also been obtained in object assembly tasks (Brunyé, Taylor, & Rapp, 2008). Thus, there is strong evidence within the psychology community that the use of visual content has a profound effect on increasing retention and comprehension when compared to text alone.

Motivated by these findings, we posit that communicating explanations for planning problems to human users should be done through a combined visual and text-based medium. Specifically, an ideal explainable AI planning system should be able to ground the explanations on the visualized action-space and state-space of the given planning problem (i.e., on the visualized plans of the human users). Being able to visualize the plans of human users provides a straightforward way for highlighting actions and/or states that are affected by the agent's explanation. In other words, the explanation can be mapped onto the human user's plan, which could assist human users in understanding why their plans are not valid (or optimal), and why the agent's plan is. This method could minimize the cognitive effort needed to understand an explanation, thus maximizing the explanation's cognitive effect.

Visualizing planning problems through the development of user interfaces has received a lot of attention (Freedman, Chakraborti, Talamadupula, Magazzeni, & Frank, 2018). While some work aim to show human users the space of alternate plans (Gopalakrishnan & Kambhampati, 2018; Chakraborti, Fadnis, Talamadupula, Dholakia, Srivastava, Kephart, & Bellamy, 2018), others aim to create systems to aid human users in the creation of plans

(e.g., Planimation (Chen, Ding, Edwards, Chau, Hou, Johnson, Sharukh Syed, Tang, Wu, Yan, Gil, & Nir, 2020)) or for assistance with domain modeling (e.g., Conductor (Bryce, Bonasso, Adil, Bell, & Kortenkamp, 2017)). These kinds of systems are essential steps towards the creation of a unified planning interface, especially when human users are involved in the loop. Indeed, in the next section, we leverage existing planning visualization work to construct a system for communicating explanations to human users.

Finally, it is worth noting that there may be cases where one would need to communicate complex explanations to human users, either because they are long, or because they involve complex information, such as properties of hybrid planning problems. In the former case, a possible resolution would be to break down the explanation in shorter chunks and communicate each chunk sequentially, with each chunk that is understood and accepted by the human user triggering the communication of the next chunk. In contrast, if an explanation includes more complex information, then it may be beneficial to expose it to human users at different levels of granularity (i.e., by abstracting the information at a level that is associated with the expertise of the user).[20]

8.1 User Study

As mentioned in Section 2, there has been some supporting evidence suggesting that explanations in the form of model reconciliation are well understood by human users when explaining classical plans to them (Chakraborti et al., 2019b; Zahedi et al., 2019). However, to the best of our knowledge, their applicability to hybrid planning problems has not been investigated thus far. Therefore, in what follows we present a human user study on a hybrid planning variant of the Logistics domain (McDermott, 2000), where we use visualizations and text for presenting explanations to human users.

8.1.1 Study Design

Recall that the model reconciliation problem requires that the explaining agent knows both its model and the model of the human user receiving the explanation. To enforce this assumption, we used variations of the Logistics domain (McDermott, 2000) as the model of the explaining agent, tweaked that model, and assigned this tweaked model to participating human users. To ensure the human users understood their assigned model, we presented it to them at the start of the study and asked them to create a plan given that model. We removed participants who created plans inconsistent with the tweaked model provided to them. Then, human users were provided an explanation in the form of model reconciliation (e.g., from our algorithms) and were asked to answer a series of questions as well as correct their plans based on their understanding of the explanation.[21] The answers to those questions as well as the human users' ability to correct their plans reflect their understanding of the explanations provided. In this study, we hypothesize that *explanations in the form of model reconciliation are effective for explaining hybrid planning plans to human users.*

20. Note that abstracting information is oftentimes domain-dependent, and should thus be considered on a case-by-case scenario. We demonstrate this in our human user study, where we simplify the presentation of some information such that the study is accessible to human users with no prior planning knowledge.
21. We used a visualization system (see Section 8.1.3) to help users create the plan as well as understand the explanation.

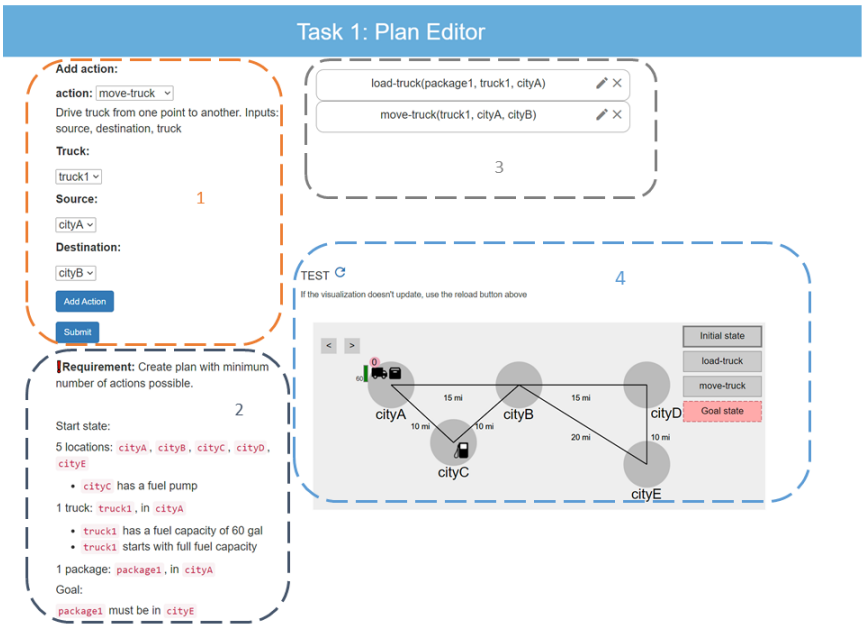

Figure 1: A view of the plan editor for the human user study. (1) Action selection; (2) The initial and goal states; (3) User's current plan; (4) Test visualization showing validity of the plan.

The user study was conducted on the online crowdsourcing platform Prolific (Palan & Schitter, 2018). We created two tasks for each participant as follows:

- **Task 1:** For this task, participants were asked to create a (shortest) plan based on the modified model provided to them using the visualization system coupled with a simple plan editing interface, as shown in Figure 1.[22] This interface also allows users to evaluate their plans, which can subsequently provide information about any errors in their plans due to their misunderstanding of the provided information in their models.
  A participant succeeds in Task 1 if they create and submit a valid plan given their model. Participants that succeeded in Task 1 continued to Task 2, and participants that failed in Task 1 were filtered out and ignored. This is important due to the assumption (in model reconciliation) that the human user's model is known a-priori. In addition, participants whose plans were valid in their model but did not require an explanation (e.g., their plans were also valid in the agent's model) were also filtered out.
- **Task 2:** For this task, we informed the participants that the initial model provided to them contained errors, and presented an explanation for those errors using visualizations as well as text. They were then asked a series of questions to evaluate their understanding of the explanation provided (**Task 2a**). The exact questions we asked the participants can be found in Appendix D. Then, they were shown the plan editor again and asked to correct their plan, this time without the ability to evaluate their plans for correctness

22. To eliminate any learning effects, participants were shown the visualization system in Task 1 to ensure that they are familiar with the system before receiving an explanation using that interface.

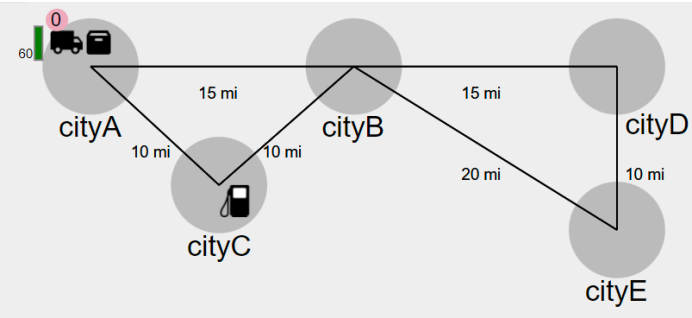

Figure 2: The initial state of the modified Logistics problem. The goal is to transport the package from `cityA` to `cityE`. A shortest plan here would have the truck load the package at `cityA` and move the truck to `cityC` to refuel before moving it to (the goal location) `cityE`.

(**Task 2b**). A participant succeeded in Task 2b if their corrected plan was valid in the agent's model.

To incentivize participants to provide answers to the best of their ability, we provided a bonus reward to participants who succeeded in Task 1, and an additional bonus to participants who also succeeded in Task 2b. Furthermore, we also included two questions for attention checks in the study, where participants were asked to type a particular string or select a particular answer in a multiple choice question. Participants who wrongly answered both of these questions were filtered out of the study.

Additionally, we performed a control experiment, where participants would receive the exact same tasks, but in Task 2a and Task 2b, they would not be shown the explanations. Instead, they are expected to correct their plan with just the knowledge of which actions were wrong. This allows us to see whether providing the explanation has any benefit, or if just the knowledge of the wrong actions is enough to help users in correcting the plan.

Finally, each participant had the following interactions in the study: (1) They arrive at the webpage following the link from Prolific, where they enter their demographics and some information on their educational background. (2) To ensure that they have the background necessary to solve the tasks, they are given tutorials on automated planning, the Logistics domain, and the plan editing interface. (3) Following the tutorials, they are asked to complete Task 1. (4) If they succeeded in Task 1, they are asked to complete Tasks 2a and 2b. (5) All participants, including those who failed Task 1, are then asked to provide feedback on the system's usability (Holzinger, Carrington, & Müller, 2020) and are informed of their payments before being redirected back to Prolific.

8.1.2 Domain and Problem

As mentioned, our choice of domain was a variant of the Logistics domain (McDermott, 2000), which we augmented to contain elements of hybrid planning problems. The particular Logistics problem involves the transportation of packages across cities via trucks. To make the domain a hybrid planning problem, we included a *process* by adding a fuel bar to the truck which is depleted by the execution of the action `move-truck`, and an *event* to stall the

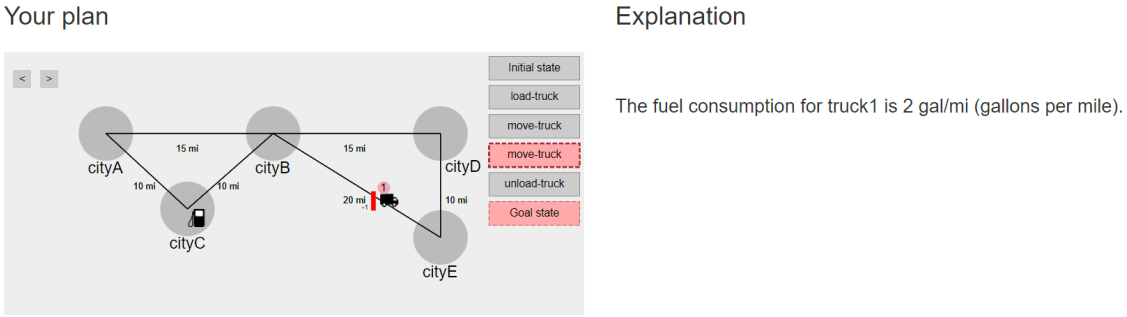

Figure 3: A view of the explanation phase for the modified Logistics problem. We used a pulsing border on the fuel bar to indicate that the truck has run out of fuel, and thus stalled. The explanation is also presented in text to the right of the visualization.

truck if it runs out of fuel midway to a city. The truck has a specified fuel consumption rate in gallons per mile (gal/mi), that is, it burns 2gal/mi. Additionally, we added the action `refuel` to refuel the truck, and a predicate `is-fuelpump` to note locations where the truck can be refueled. We created a simple problem with five cities and one fuel pump. Figure 2 shows the initial state for this problem. There is one truck available and one package that needs to be transported to the goal city without stalling the truck.

We considered the following change for the modified model of the human user: We modified the fuel consumption rate to be 1gal/mi instead of the actual 2gal/mi. This allowed users to create a shorter plan in the modified domain by directly moving the truck to the destination instead of going to `cityC` and refueling. Thus, an explanation is needed to explain why their plan is invalid, and how to compute a (shortest) valid plan.

8.1.3 Prototype Implementation

Borrowing elements from existing work in plan visualization (Bryce et al., 2017; Magnaguagno, Fraga Pereira, Móre, & Meneguzzi, 2020), we created a visualization system,[23] that combines the action-space and state-space information of the planning problem and can visualize plans and their execution as well as present explanations to human users. The inclusion of the action-space information also conveniently presents a simple way for human users to select and view different states. Actions whose preconditions are not met in the human user's plan are highlighted as red. This helps human users while creating their plan, ensuring they understand the preconditions specified in the provided model. Moreover, the state-space shows the "current" state of the world after the execution of the action selected by the human user in their plan, including information such as the positions of the truck and package, and the truck's fuel level. In addition, we used animations showing the movement of the truck between each city as well as the truck's fuel consumption.

During the explanation phase (Task 2a), actions are highlighted by using red highlights to indicate which actions are wrong, with the state-space visualization providing more details about what exactly went wrong in the human user's plan. For presenting explanations

23. We created the implementation to run on a web browser, using Flask and Python as back-end, and D3.js and JavaScript for the front-end.

	Population Size	Comprehension Score	Correction Ratio
Main Study	35	6 (out of 8)	91% (32/35)
Control	37	N/A	24% (9/37)

Table 5: Human User Study Results for the Modified Logistics Domains.

within the state-space visualization, we do the following: For each state in the execution of the plan, starting from the initial state, we display the current state with respect to the actions that are executed in the human user's plan, using the agent's domain. Figure 3 shows the explanation phase.

Note that for the purpose of this study, the text-based explanation (as seen in Figure 3) was handcrafted. Nonetheless, there may be systematic ways for translating logic-based explanations from our framework to natural language text. We provide some details in Appendix D.

8.1.4 Evaluation

The study was conducted with 100 participating users (50 in the main group and 50 in the control group). All 100 participants have at least some undergraduate education in any field. Out of all the participants, only data from 72 users was used (35 in the main group and 37 in the control group), as the rest either failed Task 1 or had created plans that did not need explanations (e.g., their plans included going to City C to refuel before going to Cities B and E).

As we mentioned earlier, the goal of this study is to identify if explanations in the form of model reconciliation can convey to humans the validity of hybrid planning plans. To evaluate our hypothesis, we used the following measures:

- **Comprehension Score:** Number of questions users answered correctly in Task 2a.
- **Correction Ratio:** Proportion of users who succeeded in Task 1 (plan creation phase) that also succeeded in Task 2b (plan correction phase).

Table 5 summarizes the results of our analysis, where we report the population size and the two measures used for evaluation. In general, the results seem to suggest that the vast majority of users in the main study population understood the explanations communicated to them. In particular, we first scored participants based on the number of correct answers they gave for the questions asked in Task 2a (comprehension score). With a maximum possible score of 8, we managed to obtain an overall mean score of 6, which indicates that the presented explanations had a positive cognitive effect, as most users used the explanation to reason through the questions and successfully answer most of them. We then measured how many users were able to accurately correct their plans in Task 2b when shown the explanation (correction ratio), i.e., the users' ability to reflect the explanation presented to them into creating the (agent's) correct plan. The percentage of users that succeeded was 91%, showing that most users indeed made use of the explanation by adding its information to their models and creating the correct plan. From the control group, we can observe that the explanation does indeed have an effect of how well the participants were able to correct their plans, as users who did not see the explanations had a much lower correction ratio (24%), as expected.

We would also like to address the potential issue of bias in the results from Task 2b. Since we removed users who failed to complete Task 1, we might have filtered out participants who did not understand planning and their assigned models or had difficulty creating plans in general, thus creating a selection bias for users who are better at understanding and solving planning problems. As mentioned earlier, it was necessary to filter out users who did not understand their assigned models, as the explanation provided is contingent on the human model as per the assumptions made by model reconciliation. However, we believe that the ability to observe errors during plan creation in Task 1 mitigates this issue to some extent, as all users could observe any errors in their plans during this phase, and as such, potentially correct them before submitting. Further, since we observe a large difference in correction ratio for the control group with a similar number of people who completed Task 1, we believe that this potential bias does not have a major effect.

In conclusion, one can see that these findings corroborate our hypothesis, that is, explanations in the form of model reconciliation are effective for explaining hybrid planning plans to human users. The results of this study, complemented by the results obtained for classical planning problems by Chakraborti et al. (2019b) and Zahedi et al. (2019), demonstrate the real-world efficacy of explanations as model reconciliation for planning problems beyond classical planning.

9. Related Work

We now provide a discussion of related work from the knowledge representation and reasoning (KR) and planning literature. We focus on these two areas as our logic-based approach bears some similarity to other logic-based approaches in KR and our application domain of explainable planning has been solved by other planning approaches.

9.1 KR Literature

As our proposed framework bears some similarities with the theory of belief change and abductive explanations, we start by describing their underlying theory. We then provide two examples that illustrate the differences between these approaches. Finally, we discuss some further related work from the KR community.

9.1.1 Belief Change

Belief change is a kind of change that can occur in a knowledge base. Depending on how beliefs are represented and what kinds of inputs are accepted, different typologies of belief changes are possible. In the most common case, when beliefs are represented by logical formulae, one can distinguish three main kinds of belief changes, namely, *expansion*, *revision*, and *contraction*. In the following, we formally describe the aforementioned notions.

Expansion: An expansion operator of a knowledge base can be formulated in a logical and set-theoretic notation:

Definition 12 (Expansion Operator) *Given a knowledge base KB and a formula ϕ, $+_e$ is an expansion operator if it expands KB by ϕ as $KB +_e \phi = \{\psi : KB \cup \phi \vdash \psi\}$.*

It is trivial to see that $KB +_e \phi$ will be consistent when ϕ is consistent with KB, and that $KB +_e \phi$ will be closed under logical consequences.

Revision: A belief revision occurs when we want to add new information into a knowledge base in such a way that, if the new information is inconsistent with the knowledge base, then the resulting knowledge base is a new consistent knowledge base. Alchourrón, Gärdenfors, and Makinson conducted foundational work on knowledge base revision, where they proposed a set of rationality postulates, called *AGM postulates*, and argued that every revision operator must satisfy them (Alchourrón & Makinson, 1985; Gärdenfors, 1986; Gärdenfors, Rott, Gabbay, Hogger, & Robinson, 1995). Although revision cannot be defined in a set-theoretic manner closed under logical consequences (like expansion), it can be defined as follows:

Definition 13 (Revision Operator) *Given a knowledge base KB and a formula ϕ, $+_r$ is a revision operator if it satisfies the AGM postulates for revision and modifies KB w.r.t. ϕ such that the resulting KB is consistent.*

The underlying motivation behind the AGM postulates is that when we change our beliefs, we want to retain as much as possible the information from the old beliefs. Thus, when incorporating new information in the knowledge base, the heuristic criterion should be the criterion of *information economy* (i.e., minimal changes to the knowledge base is preferred). As such, a model-theoretic characterization of minimal change has been introduced by Katsuno and Mendelzon (1991b), where minimality is defined as selecting the models of ϕ that are "closest" to the models of KB.

However, the AGM rationality postulates will not be adequate for every application. (Katsuno & Mendelzon, 1991a) proposed a new type of belief revision called *update*. The fundamental distinction between the two kinds of belief revision in a knowledge base, namely *revision* and *update*, is that the former consists of incorporating information about a static world, while the latter consists of inserting information to the knowledge base when the world described by it changes. As such, they claim that the *AGM postulates describe only revision* and showed that *update can be characterized by a different set of postulates called KM postulates.*

Definition 14 (Update Operator) *Given a knowledge base KB and a formula ϕ, $+_u$ is an update operator if it satisfies the KM postulates for update and modifies KB w.r.t. ϕ such that the updated KB incorporates the change in the world introduced by ϕ.*

From a model-theoretic view, the difference between revision and update, although marginal at first glance, can be described as follows: Procedures for revising KB by ϕ are those that satisfy the AGM postulates and select the models of ϕ that are "closest" to the models of KB. In contrast, update methods are exactly those that satisfy the KM postulates and select, for each model I of KB, the set of models of ϕ that are closest to I. Then, the updated KB will be characterized by the union of these models.[24]

It is worth mentioning that, on a high level, the key difference between update and revision is a temporal one: Update incorporates into the knowledge base the fact that the

24. This approach is called the *possible models approach* (Winslett, 1988).

world described by it has changed, while revision is a change to our world description of a world that has not itself changed. We refer the reader to the work by Katsuno and Mendelzon (1991a) for a comprehensive description as well as an intuitive meaning between revision and update.

Contraction: Similarly to the AGM postulates for revision, Alchourrón and Makinson (1985) proposed a set of axioms that any contraction operator must satisfy. Therefore, a contraction operator is defined by:

Definition 15 (Contraction Operator) *Given a knowledge base KB and a formula ϕ, $-_c$ is a contraction operator if it satisfies the AGM postulates for contraction, and contracts KB w.r.t. ϕ by retracting formulae in KB without adding of new ones.*

We can see that, as in the case of revision, it is not possible to define contraction in a set-theoretic manner closed under logical consequences. We illustrate this with the following example:

Example 4 *Consider the Generator domain from Section 4. Assume a subset of the original KB_a, i.e., $KB_a = [\neg gen\_running_0, gen\_running_1, \neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0]$ and $\phi = [gen\_on_0]$ that we wish to contract. Then, in order to maintain consistency in KB_a, one of $\neg gen\_running_0, gen\_running_1$, or $\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0$ must be retracted. But which one? There is no logical reason for making one choice rather than the other.*

Interestingly, it has been shown that the problems of revision and contraction are closely related (Gärdenfors, 1988). Despite the fact that the postulates that characterize revision and contraction are "independent,"[25] revision can be defined in terms of contraction (and vice versa). This is referred to as the Levi identity (Levi, 1978).

Definition 16 (Levi Identity) *Assume a knowledge base KB, a formula ϕ, and operators $+_r$ and $-_c$ that satisfy the AGM postulates for revision and contraction, respectively. Then, $KB +_r \phi = (KB -_c \neg\phi) +_e \phi$.*

Hence, a revision of a knowledge base can be viewed as contracting KB with respect to $\neg\phi$ and then expanding $(KB -_c \neg\phi)$ by ϕ.

9.1.2 Abductive Explanations

Explanations in knowledge base systems were first introduced by Levesque (1989) in terms of abductive reasoning, that is, given a knowledge base and a formula that we do not believe at all, what would it take for us to believe that formula? A more formal definition is provided below.

Definition 17 (Abductive Explanation) *Given a knowledge base KB and a query q to be explained, α is an explanation of q w.r.t. to KB iff $KB \cup \{\alpha\}$ is consistent and $KB \cup \{\alpha\} \models^s_L q$.*

25. In the sense that the postulates for revision do not refer to contraction and vice versa.

Usually, such explanations are phrased in terms of a hypothesis set H (set of atomic sentences – also called abducibles), and, generally, is an intuitive methodology for deriving root causes.

9.1.3 Two Illustrative Examples

To illustrate the differences between our approach and the KR approaches described in this section, we discuss below how they operate on two simplifications of the Generator domain example in Section 4.

Problem 1 Assume a simplified version of the Generator domain with only one action $gen_on = \{$precondition: $fuel\_full$, effect: $gen\_running\}$, and initial and goal states $fuel\_full$ and $gen\_running$, respectively. Clearly, the plan for this problem is $\pi^* = [gen\_on]$. Also, assume that the human user is not aware that action gen_on has effect $gen\_running$. Now, the knowledge bases encoding the models of the agent and the human, are respectively:

$$KB_a = [fuel\_full_0, \neg gen\_running_0, gen\_running_1, \tag{75}$$
$$gen\_on_0 \rightarrow fuel\_full_0, gen\_on_0 \rightarrow gen\_running_1, \tag{76}$$
$$\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0] \tag{77}$$

$$KB_h = [fuel\_full_0, \neg gen\_running_0, gen\_running_1, gen\_on_0 \rightarrow fuel\_full_0] \tag{78}$$

Further, without loss of generality, suppose that the explanation needed to explain π^* to KB_h is $\epsilon = [gen\_on_0 \rightarrow gen\_running_1, \neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0]$.

Abductive Explanations: Abductive explanation cannot be applied in this setting because KB_h does not contain any causal rules that can be used to abduce the query.

Revision: Since the union of ϵ and KB_h is consistent, the revision operator will yield a trivial update according to the second AGM axiom: $KB_h +_r \epsilon = KB_h \cup \epsilon$.

Update: To use the update operator, we first need to find the models of KB_h and ϵ:

- $ACC_L(KB_h)$:
  $I_1 = \{fuel\_full_0, gen\_running_1, gen\_on_0\}$,
  $I_2 = \{fuel\_full_0, gen\_running_1\}$.
- $ACC_L(\epsilon)$:
  $J_1 = \{gen\_on_0, gen\_running_1, fuel\_full_0\}$,
  $J_2 = \{gen\_on_0, gen\_running_1\}$,
  $J_3 = \{gen\_on_0, gen\_running_1, gen\_running_0, fuel\_full_0\}$,
  $J_4 = \{gen\_on_0, gen\_running_1, gen\_running_0\}$,
  $J_5 = \{gen\_running_1, gen\_running_0, fuel\_full_0\}$,
  $J_6 = \{gen\_running_1, gen\_running_0\}$,
  $J_7 = \{gen\_running_0, fuel\_full_0\}$,
  $J_8 = \{gen\_running_0\}$,
  $J_9 = \{fuel\_full_0\}$,
  $J_{10} = \{\}$.

Now, according to the KM postulates, we need to find the models of ϵ that are closest to I_1 and I_2. Then, the updated KB will be the disjunction of the conjunction of the variables in each model. Now, let the function $\mathit{Diff}(m_1, m_2)$ denote the set of propositional letters with different truth values in models m_1 and m_2.

Then, for I_1, it is easy to see that the closest model is J_1 because $\mathit{Diff}(I_1, J_1) = \emptyset < \mathit{Diff}(I_1, J_k)$ for all k. So, J_1 is selected. For I_2, we need to calculate the difference for every model of ϵ:

$\underline{\mathit{Diff}(I_2, J_1) = \{gen\_on_0\}}$,
$\mathit{Diff}(I_2, J_2) = \{gen\_on_0, fuel\_full_0\}$,
$\mathit{Diff}(I_2, J_3) = \{gen\_on_0, gen\_running_0\}$,
$\mathit{Diff}(I_2, J_4) = \{gen\_on_0, gen\_running_0, fuel\_full_0\}$,
$\underline{\mathit{Diff}(I_2, J_5) = \{gen\_running_0\}}$,
$\mathit{Diff}(I_2, J_6) = \{fuel\_full_0, gen\_running_0\}$,
$\mathit{Diff}(I_2, J_7) = \{gen\_running_0, gen\_running_1\}$,
$\mathit{Diff}(I_2, J_8) = \{gen\_running_0, gen\_running_1, fuel\_full_0\}$,
$\underline{\mathit{Diff}(I_2, J_9) = \{gen\_running_1\}}$,
$\mathit{Diff}(I_2, J_{10}) = \{fuel\_full_0, gen\_running_1\}$

where sets with the minimal elements are underlined. So, J_1, J_5, and J_9 are selected and the final result is the union of all selected models, that is, $ACC_L(KB_h +_u \epsilon) = \{J_1, J_5, J_9\}$. Thus, the resulting KB must satisfy all three models, yielding the following: $KB_h +_u \epsilon = [\big(gen\_on_0 \wedge gen\_running_1 \wedge fuel\_full_0 \wedge \neg gen\_running_0\big) \vee \big(gen\_running_1 \wedge gen\_running_0 \wedge fuel\_full_0 \wedge \neg gen\_on_0\big) \vee \big(fuel\_full_0 \wedge \neg gen\_running_1 \wedge \neg gen\_on_0 \wedge \neg gen\_running_0\big)]$.

Our Approach: As a first step, our method will first check if KB_h is consistent with the model of KB_a. Since it is, it will simply insert ϵ to KB_h, yielding $\widehat{KB}_h^{\epsilon} = KB_h \cup \epsilon$ just like revision.

In conclusion, this problem demonstrates that it is possible for *belief revision* to yield the same update as our approach, which is when $KB_h \cup \epsilon$ is consistent (per AGM postulates). It also highlights why *belief update* is not applicable for explainable planning, namely that the updated knowledge base $KB_h +_u \epsilon$ violates the action dynamics of classical planning problems (Kautz et al., 1996).

Problem 2 Now assume a version of the Generator domain which consists of two actions $gen_on = \{$precondition: $fuel\_full$, effect: $gen\_running\}$ and $gen_on_alt = \{$precondition: $fuel\_mid$, effect: $gen\_running\}$ with initial and goal states $fuel\_full$ and $gen\_running$, respectively, and a plan $\pi^* = [gen\_on]$. Also, assume that the human user is not aware that action gen_on has effect $gen\_running$. Then, the knowledge bases encoding the models of the agent and the human are respectively:

$$KB_a = [fuel\_full_0, \neg fuel\_mid_0, \neg gen\_running_0, gen\_running_1, \tag{79}$$
$$gen\_on_0 \rightarrow fuel\_full_0, gen\_on_0 \rightarrow gen\_running_1, \tag{80}$$
$$gen\_on\_alt_0 \rightarrow fuel\_mid_0, gen\_on\_alt_0 \rightarrow gen\_running_1, \tag{81}$$
$$\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0 \vee gen\_on\_alt_0, \tag{82}$$

$$\neg gen\_on_0 \vee \neg gen\_on\_alt_0] \tag{83}$$

$$KB_h = [fuel\_full_0, \neg fuel\_mid_0, \neg gen\_running_0, gen\_running_1, \tag{84}$$
$$gen\_on_0 \rightarrow fuel\_full_0, gen\_on\_alt_0 \rightarrow fuel\_mid_0, \tag{85}$$
$$gen\_on\_alt_0 \rightarrow gen\_running_1, \tag{86}$$
$$\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on\_alt_0, \neg gen\_on_0 \vee \neg gen\_on\_alt_0] \tag{87}$$

As in the previous problem, we now assume that the explanation needed is $\epsilon = [gen\_on_0 \rightarrow gen\_running_1, \neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0 \vee gen\_on\_alt_0]$.

Abductive Explanations: The method of abductive explanations will fail in this setting due to the fact that KB_h is inconsistent. Further, even if KB_h was consistent, we would still not be able to find any abductive explanations due to the lack of causal rules in KB_h.

Revision: Following AGM postulates, revision cannot be applied because KB_h is individually inconsistent.

Update: Again, as KB_h is inconsistent, and according to KM update postulates, it cannot be repaired using update.

Our Approach: As $KB_h \cup \epsilon$ is inconsistent, our approach will identify the erroneous formula $\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on\_alt_0$ and replace it with the corresponding correct formula $\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0 \vee gen\_on\_alt_0$ from KB_a, thereby restoring consistency. The updated knowledge base will be:

$$\widehat{KB}_h^{\epsilon} = [fuel\_full_0, \neg fuel\_mid_0, \neg gen\_running_0, gen\_running_1, \tag{88}$$
$$gen\_on_0 \rightarrow fuel\_full_0, gen\_on_0 \rightarrow gen\_running_1, \tag{89}$$
$$gen\_on\_alt_0 \rightarrow fuel\_mid_0, gen\_on\_alt_0 \rightarrow gen\_running_1, \tag{90}$$
$$\neg gen\_running_0 \wedge gen\_running_1 \rightarrow gen\_on_0 \vee gen\_on\_alt_0, \tag{91}$$
$$\neg gen\_on_0 \vee \neg gen\_on\_alt_0] \tag{92}$$

In summary, this problem demonstrates that when KB_h is inconsistent, abductive explanations, revision, and update cannot be applied but our approach can be applied.

Therefore, a key distinction between the previous approaches and our approach is that, historically, belief change refers to a *single agent* revising its belief after receiving a new piece of information that is in conflict with its current beliefs; so, there is a temporal dimension in belief change and a requirement that it should maintain as much as possible the belief of the agent, per AGM postulates. Our notion of explanation is done with respect to *two knowledge bases* and there is no such requirement (with respect to KB_h). For example, if the agent believes that block A is on block B, the human believes that block B is on block A, and the explanation does not remove this fact from the human's KB, then the agent and the human will still have some conflicting knowledge about the positions of blocks A and B after the update. Thus, the previous notions of belief change cannot accurately capture and characterize the MRP problem.

9.1.4 Some Further Discussion

Similar to belief change, explanation differs from other similar notions, such as diagnosis (Reiter, 1987). In general, a diagnosis is defined with respect to a knowledge base KB, a set of components H, and a set of observations O. Given that $KB \cup O \cup \{\neg ab(c) \mid c \in H\}$ is inconsistent, a diagnosis is a subset S of H such that $KB \cup O \cup \{ab(c) \mid c \in S\} \cup \{\neg ab(c) \mid c \in H \setminus S\}$ is consistent. Here, $ab(c)$ denotes that the component c is faulty. Generalizing this view, the inconsistency condition could be interpreted as the query q and $KB \cup O \models^s_L \neg q$. Then a diagnosis is a set $S \subseteq H$ such that $KB \cup O \cup S \models^s_L q$. An explanation for q from KB_a to KB_h is, on the other hand, a pair (e^+, e^-) such that $(KB_h \setminus e^-) \cup e^+ \models^s_L q$. Thus, the key difference is that an explanation might require the removal of some knowledge of KB_h while a diagnosis does not.

Another earlier research direction that is closely related to the proposed notion of explanation is that of developing explanation capabilities of knowledge-based systems and decision support systems, which resulted in different notions of explanation such as trace, strategic, deep, or reasoning explanations (see review by Moulin, Irandoust, Bélanger, and Desbordes (2002)). All of these types of explanations focus on answering why certain rules in a knowledge base are used and how a conclusion is derived, which is not our focus in this paper. The present development differs from earlier proposals in that explanations are identified with the aim of explaining a given formula to a second theory. Furthermore, the notion of a cost-optimal explanation with respect to the second theory is proposed.

There have been attempts to using argumentation for explanation (Cyras, Fan, Schulz, & Toni, 2017; Cyras, Letsios, Misener, & Toni, 2019) because of the close relation between argumentation and explanation. For example, argumentation was used by Cyras et al. (2019) to answer questions such as why a schedule does (does not) satisfy a criteria (e.g., feasibility, efficiency, etc.); the approach was to develop for each type of inquiry, an abstract *argumentation framework* (AF) that helps explain the situation by extracting the attacks (non-attacks) from the corresponding AF.

The problem of restoring consistency in a knowledge base in our framework is similar in spirit to the notion of minimal repairs/diagnoses studied by Ulbricht and Baumann (2019). However, they consider an AF as an agent's knowledge base, whereas we consider knowledge bases encoding planning problems. In addition, restoring consistency intersects with the problem of finding *minimally unsatisfiable sets* (MUSes) and *minimal correction sets* (MCSes) (Marques-Silva, 2012; Marques-Silva, Heras, Janota, Previti, & Belov, 2013). However, most MUS and MCS algorithms, with the exception of one of our prior work (Vasileiou, Previti, & Yeoh, 2021), are specialized for propositional logic and, to the best of our knowledge, are with respect to a single theory. In contrast, our notion of retracting unsatisfiable sets is with respect to two theories (i.e., KB_a and KB_h).

It is worth to point out that the problem of computing a most preferred explanation for φ from KB_a to KB_h might look similar to the problem of computing a weakest sufficient condition of φ on KB_a under KB_h as described by Lin (2001). As it turns out, the two notions are quite different. Given that $KB_a = \{p, q\}$ and $KB_h = \{p\}$. It is easy to see that q is the unique explanation for q from KB_a to KB_h. On the other hand, the weakest sufficient condition of q on KB_a under KB_h is $\bot$ (Proposition 8, (Lin, 2001)).

A recent research direction that is closely related to the proposed notion of explanation is that by Shvo, Klassen, and McIlraith (2020), where they propose a general belief-based framework for generating explanations that employs epistemic state theory to capture the models of the explainer (agent in this paper) and the explainee (human user in this paper), and incorporates a belief revision operator to assimilate explanations into the explainee's epistemic states. A main difference with our proposed framework is that our framework restricts knowledge to be stored in logical formulae, while theirs considers epistemic states that can characterize different types of problems and have no such restriction.

Finally, in another line of our work, we laid the theoretical foundations and emphasized the knowledge representation aspects of model reconciliation to problems that can be formulated as logic programs (Son, Nguyen, Vasileiou, & Yeoh, 2021). Additionally, we have also developed a dedicated ASP-based solver (Nguyen, Vasileiou, Son, & Yeoh, 2020) for solving these problems. These prior work, combined with our current work, reflect that knowledge representation and reasoning can provide a fertile ground for explanation generation in model reconciliation problems and explainable planning.

9.2 Planning Literature

As the main theme of the paper falls under the general umbrella of *explainable AI planning* (XAIP), it is important to provide a general overview of XAIP and discuss current trends as well as situate our contributions within the related work in this area.

While there is some work on adapting planning algorithms to find easily explainable plans (i.e., plans that are easily understood and accepted by a human user) (Zhang, Sreedharan, Kulkarni, Chakraborti, Zhuo, & Kambhampati, 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, researchers have tackled the problem where the model of the human user (1) must be learned (Zhang et al., 2017); and (2) is of a different form or abstraction than that of the planning agent (Tian, Zhuo, & Kambhampati, 2016; Sreedharan, Srivastava, & Kambhampati, 2018). However, when designing explanatory planning systems, one of the main considerations is taking into account the personality of the explainee (Langley, 2019). Currently in the literature, there are three personalities considered:

- **Domain designer:** The person working in acquiring the model that the system works with;
- **Algorithm designer:** The developer of the algorithms of the planning system; and
- **End user:** The person interacting/collaborating with the system in the form of a user.

Naturally, these different personas will require different kinds of explanations. These explanations fall under two primary classes of explanations: *Algorithm-based explanations* and *model-based explanations* (see the survey by Sreedharan et al. (2020) for a comprehensive discussion).

Algorithm-based explanations generally target expert users (i.e., algorithm designers) and attempt to explain the inner workings of the underlying planning algorithm. For example, Magnaguagno et al. (2020) developed a state-space search visualization that represents how the distance to the goal state changes during the search procedure by defining a heuris-

tic gradient using heat maps. The gradient colors the states based on their estimated distance to the goal state and can be used to highlight that the estimated distance can be different according to the heuristic used in the algorithm. One interesting aspect of this work is that it can visualize failed planning instances, which can be practical for debugging purposes.

In contrast, model-based explanations are considered in a very large number of XAIP papers. This category consists of algorithm-agnostic explanations that can be evaluated independently of the method used to come up with. For instance, Borgo, Cashmore, and Magazzeni (2018) and Cashmore, Collins, Krarup, Krivic, Magazzeni, and Smith (2019) created a service that allows users to hold a dialogue with the system by means of specifying contrastive questions about the plan. Essentially, they assume questions specified by users can be best understood as constraints on the plans they are expecting (i.e., a certain action to be included or excluded in the plan). The explanation is then to identify an exemplary plan that satisfies those constraints and, thus, demonstrating how the computed plan is better. Such kinds of explanations fall under the broad umbrella of *contrastive explanations*, where explanations answer questions of the form "Why not A (instead of B)?", where A is an alternative (or foil) suggested by the human to decision B taken by the agent. An explanation can explain why A is suboptimal or why the agent's decisions are better than the foil. There are multiple forms this contrastive question can take, i.e., having the user present an entire plan or specific actions as foils, and it can also take on the more general form of "Why B?", where the implicit comparison is to all possible alternatives. Contrastive explanations have been gathering interest in literature, with applications in linear temporal logic systems to answer factual questions (Kasenberg, Thielstrom, & Scheutz, 2020), and in oversubscription planning to explain goal subsets (Eifler & Hoffmann, 2020) being recent examples. There has also been work towards user interfaces for decision support that gives users suggestions in response to foils provided to AI generated plans in an interactive manner (Karthik, Sreedharan, Sengupta, & Kambhampati, 2021).

On a similar thread, Göbelbecker, Keller, Eyerich, Brenner, and Nebel (2010) have considered the case of explaining why a planning problem is unsolvable. In particular, they transform an unsolvable planning problem Π into a new problem Π' by adding predicates or objects to initial states, which they refer to as *excuses*, such that Π' becomes solvable. It is interesting to note that a logic-based method, like our proposed framework, could also be used to explain the unsolvability of planning problems; Notice that an unsolvable planning problem Π translates to an unsatisfiable knowledge base KB encoding Π. Specifically, for a given time horizon h, KB is unsatisfiable if and only if there exists no plan of length h. As such, one could find the reason for the unsatisfiability of KB by computing a *minimal unsatisfiable set* (MUS) (Ignatiev, Previti, Liffiton, & Marques-Silva, 2015) over KB, which would then serve as an explanation. Additionally, one could also compute a *maximal satisfiable set* (MSS) (Ignatiev et al., 2015) over the KB in order to find potential subproblems of Π that are solvable, and which may provide useful information for Π. We leave this interesting problem for future work.

Nevertheless, the explanations considered above do not capture directly the user's knowledge of the given planning problem and are thus not a realistic inception of a true explanatory system targeting non-expert human users. It is widely accepted that human users often come with their own preconceived notions and/or expectations of the system (Carroll & Ol-

son, 1988) and, as such, human users might evaluate plans on their own models, which may disagree with the system's outcome or quality.

A key paper that considers the mental models of human users is by Chakraborti et al. (2017), who introduced the *model reconciliation problem* that we are tackling in this paper. The high-level difference between our two approaches is that our approach is based on KR while theirs is based on automated planning and heuristic search techniques. However, both approaches share a lot of similarities. Particularly, explanations generated in both approaches can be characterized according to two properties defined in Chakraborti et al. (2017): *Completeness*, that is, the plan is valid (or optimal) in the updated human user's model; and *Monotonicity*, that is, there are no model differences in the agent's and human user's models that can change the completeness of an explantion (i.e., no further model updates can invalidate an explanation). In consequence, both approaches share similar types of explanations that can be found. For example, the $\subseteq$-minimal support in Definition 5 is equivalent to *minimally complete explanations* (MCEs) (the shortest explanation that is complete), while the $\lhd$-general support can be viewed as similar to the *minimally monotonic explanations* (MMEs) (the shortest explanation that is complete and monotonic). Additionally, *model patch explanations* (MPEs) (includes all the model updates) are trivial explanations and are equivalent to our definition that the entire KB_a itself serves as an explanation for KB_h. Note that, in our approach (as also in the original model reconciliation problem), we allow for explanations on "mistaken" expectations in the human model (Algorithm 2). However, a similar property can be seen if the mental model is not known and, therefore, by taking an "empty" model as the starting point, explanations can only add to the human's understanding but not mend mistaken ones.

Although the model reconciliation problem is a good stepping stone towards creating good explanatory planning systems, it makes a strong assumption that the system has knowledge of the human user's mental model. An alternative approach that relaxes this requirement is called *model-free model reconciliation*, which predicts how model information can affect the expectation of the human user by learning a model that characterizes the human user's expectation and using it to drive the search to determine what information should be exposed to the human user (Sreedharan et al., 2019). However, there might be caveats in going model-free. The explanations generated by such systems might be purposely false in order to satisfy the human user. For example, in the model reconciliation problem the explanation was guaranteed to be consistent with the ground truth. Researchers have showed that this guarantee can be relaxed in such a way that allows the model reconciliation process to generate erroneous explanations and, hence, create ethical quandaries that would need further investigation (Chakraborti & Kambhampati, 2019; Chakraborti, Kulkarni, Sreedharan, Smith, & Kambhampati, 2019a).

Another popular theme in XAIP is that of *plan summarization*, where the main interest is in presenting a long plan to a single human user (Myers, 2006), or to multiple human users (e.g., human teams) (Kim et al., 2020). One possible way to approach this would be to use the model reconciliation process with an empty model of the human user and compute the minimal explanation (e.g., causal links) necessary to explain every action in the plan. Another possibility would be to approach this issue through a process called *verbalization of plans*, that is, paths taken by an agent along different levels of abstraction (Rosenthal, Selvaraj, & Veloso, 2016). Interestingly, there have also been efforts on approaching plan

summarization with contrastive explanations. For example, Kim, Muise, Shah, Agarwal, and Shah (2019) proposed on a Bayesian approach to infer contrastive linear temporal logic specifications aimed at explaining how two sets of plan traces differ.

Finally, it is worth mentioning that our plan validity check bears some similarity with the validity check that is provided by VAL (Howey, Long, & Fox, 2004). The key difference is that, in case of an invalid plan, VAL identifies the first action in the plan with unsatisfied preconditions, and identifies the precondition that is not satisfied. In contrast, the validity check that is provided here identifies all the differences in the model that prevents the plan to be valid.

10. Conclusions and Future Work

When designing explanatory systems, a question that often arise is how to identify, represent, and provide explanations. There is a general belief that logic-based systems are well equipped to address this question. For example, logic-based models such as decision trees produce explanations stemming directly from the model (Lakkaraju, Bach, & Leskovec, 2016; Ignatiev, Pereira, Narodytska, & Marques-Silva, 2018). In this paper, we examined and evaluated this belief by creating a logic-based explanation generation framework for classical and hybrid planning problems for the model reconciliation problem.

The model reconciliation problem is a specific problem within the area of explainable AI planning, where the plan of a planning agent is unacceptable to a human user due to differences in their models of the problem. As such, the agent needs to provide an explanation of that plan in terms of model differences. In this context, we made the following contributions: (1) We approached the MRP problem from the perspective of *knowledge representation and reasoning* by proposing the notion of explanations and defined plan validity and optimality in terms of knowledge bases; (2) We proposed several complexity cost functions to reflect preferences between explanations; (3) We developed algorithms for computing most-preferred explanations for plan validity and optimality; and (4) We empirically showed that our approach complements the current state of the art and is able to generalize beyond classical planning to hybrid planning. Finally, through our proof-of-concept, we show that explanations from our algorithms can be effectively communicated to human users on problems beyond classical planning.

For future work, we plan to investigate an iterative conversational extension, where the agent converses with the human user and, through that process, the agent is able to update its belief of the human's knowledge base similar to Zhang et al. (2017) and Sreedharan et al. (2018). In addition, we are also interested in pursuing a more personalized approach to generating explanations. Naturally, in human-to-human explanations, the explainer will usually decide how much detail to include in the explanation by choosing a conceptual model that they think will mesh with that of the explainee. Thus, explanations can be given at different levels of abstraction, based on different conceptual models (Sreedharan et al., 2018; Sreedharan, Srivastava, Smith, & Kambhampati, 2019).

Acknowledgments

We thank the anonymous reviewers, whose suggestions improved the quality of our paper. Stylianos Loukas Vasileiou, William Yeoh, and Ashwin Kumar are partially supported by the National Science Foundation (NSF) under award 1812619. Tran Cao Son is partially supported by NSF under awards 1757207, 1812628, and 1914635. 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 United States government.

Appendix A. Working Example: PDDL+ Problem

In this example, we consider a simplified version of the Generator domain, where we are interested in explaining the validity of plan. Note that, to the best of our knowledge, there does not exist optimal PDDL+ planners, and therefore we restrict ourselves to explaining the validity of plans.

The domain and problem files are shown in Listings 3 and 4, respectively. The initial state asserts that there is one generator; the generator's capacity is 1060; and initial fuel level is 1020. The goal state is that the generator has been run for a given amount of time: *generator-ran*. The plan for this problem, as found by SMTPlan (Cashmore et al., 2020), is $\pi = generate$ [1000.0].

Listing 3: Linear generator domain.

```
(define (domain generator_linear)
(:requirements :fluents :durative-actions :duration-inequalities
               :adl :typing)
(:types generator)
(:predicates (gen-ran))
(:functions (fuelLevel ?g -generator) (capacity ?g - generator)

(:durative-action gen
  :parameters (?g - generator)
  :duration (= ?duration 1000)
  :condition (over all (>= (fuelLevel ?g) 0))
  :effect (and (decrease (fuelLevel ?g) (* #t 1))
               (at end (gen-ran))))
 )
```

The knowledge base of the agent comprise the encoding of two happenings (encoded in the quantifier-free nonlinear arithmetic theory), as described by Cashmore et al. (2016):

$$KB_a = [gen\_ran_{1,1} \rightarrow (gen\_ran_{0,1} \vee gen_{end,1}), \neg gen\_ran_{1,1} \rightarrow gen\_ran_{0,1}, \quad (93)$$

$$gen\_ran_{1,2} \rightarrow (gen\_ran_{0,2} \vee gen_{end,2}), \neg gen\_ran_{1,2} \rightarrow gen\_ran_{0,2}, \quad (94)$$

$$fuelLevel_{0,1} = fuelLevel_{1,1}, fuelLevel_{0,2} = fuelLevel_{1,2}, \quad (95)$$

Listing 4: Linear generator problem.

```
(define (problem run_generator)
(:domain generator_linear)
(:objects gen - generator)
(:init (= (fuelLevel gen) 1020)
       (= (capacity gen) 1060))
(:goal (and (gen-ran)))
 )
```

$$gen_{sta,1} \rightarrow (fuelLevel_{0,1} \geq 0), gen_{sta,2} \rightarrow (fuelLevel_{0,2} \geq 0), \tag{96}$$
$$gen_{end,1} \rightarrow (fuelLevel_{0,1} \geq 0), gen_{end,2} \rightarrow (fuelLevel_{0,2} \geq 0), \tag{97}$$
$$gen_{sta,1} \rightarrow (gen_{dur,2} = 1000.0), gen_{sta,2} \rightarrow (gen_{dur,2} = 1000.0), \tag{98}$$
$$\neg gen_{end,1}, gen_{end,2} \rightarrow gen_1, gen_{end,1} \rightarrow (gen_{dur,1} = 0.0), \tag{99}$$
$$gen_{end,2} \rightarrow (gen_{dur,2} = 0.0), gen_{end,1} \rightarrow gen\_ran_{1,1}, \tag{100}$$
$$gen_{end,2} \rightarrow gen\_ran_{1,2}, gen_1 = (gen_{dur,1} > 0), \tag{101}$$
$$\neg gen_1 = gen_{dur,1} = 0, gen_2 = (gen_{dur,2} > 0), \tag{102}$$
$$\neg gen_2 = gen_{dur,2} = 0, \neg gen\_start_i \vee \neg gen\_end_i, \tag{103}$$
$$\neg gen\_ran_{0,1}, fuelLevel_{0,1} = 1020, capacity_{0,1} = 1060, gen\_ran_{1,2}, \tag{104}$$
$$time_1 = 0, time_2 \geq time_1 + 0.001, gen\_ran_{1,1} = gen\_ran_{0,2}, \tag{105}$$
$$gen_1 \rightarrow (fuelLevel_{0,1} \geq 0), gen_2 \rightarrow (fuelLevel_{0,2} \geq 0), \tag{106}$$
$$gen_1 \rightarrow (gen_{dur,2} = gen_{dur,1} + time_1 + time_2), \tag{107}$$
$$gen_1 \rightarrow fuelLevel_{0,2} = fuelLevel_{1,1} - time_2 + time_1, \tag{108}$$
$$\neg gen_1 \rightarrow (fuelLevel_{0,2} = fuelLevel_{1,1}), capacity_{0,2} = capacity_{0,1}] \tag{109}$$

for $i = \{1, 2\}$.

Suppose now that in the human user's knowledge the duration of action *generate* is 2000 and that it is missing the effect *generator-ran* from action *generate*:

$$KB_h = [fuelLevel_{0,1} = fuelLevel_{1,1}, fuelLevel_{0,2} = fuelLevel_{1,2}, \tag{110}$$
$$gen_{sta,1} \rightarrow (fuelLevel_{0,1} \geq 0), gen_{sta,2} \rightarrow (fuelLevel_{0,2} \geq 0), \tag{111}$$
$$gen_{end,1} \rightarrow (fuelLevel_{0,1} \geq 0), gen_{end,2} \rightarrow (fuelLevel_{0,2} \geq 0), \tag{112}$$
$$gen_{sta,1} \rightarrow (gen_{dur,2} = 2000.0), gen_{sta,2} \rightarrow (gen_{dur,2} = 2000.0), \tag{113}$$
$$\neg gen_{end,1}, gen_{end,2} \rightarrow gen_1, gen_{end,1} \rightarrow (gen_{dur,1} = 0.0), \tag{114}$$
$$gen_{end,2} \rightarrow (gen_{dur,2} = 0.0), gen_1 = (gen_{dur,1} > 0), \tag{115}$$
$$\neg gen_1 = gen_{dur,1} = 0, gen_2 = (gen_{dur,2} > 0), \tag{116}$$
$$\neg gen_2 = gen_{dur,2} = 0, \neg gen\_start_i \vee \neg gen\_end_i, \tag{117}$$
$$\neg gen\_ran_{0,1}, fuelLevel_{0,1} = 1020, capacity_{0,1} = 1060, gen\_ran_{1,2}, \tag{118}$$
$$time_1 = 0, time_2 \geq time_1 + 0.001, gen\_ran_{1,1} = gen\_ran_{0,2}, \tag{119}$$

$$gen_1 \rightarrow (fuelLevel_{0,1} \geq 0), gen_2 \rightarrow (fuelLevel_{0,2} \geq 0), \tag{120}$$
$$gen_1 \rightarrow (gen_{dur,2} = gen_{dur,1} + time_1 + time_2), \tag{121}$$
$$gen_1 \rightarrow fuelLevel_{0,2} = fuelLevel_{1,1} - time_2 + time_1, \tag{122}$$
$$\neg gen_1 \rightarrow (fuelLevel_{0,2} = fuelLevel_{1,1}), capacity_{0,2} = capacity_{0,1}] \tag{123}$$

for $i = \{1, 2\}$.

It is evident that the agent's plan is invalid according to the user's knowledge base (i.e., $KB_h \not\models^c_L \pi$) and, therefore, we have to find an explanation ϵ such that $\widehat{KB}^{\epsilon}_h \models^c_L \pi$.

Let's assume that ϵ consists of the following formulae:

$$\epsilon = [gen\_ran_{1,1} \rightarrow (gen\_ran_{0,1} \vee gen_{end,1}), \neg gen\_ran_{1,1} \rightarrow gen\_ran_{0,1}, \tag{124}$$
$$gen\_ran_{1,2} \rightarrow (gen\_ran_{0,2} \vee gen_{end,2}), \neg gen\_ran_{1,2} \rightarrow gen\_ran_{0,2}, \tag{125}$$
$$gen_{sta,1} \rightarrow (gen_{dur,2} = 1000.0), gen_{sta,2} \rightarrow (gen_{dur,2} = 1000.0), \tag{126}$$
$$gen_{end,1} \rightarrow gen\_ran_{1,1}, gen_{end,2} \rightarrow gen\_ran_{1,2}] \tag{127}$$

Therefore, we can see that $\widehat{KB}^{\epsilon}_h \models^c_L \pi$. It is important to note that in order to insert this explanation to KB_h, we would first have to remove the old formulae representing the duration of the action (e.g., $gen_{sta,1} \rightarrow (gen_{dur,2} = 2000.0)$) because they contradict with the new formulae in ϵ (e.g., $gen_{sta,1} \rightarrow (gen_{dur,2} = 1000.0)$) (see Definiton 4).

Appendix B. Theoretical Analysis

We first prove the completeness and correctness of the general Algorithm 1 in Theorems 1 and 2, respectively, in the context of finding explanations for optimal (or valid) plans in model reconciliation problems. We then prove the completeness of our pre-processing Algorithm 2 in Theorem 3, before proving the completeness of the combined Algorithm 3 in Theorem 4.

Theorem 1 *Algorithm 1 is guaranteed to terminate with a solution when one exists.*

Proof. First, a solution will always exist because, in the worst case, the entire $KB_a \setminus KB_h$ will serve as an explanation to KB_h since KB_a credulously entails φ_c and skeptically entails φ_s. We now prove that it is guaranteed to terminate with a solution. As Algorithm 1 iteratively adds sets of formulae of increasing size from $KB_a \setminus KB_h$ into its priority queue q (Lines 7 and 13-16), it will eventually add the entire power set of $KB_a \setminus KB_h$ into the queue. Since each element in the queue is only evaluated exactly once (Lines 8 and 14), the set of formulae in $KB_a \setminus KB_h$ will eventually be evaluated, and when it is used to update KB_h (Line 9), the updated $\widehat{KB}^{\epsilon}_h$ will credulously entail φ_c and skeptically entail φ_s (Line 10). As a result, $\epsilon = KB_a \setminus KB_h$ will be returned as a solution upon termination (Line 11). □

Theorem 2 *Algorithm 1 is guaranteed to return a most-preferred explanation if the cost function is monotonic.*

Proof. If the cost function $\mathcal{C}_L$ used by Algorithm 1 is monotonic, then for any two explanations $\epsilon_1 \subseteq \epsilon_2$, $\mathcal{C}_L(KB_h, \epsilon_1) \leq \mathcal{C}_L(KB_h, \epsilon_2)$.

We now prove by contradiction that it is not possible for Algorithm 1 to return an explanation ϵ that is less preferred than a most-preferred explanation ϵ^* (i.e., $\mathcal{C}_L(KB_h, \epsilon) > \mathcal{C}_L(KB_h, \epsilon^*)$). Assume that the algorithm does return such an explanation ϵ. Since potential explanations are popped off the priority queue according to their costs (Line 7 and 16), it means that when the algorithm popped off explanation ϵ, the explanation ϵ^* is not in the priority queue since $\mathcal{C}_L(KB_h, \epsilon) > \mathcal{C}_L(KB_h, \epsilon^*)$. There are the following two cases:

- Explanation ϵ^* is not in the queue because it was already popped off earlier. In this case, the algorithm would have terminated and returned the explanation ϵ^*, which contradicts our assumption that the algorithm returned explanation ϵ.
- Explanation ϵ^* is not in the queue because it hasn't yet been added into the queue. This means that there exists some subset $\epsilon' \subset \epsilon^*$ that is in the queue and is not yet evaluated. Further, it must be the case that $\mathcal{C}_L(KB_h, \epsilon) \leq \mathcal{C}_L(KB_h, \epsilon')$ because, otherwise, ϵ' would have been popped off the queue and evaluated. Additionally, since the cost function is monotonic, $\mathcal{C}_L(KB_h, \epsilon') \leq \mathcal{C}_L(KB_h, \epsilon^*)$. Combining these two inequalities, we get $\mathcal{C}_L(KB_h, \epsilon) \leq \mathcal{C}_L(KB_h, \epsilon^*)$, which contradicts our assumption that $\mathcal{C}_L(KB_h, \epsilon) > \mathcal{C}_L(KB_h, \epsilon^*)$.

Therefore, it is not possible for Algorithm 1 to return an explanation ϵ that is less preferred than a most-preferred explanation ϵ^*. □

Recall from Section 6 that each formula in KB_h will have a corresponding formula in KB_a since KB_a is assumed to be complete.[26] However, it may not be true that each formula in KB_a will have a corresponding formula in KB_h since KB_h can be incomplete. We formalize this statement in the following postulate, and then use it to prove properties of our pre-processing Algorithm 2.

Postulate 1 *For a model reconciliation problem, assume that KB_a and KB_h encode the SAT (or SMT) instances of the planning agent and human user, respectively. Then, each formula in KB_h will have a corresponding formula in KB_a.*

Theorem 3 *Algorithm 2 is guaranteed to terminate with a solution when one exists.*

Proof. Algorithm 2 iteratively evaluates all formulae in KB_h with respect to KB_a's partial model (Line 20). If any formulae evaluate to false with respect to KB_a's partial model, the algorithm will replace them with the corresponding ones from KB_a (Lines 21-22). Since the number of formulae in KB_h is finite, the algorithm will eventually complete evaluating all the formulae and return the set of formulae from KB_a that correspond to the set of formulae in KB_h that evaluates to false (Line 23). In other words, if a solution exists, Algorithm 2 is guaranteed to return it. □

Theorem 4 *Algorithm 3 is guaranteed to terminate with a solution when one exists.*

Proof. As Algorithm 3 comprises of Algorithms 1 and 2, which are guaranteed to terminate with a solution when one exists (Theorems 1 and 3), the algorithm is also guaranteed to terminate with a solution when one exists. □

We now describe the worst-case time complexities of the algorithms.

26. Assuming that $KB_h \neq \emptyset$.

Theorem 5 *The time complexity of Algorithm 1 is $O(2^{|KB_a|} + 2^{|KB_a \setminus KB_h|+m})$, where m is the maximum number of variables in $\widehat{KB}_h^\epsilon$ over all candidate explanations ϵ.*

Proof. On the basic operations, the runtimes for inserting elements into sets (Line 8), checking for set memberships (Line 14), and computing costs of potential explanations (Line 15) are all $O(1)$; and the runtimes of inserting and removing elements into priority queues (Line 7 and 16) are $O(\log(n))$, where n is the size of the queue. On the entailment checks on Lines 1 and 10, their runtimes are $O(2^m)$, where m is the number of variables in the knowledge base, because that is the number of models in the knowledge base. The number of times the algorithm has to loop through Lines 6 to 17 is the size of the power set $KB_a \setminus KB_h$, which is $O(2^{|KB_a \setminus KB_h|})$, since there are that many unique subsets of potential explanations to consider. Within the loop, Line 9 takes $O(2^{|KB_h|})$ time because it has to iterate through the power set of KB_h to find the minimal set of formulae to remove according to Definition 4, and the number of times the algorithm has to loop through Lines 13 to 16 is $|KB_a \setminus KB_h|$.

Therefore, in total, the runtime of the algorithm is $O(2^m)$ (Line 1) + $O(2^n)$ (Lines 6 and 17) $\cdot\big[O(\log(n))$ (Lines 7-8) + $O(2^{|KB_h|})$ (Line 9) + $O(2^m)$ (Line 10) + $O(n)$ (Line 13) $\cdot O(\log(n))$ (Line 16) $\big] = O(2^n \cdot (2^{|KB_h|} + 2^m + n\log(n))) = O(2^{n+|KB_h|} + 2^{n+m}) = O(2^{|KB_a|} + 2^{n+m})$, where $n = |KB_a \setminus KB_h|$ and m is the maximum number of variables in $\widehat{KB}_h^\epsilon$ over all candidate explanations ϵ. □

Theorem 6 *The time complexity of Algorithm 2 is $O(2^n + |KB_h| \cdot |KB_a|)$, where n is the number of variables in KB_a.*

Proof. The time complexity of extracting the partial model (Function 1) on Line 19 is as follows. Finding a satisfying model M from KB_a for π^* (Line 25) takes $O(2^n)$, where n is the number of variables in KB_a, since it is a Boolean satisfiability problem, which is NP-complete (Cook, 1971). Extracting the relevant literals (Line 26) takes $O(l)$ time, where l is the maximum number of relevant literals. Finally, Lines 27-29 take $O(|M|)$. Therefore, the runtime for Line 19 is $O(2^n) + O(l) + O(|M|) = O(2^n)$.

The number of times the algorithm has to loop through Lines 20 to 22 is $O(|KB_h|)$. Within the loop, Line 21 takes $O(k)$ time, where k is the length of the longest formula evaluated,[27] and Line 22 takes $O(|KB_a|)$ time since it needs to loop through the entire KB_a in the worst case to find the corresponding formula.[28]

Therefore, in total, the runtime of the algorithm is $O(2^n)$ (Line 19) + $O(|KB_h|)$ (Line 20) $\cdot\big[O(k)$ (Line 21) + $O(|KB_a|)$ (Line 22) $\big] = O(2^n + |KB_h| \cdot (k + |KB_a|)) = O(2^n + |KB_h| \cdot |KB_a|)$, where n is the number of variables in KB_a. . □

Theorem 7 *The time complexity of Algorithm 3 is $O(2^{|KB_a|} + 2^{|KB_a \setminus KB_h|+m})$, where m is the maximum number of variables in $\widehat{KB}_h^\epsilon$ over all candidate explanations ϵ.*

Proof. The pre-processing call on Line 32 takes $O(2^n + |KB_h| \cdot |KB_a|)$, where n is the number of variables in KB_a (Theorem 6).

27. We assume the formula is represented in conjunctive normal form.
28. This upper bound can be tightened by using hash functions, but we consider naive implementations here.

Algorithm 4: `linear-search-update`$(L, KB_h, KB_a, \epsilon)$

Input: Logic L, knowledge bases KB_a and KB_h, and a set of formulae ϵ
Output: Updated KB with ϵ

```
1  if KB_h ∪ ϵ ⊭^s_L ⊥ then
2      KB_h ← KB_h ∪ ϵ
3      return KB_h
4  else
5      H ← ϵ ∪ {KB_h ∩ KB_a}
6      S ← KB_h \ H
7      for s ∈ S do
8          if H ∪ {s} ⊭^s_L ⊥ then
9              H ← H ∪ {s}
10     return H
```

To implement the update of the knowledge base KB_h with the explanation ϵ to $\widehat{KB}_h^{\epsilon}$ on Line 33, we loop through the explanation ϵ and, for each formula in the explanation, we loop through KB_h to find the corresponding formula. Once found, we replace the formula in KB_h with the formula from the explanation. Therefore, the runtime for this step is $O(|\epsilon| \cdot |KB_h|)$.

Finally, the runtime for Line 34 is $O(2^{|KB_a|} + 2^{|KB_a \setminus \widehat{KB}_h| + m})$, where n is the maximum number of variables in $\widehat{KB}_h^{\epsilon'}$ over all candidate explanations ϵ' (Theorem 5). Assuming $|KB_h| = |\widehat{KB}_h|$, then the runtime is $O(2^{|KB_a|} + 2^{|KB_a \setminus KB_h| + m})$.

Therefore, in total, the runtime of the algorithm is $O(2^m + |KB_h| \cdot |KB_a|)$ (Line 32) + $O(|\epsilon| \cdot |KB_h|)$ (Line 33) + $O(2^{|KB_a|} + 2^{|KB_a \setminus \widehat{KB}_h| + m})$ (Line 34) = $O(2^m + |KB_h| \cdot |KB_a| + |\epsilon| \cdot |KB_h| + 2^{|KB_a|} + 2^{|KB_a \setminus \widehat{KB}_h| + m}) = O(2^{|KB_a|} + 2^{|KB_a \setminus KB_h| + m})$, where m is the maximum number of variables in $\widehat{KB}_h^{\epsilon}$ over all candidate explanations ϵ. □

Appendix C. Knowledge Base Update with a Linear Search Algorithm

As noted in Section 7.1, we used a simple linear search algorithm to perform the knowledge base update. Specifically, such a procedure is needed to restore the consistency of the knowledge base if it becomes unsatisfiable with an explanation. Algorithm 4 describes the pseudocode of this procedure. It starts by checking if the knowledge base KB_h conjuncted with ϵ is satisfiable (Line 1). If it is, it returns the knowledge base updated with ϵ (Lines 2-3), otherwise it proceeds to its main procedure. First, it constructs the following two sets: A set `H` comprising of ϵ and the formulae that are in the intersection of KB_h and KB_a (Line 5); and a set `S` comprising all formulae in KB_h but those in `H` (Line 6). In the SAT literature, sets `H` and `S` are referred to as *hard* and *soft* constraints, respectively (Li & Manya, 2009). Intuitively, in a usual SAT problem, if a knowledge base is partitioned into hard and soft constraints, the formulae in the hard constraints must be satisfied by the SAT algorithm, while formulae in the soft constraints may not be satisfied. For the sake of Question 1 in Section 7, in this implementation, we ensure that ϵ and the formulae that are in the intersection of KB_h and KB_a will not be removed. Then, the algorithm removes

one formula s from `S` at a time (Line 6) and checks if the union of `H` and s is satisfiable (Line 8). If it is, then s is added to `H` (Line 9). Finally, the algorithm returns a satisfiable KB_h updated with ϵ (Line 10).

Appendix D. Proof-of-Concept: Communicating Explanation to Human Users

D.1 Human User Study: Comprehension Questions

Q1. *What caused the error(s) in your plan?* (Multiple Choice)

- I made a mistake in my plan.
- Wrong information provided in domain description.
- Missing information in domain description.
- There were no errors in my plan.

Q2. *Given your plan and the explanation provided, why is your plan invalid? Please be as descriptive as possible.* (Open-ended)

Q3. *Do you feel you understand what information was different in the domain Rob gave you? If yes, what was that information?* (Open-ended)

Q4. *What are the corrections needed to your plan to make it achieve the goal, with the new information in mind?* (Open-ended)

Q5. *Where do you think the error in Rob's domain description was?* (Multiple choice)

- In the description of the actions and their preconditions.
- In the description of the states (start state or goal state).

Q6. *If applicable, identify areas with wrong or missing preconditions by clicking on the corresponding region. Double click to unselect.* (Users are shown a selection area where they can click on various actions.)

Q7. *If applicable, identify areas with wrong or missing start states by clicking on the corresponding region. Double click to unselect.* (Users are shown a selection area where they can click on various states.)

These questions ensured that the participants had to think about what the explanation meant, and hence allow us to see if they really understood it.

To evaluate the user responses, we scored them for each question, where the maximum score that can be achieved is 8 points. For the open-ended questions (Q2, Q3, and Q4), we manually read through the answers and assigned a correct and incorrect flag. For the other questions, we had an answer key to check against the user responses. All questions except Q5 are worth 1 point. Q5 is worth 2 points if participants only select the correct answer, 1 point if they select both answers, and 0 otherwise.

D.2 From Logic-based Explanations to Natural Language

We now describe a simple method for transforming logic-based explanations from our framework into a human-understandable format. To do that, we leverage the expressivity and symbolic nature of logic. Notice that a knowledge base encoding a planning problem contains logical formulae that represent various phenomena of the problem. These formulae

Formula Type		Template
Initial state:	f_0	$\{f\}$.*name* must be part of the initial specification of the problem.
Goal state:	f_n	$\{f\}$.*name* must be part of the goal specification of the problem.
Action Precondition:	$a_t \Rightarrow f_t$	Action $\{a\}$.*name* requires precondition $\{f\}$.*name*.
Action Addition effect:	$a_t \Rightarrow f_{t+1}$	Action $\{a\}$.*name* requires addition effect $\{f\}$.*name*.
Action Deletion effect:	$a_t \Rightarrow \neg f_{t+1}$	Action $\{a\}$.*name* requires deletion effect $\{f\}$.*name*.
Action Duration:	$a_{t\_\text{sta}} \Rightarrow a_{t\_\text{dur}} \leq duration \wedge a_{t\_\text{dur}} \geq duration$	Action $\{a\}$.*name* has a duration of $\{duration\}$.
Process precondition:	$ps_t \Leftrightarrow f_t$	Process $\{ps\}$.*name* requires precondition $\{f\}$.*name*.

Table 6: Various formula types and their mapping onto pre-defined natural language templates.

are of a specific type, i.e., there are formulae encoding the initial and goal states, the action dynamics of the problem, and so on (see Sections 3.2.1 and 3.3.1). Each formula is grounded on propositional variables, with each variable "symbolizing" a planning element such as an action or a predicate. For example, $a_0 \rightarrow p_0$ is a formula characterizing that action a_0 has precondition p_0. As such, given an explanation consisting of a set of logical formulae, each formula's variables can be extracted and, depending on the type of the formula, be mapped onto pre-defined, natural language templates.[29] To offer some more concrete examples, Table 6 shows this method for different types of formulae that may arise in an explanation.

References

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

Baron-Cohen, S. (1999). *The Evolution of a Theory of Mind*. Oxford University Press.

Barrett, C., & Tinelli, C. (2018). Satisfiability modulo theories. In *Handbook of Model Checking*, pp. 305–343. Springer.

Bogomolov, S., Magazzeni, D., Podelski, A., & Wehrle, M. (2014). Planning as model checking in hybrid domains. In *AAAI*, pp. 2228–2234.

Borgo, R., Cashmore, M., & Magazzeni, D. (2018). Towards providing explanations for AI planner decisions. *arXiv preprint arXiv:1810.06338*.

Brunyé, T. T., Taylor, H. A., & Rapp, D. N. (2008). Repetition and dual coding in procedural multimedia presentations. *Applied Cognitive Psychology*, *22*(7), 877–895.

Bryce, D., Bonasso, P., Adil, K., Bell, S., & Kortenkamp, D. (2017). In-situ domain modeling with fact routes. In *ICAPS Workshop on User Interfaces and Scheduling and Planning*, pp. 15–22.

Carroll, J. M., & Olson, J. R. (1988). Mental models in human-computer interaction. In *Handbook of Human-Computer Interaction*, pp. 45–65. Elsevier.

Cashmore, M., Collins, A., Krarup, B., Krivic, S., Magazzeni, D., & Smith, D. (2019). Towards explainable AI planning as a service. In *ICAPS Workshop on Explainable AI Planning*.

29. Note that within our framework, this becomes relatively straightforward, as our explanations consist of macro-formulae (see end of Section 6.1), i.e., there will be no formulae repeated across multiple time steps in our explanations.

Cashmore, M., Fox, M., & Giunchiglia, E. (2012). Planning as quantified Boolean formula. In *ECAI*, Vol. 242, pp. 217–222.

Cashmore, M., Fox, M., Long, D., & Magazzeni, D. (2016). A compilation of the full PDDL+ language into SMT. In *ICAPS*, pp. 79–87.

Cashmore, M., Magazzeni, D., & Zehtabi, P. (2020). Planning for hybrid systems via satisfiability modulo theories. *Journal of Artificial Intelligence Research*, *67*, 235–283.

Chakraborti, T., Fadnis, K. P., Talamadupula, K., Dholakia, M., Srivastava, B., Kephart, J. O., & Bellamy, R. K. E. (2018). Visualizations for an explainable planning agent. In *IJCAI*, pp. 5820–5822.

Chakraborti, T., & Kambhampati, S. (2019). (When) can AI bots lie?. In *AAAI/ACM Conference on AI, Ethics, and Society*, pp. 53–59.

Chakraborti, T., Kambhampati, S., Scheutz, M., & Zhang, Y. (2017). AI challenges in human-robot cognitive teaming. *arXiv preprint arXiv:1707.04775*.

Chakraborti, T., Kulkarni, A., Sreedharan, S., Smith, D. E., & Kambhampati, S. (2019a). Explicability? Legibility? Predictability? Transparency? Privacy? Security? The emerging landscape of interpretable agent behavior. In *ICAPS*, pp. 86–96.

Chakraborti, T., Sreedharan, S., Grover, S., & Kambhampati, S. (2019b). Plan explanations as model reconciliation – an empirical study. In *HRI*, pp. 258–266.

Chakraborti, T., Sreedharan, S., & Kambhampati, S. (2019c). Balancing explicability and explanations in human-aware planning. In *IJCAI*, pp. 1335–1343.

Chakraborti, T., Sreedharan, S., Zhang, Y., & Kambhampati, S. (2017). Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*, pp. 156–163.

Chen, G., Ding, Y., Edwards, H., Chau, C. H., Hou, S., Johnson, G., Sharukh Syed, M., Tang, H., Wu, Y., Yan, Y., Gil, T., & Nir, L. (2020). Planimation. *arXiv preprint arXiv:2008.04600*.

Clark, R. C., & Mayer, R. E. (2016). *E-learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning*. John Wiley & Sons.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *ACM Symposium on Theory of Computing*, pp. 151–158.

Cyras, K., Fan, X., Schulz, C., & Toni, F. (2017). Assumption-based argumentation: Disputes, explanations, preferences. *Journal of Logics and Their Applications*, *4*(8), 2407–2456.

Cyras, K., Letsios, D., Misener, R., & Toni, F. (2019). Argumentation for explainable scheduling. In *AAAI*, pp. 2752–2759.

Davis, M., Logemann, G., Donald, & Loveland (1962). A machine program for theorem proving. *Communications of the ACM*, *5*(7), 394–397.

De Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. In *TACAS*, pp. 337–340.

Domshlak, C., Hoffmann, J., & Sabharwal, A. (2009). Friends or foes? On planning as satisfiability and abstract CNF encodings. *Journal of Artificial Intelligence Research*, *36*, 415–469.

Eifler, R., & Hoffmann, J. (2020). Iterative planning with plan-space explanations: A tool and user study. *arXiv preprint arXiv:2011.09705*.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, *20*, 61–124.

Fox, M., & Long, D. (2006). Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, *27*, 235–297.

Fox, M., Long, D., & Magazzeni, D. (2017). Explainable planning. *arXiv preprint arXiv:1709.10256*.

Freedman, R. G., Chakraborti, T., Talamadupula, K., Magazzeni, D., & Frank, J. D. (2018). User interfaces and scheduling and planning: Workshop summary and proposed challenges. In *2018 AAAI Spring Symposium Series*.

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

Gärdenfors, P. (1988). *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* The MIT press.

Gärdenfors, P., Rott, H., Gabbay, D., Hogger, C., & Robinson, J. (1995). Belief revision. *Computational Complexity*, *63*(6), 35–132.

Gerevini, A., Saetti, A., & Serina, I. (2006). An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research*, *25*, 187–231.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). PDDL – the planning domain definition language. Tech. rep. TR-98-003, Yale Center for Computational Vision and Control.

Göbelbecker, M., Keller, T., Eyerich, P., Brenner, M., & Nebel, B. (2010). Coming up with good excuses: What to do when no plan can be found. In *ICAPS*, pp. 81–88.

Gopalakrishnan, S., & Kambhampati, S. (2018). TGE-viz: Transition graph embedding for visualization of plan traces and domains. *arXiv preprint arXiv:1811.09900*.

Helmert, M. (2002). Decidability and undecidability results for planning with numerical state variables. In *AIPS*, pp. 44–53.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research*, *26*, 191–246.

Hempel, C. G. (1965). *Aspects of Scientific Explanation and other Essays in the Philosophy of Science*. New York: Free Press.

Hempel, C. G., & Oppenheim, P. (1948). Studies in the logic of explanation. *Philosophy of Science*, *15*(2), 135–175.

Henzinger, T. A. (2000). The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pp. 265–292. Springer.

Holzinger, A., Carrington, A., & Müller, H. (2020). Measuring the quality of explanations: The system causability scale (SCS). *KI-Künstliche Intelligenz*, *34*(2), 1–6.

Howey, R., Long, D., & Fox, M. (2004). VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *ICTAI*, pp. 294–301.

Ignatiev, A., Pereira, F., Narodytska, N., & Marques-Silva, J. (2018). A SAT-based approach to learn explainable decision sets. In *IJCAR*, pp. 627–645.

Ignatiev, A., Previti, A., Liffiton, M., & Marques-Silva, J. (2015). Smallest MUS extraction with minimal hitting set dualization. In *CP*, pp. 173–182.

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

Kambhampati, S. (2019). Synthesizing explainable behavior for human-AI collaboration. In *AAMAS*, pp. 1–2.

Karthik, V., Sreedharan, S., Sengupta, S., & Kambhampati, S. (2021). RADAR-X: An interactive interface pairing contrastive explanations with revised plan suggestions. In *AAAI*, pp. 16051–16053.

Kasenberg, D., Thielstrom, R., & Scheutz, M. (2020). Generating explanations for temporal logic planner decisions. In *ICAPS*, pp. 449–458.

Katsuno, H., & Mendelzon, A. O. (1991a). On the difference between updating a knowledge base and revising it. In *KR*, pp. 230–237.

Katsuno, H., & Mendelzon, A. O. (1991b). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, *52*(3), 263–294.

Kautz, H., McAllester, D., & Selman, B. (1996). Encoding plans in propositional logic. In *KR*, pp. 374–384.

Kautz, H., & Selman, B. (1992). Planning as satisfiability. In *ECAI*, pp. 359–363.

Kim, J., et al. (2020). *Plan Summarization for Decision Support in Human Team Planning*. Ph.D. thesis, Massachusetts Institute of Technology.

Kim, J., Muise, C., Shah, A., Agarwal, S., & Shah, J. (2019). Bayesian inference of linear temporal logic specifications for contrastive explanations. In *IJCAI*, pp. 5591–5598.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *ACM SIGKDD*, pp. 1675–1684.

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

Langley, P. (2019). Varieties of explainable agency. In *ICAPS Workshop on Explainable AI Planning*.

Levesque, H. J. (1989). A knowledge-level account of abduction. In *IJCAI*, pp. 1061–1067.

Levi, I. (1978). Subjunctives, dispositions and chances. *Synthese*, *34*(4), 423–455.

Li, C. M., & Manya, F. (2009). MaxSAT, hard and soft constraints. *Handbook of Satisfiability*, *185*, 613–631.

Lin, F. (2001). On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, *128*(1-2), 143–159.

Magnaguagno, M. C., Fraga Pereira, R., Móre, M. D., & Meneguzzi, F. R. (2020). Web planner: A tool to develop classical planning domains and visualize heuristic state-space search. In *Knowledge Engineering Tools and Techniques for AI Planning*, pp. 209–227.

Mandel, T. (2002). User/system interface design. *Encyclopedia of Information Systems*, *1*, 1–4.

Marques-Silva, J. (2012). Computing minimally unsatisfiable subformulas: State of the art and future directions.. *Journal of Multiple-Valued Logic & Soft Computing*, *19*(1), 163–183.

Marques-Silva, J., Heras, F., Janota, M., Previti, A., & Belov, A. (2013). On computing minimal correction subsets. In *IJCAI*, pp. 615–622.

Mayer, R. E. (1997). Multimedia learning: Are we asking the right questions?. *Educational Psychologist*, *32*(1), 1–19.

McDermott, D. M. (2000). The 1998 AI planning systems competition. *AI Magazine*, *21*(2), 35–35.

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

Moulin, B., Irandoust, H., Bélanger, M., & Desbordes, G. (2002). Explanation and argumentation capabilities: Towards the creation of more persuasive agents. *Artificial Intelligence Review*, *17*(3), 169–222.

Myers, K. L. (2006). Metatheoretic plan summarization and comparison. In *ICAPS*, pp. 182–192.

Nguyen, V., Vasileiou, S. L., Son, T. C., & Yeoh, W. (2020). Explainable planning using answer set programming. In *KR*, pp. 662–666.

Palan, S., & Schitter, C. (2018). Prolific.ac—A subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, *17*, 22–27.

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind?. *Behavioral and Brain Sciences*, *1*(4), 515–526.

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

Robinson, N., Gretton, C., Pham, D. N., & Sattar, A. (2009). SAT-based parallel planning using a split representation of actions. In *ICAPS*, pp. 281–288.

Rosenthal, S., Selvaraj, S. P., & Veloso, M. M. (2016). Verbalization: Narration of autonomous robot experience.. In *IJCAI*, pp. 862–868.

Shvo, M., Klassen, T. Q., & McIlraith, S. A. (2020). Towards the role of theory of mind in explanation. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, pp. 75–93.

Son, T. C., Nguyen, V., Vasileiou, S. L., & Yeoh, W. (2021). Model reconciliation in logic programs. In *JELIA*, pp. 393–406.

Sreedharan, S., Chakraborti, T., & Kambhampati, S. (2018). Handling model uncertainty and multiplicity in explanations via model reconciliation. In *ICAPS*, pp. 518–526.

Sreedharan, S., Chakraborti, T., & Kambhampati, S. (2020). The emerging landscape of explainable automated planning & decision making. In *IJCAI*, pp. 4803–4811.

Sreedharan, S., Hernandez, A. O., Mishra, A. P., & Kambhampati, S. (2019). Model-free model reconciliation. In *IJCAI*, pp. 587–594.

Sreedharan, S., Srivastava, S., & Kambhampati, S. (2018). Hierarchical expertise level modeling for user specific contrastive explanations. In *IJCAI*, pp. 4829–4836.

Sreedharan, S., Srivastava, S., Smith, D. E., & Kambhampati, S. (2019). Why can't you do that HAL? Explaining unsolvability of planning tasks. In *IJCAI*, pp. 1422–1430.

Tian, X., Zhuo, H. H., & Kambhampati, S. (2016). Discovering underlying plans based on distributed representations of actions. In *AAMAS*, pp. 1135–1143.

Ulbricht, M., & Baumann, R. (2019). If nothing is accepted–repairing argumentation frameworks. *Journal of Artificial Intelligence Research*, *66*, 1099–1145.

Vasileiou, S. L., Previti, A., & Yeoh, W. (2021). On exploiting hitting sets for model reconciliation. In *AAAI*, pp. 6514–6521.

Winslett, M. S. (1988). Reasoning about action using a possible models approach. In *AAAI*, pp. 1427–1432.

Zahedi, Z., Olmo, A., Chakraborti, T., Sreedharan, S., & Kambhampati, S. (2019). Towards understanding user preferences for explanation types in model reconciliation. In *HRI*, pp. 648–649.

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