Preferred Explanations: Theory and Generation via Planning

Shirin Sohrabi
Department of Computer Science
University of Toronto
Toronto, Canada

Jorge A. Baier
Depto. de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Santiago, Chile

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

In this paper we examine the general problem of generating preferred explanations for observed behavior with respect to a model of the behavior of a dynamical system. This problem arises in a diversity of applications including diagnosis of dynamical systems and activity recognition. We provide a logical characterization of the notion of an explanation. To generate explanations we identify and exploit a correspondence between explanation generation and planning. The determination of *good* explanations requires additional domain-specific knowledge which we represent as preferences over explanations. The nature of explanations requires us to formulate preferences in a somewhat retrodictive fashion by utilizing Past Linear Temporal Logic. We propose methods for exploiting these somewhat unique preferences effectively within state-of-the-art planners and illustrate the feasibility of generating (preferred) explanations via planning.

Introduction

In recent years, planning technology has been explored as a computational framework for a diversity of applications. One such class of applications is the class that corresponds to *explanation generation* tasks. These include narrative understanding, plan recognition (Ramírez and Geffner 2009), finding excuses (Göbelbecker et al. 2010), and diagnosis (e.g., Sohrabi, Baier, and McIlraith 2010; Grastien et al. 2007).[1] While these tasks differ, they share a common computational core, calling upon a dynamical system model to account for system behavior, observed over a period of time. The observations may be over aspects of the state of the world, or over the occurrence of events; the account typically takes the form of a set or sequence of actions and/or state that is extracted from the construction of a plan that embodies the observations. For example, in the case of diagnosis, the observations might be of the, possibly aberrant, behavior of an electromechanical device over a period of time, and the explanation a sequence of actions that conjecture faulty events. In the case of plan recognition, the observations might be of the actions of an agent, and the explanation a plan that captures what the agent is doing and/or the final goal of that plan.

[1](Grastien et al. 2007) characterized diagnosis in terms of SAT but employed a planning-inspired encoding.

Here we conceive the computational core underlying explanation generation of dynamical systems as a nonclassical planning task. Our focus in this paper is with the generation of *preferred* explanations – how to specify preference criteria, and how to compute preferred explanations using planning technology. Most explanation generation tasks that distinguish a subset of preferred explanations appeal to some form of domain-independent criteria such as minimality or simplicity. Domain-specific knowledge has been extensively studied within the static-system explanation and abduction literature as well as in the literature on specific applications such as diagnosis. Such domain-specific criteria often employ probabilistic information, or in its absence default logic of some notion of specificity (e.g., Brewka 1994).

In 2010, we examined the problem of diagnosis of discrete dynamical systems (a task within the family of explanation generation tasks), exploiting planning technology to compute diagnoses and suggesting the potential of planning preference languages as a means of specifying preferred diagnoses (Sohrabi, Baier, and McIlraith 2010). Building on our previous work, in this paper we explicitly examine the use of preference languages for the broader task of explanation generation. In doing so, we identify a number of somewhat unique representational needs. Key among these is the need to talk about the *past* (e.g., *"If I observe that my car has a flat tire then I prefer explanations where my tire was previously punctured."*) and the need to encode complex observation patterns (e.g., *"My brakes have been failing intermittently."*) and how these patterns relate to possible explanations. To address these requirements we specify preferences in Past Linear Temporal Logic (pLTL), a superset of Linear Temporal Logic (LTL) that is augmented with modalities that reference the past. We define a finite variant of pLTL, f-pLTL, that is augmented to include action occurrences.

Motivated by a desire to generate explanations using state-of-the-art planning technology, we propose a means of compiling our f-pLTL preferences into the syntax of PDDL3, the Planning Domain Description Language 3 that supports the representation of temporally extended preferences (Gerevini et al. 2009). Although, f-pLTL is more expressive than the preference subset of PDDL3 (e.g., f-pLTL has action occurrences and arbitrary nesting of temporal modalities), our compilation preserves the f-pLTL semantics while conforming to PDDL3 syntax. This enables us to exploit PDDL3-compliant preference-based planners for the purposes of generating preferred explanations. We also propose a further compilation to remove all temporal modalities from the syntax of our preferences (while preserving their semantics) enabling the exploitation of cost-based planners for computing preferred explanations. Additionally, we exploit the fact that observations are *known* a priori to pre-process our suite of explanation preferences prior to explanation generation in a way that further simplifies the preferences and their exploitation. We show that this compilation significantly improves the time required to find preferred explanations, sometimes by orders of magnitude. Experiments illustrate the feasibility of generating (preferred) explanations via planning.

Explaining Observed Behavior

In this section we provide a logical characterization of a preferred explanation for observed behavior with respect to a model of a dynamical system. In what follows we define each of the components of this characterization, culminating in our characterization of a preferred explanation.

Dynamical Systems

Dynamical systems can be formally described in many ways. In this paper we assume a finite domain and model dynamical systems as transition systems. For convenience, we define transitions systems using a planning language. As such transitions occur as the result of actions described in terms of preconditions and effects. Formally, a dynamical system is a tuple $\Sigma = (F, A, I)$, where F is a finite set of fluent symbols, A is a set of actions, and I is a set of clauses over F that defines a set of possible initial states. Every action $a \in A$ is defined by a precondition $prec(a)$, which is a conjunction of fluent literals, and a set of conditional effects of the form $C \rightarrow L$, where C is a conjunction of fluent literals and L is a fluent literal.

A *system state* s is a set of fluent symbols, which intuitively defines all that is true in a particular state of the dynamical system. For a system state s, we define $M_s : F \rightarrow \{true, false\}$ as the truth assignment that assigns the truth value $true$ to f if $f \in s$, and assigns $false$ to f otherwise. We say a state s is *consistent* with a set of clauses $\mathcal{C}$, if $M_s \models c$, for every $c \in \mathcal{C}$. Given a state s consistent with I, we denote Σ/s as the dynamical system $(F, A, I/s)$, where I/s stands for the set of unit clauses whose only model is M_s. We say a dynamical system $\Sigma = (F, A, I)$ has a *complete initial state* iff there is a unique truth assignment $M : F \rightarrow \{true, false\}$ such that $M \models I$.

We assume that an action a is executable in a state s if $M_s \models prec(a)$. If a is executable in a state s, we define its successor state as $\delta(a, s) = (s \setminus Del) \cup Add$, where Add contains a fluent f iff $C \rightarrow f$ is an effect of a and $M_s \models C$. On the other hand Del contains a fluent f iff $C \rightarrow \neg f$ is an effect of a, and $M_s \models C$. We define $\delta(a_0 a_1 \ldots a_n, s) = \delta(a_1 \ldots a_n, \delta(a_0, s))$, and $\delta(\epsilon, s) = s$. A sequence of actions α is *executable* in s if $\delta(\alpha, s)$ is defined. Furthermore α is executable in Σ iff it is executable in s, for any s consistent with I.

Past LTL with Action Occurrences

Past modalities have been exploited for a variety of specialized verification tasks and it is well established that LTL augmented with such modalities has the same expressive power as LTL restricted to future modalities (Gabbay 1987). Nevertheless, certain properties (including fault models and explanation models) are more naturally specified and read in this augmentation of LTL. For example specifying that every alarm is due to a fault can easily be expressed by $\square(alarm \rightarrow \blacklozenge fault)$, where $\square$ means always and $\blacklozenge$ means once in the past. Note that $\neg\mathsf{U}(\neg fault, (alarm \wedge \neg fault))$ is an equivalent formulation that uses only future modalities but is much less intuitive. In what follows we define the syntax and semantics of, f-pLTL, a variant of LTL that is augmented with past modalities and action occurrences.

Syntax Given a set of fluent symbols F and a set of action symbols A, the atomic formulae of the language are: either a fluent symbol, or $occ(a)$, for any $a \in A$. Non-atomic formulae are constructed by applying negation, by applying a standard boolean connective to two formulae, or by including the future temporal modalities "until" (U), "next" ($\bigcirc$), "always"($\square$), and "eventually"($\lozenge$), or the past temporal modalities "since" (S), "yesterday" ($\bullet$), "always in the past"($\blacksquare$), and "eventually in the past"($\blacklozenge$). Future f-pLTL is the subset of f-pLTL that contains all and only the formulae with no past temporal modalities. Past f-pLTL formulae are defined analogously. A *non-temporal* formula does not contain any temporal modalities.

Semantics Given a system Σ, a sequence of actions α, and an f-pLTL formula φ, the semantics defines when α satisfies φ in Σ. Let s be a state and $\alpha = a_0 a_1 \ldots a_n$ be a sequence of actions. We say that σ is an *execution trace* of α in s iff $\sigma = s_0 s_1 s_2 \ldots s_{n+1}$ and $\delta(a_i, s_i) = s_{i+1}$, for any $i \in [0, n]$. Furthermore, if l is the sequence $\ell_0 \ell_1 \ldots \ell_n$, we abbreviate its suffix $\ell_i \ell_{i+1} \ldots \ell_n$ by l_i.

Definition 1 (Truth of an f-pLTL Formula) *An f-pLTL formula φ is satisfied by α in a dynamical system $\Sigma = (F, A, I)$ iff for any state s consistent with I, the execution trace σ of α in s is such that $\langle \sigma, \alpha \rangle \models \varphi$, where* [2]

- $\langle \sigma_i, \alpha_i \rangle \models \varphi$, *where $\varphi \in F$ iff φ is an element of the first state of σ_i.*
- $\langle \sigma_i, \alpha_i \rangle \models occ(a)$ *iff $i < |\alpha|$ and a_i is the first action of α_i.*
- $\langle \sigma_i, \alpha_i \rangle \models \bigcirc\varphi$ *iff $i < |\sigma| - 1$ and $\langle \sigma_{i+1}, \alpha_{i+1} \rangle \models \varphi$*
- $\langle \sigma_i, \alpha_i \rangle \models \mathsf{U}(\varphi, \psi)$ *iff there exists a $j \in \{i, ..., |\sigma| - 1\}$ such that $\langle \sigma_j, \alpha_j \rangle \models \psi$ and for every $k \in \{i, ..., j-1\}$, $\langle \sigma_k, \alpha_k \rangle \models \varphi$*
- $\langle \sigma_i, \alpha_i \rangle \models \bullet\varphi$ *iff $i > 0$ and $\langle \sigma_{i-1}, \alpha_{i-1} \rangle \models \varphi$*
- $\langle \sigma_i, \alpha_i \rangle \models \mathsf{S}(\varphi, \psi)$ *iff there exists a $j \in \{0, ..., i\}$ such that $\langle \sigma_j, \alpha_j \rangle \models \psi$ and for every $k \in \{j+1, ..., i\}$, $\langle \sigma_k, \alpha_k \rangle \models \varphi$*

The semantics of other temporal modalities are defined in terms of these basic elements, e.g., $\blacksquare\varphi \overset{\text{def}}{=} \neg\blacklozenge\neg\varphi$, $\blacklozenge\varphi \overset{\text{def}}{=} \mathsf{S}(\mathsf{true}, \varphi)$, and $\lozenge\varphi \overset{\text{def}}{=} \mathsf{U}(\mathsf{true}, \varphi)$.

[2]We omit standard definitions for $\neg$, $\vee$.

It is well recognized that some properties are more naturally expressed using past modalities. An additional property of such modalities is that they can construct formulae that are exponentially more succinct that their future modality counterparts. Indeed let Σ_n be a system with $F_n = \{p_0, \ldots, p_n\}$, let $\psi_i = p_i \leftrightarrow \blacklozenge(\neg\bullet\mathsf{true} \wedge p_i)$, and let $\Psi = \Box\big(\bigwedge_{i=1}^{n} \psi_i \rightarrow \psi_0\big)$. Intuitively, ψ_i expresses that "p_i has the same truth value now as it did in the initial state".

Theorem 1 (Following Markey 2003) *Any future formula ψ, that is equivalent to Ψ (defined as above) has size $\Omega(2^{|\Psi|})$.*

In the following sections we provide a translation of formulae with past modalities into future-only formulae, in order to use existing planning technology. Despite Markey's theorem, it is possible to show that the blowup for Ψ can be avoided if one modifies the transition system to include additional predicates that keep track of the initial truth value of each of $p_0, \ldots, p_n$. Such a modification can be done in linear time.

Characterizing Explanations

Given a description of the behavior of a dynamical system and a set of observations about the state of the system and/or action occurrences, we define an explanation to be a pairing of actions, orderings, and possibly state that account for the observations in the context of the system dynamics. The definitions in this section follow (but differ slightly from) the definitions of dynamical diagnosis we proposed in (Sohrabi, Baier, and McIlraith 2010), which in turn elaborate and extend previous work (e.g., McIlraith 1998; Iwan 2001).

Assuming our system behavior is defined as a dynamical system and that the observations are expressed in future f-PLTL, we define an explanation as a tuple (H, α) where H is a set of clauses representing an assumption about the initial state and α is an executable sequence of actions that makes the observations satisfiable. If the initial state is complete, then H is empty, by definition. In cases where we have incomplete information about the initial state, H denotes assumptions that we make, either because we need to establish the preconditions of actions we want to conjecture in our explanation or because we want to avoid conjecturing further actions to establish necessary conditions. Whether it is better to conjecture more actions or to make an assumption is dictated by domain-specific knowledge, which we will encode in preferences.

Definition 2 (Explanation) *Given a dynamical system $\Sigma = (F, A, I)$, and an observation formula φ, expressed in future f-*PLTL*, an explanation is a tuple (H, α), where H is a set of clauses over F such that $I \cup H$ is satisfiable, $I \not\models H$, and α is a sequence of actions in A such that α satisfies φ in the system $\Sigma_A = (F, A, I \cup H)$.*

Example Assume a standard logistics domain with one truck, one package, and in which all that is known initially is that the truck is at loc_1. We observe pkg is unloaded from $truck_1$ in loc_1, and later it is observed that pkg is in loc_2. One can express the observation as

$$\Diamond[occ(unload(pkg, loc_1)) \wedge \bigcirc\Diamond at(pkg, loc_2)]$$

A possible explanation (H, α), is such that $H = \{in(pkg, truck_1)\}$, and α is $unload(pkg, loc_1)$, $load(pkg, loc_1)$, $drive(loc_1, loc_2)$, $unload(pkg, loc_2)$.

Note that aspects of H and α can be further filtered to identify elements of interest to a particular user following techniques such as those in (McGuinness et al. 2007).

Given a system and an observation, there are many possible explanations, not all of high quality. At a theoretical level, one can assume a reflexive and transitive preference relation $\preceq$ between explanations. If E_1 and E_2 are explanations and $E_1 \preceq E_2$ we say that E_1 is *at least as preferred as* E_2. $E_1 \prec E_2$ is an abbreviation for $E_1 \preceq E_2$ and $E_2 \not\preceq E_1$.

Definition 3 (Optimal Explanation) *Given a system Σ, E is an optimal explanation for observation φ iff E is an explanation for φ and there does not exist another explanation E' for φ such that $E' \prec E$.*

Complexity and Relationship to Planning

It is possible to establish a relationship between explanation generation and planning. Before doing so, we give a formal definition of planning.

A *planning problem with temporally extended goals* is a tuple $P = (\Sigma, G)$, where Σ is a dynamical system, and G is a future f-PLTL goal formula. The sequence of actions α is a *plan* for P if α is executable in Σ and α satisfies G in Σ. A planning problem (Σ, G) is *classical* if Σ has a complete initial state, and *conformant* otherwise.

The following is straightforward from the definition.

Proposition 1 *Given a dynamical system $\Sigma = (F, A, I)$ and an observation formula φ, expressed in future f-*PLTL*, then (H, α) is an explanation iff α is a plan for conformant planning problem $P = ((F, A, I \cup H), \varphi)$ where $I \cup H$ is satisfiable and where φ is a temporally extended goal.*

In systems with complete initial states, the generation of a single explanation corresponds to classical planning with temporally extended goals.

Proposition 2 *Given a dynamical system Σ such that Σ has complete initial state, and an observation formula φ, expressed in future f-*PLTL*, then $(\emptyset, \alpha)$ is an explanation iff α is a plan for classical planning problem $P = (\Sigma, \varphi)$ with temporally extended goal φ.*

Indeed, the complexity of explanation existence is the same as that of classical planning.

Theorem 2 *Given a system Σ and a temporally extended formula φ, expressed in future f-*PLTL*, explanation existence is PSPACE-complete.*

Proof sketch. For membership, we propose the following NPSPACE algorithm: guess an explanation H such that $I \cup H$ has a unique model, then call a PSPACE algorithm (like the one suggested by de Giacomo and Vardi (1999)) to decide (classical) plan existence. Then we use the fact that NPSPACE=PSPACE. Hardness is given by Proposition 2 and the fact that classical planning is PSPACE-hard (Bylander 1994). §

The proof of Theorem 2 appeals to a non-deterministic algorithm that provides no practical insight into how to translate plan generation into explanation generation. At a more practical level, there exists a deterministic algorithm that maps explanation generation to classical plan generation.

Theorem 3 *Given an observation formula φ, expressed in future f-*PLTL*, and a system Σ, there is an exponential-time procedure to construct a classical planning problem $P = (\Sigma', \varphi)$ with temporally extended goal φ, such that if α is a plan for P, then an explanation (H, α') can be generated in linear time from α.*

Proof sketch. Σ', the dynamical system that describes P is the same as $\Sigma = (F, A, I)$, augmented with additional actions that "complete" the initial state. Essentially, each such action generates a successor state s that is consistent with I. There is an exponential number of them. If $a_0 a_1 \dots a_n$ is a plan for P, we construct the explanation (H, α') as follows. H is constructed with the facts true in the state s that a_0 generates. α' is set to $a_1 \dots a_n$. §

All the previous results can be re-stated in a rather straightforward way if the desired problem is to find an optimal explanation. In that case the reductions are made to preference-based planning (Baier and McIlraith 2008).

The proofs of the theorems above unfortunately do not provide a *practical* solution to the problem of (high-quality) explanation generation. In particular, we have assumed that planning problems contain temporally extended goals expresed in future f-PLTL. No state-of-the-art planner that we are aware of supports these goals directly. We have not provided a compact and useful way to represent the $\preceq$ relation.

Specifying Preferred Explanations

The specification of preferred explanations in dynamical settings presents a number of unique representational requirements. One such requirement is that preferences over explanations be **contextualized with respect to observations**, and these observations themselves are not necessarily single fluents, but rich temporally extended properties – sometimes with characteristic forms and patterns. Another unique representational requirement is that the generation of explanations (and preferred explanations) necessitates **reflecting on the past**. Given some observations over a period of time, we wish to conjecture what preceded these observations in order to account for their occurrence. Such explanations may include certain system state that explains the observations, or it may include action occurrences. Explanations may also include reasonable facts that we wish to posit about the initial state (e.g., that it's below freezing outside – a common precursor to a car battery being dead).

In response to the somewhat unique representational requirements, we express preferences in f-PLTL. In order to generate explanations using state-of-the-art planners, an objective of our work was to make the preference input language PDDL3 compatible. However, f-PLTL is more expressive than the subset of LTL employed in PDDL3, and we did not wish to lose this expressive power. In the next section we show how to compile away some or all temporal modalities by exploiting the correspondence between past and future modalities and by exploiting the correspondence between LTL and Büchi automata. In so doing we preserve the expressiveness of f-PLTL within the syntax of PDDL3.

Preferred Explanations

A high quality explanation is determined by the optimization of an objective function. The PDDL3 metric function we employ for this purpose is a weighted linear sum of formulae to be minimized. I.e., (minimize (+ ($*$ w_1 ϕ_1) ...($*$ w_k ϕ_k))) where each ϕ_i is a formula that evaluates to 0 or 1 depending on whether an associated preference formula, a property of the explanation trajectory, is satisfied or violated; w_i is a weight characterizing the importance of that property (Gerevini et al. 2009). The key role of our preferences is to convey domain-specific knowledge regarding the most preferred explanations for particular observations. Such preferences take on the following canonical form.

Definition 4 (Explanation Preferences) $\Box(\phi_{obs} \rightarrow \phi_{expl})$ *is an explanation preference formula where ϕ_{obs}, the observation formula, is any future f-*PLTL *formula, and ϕ_{expl}, the explanation formula, is any past f-*PLTL *formula. Non-temporal expressions may appear in either formula.*

An observation formula, ϕ_{obs}, can be as simple as the observation of a single fluent or action occurrence (e.g., *my car won't start.*), but it can also be a complex future f-PLTL. In many explanation scenarios, observations describe a telltale ordering of system properties or events that suggest a unique explanation such as a car that won't start every time it rains. To simplify the description of observation formulae, we employ precedes as a syntactic constructor of observation patterns. φ_1 precedes φ_2 indicates that φ_1 is observed before φ_2. More generally, one can express ordering among observations by using formula of the form (φ_1 precedes φ_2 ... precedes φ_n) with the following interpretation:

$$\varphi_1 \wedge \bigcirc\Diamond(\varphi_2 \wedge \bigcirc\Diamond(\varphi_3 ...(\varphi_{n-1} \wedge \bigcirc\Diamond\varphi_n)...)) \quad (1)$$

Equations (2) and (3) illustrate the use of precedes to encode a total (respectively, partial) ordering among observations. These are two common forms of observation formulae.

$$(\mathsf{obs}_1 \text{ precedes } \mathsf{obs}_2 \text{ precedes } \mathsf{obs}_3 \text{ precedes } \mathsf{obs}_4) \quad (2)$$

$$(\mathsf{obs}_3 \text{ precedes } \mathsf{obs}_4) \wedge (\mathsf{obs}_1 \text{ precedes } \mathsf{obs}_2) \quad (3)$$

Further characteristic observation patterns can also be easily described using precedes. The following is an example of an intermittent fault.

(alarm precedes no_alarm precedes alarm precedes no_alarm)

Similarly, explanation formulae, ϕ_{exp}, can be complex temporally extended formulae over action occurrences and fluents. However, in practice these explanations may be reasonably simple assertions of properties or events that held (resp. occurred) in the past. The following are *some* canonical forms of explanation formulae: $(\blacklozenge e_1 \wedge ... \wedge \blacklozenge e_n)$, and $(\blacklozenge e_1 \otimes ... \otimes \blacklozenge e_n)$, where $n \geq 1$, and e_i is either a fluent $\in F$ or $occ(a)$, $a \in A$ and $\otimes$ is exclusive or.

Computing Preferred Explanations

In previous sections we addressed issues related to the specification and formal characterization of preferred explanations. In this section we examine how to effectively *generate* explanations using state-of-the-art planning technology.

Propositions 1 and 2 establish that we can generate explanations by treating an observation formula φ as the temporally extended goal of a conformant (resp. classical) planning problem. Preferred explanations can be similarly computed using preference-based planning techniques. To employ state-of-the-art planners, we must represent our observation formulae and the explanation preferences in syntactic forms that are compatible with some version of PDDL. Both types of formulae are expressed in f-PLTL so PDDL3 is a natural choice since it supports preferences and some LTL constructs. However, f-PLTL is more expressive than PDDL3, supporting arbitrarily nested past and future temporal modalities, action occurrences, and most importantly the *next* modality, $\bigcirc$, which is essential to the encoding of an ordered set of properties or action occurrences that occur over time. As a consequence, partial- and total-order observations are *not* expressible in PDDL3's subset of LTL, and so it follows that the precedes constructor commonly used in the ϕ_{obs} component of explanation preferences is not expressible in the PDDL3 LTL subset. There are similarly many typical ϕ_{expl} formulae that cannot be expressed directly in PDDL3 because of the necessity to nest temporal modalities. So to generate explanations using planners, we must devise other ways to encode our observation formulae and our explanation preferences.

Approach 1: PDDL3 via Compilation

Although it is not possible to express our preferences directly in PDDL3, it is possible to compile unsupported temporal formulae into other formulae that *are* expressible in PDDL3. To translate to PDDL3, we utilize Baier and McIlraith's future LTL compilation approach (2006), which we henceforth refer to as the BM compilation. Given a future LTL goal formula ϕ, and a planning problem P, the BM compilation executes the following two steps: (*phase 1*) generates a finite state automaton for ϕ, and (*phase 2*) encodes the automaton in the planning problem by adding new predicates to describe the changing configuration of the automaton as actions are performed. The result is a new planning problem P' that augments P with a newly introduced *accepting predicate* $accept_\phi$ that becomes true after performing a sequence of actions α in the initial state if and only if α satisfies ϕ in P's dynamical system. Predicate $accept_\phi$ is the (classical) goal in problem P'. Below we introduce an extension of the BM compilation that allows compiling away *past* f-PLTL formulae.

Our compilation takes dynamical system Σ, an observation φ, a set Γ of formulae corresponding to explanation preferences, and produces a PDDL3 planning problem.

Step 1 Takes Σ and φ and generates a classical planning problem P_1 with temporally extended goal φ using the procedure described in the proof for Theorem 3.

Step 2 Compiles away *occ* in P_1, generating P_2. For each occurrence of $occ(a)$ in Γ or φ, it generates an additional fluent $happened_a$ which is made true by a and is deleted by all other actions. Replace $occ(a)$ by $happened_a$ in Γ and φ.

Step 3 Compiles away all the *past* elements of preferences in Γ. It uses the BM compilation over P_2 to compile away past temporal operators in preference formulae of the form $\square(\phi_{obs} \rightarrow \phi_{expl})$, generating P_3. For every past explanation formula ϕ_{expl} in Γ we do the following. We compute the reverse of ϕ_{expl}, ϕ^r_{expl}, as a formula just like ϕ_{expl} but with all past temporal operators changed to their future counterparts (i.e., $\bullet$ by $\bigcirc$, $\blacklozenge$ by $\lozenge$, S by U). Note that ϕ_{expl} is satisfied in a trajectory of states σ iff ϕ^r_{expl} is satisfied in the reverse of σ. Then, we use phase 1 of the BM compilation to build a finite state automaton $A_{\phi^r_{expl}}$ for ϕ^r_{expl}. We now compute the reverse of $A_{\phi^r_{expl}}$ by switching accepting and initial states and reversing the direction of all transitions. Then we continue with phase 2 of the BM compilation, generating a new planning problem for the reverse of $A_{\phi^r_{expl}}$. In the resulting problem the new predicate $accept_{\phi_{expl}}$ becomes true as soon as the past formula ϕ_{expl} is made true by the execution of an action sequence. We replace any occurrence of ϕ_{expl} in Γ by $accept_{\phi_{expl}}$. We similarly use the BM compilation to remove future temporal modalities from φ and ϕ_{obs}. This is only necessary if they contain nested modalities or $\bigcirc$, which they often will. The output of this step is PDDL3 compliant. To generate PDDL3 output without *any* temporal operators, we perform the following further step.

Step 4 (optional) Compiles away temporal operators in Γ and φ using the BM compilation, ending with simple preferences that refer only to the final state.

Theorem 4 *Let P_3 be defined as above for a description Σ, an observation φ, and a set of preferences Γ. If α is a plan for P_3 with an associated metric function value M, then we can construct an explanation (H, α) for Σ and φ with associated metric value M in linear time.*

Although Step 4 is not required, it has practical value. Indeed, it enables potential application of other compilation approaches that work directly with PDDL3 without temporal operators. For example, it enables the use of Keyder and Geffner's compilation (2009) to compile preferences into corresponding actions costs so that standard cost-based planners can be used to find explanations. This is of practical importance since cost-based planners are (currently) more mature than PDDL3 preference-based planners.

Approach 2: Pre-processing (Sometimes)

The compiled planning problem resulting from the application of Approach 1 can be employed with a diversity of planners to generate explanations. Unfortunately, the preferences may not be in a form that can be effectively exploited by delete relaxation based heuristic search. Consider the preference formula $\gamma = \square(\phi_{obs} \rightarrow \phi_{expl})$. Step 4 culminates in an automaton with accepting predicate $accept_\gamma$. Unfortunately, $accept_\gamma$ is generally true at the outset of plan construction because ϕ_{obs} is false – the observations have not yet occurred in the plan – making $\phi_{obs} \rightarrow \phi_{expl}$, and thus γ, trivially true. This deactivates the heuristic search to achieve $accept_\gamma$ and thus the satisfaction of this preference does not benefit from heuristic guidance. For a restricted but compelling class of preferences, namely those of the form $\square(\phi_{obs} \rightarrow \bigwedge_i \blacklozenge e_i)$ with e_i a non-temporal formula, we can pre-process our preference formula in advance of applying Approach 1, by exploiting the fact that we *know* a priori what observations have occured. Our pre-processing utilizes the following LTL identity:

$$\Box(\phi_{obs} \rightarrow \bigwedge_i \blacklozenge e_i) \wedge \Diamond\phi_{obs} \equiv \bigwedge_i \neg\phi_{obs}\mathsf{U}(e_i \wedge \Diamond\phi_{obs}).$$

Given a preference in the form $\Box(\phi_{obs} \rightarrow \bigwedge_i \blacklozenge e_i)$ we determine whether ϕ_{obs} is entailed by the observation φ (this can be done efficiently given the form of our observations). If this is the case, we use the identity above to transform our preferences, followed by application of Approach 1. The accepting predicate of the resulting automaton becomes true if ϕ_{expl} is satisfied prior to ϕ_{obs}. In the section to follow, we see that exploiting this pre-processing can improve planner performance significantly.

Experimental Evaluation

The objective of our experimental analysis was to gain some insight into the behavior of our proposed preference formalism, specifically, we wanted to: 1) develop a set of somewhat diverse benchmarks and illustrate the use of planners in the generation of explanations; 2) examine how planners perform when the number of preferences is increased; and 3) investigate the computational time gain resulting from Approach 2. We implemented all compilation techniques discussed in Section 5 to produce PDDL3 planning problems with simple preferences that are equivalent to the original explanation generation problems.

We used four domains in our experiments: a computer domain (see Grastien et al. 2007), a car domain (see McIlraith and Scherl 2000), a power domain (see McIlraith 1998), and the trucks domain from IPC 2006. We modified these domains to account for how observations and explanations occur within the domain. In addition, we created two instances of the same problem, one with total-order observations and another with partial-order observations. Since the observations we considered were either total- or partial-order, we were able to compile them away using a technique that essentially makes an observation possible only after all preceding observations have been observed (Haslum and Grastien 2009; 2011). Finally, we increased problem difficulty by increasing the number of observations in each problem.

To further address our first objective, we compared the performance of FF (Hoffmann and Nebel 2001), LAMA (Richter, Helmert, and Westphal 2008), SGPlan$_6$ (Hsu and Wah 2008) and HPLAN-P (Baier, Bacchus, and McIlraith 2009) on our compiled problems but with no preferences. The results show that in the total-order cases, all planners except HPLAN-P solved all problems within seconds, while HPLAN-P took much longer, and could not solve all problems (i.e., it exceeded the 600 second time limit). The same results were obtained with the partial-order problems, except that LAMA took a bit longer but still was far faster than HPLAN-P. This suggests that our domains are reasonably challenging.

To address our second objective we turned to preference-based planner HPLAN-P. We created different versions of the same problem by increasing the number of preferences they used. In particular, for each problem we tested with 10, 20, and 30 preferences. To measure the change in computation time between problems with different numbers of preferences, we calculated the percentage difference between

	Total-Order				Partial-Order			
	HPLAN-P		LAMA		HPLAN-P		LAMA	
	Appr 1	Appr 2	Appr 1	Appr 2	Appr 1	Appr 2	Appr 1	Appr 2
computer-1	1.05	0.78	5.29	0.25	2.26	0.58	5.93	0.57
computer-2	5.01	4.88	0.19	0.41	1.49	1.42	0.50	0.42
computer-3	23.44	22.85	0.75	0.75	15.92	15.57	1.02	1.94
computer-4	55.69	51.98	6.94	4.58	15.97	13.93	3.64	4.33
computer-5	128.50	125.98	2.05	3.20	57.28	56.19	3.42	6.12
computer-6	83.17	82.78	2.64	4.63	43.92	43.86	16.99	16.27
computer-7	505.73	484.68	4.23	5.85	188.45	181.44	89.03	68.47
computer-8	236.03	205.81	3.75	6.13	159.92	152.49	29.35	28.91
car-1	1.60	1.53	0.66	0.08	0.60	0.53	2.11	0.07
car-2	8.96	8.31	10.72	0.20	3.04	2.59	15.14	0.25
car-3	563.60	40.17	13.98	0.59	593.10	15.06	16.51	0.62
car-4	NF	103.80	24.00	1.41	NF	38.48	33.79	0.95
car-5	NF	245.69	35.93	1.56	NF	103.18	NF	1.23
car-6	NF	522.50	117.45	2.44	NF	176.11	NF	1.56
car-7	NF	NF	62.00	3.47	NF	170.54	NF	2.02
car-8	NF	NF	108.07	4.46	NF	257.10	NF	2.94
power-1	0.02	0.01	0.02	0.02	0.82	0.53	0.02	0.02
power-2	0.18	0.18	0.13	0.06	0.14	0.13	0.40	0.06
power-3	0.47	0.50	0.13	0.13	0.31	0.33	3.50	0.11
power-4	1.62	1.52	0.63	0.58	75.37	69.85	14.92	18.37
power-5	26.98	24.60	5.97	0.97	NF	NF	46.43	0.64
power-6	51.65	51.48	11.26	6.84	NF	NF	NF	NF
power-7	177.58	177.42	15.09	9.42	NF	NF	NF	NF
power-8	565.77	564.71	30.67	16.38	NF	NF	NF	NF
truck-1	1.90	1.62	0.13	0.29	3.08	1.98	0.24	0.24
truck-2	5.25	5.10	0.85	0.74	3.12	3.07	0.32	0.49
truck-3	108.83	92.57	0.38	1.07	36.92	27.57	0.59	1.15
truck-4	323.18	323.06	2.03	2.48	402.96	219.73	2.15	1.87
truck-5	177.68	174.22	3.31	4.93	NF	NF	2.71	3.90
truck-6	NF	NF	2.69	1.88	NF	NF	8.53	6.14
truck-7	NF	NF	10.23	11.92	NF	NF	8.42	8.41
truck-8	NF	NF	11.60	8.76	NF	NF	8.19	11.81

Figure 1: Runtime comparison between HPLAN-P and LAMA on problems of known optimal explanation. NF means the optimal explanation was not found within the time limit of 600 seconds.

the computation time for the problem with the larger and with the smaller number of preferences, all relative to the computation time of the larger numbered problem. The average percentage difference was 6.5% as we increased the number of preferences from 10 to 20, and was 3.5% as we went from 20 to 30 preferences. The results suggest that as we increase the number of preferences, the time it takes to find a solution does increase but this increase is not significant.

As noted previously the Approach 1 compilation technique (including Step 4) results in the complete removal of temporal modalities and therefore enables the use of the Keyder and Geffner's compilation technique (2009). This technique supports the computation of preference-based plans (and now preferred explanations) using cost-based planners. However, the output generated by our compilation requires a planner compatible with ADL or derived predicates. Among the rather few that support any of these, we chose to experiment with LAMA since it is currently the best-known cost-based satisficing planner. Figure 1 shows the time it takes to find the optimal explanation using HPLAN-P and LAMA as well as the time comparison between our "Approach 1" and "Approach 2" encodings (Section 5). To measure the gain in computation time from the "Approach 2" technique, we computed the percentage difference between the two, relative to "Approach 1". (We assigned a time of 600 to those marked NF.) The results show that on average we gained 22.9% improvement for HPLAN-P and 29.8 % improvement for LAMA in the time it takes to find the optimal solution. In addition, we calculated the time ratio ("Approach 1"/ "Approach 2"). The results show that on average HPLAN-P found plans 2.79 times faster and LAMA found plans 31.62 times faster when using "Approach 2". However, note that "Approach 2" does not always improve the performance. There are a few cases where the planners take longer when using "Approach 2". While the definite cause of this decrease in performance is currently unknown, we believe this decrease may depend on the structure of the problem and/or on the difference in the size of the translated domains. On average the translated problems used in "Approach 2" are 1.4 times larger, hence this increase in the size of the problem may be one reason behind the decrease in performance. Nevertheless, this result shows that "Approach 2" can significantly improve the time required to find the optimal explanation, sometimes by orders of magnitude, in so doing it allows us to solve more problem instances than with "Approach 1" alone (see car-4 and car-5).

Summary

In this paper, we examined the task of generating preferred explanations. To this end, we presented a logical characterization of the notion of a (preferred) explanation and established its correspondence to planning, including the complexity of explanation generation. We proposed a finite variant of LTL, f-PLTL, that includes past modalities and action occurrences and utilized it to express observations and preferences over explanation. To generate explanations using state-of-the-art planners, we proposed and implemented a compilation technique that preserves f-PLTL semantics while conforming to PDDL3 syntax. This enables computation of preferred explanations with PDDL3-compliant preference-based planners as well as with cost-based planners. Exploiting the property that observations are known a priori we transformed explanation preferences into a form that was amenable to heuristic search. In so doing, we were able to reduce the time required for explanation generation by orders of magnitude, sometimes. A number of practical problems can be characterized as explanation generation problem. The work presented in this paper opens the door to a suite of promising techniques for explanation generation.

Acknowledgements: We thank Alban Grastien and Patrik Haslum for providing us with an encoding of the computer problem, which we modified and used in this paper for benchmarking. We also gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC). Jorge Baier was funded by the VRI-38-2010 grant from Universidad Católica de Chile.

References

Baier, J., and McIlraith, S. 2006. Planning with first-order temporally extended goals using heuristic search. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*, 788–795.

Baier, J., and McIlraith, S. 2008. Planning with preferences. *AI Magazine* 29(4):25–36.

Baier, J.; Bacchus, F.; and McIlraith, S. 2009. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence* 173(5-6):593–618.

Brewka, G. 1994. Adding priorities and specificity to default logic. In *Proc. of the Logics in Artificial Intelligence, European Workshop (JELIA)*, 247–260.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

de Giacomo, G., and Vardi, M. Y. 1999. Automata-theoretic approach to planning for temporally extended goals. In Biundo, S., and Fox, M., eds., *ECP*, volume 1809 of *LNCS*, 226–238. Durham, UK: Springer.

Gabbay, D. M. 1987. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Temporal Logic in Specification*, 409–448.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the 5th int'l planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5-6):619–668.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up with good excuses: What to do when no plan can be found. In *Proc. of the 20th Int'l Conference on Automated Planning and Scheduling (ICAPS)*, 81–88.

Grastien, A.; Anbulagan; Rintanen, J.; and Kelareva, E. 2007. Diagnosis of discrete-event systems using satisfiability algorithms. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI)*, 305–310.

Haslum, P., and Grastien, A. 2009. Personal communication.

Haslum, P., and Grastien, A. 2011. Diagnosis as planning: Two case studies. In *Proc. of the Int'l Scheduling and Planning Applications workshop (SPARK)*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hsu, C.-W., and Wah, B. 2008. The SGPlan planning system in IPC-6. In *6th Int'l Planning Competition Booklet (IPC-2008)*.

Iwan, G. 2001. History-based diagnosis templates in the framework of the situation calculus. In *Proc. of the Joint German/Austrian Conference on Artificial Intelligence (KR/ÖGAI)*. 244–259.

Keyder, E., and Geffner, H. 2009. Soft Goals Can Be Compiled Away. *Journal of Artificial Intelligence Research* 36:547–556.

Markey, N. 2003. Temporal logic with past is exponentially more succinct, concurrency column. *Bulletin of the EATCS* 79:122–128.

McGuinness, D. L.; Glass, A.; Wolverton, M.; and da Silva, P. P. 2007. Explaining task processing in cognitive assistants that learn. In *Proc. of the 20th Int'l Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 284–289.

McIlraith, S., and Scherl, R. B. 2000. What sensing tells us: Towards a formal theory of testing for dynamical systems. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI)*, 483–490.

McIlraith, S. 1998. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. of the 6th Int'l Conference of Knowledge Representation and Reasoning (KR)*, 167–179.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proc. of the 21st Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1778–1783.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. of the 23rd National Conference on Artificial Intelligence (AAAI)*, 975–982.

Sohrabi, S.; Baier, J.; and McIlraith, S. 2010. Diagnosis as planning revisited. In *Proc. of the 12th Int'l Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 26–36.