Making Hybrid Plans More Clear to Human Users — A Formal Approach for Generating Sound Explanations

Bastian Seegebarth and **Felix Müller** and **Bernd Schattenberg** and **Susanne Biundo**

Institute of Artificial Intelligence,
Ulm University, D-89069 Ulm, Germany,
email: forename.surname@uni-ulm.de

Abstract

Human users who execute an automatically generated plan want to understand the rationale behind it. Knowledge-rich plans are particularly suitable for this purpose, because they provide the means to give reason for causal, temporal, and hierarchical relationships between actions. Based on this information, focused arguments can be generated that constitute explanations on an appropriate level of abstraction. In this paper, we present a formal approach to plan explanation. Information about plans is represented as first-order logic formulae and explanations are constructed as proofs in the resulting axiomatic system. With that, plan explanations are provably correct w.r.t. the planning system that produced the plan. A prototype plan explanation system implements our approach and first experiments give evidence that finding plan explanations is feasible in real-time.

Introduction

In many real-world application areas, AI planning techniques are deployed to support human users in their decision making. Prominent such areas are emergency planning, where disaster missions such as fire brigade operations and evacuation measures are planned (de la Asunción et al. 2005; Muñoz-Avila et al. 1999), and the system-supported assistance of (cognitively impaired and elderly) people in their daily activities (Cesta et al. 2011; Pollack 2002; Biundo et al. 2011). In these *user-centered planning* settings, plans are made for human users, i.e. the systems provide recommendations when and how to act to subjects who are competent themselves and might scrutinize the systems' suggestions. Therefore, it is essential that a system is able to make its decisions transparent, to give good reasons for them, and to present these reasons in a comprehensible manner. Otherwise, mistrust in the system's competence might be provoked, and this might even lead to its disregard, in particular if the situation is critical. However, the issue of explaining the results produced by planning systems has only rarely been addressed so far, as was also recently discussed (Smith 2010).

In this paper, we introduce a formal approach to plan explanation. It relies on so-called *knowledge-rich plans* as they are generated by hybrid planning systems – systems that combine the paradigms of both partial-order-causal-link planning and hierarchical planning. The plans comprise a set of actions to be executed, a partial order on them, and so-called *causal links*, which explicitly state the causal relationships that exist between the actions. In addition, information about hierarchical dependencies between actions is available from the underlying planning process. In order to explain certain properties of a plan, like the existence of a particular action in this plan or the particular ordering between some of the actions, our plan explanation approach uses a first-order logic axiomatic system that formalizes basic arguments about a plan and the logical consequences that can be derived from them. The axiomatic system is generated from the plan to be explained and from information about the planning process that led to it. In this framework, explanation consists of two steps. First, a proof, which we call a raw explanation, of the plan properties in question is constructed. With that, plan explanations are provably correct w.r.t. to the underlying planner. This means, if the planner is sound, the explanation system always produces valid explanations. Hence, a user can always feel certain about the explanations she is given, which in turn helps to establish her trust in the planning system. Second, an actual explanation is constructed by transforming the raw explanation to text, speech, or a graphical representation. In this paper we address the first of these steps.

We have implemented a prototype system for the generation of raw explanations based on our framework. First results show that they can be generated within a few seconds. This gives evidence that it is feasible to find plan explanations in real-time, a fact that is of particular importance in user-centered planning.

The general question of how automated systems should present their results has already been addressed in the past (Moore and Swartout 1989). Here, the presentation of results is seen as an interactive process, where the user may ask questions, receive answers, and pose follow-up questions. In automated reasoning, one way to create explanations for the results a system produces is to extract them from the steps performed to solve the reasoning task. This method has been used to generate explanations for subsumption in description logics (McGuinness and Borgida 1995) and for proofs found by an automated theorem prover (Horacek 2007). In the context of planning, an approach for

explaining the causes of discrepancies between the expected and the observed world state during the execution of a plan is presented by Molineaux, Kuter, and Klenk (2011). Here, an explanation is a partially ordered set of actions, observations, and unobservable exogenous events that would generate the observed world state. In domains with uncertain action effects that can be represented as Markov Decision Processes (Puterman 1994), the reasons for executing an action can be explained based on the frequency with which they help reaching a state with a high reward (Khan, Poupart, and Black 2009). The challenge of explaining why a planning system failed to produce a plan for a given planning task is discussed by Göbelbecker et al. (2010): the problem is solved by making targeted modifications to the task and solving the new task using optimal planning. Another related line of work concerns the ICEE explanation system, which adds task explanation functionality to the personal software assistant CALO (McGuinness et al. 2007; 2005). The ICEE system allows a user among other things to ask the assistant why it is executing certain tasks and why tasks are executed in a certain order.

The rest of the paper is organized as follows. After a brief introduction to hybrid planning we present our formal framework for plan explanation. It includes the explanation of plan steps and ordering constraints, based on causal and decomposition relationships. We then present our prototype plan explanation system and some first experimental results. The paper concludes with a short discussion of prospective issues.

Hybrid Planning

The hybrid planning framework is a combination of POCL and HTN planning (Estlin, Chien, and Wang 1997; Kambhampati, Mali, and Srivastava 1998; Biundo and Schattenberg 2001; Gerevini et al. 2008).

A POCL planning problem (McAllester and Rosenblitt 1991) is a triple $\Pi = (\mathcal{O}, s_{init}, g)$. Operators are supplied in the set $\mathcal{O}$. Each operator is a triple $(o(\overline{v}), pre, \mathit{eff})$, where o is an operator symbol, $\overline{v} = v_1, \ldots, v_n$ a list of variables, and pre and eff the preconditions and effects of the operator given as conjunctions of literals over the variables $\overline{v}$. The initial state s_{init} and the goal description g are given as conjunctions of ground literals.

A partial plan is a 4-tuple $(PS, \prec, V, C)$. PS is a set of plan steps, where each plan step $s{:}o$ consists of a unique label s and a (partially) instantiated operator o. We will usually refer to a plan step by its label. The relation $\prec$ defines a partial order on the plan steps in PS. The set V contains constraints on variables, appearing in partially instantiated plan steps in the plan. C is a set of causal links; a causal link $(s \rightarrow_p s')$ indicates that the literal p from the preconditions of the operator o' from the plan step s' is established by the effects of the operator o from the plan step s. We also call s the producer and s' the consumer of the causal link.

Given a POCL planning problem Π, a solution is a partial plan $P = (PS, \prec, V, C)$ satisfying the following criteria:

- PS contains a plan step $init$ that has the initial state s_{init} as effects and a plan step $goal$ that has the goal state description g as preconditions,
- for every precondition p of a plan step s' there is a causal link $(s \rightarrow_p s')$ such that s has p as an effect and s is ordered before s' by the ordering constraints,
- no causal links are threatened, i.e., for each causal link $(s \rightarrow_p s') \in C$ the ordering constraints ensure that no plan step s'' with a literal $\neg p$ in the effects can be ordered between s and s'.

These solution criteria ensure that every linearization of a solution to a POCL planning problem must be executable in the initial state and generates a state satisfying the goal description.

HTN planning (Yang 1990; Erol, Hendler, and Nau 1994) is based on the concepts of primitive and abstract tasks. Primitive tasks correspond to operators from POCL planning. Abstract tasks represent high-level activities, such as composing a message on a smart phone. Tasks are arranged in task networks, which contain a set of tasks and a partial order on them. For each abstract task the domain provides a number of task networks as predefined solutions, called methods. To solve an abstract task it is replaced by the task network from one of its corresponding methods. This process is called task decomposition. A planning problem is then specified as an initial task network. It is solved by repeatedly decomposing abstract tasks from the task network until it contains only primitive tasks and is consistent w. r. t. their ordering and causal structure.

Hybrid planning combines POCL and HTN planning as follows. Firstly, we adopt the concepts of primitive and abstract tasks and the decomposition of abstract tasks via methods from HTN planning. These tasks may, however, be inserted into plans at any point without being introduced via decomposition. Secondly, abstract tasks now carry preconditions and effects as in POCL planning. These may be expressed on an abstract level by using abstract state features provided by the hybrid planning problem. Lastly, in addition to the initial task network from HTN planning, we also allow problems to contain a goal description as in POCL planning.

In hybrid planning, a partial plan is a 4-tuple $P = (PS, \prec, V, C)$. The difference to partial plans from POCL planning is that plan steps now contain (partially) instantiated tasks instead of operators. We will call a plan step primitive or abstract if its task is primitive or abstract.

A hybrid planning problem is given as a 6-tuple $\Pi = (\mathcal{T}, \mathcal{M}, \Delta, s_{init}, P_{init}, g)$. The set of available task schemata is given by $\mathcal{T}$. Task schemata are defined analogously to operators as a triple $(t(\overline{v}), pre, \mathit{eff})$ with the same meaning. Methods for abstract tasks are contained in $\mathcal{M}$. A method $m = (t, P)$ provides a predefined solution for an abstract task t, given as a partial plan P. The set Δ contains decomposition axioms. A decomposition axiom is a formula of the form $\forall \overline{x}.[\phi(\overline{x}) \Leftrightarrow \psi_1(\overline{x}) \vee \cdots \vee \psi_n(\overline{x})]$ where $\overline{x} = x_1, \ldots, x_n$ is a list of variables, $\phi(\overline{x})$, called abstraction, is an atomic formula over the variables from $\overline{x}$, and each $\psi_i(\overline{x})$, called formula decomposition, is an existentially quantified conjunction of literals. Abstract literals may only be used in the preconditions and effects of abstract tasks. As in POCL planning, the initial state and goal state

description are given by s_{init} and g. Finally, P_{init} specifies the initial partial plan for the planning problem.

Given a hybrid planning problem Π, a solution is a partial plan $P = (PS, \prec, V, C)$ if it satisfies the above solution criteria for plans in POCL planning and one more criterion:

- P contains no abstract plan steps and can be obtained from P_{init} by repeated decomposition of plan steps and insertion of plan steps, causal links, ordering, and variable constraints.

Formal Framework

In order to clarify, why the plan data structure at hand actually is a solution to the proposed problem, we rely on formal proofs as a means for establishing coherent sequences of arguments. To this end, we developed an axiomatic system of first-order logic formulae, which formalize basic arguments about a plan and their logical consequences. In this setting, constructing an explanation for any element of a plan (the plan steps, their causal structure, etc.) means to find a sequence of "applications" of the axioms that entails the requested aspect and that is supported by the problem specification – the most fundamental, unquestionable raison d'être for any plan. The axioms are thereby derived from the solution criteria above, stating explicitly every possible rationale for generating plans the way they are.

Explanation of Plan Steps

We will begin with explaining why a given plan step s is present in a given plan P, viz., why s is "necessary" for P to constitute a solution. According to the solution criteria for hybrid plans, one of the following may be the reason: First, s is trivially either $init$, $goal$, or any plan step from the problem specification. Second, s establishes a precondition of some other plan step s'. Third, s has been introduced by decomposing an abstract plan step s' during the construction of P. The latter two reasons recursively imply follow-up explanations for s'.

The following sections define an axiomatic system Σ that captures all relevant facts and axioms for deriving explanations as proofs. Its most basic elements are defined by the causal structure of a plan $P = (PS, \prec, V, C)$, i.e., its set of causal links: for every causal link $(s \rightarrow_p s') \in C$, Σ contains an atom $CR(s, p, s')$, representing the *causal relation* between s and s'. This also includes causal relations of the form $CR(s, p, goal)$, denoting the goal description.

We also make use of the decomposition structure behind P, i.e., the substitution of abstract plan steps by their implementations during P's generation out of P_{init}. This information is, however, not represented explicitly in the solution plan and has to be (re-)constructed from the knowledge about the plan generation process: for each decomposition of an abstract plan step s' via a method m, performed during the construction of P from P_{init}, Σ contains an atom $DR(s, m, s')$ for each plan step s introduced by m, representing the *decomposition relation* between s and s'. On a technical note, we treat plan steps in the initial plan as results from decomposing a virtual top-level step top via a special method m_{top}; we therefore add $DR(s, m_{top}, top)$ to Σ for each step s in the initial plan.

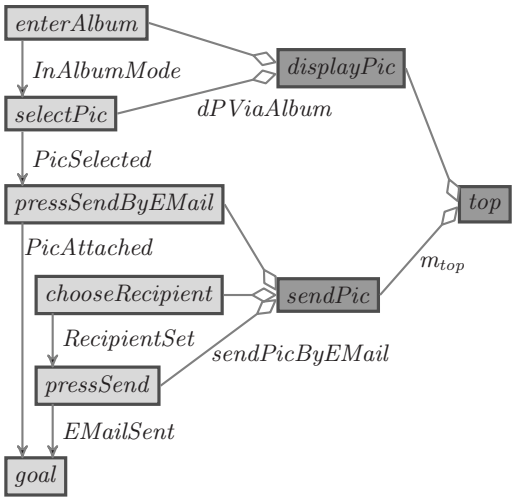

Figure 1: The causal and decomposition structure of an example plan. Primitive plan steps are shown on the left hand side, abstract plan steps on the right. Selected causal relations are depicted as arrows, decomposition relations as arrows with diamond tips.

In order to illustrate the contents of Σ, we introduce an example plan in a simplified version of a domain describing various user assistance task (Biundo et al. 2011). It addresses the task of selecting a picture from the photo album of a smart phone and sending it to some other person, i.e., the goal of the plan is to have the picture transferred to that person in some way. Fig. 1 visualizes its causal and decomposition structure. In this example, Σ contains $CR(enterAlbum, InAlbumMode, selectPic)$, because plan step $enterAlbum$ establishes the precondition $InAlbumMode$ of its successor $selectPic$. Also, in the decomposition of the abstract plan step $displayPic$ the method $dPViaAlbum$ was used and steps $enterAlbum$ and $selectPic$ were added to the plan; Σ therefore contains both axioms $DR(enterAlbum, dPViaAlbum, displayPic)$ and $DR(selectPic, dPViaAlbum, displayPic)$.

Basic Explanations for Plan Steps We use the literal $Nec(s)$ for representing that a step s is necessary for a given plan P to constitute a solution. The reasons for why this is the case are given by axioms of the form $\forall s.[\phi \Rightarrow Nec(s)]$ to Σ, where ϕ is a formula encoding an argument that supports the presence of s in P. Because all of our axioms will have this form, the only inference rule we need to derive proofs from such specifications is *modus ponens*.

Let us begin with axioms that give reason to plan steps based on the plan's causal structure:

(1) $\forall s.[[\exists g.CR(s, g, goal)] \Rightarrow Nec(s)]$

(2) $\forall s.[[\exists s', p.[CR(s, p, s') \wedge Nec(s')]] \Rightarrow Nec(s)]$

Axiom 1 is the base case of causality-related arguments: it encodes that plan step s can be explained if it is found establishing g, one of the goals. Axiom 2 takes into account the

transitive closure of causal arguments; the reason for a plan step s is to establish a precondition of some other plan step s', given that there is a reason for s'. The second condition is required because if it turned out that s' is not necessary for the plan then arguing for s based on the establishment of a precondition for s' would be rendered irrelevant.

The next axioms deal with the decomposition of abstract tasks:

(3) $\forall s.[DR(s, m_{top}, top) \Rightarrow Nec(s)]$

(4) $\forall s.[[\exists s', m.[DR(s, m, s') \wedge Nec(s')]] \Rightarrow Nec(s)]$

Axiom 3 states that the reason for every (possibly abstract) plan step in the initial plan P_{init} is given by the problem specification per se. In a similar fashion to the transitive closure of causality above, Axiom 4 encodes that decomposing a plan step s' gives reason to all steps s introduced in its decomposition, given that there is some reason for s'.

We can now generate an explanation for a given plan step s by proving that the formula $Nec(s)$ is true for a given Σ. A verbalization of the explanation can then be extracted from the proof. The following example shows a proof for $Nec(enterAlbum)$, an explanation for plan step $enterAlbum$ in Fig. 1; referenced axioms and proof steps are annotated.

(5) $Nec(enterAlbum)$ [2,6,7]

(6) $CR(enterAlbum, InAlbumMode, selectPic)$

(7) $Nec(selectPic)$ [2,8,9]

(8) $CR(selectPic, PicSelected, pressSendByEMail)$

(9) $Nec(pressSendByEMail)$ [4,10,11]

(10) $DR(pressSendByEMail, sendPicByEMail, sendPic)$

(11) $Nec(sendPic)$ [3,12]

(12) $DR(sendPic, m_{top}, top)$

Steps 6, 8, 10, and 12 are elements of the basic causal and decomposition structure; they are given explicitly in Fig. 1. In step 5, we use the atoms 6 and 7 and instantiate Axiom 2 to $[CR(enterAlbum, InAlbumMode, selectPic) \wedge Nec(selectPic)] \Rightarrow Nec(enterAlbum)$. From that we derive $Nec(enterAlbum)$ by applying modus ponens. Analogously, we construct the reasons for $selectPic$ and $pressSendByEMail$ via causal and decomposition relations, respectively. Finally, steps 11 and 12 finish the proof by stating that the reason for $sendPic$ is given by the initial plan, i.e., by using Axiom 3.

Causal Relations for Abstract Plan Steps A solution to a planning problem does not contain abstract plan steps and its causal structure is hence given exclusively in terms of primitive actions. Any explanations for an abstract plan step would therefore consist of pure decomposition-based reasons. This would be inadequate for two reasons.

First, assume that an abstract plan step is identified as the reason for some plan step by decomposition. In hybrid planning, such an abstract plan step can be inserted in a POCL-like fashion for establishing preconditions of some other step. It hence does not occur as first argument in any decomposition relation and cannot be explained by Axioms 3 and 4. As no causal relation for the abstract step is given in the plan either, the current explanation cannot be completed.

Second, the decomposition structure of a plan is typically only a few layers deep and explanations that exclusively rely on decomposition relations would be very short. Although briefness is on the one hand a desired quality of explanations, it also implies the risk of producing overly short and uninformative explanations.

These considerations motivate to support inference of causal relations on any level of abstraction, thereby allowing to flexibly interleave decompositional and causal arguments. To this end, we adopt the notion of the legality of methods and consequently interpret abstract plan steps with their preconditions and effects as specifications for their implementing decompositions (Biundo and Schattenberg 2001). Informally, this means that an abstract plan step s "inherits" the causal relations from the actions in its decomposition. For example, the primitive plan step $pressSendByEMail$ in Fig. 1 has the precondition $PicSelected$ and $pressSend$ has the effect $EMailSent$. Thus, the abstract plan step $sendPic$ is associated with the precondition $PicSelected$ and the effect $EMailSent$. However, $pressSend$ also has the precondition $RecipientSet$. This is not considered a precondition of $sendPic$ because it is established by $chooseRecipient$, which is from within the decomposition of $sendPic$. Analogously, causal relations for abstract plan steps can be derived from the causal relations that are produced or consumed by the steps in their decomposition. We call this process propagation of causal relations and represent it with the following axioms:

(13) $\forall s, p, s'.[[\exists m, s''.[DR(s'', m, s') \wedge CR(s, p, s'')]] \Rightarrow CR(s, p, s')]$

(14) $\forall s, p, s'.[\exists m, s''.[DR(s'', m, s) \wedge CR(s'', p, s')]] \Rightarrow CR(s, p, s')]$

From Axiom 13 we can derive a causal relation between two plan steps s and s' over a precondition p of s' if there is a plan step s'' such that s'' occurs in the decomposition of s' and s establishes p for s''. Analogously, Axiom 14 allows us to derive causal relations where an abstract plan step establishes a precondition of another plan step. With these axioms, we can lift any causal relation to the abstract plan step level. For example, we can derive the causal relation $CR(displayPic, PicSelected, sendPic)$, which is defined between two abstract plan steps (see Fig. 1), as follows:

(15) $CR(displayPic, PicSelected, sendPic)$ [13,16,17]

(16) $DR(pressSendByEMail, sendPicByEMail, sendPic)$

(17) $CR(displayPic, PicSelected, pressSendByEMail)$ [14,18,19]

(18) $DR(selectPic, dPViaAlbum, displayPic)$

(19) $CR(selectPic, PicSelected, pressSendByEMail)$

Proof step 17 uses effect propagation, Axiom 14, to lift the causal relation handled in step 19; with this argument, the abstract plan step $displayPic$ is regarded as the effect producer. The causal relation is lifted again in proof step 15, this time by using precondition propagation, Axiom 13.

Causal Relations over Abstract State Features When explaining a plan to a human user, it is obviously important that the explanation is presented on an adequate level of abstraction: large plans may contain hundreds of plan steps, and explaining one of them based on the most primitive level may yield explanations that consist of dozens of causal relations. The above introduced causal relation propagation addresses this issue to some extent, since giving reasons on a more abstract task level leads to more concise explanations. However, this technique alone is not sufficient: in order to explain $sendPic$ (Fig. 1), we can use the propagated causal relations $CR(sendPic, EMailSent, goal)$ and $CR(sendPic, PicAttached, goal)$, that means, we either explain that we need to send the picture because it is a goal to send that e-mail or because it is a goal to attach the picture to the e-mail. Although sound, both explanations do not seem reasonable. The reason for sending a picture to someone is not to have it attached to an e-mail and attaching the picture is rather a part of the general –more abstract– goal to have the picture *transferred* to someone. In particular the latter may not be obvious to the human user who receives the explanation. The following section will explore how we can use causal relations expressed over abstract literals in order to create a more plausible explanation in these situations.

As a first technical step, the following axiom introduces combined causal relations that subsume multiple causal relations by dealing with conjunctions of literals:

$$(20)\quad \forall s, s', p, p'.[[CR(s, p, s') \wedge CR(s, p', s')] \Rightarrow CR(s, p \text{ and } p', s')]$$

We can now introduce axioms for abstract literals in Σ, that correspond to the decomposition axioms in the domain model, for example:

$$(21)\quad daPicTransf : PicTransferred \Leftrightarrow [PicAttached \wedge EMailSent] \vee [PicPrinted \wedge FaxSent] \vee [PicAttached \wedge MMSSent]$$

The axiom specifies three more concrete interpretations for satisfying the abstract goal $PicTransferred$; it bears a label $daPicTransf$ as it will be an argument in the later explanation. Let us assume that one way of transferring the picture is to attach it to an e-mail and sending that e-mail to the recipient of the picture, implemented by a method $sendPicByEMail$ (Fig. 1). By propagation of causal relations, $sendPic$ establishes the goals $PicAttached$ and $EMailSent$, and according to the above decomposition axiom it becomes clear that it also establishes the abstract goal $PicTransferred$. These considerations lead to the definition of the following axiom for lifting state features:

$$(22)\quad \forall s, s', a.[[\exists da, d.[FDec(d, da) \wedge AbsL(a, da) \wedge CR(s, d, s')]] \Rightarrow CR(s, a, s')]$$

The existentially quantified variable da denotes a decomposition axiom and d a conjunction of literals. The atom $FDec(d, da)$ indicates that d is one of the formula decompositions from da and $AbsL(a, da)$ that a is the abstract literal defined by da. Using the above three axioms, we can finally infer the causal relation $CR(sendPic, PicTransferred, goal)$ as follows:

(23) $CR(sendPic, PicTransferred, goal)$ [22,24,25,26]

(24) $FDec(PicAttached \text{ and } EMailSent, daPicTransf)$

(25) $AbsL(PicTransferred, daPicTransf)$

(26) $CR(sendPic, PicAttached \text{ and } EMailSent, goal)$ [20,27,28]

(27) $CR(sendPic, PicAttached, goal)$

(28) $CR(sendPic, EMailSent, goal)$

In step 26 we compose a conjunction of two literals from causal relations (Axiom 20) in order to match the definition of the respective decomposition axiom. Step 23 uses Axiom 22 to derive a lifted causal relation over the abstraction of the decomposition axiom. We can then use that relation to finally give the plausible, albeit trivial reason to the $sendPic$ plan step, namely that we need to send the picture because it is our goal to transfer the picture to someone.

Explanation of Ordering Constraints

This section introduces a set of axioms in Σ that is used to explain why a given plan step is ordered before some other plan step. Please note the subtle difference between the semantics of primitive and abstract plan steps: the former are interpreted classically as atomic, instantaneous state transitions, i.e., their start and end points in time coincide. Abstract tasks are however specifications for the set of predefined plans that are given in their decomposition methods, and thus their occurrence in an explanation has to take into account the reasons for all the implied primitive plan steps. This has to be considered in the following formalizations.

Since plan steps under explanation can extend over an interval, from one qualitative point in time to another, we deploy a time-point algebra (Vilain, Kautz, and van Beek 1990). The starting point of a plan step s is denoted by s^- and its end point by s^+. For two time points i and j we define the usual *temporal relations* $i < j$, $i \leq j$, and $i = j$ in order to express that i is before, not after, or the same as j. We can now axiomatize some basic temporal relationships as follows:

(29) $\forall i, j, k.[[i < j \wedge j < k] \Rightarrow i < k]$

(30) $\forall s.[s^- \leq s^+]$

(31) $\forall s.[Prim(s) \Rightarrow s^- = s^+]$

Furthermore, Σ contains an atom $Prim(s)$ for every plan step s that is primitive. Axiom 29 handles the transitivity of the '$<$'-relation. Axiom 30 specifies that the beginning of a plan step cannot occur after its end, with those time points coninciding for primitive plan steps.

Questions about the ordering of plan steps are formulated in terms of temporal relations between the corresponding points, e.g., if we want to explain why the execution of a plan step s has to begin before the execution of a plan step s', we have to explain the temporal relation $s^- < s'^-$. Analogously to the explanation of plan steps above, the possibilities for explaining why a temporal relation between two

points is necessary are formalized as axioms $\forall i, j.[\phi \Rightarrow iRj]$ where $R \in \{<, \leq, =\}$ and ϕ encodes an argument that supports iRj to be necessary for the plan to constitute a solution.

Temporal Relations from Decomposition Relations The first opportunity to derive temporal relations is given by the decomposition structure of a plan: an abstract plan step s spans from the beginning of the first to the last step of its decomposition. In time-point algebra, this translates into the starting and end point of s never occurring after any starting point or before any end point in its decomposition, respectively. The following two axioms express this relationship:

(32) $\forall s, s'.[[\exists m.DR(s', m, s)] \Rightarrow s^- \leq s'^-]$

(33) $\forall s, s'.[[\exists m.DR(s, m, s')] \Rightarrow s^+ \leq s'^+]$

Temporal Relations from Causal Relations The temporal relations that can be derived from causal relations depend on whether the involved plan steps are primitive or abstract. Given a causal relation $CR(s, p, s')$ such that s and s' are both referring to primitive plan steps, it directly follows that s has to occur before s' in order for the effects of s to manifest before they are required by s', i.e., we infer the temporal relation $s^+ < s'^-$:

(34) $\forall s, s'.[[Prim(s) \wedge Prim(s') \wedge \exists p.CR(s, p, s')] \Rightarrow s^+ < s'^-]$

However, if we encounter an abstract plan step s' during explanation generation, we cannot derive this temporal relation in the same way. The reason can be seen in Fig. 2: it shows the causal and decomposition structure of a slightly different plan in the example scenario (cf. Fig. 1). Here, after entering the album to select the picture it appears that an appropriate picture is not available, so a new one has to be taken in the camera mode before selecting it. The two primitive plan steps to enter the camera mode and take the picture were added through the decomposition of the abstract plan step *obtainPicture*. Using the axioms for the propagation of causal relations, we can derive the causal relation $CR(takePicture, HavePicture, displayPicture)$ between the primitive plan step *takePicture* and the abstract plan step *displayPicture*. If *displayPicture* were primitive, then Axiom 34 would allow us to derive $takePicture^+ < displayPicture^-$. However, we can also see in the figure that *enterAlbum* is executed before *takePicture*, and since *enterAlbum* is part of the decomposition of *displayPicture*, the execution of *displayPicture* also begins before *takePicture*, i.e., the temporal relation $takePicture^+ < displayPicture^-$ apparently does not hold. On the other hand, the causal relation was derived using the causal relation between *takePicture* and *selectPicture* and the decomposition relation between *selectPicture* and *displayPicture*. The former lets us derive $takePicture^+ < selectPicture^-$ and the latter, using Axiom 33, $selectPicture^+ \leq displayPicture^+$. Since the relation $selectPicture^- = selectPicture^+$ trivially holds as well, we can combine these three temporal relations in order

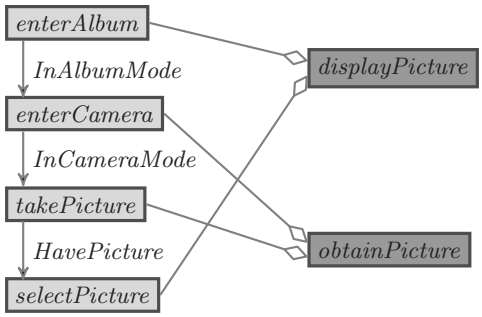

Figure 2: The causal and decomposition structure of a plan in which the execution of implementations of abstract plan steps interleave.

to obtain $takePicture^+ < displayPicture^+$. The same kind of inference can be applied for any causal relation with an abstract plan step as consumer. We represent this in Σ by the following axiom:

(35) $\forall s, s'.[[Prim(s) \wedge \exists p.CR(s, p, s')] \Rightarrow s^+ < s'^+]$

Analogously, given the causal relation $CR(s, p, s')$, we can derive $s^- < s'^-$ if only the consumer s' is primitive and $s^- < s'^+$ if neither of the two involved plan steps is primitive:

(36) $\forall s, s'.[[Prim(s') \wedge \exists p.CR(s, p, s')] \Rightarrow s^- < s'^-]$

(37) $\forall s, s'.[[\exists p.CR(s, p, s')] \Rightarrow s^- < s'^+]$

The following proof illustrates the explanation of a temporal relation $enterAlbum^+ < sendPic^+$: why does *enterAlbum* end before *sendPic* does (Fig. 1)?

(38) $enterAlbum^+ < sendPic^+$ [29,39,43]

(39) $enterAlbum^+ < selectPic^+$ [29,40,42]

(40) $enterAlbum^+ < selectPic^-$ [34,41]

(41) $CR(enterAlbum, InAlbumMode, selectPic)$

(42) $selectPic^- = selectPic^+$ [31]

(43) $selectPic^+ < sendPic^+$ [35,44]

(44) $CR(selectPic, PicSelected, sendPic)$ [13,45,46]

(45) $CR(selectPic, PicSelected, pressSendbyEMail)$

(46) $DR(pressSendbyEMail, sendPicByEMail, sendPic)$

In proof steps 40 and 43 we use a causal relation to derive a temporal relation, first between two primitive plan steps and second with an abstract plan step as consumer of the causal relation. In step 42 we derive the trivial temporal relation that the start of *selectPic* is the same as its end. These three temporal relations are then combined twice, using the transitivity of the '<'-Relation defined by Axiom 29, to derive the desired temporal relation $enterAlbum^+ < sendPic^+$.

Temporal Relations for Abstract Plan Steps With the axioms presented so far, we cannot yet derive a temporal relation $s^+ < s'^-$ if the plan step s' is an abstract one. For example, $enterAlbum$ in Fig. 2 is ordered before any of the steps from the decomposition of $obtainPicture$, so $enterAlbum^+ < obtainPicture^-$ must hold, which we cannot support yet. But the example indicates how we can solve this problem: any time point i, which may be the start or end point of any plan step, is ordered before the start of an abstract plan step if it is ordered before all start points of the steps from the decomposition of that abstract plan step. The following axiom formalizes this notion:

$$(47) \quad \forall i, s.[[Abs(s) \wedge \forall s', m.[DR(s', m, s) \Rightarrow i < s'^-]] \Rightarrow i < s^-]$$

The atom $Abs(s)$ thereby denotes that s is an abstract plan step. We treat the symmetric temporal situation analogously:

$$(48) \quad \forall i, s.[[Abs(s) \wedge \forall s', m.[DR(s', m, s) \Rightarrow s'^+ < i]] \Rightarrow s^+ < i]$$

Temporal Relations by Threat Resolution Given a causal link $(s \rightarrow_p s')$ and a plan step s'' with an effect $\neg p$ we call s'' a *threat* to the causal link if s'' can be ordered between s and s'. In POCL or hybrid planning systems, threats are usually resolved by adding an ordering constraint $s'' \rightarrow s$, called demotion, or $s' \rightarrow s''$, called promotion, to the plan. This prevents s'' from being ordered between s and s' and hence the causal conflict does not materialize. Concerning plan explanation, this means that there may be valid temporal relations that are not justified by the causal or decomposition structure but only by taking into account the threats that have been encountered during the plan generation process. In order to incorporate explanations for temporal relations based on threat resolution we introduce the post-hoc threat structure of a plan. The threat structure indicates occurrences of ordering constraints that were only added to the plan to resolve a threat. For each such ordering constraint we add an atom $Dem(s'', s)$ or $Pro(s'', s')$ to our axiomatic system Σ in order to indicate that s'' has been ordered before s or after s' to resolve a threat by s'' to a causal link between s and s'. The symbols Dem and Pro stand for demotion and promotion. As with the temporal relations derived from causal relations, given a demotion or promotion between two plan steps, the temporal relations that can be derived from it depend on whether the plan steps are primitive or abstract. If both are primitive, the execution of the predecessor has to be finished before the execution of the successor can begin. Thus, we get the axiom:

$$(49) \quad \forall s, s'.[[Prim(s) \wedge Prim(s') \wedge [Dem(s, s') \vee Pro(s', s)]] \Rightarrow s^+ < s'^-]$$

Concerning threats between abstract plan steps, the situation is similar to that of causal relations: the solution criteria for hybrid planning problems do not address threats between abstract plan steps. We thus include axioms for the propagation of demotions and promotions in the same way we did with Axioms 13 and 14, based on which we then infer propagated temporal relations like in Axioms 35, 36, and 37. We have to omit these axioms due to lack of space.

Experimental System

We have implemented a prototype system that generates raw explanations as specified by the formal framework. The system consists of three main components: first, a reasoning component to derive all possible causal relations, decomposition relations, temporal relations, and so on, i.e., all the building blocks for the construction of explanations. Second, a management component that combines the building blocks to explanations as specified by the axioms from the previous sections. As there are usually several ways for explaining a plan step or temporal relation, the management component effectively performs a search in the space of (incomplete) explanations. Third, a strategy component guides the search for explanations.

Generally speaking, finding an explanation is an easy task because we can never reach a dead end while constructing explanations, because in POCL or hybrid planning every plan step was usually added through decomposition or as producer of some causal link. Therefore, if we wanted to explain a plan step we could just follow an arbitrary chain of causal and decomposition relations to the goal or until we reach a plan step from the initial plan without any backtracking. Some explanations may be considered better than others though. Thus, finding an explanation that is well-suited for a specific situation requires an appropriate search strategy. Performing an exhaustive study in order to determine what makes an explanation well-suited, however, is beyond the scope of this paper.

For the time being, we let the system generate as many different raw explanations as possible within a given time limit. We did so on a plan generated in the example user-assistance domain. The plan contains 26 primitive plan steps, and 21 abstract plan steps were decomposed during its construction. For each of these 47 plan steps, we gave our system a time limit of two seconds, not including setup time but including any kind of preprocessing, on a normal desktop PC to generate as many explanations as possible and measured the number of generated explanations and used time. Table 1 shows the results. The number of generated explanations for a plan step ranged between 1 and 8,147 with an average of 961. This variance is mostly due to the position of the plan steps within the plan and the complexity of the causal structure: a step which is near the beginning of the plan can be explained through a long chain of causal and decomposition relations. Because many plan steps produce several causal relations we often have several possible ways to build the explanation argument, i.e., the number of possible explanations can grow exponentially in the distance of the plan step to the goal. For 37 of the 47 plan steps the explanation system terminated within the two seconds time limit, i.e., the system found all possible explanations, and since the domain is rather complex with over 130 task schemata and a deep hierarchical structure we consider it as a valid benchmark for this task.

PS#	#E	t(ms)
1	21	218
2	42	202
3	2095	2013
4	6429	2004
5	3	156
6	10	156
7	5	156
8	150	202
9	466	296
10	379	2012
11	733	2043
12	22	172
13	1784	593
14	7325	1498
15	264	422
16	4	436
17	267	343
18	19	299
19	15	294
20	19	281
21	15	266
22	142	408
23	1	187
24	1	281
25	1	289
26	1	187
27	1	328
28	2	296
29	16	242
30	58	218
31	8147	2012
32	1485	2013
33	2074	2012
34	2	203
35	106	203
36	286	202
37	793	2012
38	1047	2006
39	1168	2013
40	12	187
41	1163	624
42	3217	1545
43	5271	1232
44	3	281
45	15	296
46	100	265
47	1	203

Table 1: Number of explanations (#E) generated by our experimental system and time needed to generate the explanations in milliseconds for 47 plan steps.

As it can be seen, a rather large number of explanations can be found usually within only a few seconds. This supports our claim that the task of finding candidates for good explanations within a reasonably short amount of time is feasible. However, the number of explanations generated per plan step will vary significantly in general, so developing quality measures to select the best explanation among the generated ones appears necessary.

Discussion

User-adaptive Plan Explanation While it is beyond the scope of this paper to thoroughly investigate quality measures for selecting raw explanations, there are some general aspects that influence the appropriateness of explanations: the knowledge of the user about the planning domain and problem, the available means for presenting an explanation (graphics, speech (Bidot et al. 2010), or written text), etc.

Given such context information, various quality features of explanations can be identified. For example, a basic feature is the length of the explanation. Generally speaking, the longer an explanation, the more information it will contain and provide a better explanation for the user. But overburdening the user with too much information can be counterproductive. Also, the length of an explanation may be limited by the available presentation means.

Another quality feature are the plan steps, literals, and methods referenced by the explanation. For various reasons, it might be advisable to avoid or encourage the use of some of these elements. For example, using plan steps over tasks that the user has never seen before in an explanation might be confusing, but using plan steps that are well-understood by the user could improve the quality of the explanation. A related feature is the importance of plan steps for the plan: some plan steps could be mission-critical while others are of low importance. Finally, the level of abstraction of the explanation can be adapted to the user. If he is experienced in the application domain, an explanation on the most primitive level might contain many unnecessary details, e.g., if he has previously taken many pictures on his smart phone we should not explain to him to first enter the camera mode, configure the camera, press the right buttons, etc.; just telling him to obtain a picture should be sufficient. All this information can easily be annotated in the planning domain model.

Another possibility to improve the quality of an explanation is to manipulate it to better fit the given context. For example, steps of the explanation that should be avoided for one of the reasons mentioned above can be bridged. Technically, the correctness of the explanation would be destroyed by such a manipulation, but depending on the situation the user might be able to fill the gap in the explanation, using her knowledge on the application domain.

Formal Aspects Concerning "plan step necessity", please note that this merely refers to plan steps that cannot be removed from the plan without spoiling executability. This does not exclude the possibility that there is a different solution that does not contain the plan step. In other words, a necessary plan step does not have to be an action landmark (Vidal and Geffner 2006).

While the causal links are readily available in plans generated via hybrid planning, this is not the case for all planning paradigms. Planners that employ forward-search in the state space, for example, usually do not explicitly represent causal links in the plans they generate. Still, our explanation framework can be used in conjunction with these planners as causal links can be post-hoc calculated via causal link analysis (Karpas and Domshlak 2011).

Conclusion

We have presented a formal framework for constructing explanations as proofs for the necessity of plan steps and orderings on plan steps in the plan at hand. To this end, we have shown how to construct an axiomatic system that formally describes the plan structures and their generation rationale. Possible types of explanations are derived from the solution criteria for hybrid plans. We have implemented a prototype system that constructs raw explanations as specified by the framework. First results suggest that finding good explanations is a feasible task. Future research will be dedicated to conducting user studies to determine adequate quality measures of explanations in a given application context and how to transform raw explanations in an informative manner into a human-readable format, e.g., speech, text, or graphics.

Acknowledgment

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

Bidot, J.; Biundo, S.; Heinroth, T.; Minker, W.; Nothdurft, F.; and Schattenberg, B. 2010. Verbal explanations for hybrid planning. In *Proc. of the Conference "Multikonferenz Wirtschaftsinformatik" (MKWI 2010), Teilkonferenz "Planen, Scheduling und Konfigurieren, Entwerfen" (PuK 2010)*, 2309–2320. Universitätsverlag Göttingen.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief (a preliminary report on combining state abstraction and HTN planning). In *Proc. of the Sixth European Conference on Planning (ECP 2001)*, 157–168.

Biundo, S.; Bercher, P.; Geier, T.; Müller, F.; and Schattenberg, B. 2011. Advanced user assistance based on AI planning. *Cognitive Systems Research* 12(3-4):219–236. Special Issue on Complex Cognition.

Cesta, A.; Cortellessa, G.; Rasconi, R.; Pecora, F.; Scopelliti, M.; and Tiberio, L. 2011. Monitoring elderly people with the robocare domestic environment: Interaction synthesis and user evaluation. *Computational Intelligence* 27(1):60–82.

de la Asunción, M.; Castillo, L.; Fdez.-Olivares, J.; García-Pérez, O.; González, A.; and Palao, F. 2005. SIADEX: An interactive knowledge-based planner for decision support in forest fire fighting. *AI Communications* 18:257–268.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the Second International Conference on AI Planning Systems (AIPS 1994)*, 249–254.

Estlin, T. A.; Chien, S. A.; and Wang, X. 1997. An argument for a hybrid HTN/operator-based approach to planning. In *Proc. of the Fourth European Conference on Planning (ECP 1997)*, 182–194.

Gerevini, A.; Kuter, U.; Nau, D. S.; Saetti, A.; and Waisbrot, N. 2008. Combining domain-independent planning and HTN planning: The duet planner. In *Proc. of the 18th European Conference on AI (ECAI 2008)*, 573–577.

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 International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 81–88.

Horacek, H. 2007. How to build explanations of automated proofs: A methodology and requirements on domain representations. In *ExaCt Workshop on Explanation-aware Computing*, 34–41.

Kambhampati, S.; Mali, A.; and Srivastava, B. 1998. Hybrid planning for partially hierarchical domains. In *Proc. of the 15th National Conference on AI (AAAI 1998)*, 882–888.

Karpas, E., and Domshlak, C. 2011. Living on the edge: Safe search with unsafe heuristics. In *HDIP 2011 Third Workshop on Heuristics for Domain-independent Planning*, 53–58.

Khan, O.; Poupart, P.; and Black, J. 2009. Minimal sufficient explanations for factored markov decision processes. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 194–200.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proc. of the Ninth National Conference on Artificial Intelligence (AAAI 1991) - Volume 2*, 634–639.

McGuinness, D. L., and Borgida, A. T. 1995. Explaining subsumption in description logics. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995) - Volume 1*, 816–821.

McGuinness, D.; da Silva, P.; Wolverton, M.; and Center, A. 2005. Plan for explaining task execution in CALO. Technical report, Stanford KSL Technical Report. KSL-05-11.

McGuinness, D.; Glass, A.; Wolverton, M.; and Da Silva, P. 2007. Explaining task processing in cognitive assistants that learn. In *Proc. of the 20th International FLAIRS Conference (FLAIRS-20)*, 284–289.

Molineaux, M.; Kuter, U.; and Klenk, M. 2011. What just happened? Explaining the past in planning and execution. In *ExaCt Workshop on Explanation-aware Computing*, 31–40.

Moore, J. D., and Swartout, W. R. 1989. A reactive approach to explanation. In *Proc. of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989)*, 1504–1510.

Muñoz-Avila, H.; Aha, D. W.; Breslow, L.; and Nau, D. S. 1999. HICAP: an interactive case-based planning architecture and its application to noncombatant evacuation operations. In *Proc. of the 11th Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, 870–875.

Pollack, M. E. 2002. Planning technology for intelligent cognitive orthotics. In *Proc. of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS 02)*, 322–331.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc.

Smith, D. E. 2010. Thinking outside the "box". ICAPS-2010 Panel on The Present and Future(s) of Planning.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence* 170:298–335.

Vilain, M.; Kautz, H.; and van Beek, P. 1990. *Readings in Qualitative Reasoning about Physical Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. chapter Constraint propagation algorithms for temporal reasoning: a revised report, 373–381.

Yang, Q. 1990. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence* 6(1):12–24.