Explanation-Based Generalization of Partially Ordered Plans

Subbarao Kambhampati*
Center for Design Research
and Dept. of Computer Science
Stanford University
Bldg 530, Duena Street, Stanford CA 94305-4026

Smadar Kedar
Sterling Federal Systems
AI Research Branch
NASA AMES Research Center
Moffett Field CA 94035

Abstract

Most previous work in analytic generalization of plans dealt with totally ordered plans. These methods cannot be directly applied to generalizing partially ordered plans, since they do not capture all interactions among plan operators for all total orders of such plans. In this paper we introduce a new method for generalizing partially ordered plans. This method is based on providing EBG with explanations which systematically capture the interactions among plan operators for all the total orders of a partially-ordered plan. The explanations are based on the Modal Truth Criterion [2], which states the necessary and sufficient conditions for ensuring the truth of a proposition at any point in a plan (for a class of partially ordered plans). The generalizations obtained by this method guarantee successful and interaction-free execution of any total order of the generalized plan. In addition, the systematic derivation of the generalization algorithms from the Modal Truth Criterion obviates the need for carrying out a separate formal proof of correctness of the EBG algorithms.

1 Introduction

Creating and using generalized plans is a central problem in machine learning and planning. This paper addresses the problem of generalizing a class of plans known as *partially ordered plans* that have been extensively investigated in the planning literature [16] [17] [18] [2]. A partially ordered plan corresponds to a set of total orderings, called *completions*, each corresponding to a topological sort of the plan. A partially ordered plan is considered correct if and only if each of its completions will be able to achieve the desired goals (that is, the plan should be executable in any total order consistent with the partial ordering, without any subgoal interactions).

The problem of generalizing a plan has traditionally been characterized as that of computing the *weakest* (most general) initial conditions of a sequence of operators. The computed conditions are required to describe exactly the set of initial states such that the generalized plan applicable in those states is guaranteed to achieve a state matching the goals.

*Kambhampati was partially supported by the Office of Naval Research under contract N00014-88-K-0620. The authors' *email* addresses are *rao@cs.stanford.edu* and *kedar@ptolemy.arc.nasa.gov.*

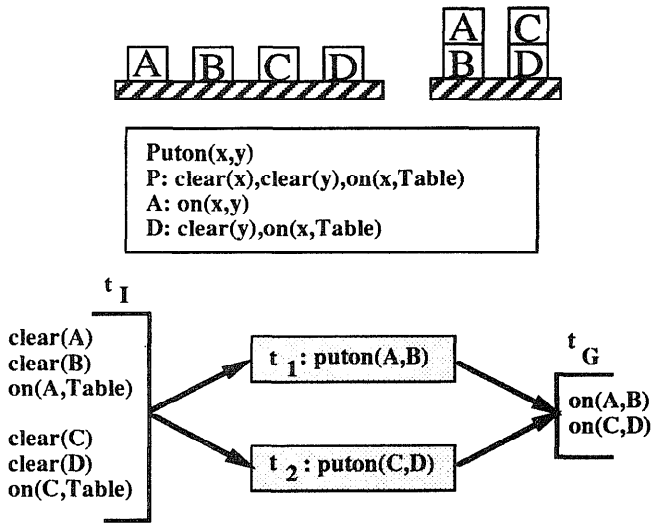

Figure 1: Four Block Stacking Problem (4BSP)

Goal regression [15], explanation-based generalization (EBG) [12] [4] [14], and macro-operator formation [5], are some previous analytic solutions to the plan generalization problem. These methods were developed for totally ordered plans. They typically compute the weakest conditions of such plans by regressing variablized goals back through the plan operator sequence to ensure appropriate *producer-consumer* dependencies among effects and preconditions in the generalized plan, and to prevent deletions of needed literals.

These methods cannot be directly applied to generalizing partially ordered plans, since they do not capture all interactions among plan operators for all total orders of such plans. To illustrate this limitation, consider the simple blocks world problem for stacking four blocks (4BSP) and a partially ordered plan for solving it, shown in Figure 1. Given an initial state where four blocks A, B, C, D are on the table and clear, the goal $On(A,B) \wedge On(C,D)$ can be achieved by the partially ordered plan corresponding to two total orders, or completions, $Puton(A,B) \rightarrow Puton(C,D)$, and $Puton(C,D) \rightarrow Puton(A,B)$ (where the operator template $Puton(x,y)$ is specified as shown in the figure).

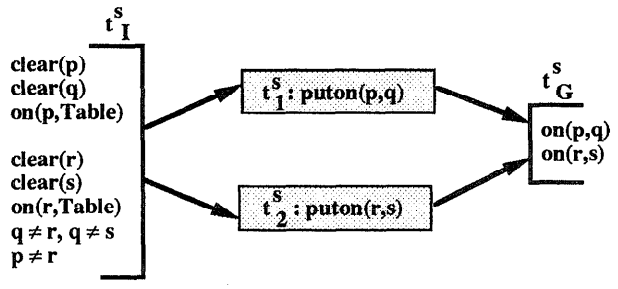

Figure 2: An incorrect generalization of 4BSP

For this problem, the generalization algorithms discussed above produce a generalized plan such as the one shown in Figure 2. However, if we would like to guarantee that *any* total order of this partial order will succeed, the generalized conditions are incorrect. The reason is that the plan was generalized guided by one specific total order, so constraints for other total orders were not accounted for. For example, if the problem were to stack three blocks A, B and C on top of each other, this generalized plan would be applicable, and yet fail for one of the two total orders (as it includes an incorrect completion $Puton(A,B) \rightarrow Puton(B,C)$). What is missing is the constraint that s be not the same as p (whereas both are codesignating with B in this case).

To avoid this problem, the EBG algorithm needs to be more systematic in accounting for all possible interactions among operators corresponding to *all* possible total orders consistent with the partial ordering. There are two options for doing this. One is to *modify the algorithm:* For instance, repeatedly compute weakest conditions of all total orders of the partial order and then conjoin them in some way. Another option is to *modify the input*: Provide a full explanation of correctness of the instantiated partially ordered plan, and use that explanation to produce the correct generalized initial conditions for the generalized partially ordered plan.

In this paper, we describe a technique for solving this problem by the latter approach, viz., by modifying the input to generalization algorithm (in particular, to EBG). By modifying the input to EBG, rather than the EBG algorithm, we retain the broad applicability of the algorithm (for different classes of plans, different generalizations can be produced). In addition, a partially plan can correspond to an exponential number of totally ordered completions, while weakest conditions are more directly related to the causal structure of the plan. Thus, computing the weakest conditions on each total order separately (and conjoining them to get the correct generalization of the plan) would involve an exponential amount of redundant computation [2]. By computing and using the explanation of correctness of the partially ordered plan directly, we can avoid this redundant computation.

Our approach is to provide EBG with explanations of correctness of partially ordered plans based on the Modal Truth Criterion [2], which states the necessary and sufficient conditions for ensuring the truth of a proposition at any point in a plan for a class of partially ordered plans. These explanations are then used as the basis for generalization. The generalizations obtained by this method guarantee successful and interaction-free execution of all total orders of the generalized plan. In addition, the systematic derivation of the generalization algorithms from the Modal Truth Criterion obviates the need for carrying out a separate formal proof of correctness of the EBG algorithms. Finally, the methodology can be extended in a straightforward fashion to handle other types of generalizations of partially ordered plans (such as computing conditions under which at least some completion of the plan can possibly execute; see Section 5).

In the rest of the paper, we introduce the notion of truth criterion and present the explanation of correctness of a partially ordered plan based on the Modal Truth Criterion. We then describe how these explanations form the basis for the generalization. We conclude by describing related work and examining the contributions of this paper. The main focus of this paper is development of systematic methods for generalizing partially ordered plans. The complementary issue of tradeoffs involved in synthesizing and generalizing partially ordered vs. totally ordered plans is discussed briefly in Section 5.

1.1 Terminology

Given a planning problem $[\mathcal{I},\mathcal{G}]$ where $\mathcal{I}$ is a conjunction of literals specifying the initial state and $\mathcal{G}$ is a conjunction of literals specifying the desired goal state, a *partially ordered plan* $\mathcal{P}$ is a 2-tuple $\mathcal{P} : \langle T, O\rangle$, where T is the set of actions in the plan, and O is a partial ordering relation over T. T contains two distinguished nodes t_I and t_G, where the effects of t_I and the preconditions of t_G correspond to the initial and final states of the plan, respectively. The actions are represented by instantiated STRIPS-type operators with *Add*, *Delete* and *Precondition* lists, all of which are conjunctions of first order literals[1]. O defines a *partial ordering* over T: $O = \{(t_i,t_j) \mid t_i,t_j \in T\}$. We write $t_i \prec t_j$ if either $(t_i,t_j) \in O$, or there exist a sequence of operators $t_1 \cdots t_n \in T$, such that $(t_i,t_1),(t_1,t_2)\cdots(t_n,t_j) \in O$. (Thus, the "$\prec$" relation corresponds to the transitive closure of O.) If t_i and t_j are unordered with respect to each other (i.e., $t_i \not\prec t_j$ and $t_j \not\prec t_i$), then we say $t_i \parallel t_j$.

The modal operators "$\Box$" and "$\Diamond$" are used to denote the *necessary* and *possible* truth of a statement. In particular $\Diamond(t_i \prec t_j)$ if and only if t_i can possibly precede t_j in some total ordering of the partially ordered plan (which means that either $(t_i \prec t_j)$ or $(t_i \parallel t_j)$). Finally, we define *codesignation* and *non-codesignation* constraints among literals as follows: If a literal p_i is constrained to codesignate with another literal p_j (written as $p_i \approx p_j$), then p_i and p_j must be unifiable. Similarly, if p_i is constrained *not* to codesignate with p_j (written as $p_i \not\approx p_j$), then p_i must not unify with p_j. Codesignation constraints among literals translate into equalities among variables and constants (domain objects), while the non-codesignation constraints translate into disequalities among variables. For example, $On(A,B) \approx On(x,y)$ if and only if $eq(A,x) \wedge eq(B,y)$ (since the most general unifier of the two literals is $\theta = ((A\,x)(B\,y))$. Similarly, $On(A,B) \not\approx On(x,y)$ if and only if $\neg[eq(A,x) \wedge eq(B,y)]$ (that is $neq(A,x) \vee neq(B,y)$).

[1] We shall use upper case letters for constants and the lower case ones for variables.

2 Explanation of Correctness using the Modal Truth Criterion

In [2], Chapman provided a formal means of reasoning about partially ordered plans called the Modal Truth Criterion (MTC). The MTC provides necessary and sufficient conditions for ensuring the truth of a proposition C before an action t in a partially ordered plan. In this section, we shall develop the explanation of correctness of a partially ordered plan in terms of this truth criterion. For plans containing STRIPS-type operators whose precondition, add and delete lists contain first order literals, the MTC can be stated as follows:[2]

$$\begin{aligned} &holds(C,t,\mathcal{P}) \iff \\ &\quad \exists t' \text{ s.t. } \Box(t' \prec t) \wedge e \in effects(t') \wedge \Box(e \approx C) \bigwedge \\ &\quad \forall t'' \text{ s.t. } \Diamond(t' \prec t'' \prec t) \\ &\qquad \forall d \in delete(t'')\, \Box(d \not\approx C) \end{aligned} \quad (1)$$

It states that a proposition C holds before an operator t in a partially ordered plan $\mathcal{P} : \langle T, O\rangle$ if and only if there exists an operator t' such that an effect e of t' necessarily codesignates with C, and for every operator t'' of the plan that may possibly fall between t' and t, every proposition belonging to the delete list of t'' will necessarily *not* codesignate with C. The truth criterion can usefully be thought of as a completeness/soundness theorem for a version of the situation calculus (*cf.* [2], pp. 340). Alternatively, it can be thought of as a method for doing goal-regression [15] over a class of partially ordered plans.

In planning, the intended use of the MTC is as a prescription of all possible ways of making a proposition of a partially ordered plan true during *plan synthesis*. However, the MTC can also be used as the formal basis solely for proving *plan correctness*. In particular, a partially ordered plan $\mathcal{P} : \langle T, O\rangle$ is considered *correct* according to the modal truth criterion, if and only if all the goals of the plan, as well as all the preconditions of the individual plan steps can be shown to hold according to the criterion given in equation 1 without extending or modifying $\mathcal{P}$ in anyway.

The explanation of correctness of a plan can then be characterized as a "proof" that the plan satisfies this criterion for correctness. The algorithm EXP-MTC shown in Figure 3 constructs the explanation of correctness given a partially ordered plan, by interpreting equation 1 for each precondition and goal of the plan. It returns failure if the plan turns out to be incorrect according to MTC.

Note that this algorithm represents the computed explanation by a set $\mathcal{V}$ of dependency links. The individual dependency links are of the form $\langle e, t', C, t\rangle$. We shall refer to these links as **validations** of the plan [9]. Intuitively, these represent the interval of operators t' and t over which a literal C needs to hold. C is made true by the effect e of operator t', and is needed at t. It is protected throughout that interval (t', t) from being clobbered, that is, any operator t'' that may *possibly* come

[2]For ease of exposition, in this paper we will be using a version of Chapman's truth criterion [2] without the *white-knight* clause. The development for the more general version can be carried out in a very similar fashion, and with the same complexity bounds [10].

Algorithm EXP-MTC $(\mathcal{P} : \langle T, O\rangle)$
  foreach $t \in T$ **do**
    foreach $\langle C, t\rangle$ (where $C \in precond(t)$) **do**
      Traverse $\mathcal{P}$ in the reverse topological order and find the first operator t' s.t.
        $t' \prec t \wedge \exists e \in effects(t') \wedge \Box(e \approx C) \bigwedge$
        $\forall t''$ s.t. $\Diamond(t' \prec t'' \prec t)$,
          $\forall d \in delete(t'')\, \Box(d \not\approx C)$
      if such a t' is found
        then $\mathcal{V} \leftarrow \mathcal{V} \cup \{\langle e, t', C, t\rangle\}$
        else *return failure*
  od od

Figure 3: Explanation Construction Algorithm

between t' and t in some total ordering must not violate the condition C. The semantics of validations therefore capture both the traditional precondition-effect dependencies and protection violations across all total orderings. In particular,

$$\begin{aligned} v : \langle e, t', C, t\rangle \text{ is a validation of } \mathcal{P} : \langle T, O\rangle \iff \\ \Box(e \approx C) \wedge \Box(t' \prec t) \wedge \\ \forall t'' \in T \text{ s.t. } \Diamond(t' \prec t'' \prec t) \\ \forall d \in delete(t''),\ \Box(d \not\approx C) \end{aligned} \quad (2)$$

For the 4BSP plan shown in Figure 1, the explanation of correctness found by this algorithm would consist of the following validations:

$$\begin{aligned} v_1 &: \langle On(A, Table), t_I, On(A, Table), t_1\rangle \\ v_2 &: \langle Clear(A), t_I, Clear(A), t_1\rangle \\ v_3 &: \langle Clear(B), t_I, Clear(B), t_1\rangle \\ v_4 &: \langle On(C, Table), t_I, On(C, Table), t_2\rangle \\ v_5 &: \langle Clear(C), t_I, Clear(C), t_2\rangle \\ v_6 &: \langle Clear(D), t_I, Clear(D), t_2\rangle \\ v_7 &: \langle On(A, B), t_1, On(A, B), t_G\rangle \\ v_8 &: \langle On(C, D), t_2, On(C, D), t_G\rangle \end{aligned}$$

Discussion: The EXP-MTC algorithm shown in Figure 3 finds only *one* out of the possibly many explanations of correctness of the plan. In particular, for each precondition C of an operator t, there might possibly be many operators that can contribute C (according to the criteria stated by MTC). Of these, the algorithm records only the first operator t' encountered in a reverse topological order scan of $\mathcal{P}$ that satisfies the conditions of MTC[3]. It is perfectly reasonable to choose another explanation of correctness (i.e., another set of validation links $\mathcal{V}$) over the one given by this algorithm as long as that explanation also satisfies the MTC. It should however be noted that the generalization phase will be guided by the particular explanation that is chosen at this step (rather than by all possible explanations). This corresponds to a common restriction for EBG termed "generalizing with respect to the explanation structure", or "following the example" [12]

Complexity: The cost of finding a validation link in the above algorithm is $O(n^2\zeta)$, where ζ is an upper bound on the number of delete literals per operator, and n is

[3]If no such t' is found, the algorithm returns failure, which means that there exists at least one linearization of $\mathcal{P}$ that will have subgoal interactions.

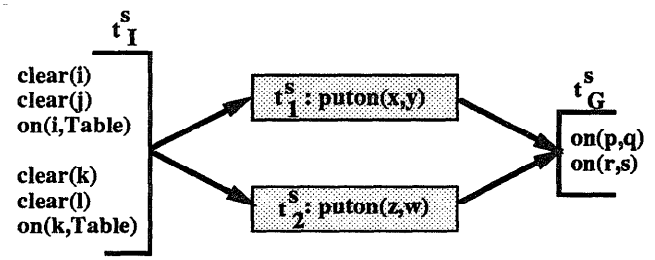

Figure 4: Schematized Plan for 4BSP

the number of operators in the plan[4]. If ξ is the upper bound on the number of preconditions per operator, then there must be $O(\xi n)$ validations links in the explanation. Thus the total cost of explanation construction is $O(n^3)$.

3 Generalization using Explanation

In this section, we will first use the explanation of correctness developed in the previous section to derive a declarative specification for the generalization phase of EBG for partially ordered plans. We will then provide an algorithm that interprets this specification.

The generalization process consists of schematizing[5] the plan $\mathcal{P}$ to produce $\mathcal{P}^s$, and determining the weakest initial conditions under which $\mathcal{P}^s$ will be correct according to the MTC, with respect to the same explanation structure as that used to explain the correctness of $\mathcal{P}$.

Given a plan $\mathcal{P} : \langle T, O \rangle$, we construct its *schematized version*, $\mathcal{P}^s : \langle T^s, O^s \rangle$ by replacing each instantiated operator $t \in T$ by the corresponding operator template t^s (with unique variables). (For t_I and t_G, we replace their literals by their variablized versions.) O^s defines a partial ordering on T^s that is isomorphic to O. Figure 4 shows the schematized plan corresponding to the 4BSP plan shown in Figure 1.

The schematization process defines a one-to-one mapping between the add, delete and precondition lists of each step t of $\mathcal{P}$ and those of the corresponding operator template t^s of $\mathcal{P}^s$. Let LMAP denote this mapping. For example, the literal $On(A, Table)$ in the preconditions of operator t_1 in the 4BSP plan shown in Figure 1 corresponds to the literal $On(x, Table)$ in the schematized plan. Given this mapping, a set of explanation links $\mathcal{V}^s$ for $\mathcal{P}^s$ can be constructed such that they are isomorphic to $\mathcal{V}$ of $\mathcal{P}$. For each validation $v : \langle e, t', C, t'' \rangle \in \mathcal{V}$, there will be a validation $v^s : \langle e^s, t'^s, C^s, t''^s \rangle \in \mathcal{V}^s$ such that t'^s and t''^s are operator templates in the schematized plan corresponding to t' and t'' respectively, and e^s and C^s are the literals corresponding to e and C according to LMAP defined above. For the 4BSP schematized plan shown in Figure 4, the explanation links in $\mathcal{V}^s$ are:

$v^s_1 : \langle On(i, Table), t^s_I, On(x, Table), t^s_1 \rangle$
$v^s_2 : \langle Clear(i), t^s_I, Clear(x), t^s_1 \rangle$
$v^s_3 : \langle Clear(j), t^s_I, Clear(y), t^s_1 \rangle$
$v^s_4 : \langle On(k, Table), t^s_I, On(z, Table), t^s_2 \rangle$
$v^s_5 : \langle Clear(k), t^s_I, Clear(z), t^s_2 \rangle$
$v^s_6 : \langle Clear(l), t^s_I, Clear(w), t^s_2 \rangle$
$v^s_7 : \langle On(x, y), t^s_1, On(p, q), t^s_G \rangle$
$v^s_8 : \langle On(z, w), t^s_2, On(r, s), t^s_G \rangle$

Notice that after the schematization, $\mathcal{P}^s$ and $\mathcal{V}^s$ are *over general* in that the links in $\mathcal{V}^s$ may no longer constitute an explanation of correctness of $\mathcal{P}^s$ according to the MTC. The objective of the generalization phase is to post constraints (codesignation and non-codesignation) on the variable bindings to specialize this over general schematized plan and validations just enough so that $\mathcal{V}^s$ is an explanation of correctness for $\mathcal{P}^s$ according to MTC. Extracted initial conditions on $\mathcal{P}^s$ are then the weakest (most general) conditions for which $\mathcal{P}^s$ can be executed in any total order consistent with the partial order O^s, according to the same explanation structure

We now develop the declarative specification of the necessary and sufficient conditions under which $\mathcal{V}^s$ will be an explanation of correctness of $\mathcal{P}^s$ according to the MTC. We do this by expressing the conditions under which each element $v^s \in \mathcal{V}^s$ is a validation of $\mathcal{P}^s$. From the semantics of the validations provided in equation 2, these conditions can be stated as the conjunction of codesignation and non-codesignation constraints shown in expression 3 in Figure 5.[6]

Essentially, the validations offer an "interval" view on the explanation – the intervals in which literals have to hold. For our generalization algorithm to mirror standard EBG algorithms, we regroup the validations to reflect what needs to hold for each operator (the "operator" view). The validations grouped for each operator $t^s \in T^s$, describe validations it is required to *support* and *preserve* in the explanation of correctness. The "interval" view in expression 3 can thus be re-expressed in an "operator" view by grouping the validations at each operator, as shown in expression 4 in Figure 5.

Informally, expression 4 states that every operator in the schematized plan should: (*i*) necessarily support the conditions that its counterpart in the specific plan was required to support according to the explanation and (*ii*) necessarily preserve all the conditions that its counterpart in the specific plan was required to preserve. In particular, we can define the *e-conditions* (for relevant effect conditions) of an operator as the set of *validations* it is required to *support* in the explanation of correctness (equation 5 in Figure 5), and *p-conditions* (for preservable conditions) of an operator as the set of *validations* it is required to *protect* (equation 6). Using equations 5 and 6, we can now rewrite expression 4 as expression 7.

Expression 7 is then the declarative specification of the necessary and sufficient conditions under which the schematized plan $\mathcal{P}^s$ is correct according to MTC, given the same explanation structure $\mathcal{V}^s$. $\mathcal{P}^s$ can be used in any initial state S that satisfies the conditions shown in

[4]This assumes that the transitive closure of the partial ordering relations among plan steps is available (for an n step plan, this can be computed in $O(n^3)$ time), thereby allowing the checks on ordering relations during explanation construction to be done in constant time.

[5]We shall use the superscript "s" to distinguish entities corresponding to the schematized plan.

[6]Since there is a direct correspondence between $\mathcal{V}^s$ and $\mathcal{V}$, and O^s is isomorphic to O, for each $v^s : \langle e^s, t'^s, C^s, t''^s \rangle \in \mathcal{V}^s$, we already have $t'^s \prec t''^s$ (see equation 2)

$$\bigwedge_{\forall v^s:\langle e^s,t'^s,C^s,t''^s\rangle \in \mathcal{V}^s} \left[\Box(e^s \approx C^s) \bigwedge \forall t^s \in T^s \; s.t. \; \begin{array}{l}\Diamond(t'^s \prec t^s \prec t''^s),\\ \forall d^s \in delete(t^s)\; \Box(d^s \not\approx C^s)\end{array}\right] \tag{3}$$

$$\bigwedge_{\forall t^s \in T^s} \left[\begin{array}{l} \underbrace{\forall v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in \mathcal{V}^s \; s.t. t'^s = t^s}_{e-conditions(t^s)}, \; \Box(C^s \approx e^s) \wedge \\ \underbrace{\forall v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in \mathcal{V}^s \; s.t. \; \Diamond(t'^s \prec t^s \prec t''^s)}_{p-conditions(t^s)} \; \forall d^s \in delete(t^s)\; \Box(d^s \not\approx C^s) \end{array}\right] \tag{4}$$

$$e - conditions(t^s) \;=\; \{v^s \mid v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in \mathcal{V}^s \; s.t. t'^s = t^s\} \tag{5}$$

$$p - conditions(t^s) \;=\; \{v^s \mid v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in \mathcal{V}^s \; s.t. \; \Diamond(t'^s \prec t^s \prec t''^s)\} \tag{6}$$

$$\bigwedge_{\forall t^s \in T^s} \left[\begin{array}{l} \forall v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in e - conditions(t^s),\; \Box(C^s \approx e^s) \wedge \\ \forall v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in p - conditions(t^s)\; \forall d^s \in delete(t^s)\; \Box(d^s \not\approx C^s) \end{array}\right] \tag{7}$$

Figure 5: Derivation of the generalization algorithm (see text)

Algorithm EXP-GEN $(\mathcal{P}^s : \langle T^s, O^s\rangle, \mathcal{V}^s)$
Initialize: Let β be a null substitution and γ be *True*
foreach $t^s \in T^s$ **do**
 foreach $v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in e - conditions(t^s)$ **do**
 Let β' be the substitution under which $\Box(e^s \approx C^s)$
 $\beta \leftarrow \beta \circ \beta'$
 foreach $v^s : \langle e^s,t'^s,C^s,t''^s\rangle \in p - conditions(t^s)$ **do**
 Let γ' be the condition under which
 $\forall d^s \in delete(t^s)\; \Box(d^s \not\approx C^s)$
 $\gamma \leftarrow \gamma \wedge \gamma'$
Substitute β into all the literals of $\mathcal{P}^s$ and $\mathcal{V}^s$
Weakest preconditions $\leftarrow effects(t_I^s) \wedge \gamma \circ \beta$

Figure 6: Generalization Algorithm

expression 7. For such states, the plan is guaranteed to succeed in *all* of its total orderings. Furthermore, note that expression 7 computes exactly those conditions that are *required* (by MTC) to make $\mathcal{V}^s$ an explanation of correctness of $\mathcal{P}^s$. In this sense, the computed conditions are the weakest preconditions (modulo the given explanation) for $\mathcal{P}^s$

The algorithm EXP-GEN shown in Figure 6 implements expression 7 procedurally in a straightforward manner. The algorithm makes one pass through the plan, visiting each operator, computing the codesignation and non-codesignation constraints imposed on the generalized plan. The codesignation constraints are maintained as substitutions in β, and the non-codesignation constraints are maintained as disequality constraints on the variables in γ. At the end of the generalization phase, the substitution list β is applied to all the literals in the schematized plan $\mathcal{P}^s$ and its explanation structure $\mathcal{V}^s$. Finally, the equality and disequality constraints imposed by β and γ respectively are conjoined with the initial state specification[7] of the generalized plan to get the weakest preconditions for the generalized plan.

Complexity: The generalization algorithm runs in polynomial time. In particular, the *e-conditions* and *p-conditions* of all the operators in $\mathcal{P}^s$, as required by the algorithm, can be precomputed in $O(|T^s||\mathcal{V}^s|)$ or $O(n^2)$ time (where n is the number of operators of the plan), and the propositional unification required to compute β' and γ' itself can be done in polynomial time.

3.1 Example

Let us now follow the generalization of the schematized plan for 4BSP by the algorithm EXP-GEN. Following the definitions in equations 5 and 6, and the schematized validations in Section 3, the *e-conditions* and *p-conditions* of the operators in the schematized plan can be computed as:

$\mathbf{e-conditions}(t_1^s) : v_7^s : \langle On(x,y), t_1^s, On(p,q), t_G^s\rangle$

$\mathbf{e-conditions}(t_2^s) : v_8^s : \langle On(z,w), t_2^s, On(r,s), t_G^s\rangle$

$\mathbf{p-conditions}(t_1^s) : v_8^s : \langle On(z,w), t_2^s, On(r,s), t_G^s\rangle$
$v_5^s : \langle Clear(k), t_I^s, Clear(z), t_2^s\rangle$
$v_6^s : \langle Clear(l), t_I^s, Clear(w), t_2^s\rangle$
$v_4^s : \langle On(k,Table), t_I^s, On(z,Table), t_2^s\rangle$

$\mathbf{p-conditions}(t_2^s) : v_7^s : \langle On(x,y), t_1^s, On(p,q), t_G^s\rangle$
$v_2^s : \langle Clear(i), t_I^s, Clear(x), t_1^s\rangle$
$v_3^s : \langle Clear(j), t_I^s, Clear(y), t_1^s\rangle$
$v_1^s : \langle On(i,Table), t_I^s, On(x,Table), t_1^s\rangle$

$\mathbf{e-conditions}(t_I^s) : v_5^s : \langle Clear(k), t_I^s, Clear(z), t_2^s\rangle$
$v_6^s : \langle Clear(l), t_I^s, Clear(w), t_2^s\rangle$
$v_4^s : \langle On(k,Table), t_I^s, On(z,Table), t_G^2\rangle$
$v_2^s : \langle Clear(i), t_I^s, Clear(x), t_1^s\rangle$
$v_3^s : \langle Clear(j), t_I^s, Clear(y), t_1^s\rangle$
$v_1^s : \langle On(i,Table), t_I^s, On(x,Table), t_1^s\rangle$

Recall that $e-conditions$ of an operator t describe those literals which t supports, and $p-conditions$ are those literals it is required to preserve (these would include the preconditions and useful effects of other operators parallel to t; for example, among the four $p-condition$ validations of t_1^s, the first one corresponds to preserving the required effect of t_2^s and the other three correspond to preserving the preconditions of t_2^s). Note that by definition, t_G^s will have no *e-conditions* or *p-conditions*, and t_I^s will only have *e-conditions* (since no plan operator can precede t_I or follow t_G).

[7] the literals in the *e-conditions* of t_I, to be precise

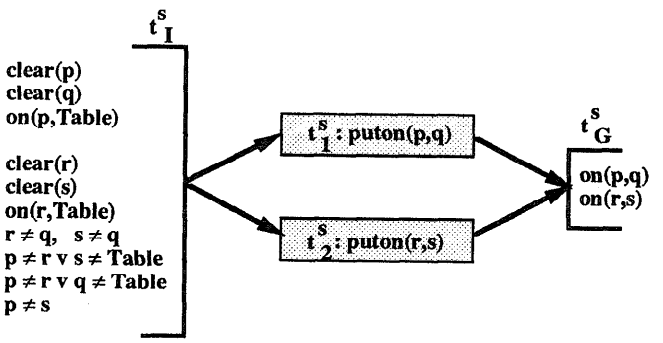

Figure 7: Generalized Plan for 4BSP

The EXP-GEN algorithm computes β'_1 for t^s_1 : $Puton(x,y)$ by unifying $On(x,y)$ and $On(p,q)$. Thus at this point, β'_1 (and therefore β) is $((x\,p)(y\,q))$. Next, γ'_1 for t^s_1 : $Puton(x,y)$ is computed by ensuring that its delete literals $on(x,Table) \wedge clear(y)$ do not unify with the literals of its *p-conditions*. Thus γ'_1 can be computed as $[neq(x,z) \vee neq(Table,w)] \wedge neq(y,z) \wedge neq(y,w)$.

Similar processing for t^s_2 : $Puton(z,w)$ yields β'_2 as $((z\,r)(w\,s))$, and γ'_2 as $[neq(z,x) \vee neq(Table,y)] \wedge neq(w,x) \wedge neq(w,y)$. Finally, the processing for t^s_I yields β'_3 as $((i\,x)(j\,y)(k\,z)(l\,w))$ (there are no *p-conditions* for t^s_I and so γ'_3 is trivially *True*).

The resultant global substitution β is thus $\beta'_1 \circ \beta'_2 \circ \beta'_3$, or:

$$\beta = ((i\,p)(x\,p)(j\,q)(y\,q)(k\,r)(z\,r)(l\,s)(w\,s))$$

Similarly the global non-codesignation constraints on variables γ is computed by conjoining γ'_1, γ'_2 and γ'_3 as:

$$\gamma = [neq(x,z) \vee neq(Table,w)] \wedge neq(y,z) \wedge neq(y,w) \wedge [neq(z,x) \vee neq(Table,y)] \wedge neq(w,x)$$

Figure 7 shows the generalized plan (computed by substituting β into the literals of schematized 4BSP plan shown in Figure 4), and its weakest preconditions, computed by conjoining $\gamma \circ \beta$ with the effects of t^s_I in the plan. In particular, we have:

$$\gamma \circ \beta = [neq(p,r) \vee neq(Table,s)] \wedge neq(q,r) \wedge neq(q,s) \wedge [neq(r,p) \vee neq(Table,q)] \wedge neq(s,p)$$

Notice that the weakest preconditions rightly prohibit the use of this plan in a situation where the goal is $On(A,B) \wedge On(B,C)$, because they explicitly prohibit codesignation of q and r, and p and s (see $\gamma \circ \beta$). Thus, the algorithm avoids the overgeneralization discussed in Section 1.

4 Related Work

Our algorithms directly correspond to the EBG explanation and generalization steps, but work on specialized explanations of correctness tailored to plans, rather than arbitrary proofs. It is possible to use the standard EBG algorithm [12] itself for this purpose, by proving (explaining) correctness of a plan directly from first order situation calculus. The advantage of dealing with specialized explanations is that they often can be produced much more efficiently. In particular, we have seen that explanations of correctness (validations) based on MTC (which states soundness/completeness theorem for a version of situation calculus) can be generated in polynomial time (Section 2). In contrast, generating proofs in full situation calculus is undecidable. In addition, by starting with a provably sound and complete truth criterion and deriving the EBG algorithms directly from that, we obviate the need to carry out a separate formal proof of correctness of the algorithms (e.g. [1]).

There are interesting similarities between our computation of generalized protection violations, and that performed by other plan generalization methods (although some plan generalization methods such as [12] and [14] omitted this). STRIPS' generalized macro-operators [5] handle protection violations by unifying delete lists with the literals in a "lifted," or generalized, triangle table, adding non-codesignation constraints. Minton [11] specified protection violations as a meta-level axiom to be used as part of a proof of correctness of plans by EBG. Goal regression [15] computes protection violations by unifying delete lists with regressed conditions. As noted earlier, none of these deal with partially ordered plans. In comparison, we provide a systematic way of doing this analysis for a class of partially ordered plans, with the help of MTC.

The work reported here is also related the "operator order generalization" algorithms such as [13] and [3]. After generalizing a totally ordered plan using the EGGS algorithm [14], these algorithms further generalize the structure of the plan by removing any redundant orderings. In contrast, we start with a correct partially ordered plan (generated by any classical partial-order planner–such as NONLIN [17]), and compute a generalized partially ordered *macro-operator* which represents the weakest conditions under which the generalized plan can be successfully executed. However, the methodology that we developed here can be extended to allow a broader class of generalizations. In fact, by relaxing the notion of "following the example" (Section 3), we can systematically develop a spectrum of generalization algorithms to allow a variety of structural generalizations (Section 5). In this sense, the methodology presented here could be used to systematically characterize the EBG, order generalization and structure generalization [13] algorithms as different points on a spectrum of generalizations (with varying amounts of emphasis on plan-time vs. generalization-time analysis).

5 Concluding Remarks

This paper addresses the problem of generalizing partially ordered plans – a class of plans which have been extensively investigated in the planning literature. We have developed the formal notion of explanation of correctness for partially ordered plans based on the MTC, and used this definition to derive a declarative specification for explanation-based generalization of partially ordered plans. The generalized plans that are produced by procedurally interpreting this declarative specification are guaranteed to execute successfully in any total order consistent with the partial ordering, in any situation matching the weakest preconditions computed by the generalization.

While the development here provided a separate algorithm to compute the explanation of correctness of a

partially ordered plan, often the explanation construction phase can be integrated with the generation of the plan. In particular, most partial-order planners (such as NONLIN [17], SIPE [18]) keep track of the "validation structure" of the plan being developed to aid in plan generation. Furthermore, in [8, 9], we discuss a plan modification framework that utilizes the validation structure-based explanation of correctness of the plan to guide incremental plan modification. This opens up the possibility of integrating generalization with modification [6]: In situations where a generalized plan is only partially applicable, it can be modified appropriately.

The generalization phase in Section 3 is "conservative" in that it follows the explanation structure of the given plan in a very strict sense. It only allows the generalized plan to have a validation structure that is isomorphic to the validation structure of the specific plan. It does not find alternative validation structures of the plan, or make any modifications/extensions to the structure of the plan. However, within the framework developed in this paper, it is possible to relax the generalization phase in a variety of ways, giving rise to a spectrum of generalizations of the given plan. These include allowing new ordering constraints, or allowing new planning steps during the generalization phase. This would essentially involve using the truth criterion as a way of establishing the truth of a proposition (synthesis) [2] rather than merely as a way of testing for correctness.

The algorithms presented in this paper compute the weakest conditions under which *all* topological sorts of a partially ordered plan can be guaranteed to execute successfully. Sometimes, it may be useful to compute weakest conditions under which at least some topological sort of the plan can possibly execute. Generalization algorithms for this case can be developed in a very similar fashion. In particular, the truth criterion for guaranteeing the possible truth of a proposition is specified by reversing the modalities in the equation 2 (substitute "$\Diamond$" for "$\Box$" and vice versa) [2]. Using this truth criterion, we can then develop similar polynomial time EBG algorithms for possible correctness [10].

The truth criterion that we have used in this paper assumes an operator representation that is free of conditional effects and conditional preconditions. We believe that similar methodology can also be used for more expressive operator representations. It must however be noted that increased expressiveness of the operators necessitates truth criteria that are correspondingly expensive to interpret, increasing the cost of EBG.

Finally, an issue that is orthogonal to the problem of correctly generalizing a given partially ordered plan, is the *utility* of generalizing and using partially ordered plans as compared to totally ordered plans. Some points in favor of the former include compactness of representation, and a greater flexibility of modification and reuse (*cf* [7]). In particular, by integrating our generalization process with a plan modification framework such as PRIAR [8], we can store and reuse generalized plans in a variety of new situations (including those in which they are only partially applicable). This presents interesting tradeoffs compared to a MACROP [5] based reuse mechanism (for example, given the flexibility of reuse, exact match retrieval may no longer be critical). Our future plans include implementing the generalization algorithms on top of PRIAR modification framework [6, 8, 9], and carrying out empirical experimentation to get a clearer understanding of these tradeoffs.

Acknowledgements: Discussions with Steve Minton, Mark Drummond and John Bresina helped significantly in clarifying our presentation. We also acknowledge helpful suggestions from Peter Friedland, Pat Langley, Amy Lansky, Rich Levinson and Prasad Tadepalli.

References

[1] N. Bhatnagar. A correctness proof of explanation-based generalization as resolution theorem provi ng. In *Proc. AAAI Spring Symp. on Exp. Based Learning*, 1988.

[2] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[3] S.A. Chien. Using and refining simplifications: Explanation-based learning of plans in intractable domains. In *Proc. 11th IJCAI*, 1989.

[4] G. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145 – 176, 1986.

[5] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[6] S. Kambhamapti. Supporting plan reuse. In S. Minton and P. Langley, editors, *Learning Methods for Planning*. Morgan Kaufmann, Palo Alto, Stanford, 1991 (in press).

[7] S. Kambhampati. Mapping and retrieval during plan reuse: A validation-structure based approach. In *Proc. 8th AAAI*, 1990.

[8] S. Kambhampati. A theory of plan modification. In *Proc. 8th AAAI*, 1990.

[9] S. Kambhampati and J.A. Hendler. A validation structure based theory of plan modification and reuse. Tech. Rep. STAN-CS-90-1312, Comp. Sci., Stanford Univ., 1990. (To appear in *Artificial Intelligence*).

[10] S. Kambhampati and S. Kedar. Explanation-based generalization of partially ordered plans. Technical report, Stanford Univ., 1991.

[11] S. Minton and J.G. Carbonell. Strategies for learning search control rules: An explanation-based approach. In *Proc. 10th IJCAI*, 1987.

[12] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based learning: A unifying view. *Machine Learning*, 1(1):47 – 80, 1986.

[13] R. J. Mooney. Generalizing the order of operators in macro-operators. In *Proc. 5th Intl. Conf. on Machine Learning*, 1988.

[14] R. J. Mooney and S. W. Bennett. A domain independent explanation-based generalizer. In *Proc. 5th AAAI*, 1986.

[15] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishers, 1980.

[16] E. D. Sacerdoti. *A Structure for Plans and Behavior*. American Elsevier, New York, NY, 1977.

[17] A. Tate. Generating project networks. In *Proc. 5th IJCAI*, 1977.

[18] D. E. Wilkins. Domain independent planning: Representation and plan generation. *Artificial Intelligence*, 22:269–301, 1984.