Characterizing Multi-Contributor Causal Structures for Planning

Subbarao Kambhampati
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406
email: rao@asuvax.asu.edu

Abstract

Explicit causal structure representations have been widely used in classical planning systems to guide a variety of aspects of planning, including plan generation, modification and generalization. For the most part, these representations were limited to single-contributor causal structures. Although widely used, single-contributor causal structures have several limitations in handling partially ordered and partially instantiated plans. The foremost among these is that they are incapable of exploiting redundancy in the plan. In this paper, we explore multi-contributor causal structures as a way of overcoming these limitations. We will provide a general formulation for multi-contributor causal links, and explore the properties of several special classes of this formulation. We will also discuss their applications in plan generation, modification and generalization.

1 Introduction

Representation and use of causal structure of the plans has been a long-standing theme in classical planning. Some of the specific (though similar) causal structure representations that have been proposed include *Protection intervals* [19, 20] [23], *goal structure* [21], *plan rationale* [24], *causal links* [12], and *validations* [9]. Such representations have been used extensively in plan generation to keep track of interactions and to systematize the search process [21][12]; in plan recognition to capture the plan rationale [14]; in replanning, plan modification and abstraction planning to justify individual planning decisions and to retract unjustified ones [9][5][25]; in plan debugging to characterize the plan failures [19][17]; and in plan generalization to explain plan correctness and use that explanation as a basis for generalization [10][2].

Most of the previous work modeled plan causal structures in terms of single-contributor causal links, which capture the dependencies between a consumer step requiring a prerequisite, and a *single* producer step which contributes that prerequisite. Such single contributor causal structures suffer from several disadvantages in capturing the causal structure of partially ordered partially instantiated plans. The biggest problem is their inability to deal with, and exploit redundant contributors. Consider, for example, the prerequisite R of the step fin in the plan MP shown in Figure 1 (the add and delete lists literals of each step are shown above the step with + and $-$ signs, while the prerequisites of the step are in parentheses under the step). The steps $w0$, $s5$ and $w2$ can all independently provide R to fin. This type of redundancy in the causal structure of plan can be gainfully utilized to make interaction resolution more efficient during planning, as well as to facilitate more efficient plan modification and replanning strategies. Unfortunately however, we cannot do this in a planner using single-contributor causal structures. In the example above, a planner using single contributor causal links will have to commit to one of $w0$, $s5$ or $w2$ as the contributor.

Apart from failing to exploit the redundancy in the causal structure, such a premature commitment may also lead to unnecessary backtracking in the planning process (especially in planners that employ goal protection). Especially troublesome in this respect are planners such as [12][18][13] that widen the definition of protection violation to include both those nodes that intervene and delete the protected condition and those which intervene and merely reassert the protected condition. This stronger definition of un-threatened and un-usurped causal links is introduced to avoid redundancy in the planner's search process (i.e., to make sure that the planner will not visit two plans with overlapping completions). Our empirical studies with [18] in a variant of blocks world (that induces multiple redundant contributors for some pre-requisites) demonstrate that not only do such planners fail to exploit redundant contributors, but their performance actually *worsens* in the presence of redundant contributors [11][1].

A way of avoiding this overcommitment, while still keeping down the redundancy in the planner's search space (See Section 5), is to model causal links as dependencies between a prerequisite and several contributor nodes such that one of those contributor nodes is guaranteed to provide the prerequisite for every execution sequence of the plan.

Although the idea of multi-contributor causal links has been first introduced in Tate's NONLIN [21] (see Section 5), there has not been any systematic study of their properties.

[1]Note that use of abstraction techniques, such as precondition abstraction[16, 25] can mitigate this to some extent by postponing achievement of preconditions which are likely to have multiple contributors (and thus are easily achievable).

The primary goal of this paper is to develop a formal characterization of multi-contributor causal structures for planning. We will present a general formulation for multi-contributor causal links, and within that formulation explore several sub-classes with useful properties. We will then describe the advantages of using these causal structures in plan generation, modification and generalization. Specifically, we will describe a planning algorithm based on them, and will develop a justification framework based on multi-contributor causal structures that can form the basis for plan modification and generalization.

Guide to the paper: The rest of the paper is organized as follows: Section 1.1 introduces some preliminary terminology that is used throughout the rest of the paper. Section 2 provides the general formulation of multi-contributor causal links, and characterizes the correctness of a plan with respect to this formulation. Section 3 defines and explores a variety of special classes of of multi-contributor validation structures with useful properties. Section 4 discusses the applications of these causal structure representations in planning. Section 4.1 describes a planning algorithm that can exploit the multi-contributor causal structures described in preceding sections. Section 4.2 develops the notion of justifying individual planning decisions with respect to causal structure, which is useful in plan modification and generalization. Section 5 discusses the relations with past research, and Section 6 summarizes the paper and discusses some outstanding issues. Readers eager to understand the applications of multi-contributor causal structures before looking at the detailed formulations may want to go directly to Section 4.1 after Section 2 on their first reading.

1.1 Terminology

In this paper, we will be using the following terminology for partially ordered partially instantiated plans (widely referred to also as nonlinear plans): A partially ordered partially instantiated plan $\mathcal{P}$ is represented as a 3-tuple $\langle T, O, \Pi \rangle$, where: T is the set of actions in the plan, and O is a partial ordering relation over T, and Π is a set of codesignation (binding) and non-codesignation (prohibited bindings) constraints on the variables in $\mathcal{P}$. 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. Actions are represented by instantiated STRIPS-style operators with *Add*, *Delete* and *Precondition* lists, all of which are conjunctions of functionless first order literals.

A partially ordered partially instantiated plan corresponds to a set of totally ordered totally instantiated execution sequences, called *completions*, each corresponding to a fully instantiated topological sort of the plan that is consistent with the binding constraints Π and ordering constraints O. Such a plan is considered correct if and only if each of its completions is an executable STRIPS plan capable of transforming the world to a state where all the prerequisites of t_G are satisfied. For plans involving strips type operators without conditional or deductive effects, the TWEAK truth criterion [1] provides a set of necessary and sufficient conditions under which every completion of a partially ordered partially instantiated plan will be correct in the above sense.

2 Formulating Multi-contributor causal links

In formulating multi-contributor validation structures we are faced with a choice as to how conservative our formulation should be. In particular, given a prerequisite p of a step w which needs contributors, we can be very conservative and decide to include those and only those contributors that are absolutely necessary to support p in all the completions of the plan. On the other hand, we can also be anti-conservative and include in our formulation any step which can *possibly* contribute the prerequisite p (i.e., the step does not follow w, and it has an effect that can possibly codesignate with p). Our formulation below strikes a middle ground:

Definition 1 (Causal Link/Protection Interval) *A causal link (or protection interval) of a plan $\mathcal{P}$ is a 3-tuple $\langle \mathcal{S}, p, w \rangle$ (or $\mathcal{S} \xrightarrow{p} w$ in McAllester's notation [12]) where (i) w is an individual step and $\mathcal{S}$ is a set of steps belonging to plan $\mathcal{P}$, (ii) w requires a condition p (iii) $\forall s \in \mathcal{S}\ \Box(s \prec w)$ and (iv) for each step $s \in \mathcal{S}$, there exists an effect $e \in effects(s)$ such that $\Box(e \approx p) \in \Pi$.*[2]

Definition 1.1 (Validation) *Corresponding to each causal link $\langle \mathcal{S}, p, w \rangle$, we associate the notion of a validation $\langle \mathcal{SE}, p, w \rangle$, where $\mathcal{SE}$ is a set of tuples: $\{\langle s, e \rangle | s \in \mathcal{S}, e \in effects(s), \Box(e \approx p)\}$.*

Consider again the plan MP shown in Figure 1. From our definitions, $\langle \{w0, w2\}, R, fin \rangle$ is a causal link for this plan, and $\langle \{\langle w0, R \rangle, \langle w2, R \rangle\}, R, fin \rangle$ is the corresponding validation. Thus the only difference between a causal link and the validation is that the latter explicitly lists the effects of the individual contributors that actually supply the condition being supported by the causal link. In view of this tight correspondence, we will use the words **causal link, protection interval** and **validation** interchangeably in the rest of the paper.

Definition 2 (Violated Causal Links) *A causal link $\langle \mathcal{S}, p, w \rangle$ of a plan $\mathcal{P}$ is said to be violated if there are completions of $\mathcal{P}$ where none of the contributors of $\mathcal{S}$ can successfully provide the effect p to w, without some other step s' of the plan possibly intervening and* **deleting** *p. A causal link is said to* hold *if and only if it is not violated.*

From the definition, we have the following necessary and sufficient conditions to ensure that a validation is not violated:

Property 2.1 *A validation $\langle \mathcal{S}, p, w \rangle$ is not violated if and only if $\forall s' \in \mathcal{P}$ s.t. $\neg q \in effects(s')$ and $\Diamond(q \approx p)$, either $w \prec s'$ or $\exists s \in \mathcal{S}$ such that $s' \prec s$*

[2]Note that requiring that all the contributors of a causal link must provide an effect that will necessarily codesignate with the condition being supported, is in general stronger than the constraints imposed by TWEAK truth criterion [1]. Consider, for example, the case of a plan where a step w needs a condition $P(v)$, $s1$ has an effect $P(x)$, s_2 has an effect $P(z)$, $s_1 \prec w$, $s_2 \prec w$, s_1 and s_2 are unordered w.r.t. to each other. We also have two other steps s_g and s_d such that $s_g \prec s_1$, $s_d \prec s_2$, and s_g negates $P(u)$, and s_d negates $P(y)$. Now, if we use a multi-contributor validation link to support $P(v)$ at w, then the weakest constraints we need are $[x \approx v \wedge z \approx v]$. This is stronger than what is required by the modal truth criterion for guaranteeing the truth of $P(u)$ at w, which is $[x \approx v \wedge z \approx v] \vee [x \approx v \wedge (y \not\approx v \vee z \approx v)] \vee [z \approx v \wedge (u \not\approx v \vee x \approx v)]$.

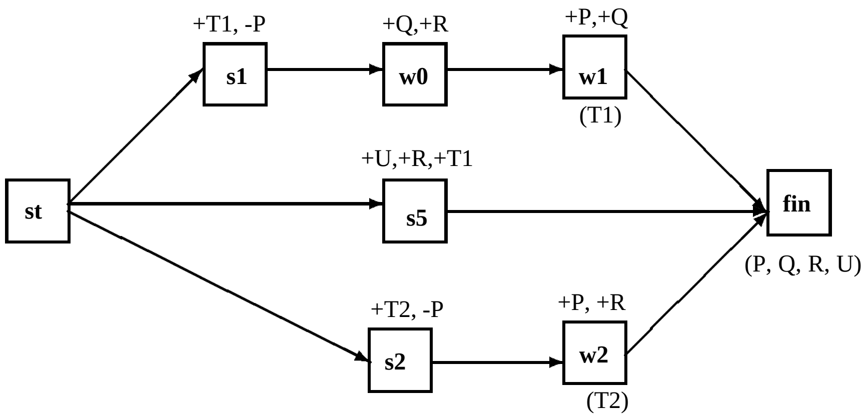

Figure 1: MP: A partially ordered plan

$\langle\{s1\}, T1, w1\rangle$	$\langle\{s2\}, T2, w2\rangle$
$\langle\{w1, w2\}, P, fin\rangle$	$\langle\{w0, w1\}, Q, fin\rangle$
$\langle\{w2, w0\}, R, fin\rangle$	$\langle\{s5\}, U, fin\rangle$

Figure 2: A multi-contributor validation structure for the plan MP

The above formulation explicitly admits the possibility that p is contributed by different members of $\mathcal{S}$ for different completions of the plan. This facilitates a more flexible way of accommodating plans that are correct by the *white-knight* clause of TWEAK truth criterion [1]. Consider, for example, the prerequisite P of step fin in the plan MP shown in Figure 1. Although both $w1$ and $w2$ provide P, neither of them can do it in all the completions of MP. The standard way of accommodating this type of situations within single-contributor causal structure representations is to utilize the white-knight declobbering clause, and to consider one of $w1$ and $w2$ as the node contributing P for fin and the other node as the white-knight. The choice here, of which node to refer to as the contributor and which to refer to as the white knight, is completely arbitrary. It hides the fact that $w1$ and $w2$ are in fact complementary contributors for different completions. Multi-contributor causal links obviate this problem. In particular, the causal link $\langle\{w1, w2\}, P, fin\rangle$ can support p for fin according to our formulation.

2.1 Causal Structure and Plan Correctness

Using the definitions above, we can now characterize the correctness of the plan with respect to a set of causal links in a straightforward fashion.

Definition 3 (Plan Correctness w.r.t. Validation Structure) *A plan $\mathcal{P}$ is said to be correct with respect to a set of causal links (validations) $\mathcal{V}$, if and only if (i) For each precondition p of each step w of the plan, there exists a causal link $\langle \mathcal{S}, p', w\rangle \in \mathcal{V}$ such that $(p' \approx p) \in \Pi$ and (ii) None of the causal links of $\mathcal{V}$ are violated. The set $\mathcal{V}$ is called a causal structure or the validation structure for the plan $\mathcal{P}$.*

It can be easily be shown that if a partially ordered partially instantiated plan $\mathcal{P}$ is correct by the above definition, then every completion of it will constitute a executable solution for the corresponding planning problem. In particular, we can show that in any completion of the plan, corresponding to each prerequisite p of each step w, there will necessarily be a step s of the plan that provides the prerequisite p to w without any intervening steps of the plan denying it.[3] Figure 2 shows a set of valid (multi-contributor) causal links under which the plan MP shown in Figure 1 is correct. The converse of the above is however not necessarily true. In particular, a plan may be incorrect according to a validation structure $\mathcal{V}$ and may still be correct according to the modal truth criterion.

3 Specializations of Multi-contributor causal structures

The formulation of multi-contributor validation structure developed in the previous section is too general in that it does not differentiate between causal links with irrelevant contributors (i.e., contributors which will always be superseded by other contributors), redundant contributors (i.e., contributors which can be removed without affecting the correctness of the plan), and irredundant contributors (i.e., contributors which cannot be removed without affecting the correctness of the plan). In the following we tighten this formulation by imposing some additional restrictions to derive special classes of multi-contributor causal structures with interesting properties.

3.1 Irredundant Validation Structures

The most stringent restriction on multi-contributor causal links would be to stipulate that every contributor in the

[3]To see this, consider a completion T' of the plan. Let v be the last step before w in T' such that v deletes p. By definition 3, there must be a causal link $\langle \mathcal{S}, p', w\rangle$ in the plan such that $p' = p$. Further more, since the $\langle \mathcal{S}, p', w\rangle$ is not violated, by definition 2 there must be a step $s \in \mathcal{S}$ such that $v \prec s$. Thus, s can contribute p to w without any intervening nodes denying it.

causal link is *required* in some strong sense. This is captured by the notion of irredundant validation structures defined below:

Definition 4 (Irredundant Contributors) *A contributor s of an initially un-violated causal link $\langle \mathcal{S}, p, w\rangle$ is said to be irredundant if removing s from $\mathcal{S}$ will violate the validation (according to Definition 2).*

Property 4.1 *Given a validation $\langle \mathcal{S}, p, w\rangle$, a contributor $s \in \mathcal{S}$ is irredundant if and only if* **either** *$\mathcal{S}$ is a singleton set,* **or** *there exists some step n in the plan such that: (i) $(n \prec s) \wedge (w \not\prec n)$ and (ii) $\neg d \in effects(n)$ such that $\Diamond(d \approx p)$ and (iii) $\forall s_i \in \mathcal{S}$ if $(s_i \not= s)$ then $(n \not\prec s_i)$ (In other words, s is the only step in the plan capable of foiling the interaction caused by n.)*

Definition 4.1 (Irredundant Validation Structure) *A validation structure $\mathcal{V}$ is said to be* **irredundant** *for a plan $\mathcal{P}$ if and only if each contributor of every validation belonging to $\mathcal{V}$ is* irredundant.

Property 4.2 *If $\mathcal{V}$ is an irredundant validation structure for a plan $\mathcal{P}$, then a validation $\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ will have multiple contributors (i.e. $\mathcal{S}$ is not a singleton) if and only if proving the truth of p at w in the plan $\mathcal{P}$ would require using the white-knight clause in* TWEAK*'s truth criterion [1].*

Thus, *irredundancy* generalizes the single contributor validation structure (cf [9][12]) just enough to allow plans that have white-knight interactions. In particular, irredundant validation structures admit multiple contributors in causal links only when they are absolutely necessary. Since they *do not* admit any form of redundant contributors, a planning algorithm that constructs plans with irredundant validation structures can suffer from the some of the same drawbacks as the planner's using single contributor causal links in terms of premature commitment to contributors (see Section 1) [12].

3.2 Relevant Validation Structures

Next, we will look at the notion of *relevant validation structure* which is more general than irredundant validation structures --- in that it allows redundant contributors, but more specific than the formulation in Section 2 as it stipulates that each contributor should be an effective contributor in at least one completion of the partially ordered plan. To define this formally, we need the notion of *effective contributor in a completion*:

Definition 5 (Effective Contributor in a completion) *Let $\mathcal{CP}$ be a completion of plan $\mathcal{P}$. For every step w in $\mathcal{P}$, and a precondition p of that step, p holds at w in the completion $\mathcal{CP}$, if and only if there exists a step s such that s is the last step before w in $\mathcal{CP}$ which asserts p without any other step between s and w asserting or deleting p. If such a step s exists (and it is guaranteed to exist, if the plan $\mathcal{P}$ is correct) then that step is called the* **effective (or relevant) contributor** *of p to w in the completion $\mathcal{CP}$.*

Definition 6 (Irrelevant Contributors) *A step s is said to be an* **irrelevant** *contributor of a validation $\langle \mathcal{S}, p, w\rangle$ of a plan $\mathcal{P}$ if and only if $s \in \mathcal{S}$ and s cannot be an* effective contributor *of p to w in any completion of the plan $\mathcal{P}$.*

Property 6.1 *Given a validation $\langle \mathcal{S}, p, w\rangle$, a step s belonging to $\mathcal{S}$ is an* **irrelevant** *contributor if there exists a step u in the plan $\mathcal{P}$, such that $\Box(s \prec u \prec w)$ and either $e \in effects(u) \wedge \Box(e \approx p)$ or $\neg e \in effects(u) \wedge \Box(e \approx p)$ (i.e., u comes after s and either reasserts or deletes p).*

In the validation structure shown in Figure 2, $w0$ is an irrelevant contributor for the causal link $\langle \{w0, w1\}, Q, fin\rangle$. This is because $w1$ follows $w0$ in every completion of the plan and thus the latter can never be the last step to assert Q before fin in any completion of MP.

Definition 6.1 (Relevant Validation Structure) *If a plan $\mathcal{P}$ has a validation structure $\mathcal{V}$ such that no causal link in $\mathcal{V}$ has any irrelevant contributors, then $\mathcal{V}$ is said to be a* relevant *validation structure for $\mathcal{P}$.*

Property 6.2 *All the contributors of a relevant validation are necessarily unordered with respect to each other*

This property can be derived as a corollary of property 6.1, by selecting the step u from $\mathcal{S}$. The validation structure in Figure 2 is not a relevant validation structure for MP. However, it can be made relevant by removing $w0$ from the validation $\langle \{w0, w1\}, Q, fin\rangle$.

It can be easily seen that *irredundancy* of validation structure is a stronger condition than *relevance*. Consequently, the properties 6.1 and 6.2 also hold for irredundant validation structures.

3.3 Exhaustive Validation Structures

We will now look at a specialization of relevant validation structures called *exhaustive validation structures*. These have the useful property that for every prerequisite in the plan, the validation structure will account for every step in the plan that can possibly be a contributor of that prerequisite.

Definition 7 (Exhaustive Validation) *A validation $\langle \mathcal{S}, p, w\rangle$ of a plan $\mathcal{P}$ is said to be* **exhaustive** *if the validation is relevant, and for every completion $\mathcal{CP}$ of the plan $\mathcal{P}$, the effective contributor of p to w in $\mathcal{CP}$ belongs to $\mathcal{S}$.*

From the definition, we have the following necessary and sufficient conditions for exhaustiveness of a validation.

Property 7.1 *A validation $\langle \mathcal{S}, p, w\rangle$ of a plan $\mathcal{P}$ is exhaustive if and only if it is relevant, and $\forall n \in \mathcal{P}$, if n has an effect e such that $\Diamond(e \approx p)$, then it must* **either** *be the case that $n \in \mathcal{S}$* **or** *it must be the case that $w \prec n$* **or** *it must be the case that $\exists s \in \mathcal{S}$ such that $n \prec s$.*

In Figure 2, the validation $\langle \{w2, w0\}, R, fin\rangle$ is not a exhaustive validation since $s5$ also provides R to Fin and it is not included in the validation. (It can also be seen that except for this validation and $\langle \{s1\}, T1, w1\rangle$, the rest of the validations are exhaustive).

Definition 7.1 (Exhaustive Validation Structure) *A validation structure $\mathcal{V}$ is said to be* exhaustive *with respect to a plan $\mathcal{P}$ if and only if all the causal links in $\mathcal{V}$ are exhaustive.*

Unlike irredundance and relevance, exhaustiveness imposes additional constraints on a plan. Given a plan $\mathcal{P}$, it is not always possible to find a exhaustive validation structure for $\mathcal{P}$

without adding additional constraints to it.[4] Consider the example of the validation $\langle\{w2, w0\}, R, fin\rangle$ in Figure 2. We can make this validation exhaustive by simply adding $s5$ as an additional contributor (making it $\langle\{w2, w0, s5\}, R, fin\rangle$). However, the validation $\langle\{s1\}, T1, w1\rangle$ cannot be made exhaustive without also adding an ordering constraint $s5 \prec w1$ to MP. In other words, we cannot find a exhaustive validation structure for MP as it stands.

In spite of their restrictiveness, exhaustive validation structures are useful because of their uniqueness: A plan may have many different relevant or irredundant validation structures, but it can only have *at most one* exhaustive validation structure:

Property 7.2 *If $\mathcal{V}$ and $\mathcal{V}'$ are two exhaustive validation structures for a plan $\mathcal{P}$, then it must be the case that $\mathcal{V} = \mathcal{V}'$.*

An interesting corollary of this property is that in the single-contributor case, exhaustive validation structures can be used to define a equivalence class relation between nonlinear plans and their completions. In particular, given a validation structure $\mathcal{V}$, and a totally ordered totally instantiated plan $\mathcal{CP}$, we can uniquely determine the nonlinear abstraction of $\mathcal{CP}$ to which $\mathcal{V}$ is an exhaustive validation structure. This tight correspondence is used in McAllester's planner [12] to ensure the systematicity of the planning algorithm.

4 Applications of Multi-contributor Causal Links

4.1 Planning with Multi-Contributor Causal Links

In this section, we will describe how multi-contributor validation structures described in the previous sections can be used in planning. In particular, we will provide a McAllester style planning algorithm [12] that is capable of generating plans with *relevant* and *exhaustive*[5] validation structures. One of the advantages of this algorithm is that it enables the planner to maintain multiple parallel contributors without committing to any one of them prematurely (thereby avoiding the limitations of the single-contributor validation structures, described in Section 1).

Before we describe the planning algorithm, we need to specify the notion of conflict/interaction used by the planner, and the termination condition for the planner. Since we want to maintain exhaustive (and thus relevant) validation structure, from property 7.1, we see that a validation $\langle\mathcal{S}, p, w\rangle$ is threatened by any step $v \notin \mathcal{V}$ that either asserts or deletes p.[6] Thus, we define the notion of *threat* of a validation as follows:

Definition 8 (Threat for a Validation) *A step v is called a* threat *to a causal link $\langle\mathcal{S}, p, w\rangle$ if v is a step other than w, and $v \notin \mathcal{S}$ and either $q \in effects(v)$ or $\neg q \in effects(v)$ such that $\Diamond(q \approx p)$.*

Although this definition of threat is stronger than that used in most classical planners, we will see that the use of multi-contributor causal structures makes sure that it does not cause any excessive backtracking (as was the case in [12, 13]). Using this definition, we can now develop the termination condition for the planner. In particular, we define a complete plan as follows:

Definition 9 (Complete Plans) *A plan $\mathcal{P} : \langle T, O, \Pi\rangle$ is called* complete *with respect to a validation structure $\mathcal{V}$ if the following conditions hold*

- *If w is a step in $\mathcal{P}$, and w has a prerequisite p', then $\mathcal{V}$ contains some causal link of the form $\langle\mathcal{S}, p, w\rangle$, such that $(p' \approx p) \in \Pi$.*
- *If $\mathcal{P}$ contains a causal link $\langle\mathcal{S}, p, w\rangle$, and a step v that is a threat to the causal link $\langle\mathcal{S}, p, w\rangle$, then O contains either $v \succ w$ or $v \prec s$ for some $s \in \mathcal{S}$.*
- *For every causal link $\langle\mathcal{S}, p, w\rangle \in \mathcal{V}$, the members of $\mathcal{S}$ are unordered with respect to each other.*

It can be easily seen that completeness is a specialization of the notion of correctness (definition 3). In particular, any plan that is complete by the above definition is also *correct* by definition 3, has a *relevant* validation structure by property 6.2, and has an *exhaustive* validation structure by definition 7.1.

The completeness condition should be seen as imposing a particular *bias* (c.f. [15]) on the search space of partially ordered plans defined by Chapman's truth criterion [1]. It is easy to show that this bias does not affect the completeness of the planner (in the sense that for every planning problem solvable by TWEAK, there exists a plan for that problem which satisfies the termination condition specified in definition 9).[7]

Figure 3 shows a McAllester style planning algorithm [12] that generates plans that are complete by this definition. To simplify discussion, we only show the procedure for generating ground partially ordered plans. The procedure for generating partially instantiated and partially ordered plans can be obtained in a straightforward fashion using the lifting transformation discussed in [12]. The important difference between our algorithm and the one proposed in [12] is that ours maintains multiple possible contributors for each pre-requisite in a systematic fashion. Because of this, its treatment of unsafe causal links is different.

When the procedure finds an unsafe causal link $\langle\mathcal{S}, p, w\rangle$ (in Step 3), it has three choices: The threat can either be promoted to come after w (3(a)), or be demoted to come before *one* of the the steps in $\mathcal{S}$ (3(b)). In addition, if the threat is adding the prerequisite, then the procedure also has the choice of merging the threat into the contributor set $\mathcal{S}$.[8] This is what is done in step 3(c). Although all

[4]For the special case of single contributor causal structures, the plans produced by NONLIN [21] do not in general have exhaustive validation structures, while those produced by McAllester's planner [12], and Minton et al's UA planner [13] have exhaustive validation structures.

[5]See Section 5 for a rationale for maintaining exhaustive validation structures

[6]Note that by Definition 2, a step asserting p will not violate the validation $\langle\mathcal{S}, p, w\rangle$. Thus the notion of threat is stronger than that of violation.

[7]However, because of the restrictive nature of exhaustive validation structures, it is possible that there exist plans which are correct according to the TWEAK truth criterion, but do not satisfy the completeness criterion.

[8]By this stage in the procedure, we know that $\nexists s \in \mathcal{S}\ s.t.\ v \prec s$. From this it can easily be shown that no node in $\mathcal{S}$ necessarily follows v. Thus, v can be included in the contributor list, as v is the last such contributor

The Procedure FindCompletion($\mathcal{P}$, $\mathcal{V}$, c)

1. If the partially ordered plan $\mathcal{P}$ is order inconsistent, or the total cost of the steps in $\mathcal{P}$ is greater than c, then fail.
2. If $\mathcal{P}$ is *complete* (by definition 9), then return $\langle \mathcal{P}, \mathcal{V} \rangle$.
3. *Threatened Causal Links:* If there is a causal link $\langle \mathcal{S}, p, w \rangle$ in $\mathcal{V}$ and a **+ve** or **-ve** threat v to this link in the plan $\mathcal{P}$, such that $\mathcal{P}$ does not contain either $(v \succ w)$ or $(v \prec s)$ for some $s \in \mathcal{S}$, then nondeterministically return one of the following:
   (a) **FindCompletion**($\mathcal{P}$ + $(w \prec v)$, **MakeRelevant**($\mathcal{V}, (w \prec v)$), c)
   (b) if v deletes p, non-deterministically choose some s from $\mathcal{S}$ and return
   FindCompletion($\mathcal{P}$ + $(v \prec s)$, **MakeRelevant**($\mathcal{V}$, $(v \prec w)$), c)
   (c) If v adds p, then return
   FindCompletion($\mathcal{P}$ + $(v \prec w)$, **MakeRelevant**($\mathcal{V} - \langle \mathcal{S}, p, w \rangle + \langle \mathcal{S} + v, p, w \rangle$, $(v \prec w)$), c)
4. There must now exist some open prerequisite (a step w and a prerequisite p of w, such that there is no causal link of the form $\langle \mathcal{S}, p, w \rangle$ in $\mathcal{V}$). In this case, nondeterministically do one of the following:
   (a) Let s be (nondeterministically) some step in $\mathcal{P}$ that adds p. Return the plan
   FindCompletion($\mathcal{P}$ + $(s \prec w)$, **MakeRelevant**($\mathcal{V} + \langle \{s\}, p, w \rangle$, $(s \prec w)$), c)
   (b) Select (nondeterministically) an operator O_i from the allowed set of operations such that O_i adds p. Create a new step s in $\mathcal{P}$ corresponding to the operator O_i. Then return the plan
   FindCompletion($\mathcal{P}$ + $\langle \{s\}, p, w \rangle$ + $(s \prec w)$, c)

The Procedure MakeRelevant($\mathcal{V}$, $(s_1 \prec s_2)$)

```
foreach $\langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ do
    If $prec(s_1) \cap \mathcal{S} \neq \emptyset$ and $\mathcal{S} \cap succ(s_2) \neq \emptyset$
      then $\mathcal{V} \leftarrow \mathcal{V} - \langle \mathcal{S}, p, w \rangle + \langle \mathcal{S} \setminus prec(s_1), p, w \rangle$
od
Return $\mathcal{V}$.
```

Figure 3: MP : A procedure for generating ground plans with exhaustive and relevant multi-contributor causal structures

3. *Threatened Causal Links:* If there is a causal link $\langle \mathcal{S}, p, w \rangle$ in $\mathcal{V}$ and a **-ve threat** v to this link in the plan $\mathcal{P}$, such that $\mathcal{P}$ does not contain either $(v \succ w)$ or $(v \prec s)$ for some $s \in \mathcal{S}$, then nondeterministically return one of the following:
   (a) **FindCompletion**($\mathcal{P}$ + $(w \prec v)$, **MakeRelevant**($\mathcal{V}, (w \prec v)$), c)
   (b) Non-deterministically choose some s from $\mathcal{S}$ and return
   FindCompletion($\mathcal{P}$ + $(v \prec s)$, **MakeRelevant**($\mathcal{V}$, $(v \prec w)$), c)
   (c) Non-deterministically choose some step v' (if any) in the plan such that $(v \prec v')$ and v' *adds* p, and return
   FindCompletion($\mathcal{P}$ + $(v' \prec w)$, **MakeRelevant**($\mathcal{V} - \langle \mathcal{S}, p, w \rangle + \langle \mathcal{S} + v', p, w \rangle$, $(v' \prec w)$), c)

Figure 4: Treatment of threatened causal links in MP-I

the three choices are applicable to a threat that adds the prerequisite, it is obviously heuristically advantageous to prefer the alternative 3(c).

Finally, in step 4.a. when the procedure establishes an open prerequisite with the help of existing steps of the plan, it simply selects one of the possible contributors nondeterministically. The contributor set will grow appropriately at a later point, when threats are discovered and merged.[9]

Any time we introduce ordering constraints between two existing steps of the plan (as is done in steps 3(a), 3(b), 3(c) and 4(a)), it is possible to make some contributors of some causal links irrelevant, thereby affecting the relevance of the validation structure. We use the sub-routine called **MakeRelevant** to maintain the relevance of the plan validation structure all through the planning cycle.[10] This procedure takes the existing causal structure and the newly introduced ordering relation as inputs, removes any irrelevant contributors from the causal links, and returns the resultant causal structure (which is guaranteed to be relevant with respect to the current plan). The algorithm uses the functions $prec(s)$ and $succ(s)$. The former comprises of s and all the nodes that necessarily precede s, while the latter comprises of s and all its the nodes that necessarily follow s. The idea behind this procedure is the following: When we add an ordering between two steps s_1 and s_2, we essentially have to be worried about the situation where we have a validation $\langle \mathcal{S}, p, w \rangle$ such that $\mathcal{S}$ contains both s_1 or some of its predecessors, and s_2 or some of its successors. When this happens, the members of $\mathcal{S}$ are no longer unordered with respect to each other, and thus $\langle \mathcal{S}, p, w \rangle$ will no longer be relevant. We can however make it relevant by removing s_1 and its predecessors from $\mathcal{S}$.

Note that since the procedure removes only irrelevant contributors, for any step s that is removed from any validation $\langle \mathcal{S}, p, w \rangle$, there will be a step $s' \in \mathcal{S}$ such that $s \prec s'$. Further, the test at the beginning of step 3 in the procedure ignores any threat that necessarily precedes any of the current contributors of the validation. Thus, once a step has been removed from a causal link by **MakeRelevant**, it will *never be* reintroduced into that link in that branch of the search process (in other words, there will not be any looping behavior because of removal of irrelevant contributors).

When the procedure **FindCompletion** terminates successfully, it returns a plan $\mathcal{P}$ and a validation structure $\mathcal{V}$. It can be easily shown that $\mathcal{P}$ will be correct, and that $\mathcal{V}$ will be a relevant and exhaustive validation structure for $\mathcal{P}$. The advantage of this algorithm is that it exploits redundancy in the plan causal structure, and avoids excessive backtracking. At the same time, by maintaining exhaustive (and relevant) validation structures, it keeps the redundancy in the planner's search space low. We are currently conducting empirical experimentation with an implementation algorithm to characterize the computational benefits it can offer.

There is one other point to be noted about the treatment of unsafe causal links in algorithm in Figure 3. In step 3 of the algorithm, when the threat v is deleting the condition being supported by the causal link $\langle \mathcal{S}, p, w \rangle$, one possibility is to see if there is a way of eliminating this interaction by *removing* some non-irredundant contributor from $\mathcal{S}$ (cf. [21]). If this is possible, then we can eliminate the interaction without putting any further constraints on the plan (see Section 5). Such a step, however, introduces two complications: (i) Allowing removal of contributors as a way of resolving interactions introduces *retraction* into the truth criterion of the planner. This is contrary to the philosophy of most first principles generative planners (such as TWEAK [1]) which have *monotonic refinement* truth criteria. (ii) Even if the planner were to allow retraction as a part of its truth criterion, retraction in general introduces superfluous constraints into the plan, affecting its minimality. Failing to remove these superfluous constraints in turn can lead to loss of completeness[11]. To deal with this, we need a way of keeping track of the dependencies between the planning decisions. For both these reasons, in the algorithm shown in Figure 3, we avoided doing interaction resolution through retraction. We will however discuss how retraction can be accomplished without loss of completeness in the next section, where we will describe a validation-structure based justification framework that can be used for this purpose.

4.2 Justifying Plans with Multi-contributor Causal Structures

Causal structure representations have been shown to be very valuable in guiding plan modification [9][5], and generalization [10]. From a first principles perspective, the only augmentation that is needed to enable a generative planner to modify a given plan to solve a new problem, or to generalize a given plan by removing unnecessary constraints, is the ability to *retract* some constraints on the plan. Retracting decisions from a plan typically may introduce inconsistencies and/or superfluities into the plan which need to be handled appropriately.

Causal structures can help in this process by serving as a basis to justify individual planning constraints (steps, ordering constraints and binding constraints) of a plan. In particular, we can justify causal links in terms of the over all goals of the plan, and then justify the other constraints in the

[9]In implementing this procedure, it is possible to reduce some of this later interaction resolution by setting $\mathcal{S}$ initially to the set of steps that are the last incoming contributors of p in each branch. (Such steps are called the critical PV nodes in NONLIN terminology [21]).

[10]An alternative to maintaining a relevant validation structure all through the planning cycle is to wait until a correct plan is generated and then check for irrelevant contributors. However, this latter alternative can produce spurious interactions involving irrelevant contributors during planning and bog down the planner.

[11]Note however, that the removal of irrelevant contributors in the procedure **MakeRelevant** in Figure 3 **does not** introduce any superfluous constraints into the plan. To see why, consider the case where a step s_1 has just been removed from a causal link $\langle \mathcal{S}, p, w \rangle$. Then it must have been the case that some node s_2 has just been added to $\mathcal{S}$ such that $s_1 \prec s_2$. The question we need to answer is whether there are any constraints that we imposed when making s_1 a contributor of p that we can remove now. Obviously, the constraint $s_1 \prec w$ cannot be removed since $s_1 \prec s_2 \prec w$. The only other constraints may have been some orderings imposed on the plan to allow s_1 to contribute p to w. Suppose, at the time we decided to have s_1 as a contributor, we had threat v such that v deletes p. Obviously, if it was the case that $s_1 \prec v \prec w$, then we could not have avoided that interaction anyway. The only other possibility is that v was unordered with respect to either s_1 or w. In either of these cases, v will still be a threat to the new validation, and thus any constraints added to the plan to deal with v will still remain justified.

plan in terms of the causal links they support [9][25][4]. Such a justification structure allows the planner to locate parts of the plan that become superfluous whenever a particular retraction occurs.

When we allow multi-contributor causal structures, however, the mere fact that a step is supporting a validation does not necessarily mean that it is justified. This is because the step could be a redundant or irrelevant contributor of the validation and consequently removing it will not lead to incorrectness of the plan. In the following, we will develop a framework for justifying a plan in terms of a multi-contributor causal structure. We will start by justifying causal links in terms of plan correctness, and then go on to justify individual constraints in terms of the causal links.

Definition 10 (Causal Link Justification) *Given a plan $\mathcal{P}$ with a validation structure $\mathcal{V}$, a causal link $\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ is justified if and only if it ultimately supports a prerequisite of the goal step t_G. That is either $w = t_G$ and $\exists q \in prerequisites(t_G)$ such that $(q \approx p)$ or there exists another justified causal link $\langle \mathcal{S}', p', w'\rangle$ such that $w \in \mathcal{S}'$.*

Definition 11 (Step Justification) *A step s of a plan $\mathcal{P}$ is said to be* justified *with respect to a relevant validation structure $\mathcal{V}$ if and only if there exists a validation $\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ such that $s \in \mathcal{S}$. In particular, the set of such validations for which s is a contributor is defined as its justification.* [12]

A step s is said to be strongly justified if s is an irredundant contributor for at least one validation.

A step s is said to be weakly justified if s is not strongly justified, and every co-contributor of s in any causal link that s participates in is strongly justified.

A justified step that is neither strongly justified nor weakly justified is said to be conditionally justified.

In the validation structure of Figure 2 for the plan MP, the steps $w1$, $w2$, $s1$, $s2$ and $s5$ are all strongly justified. But, the step $w0$ is not strongly justified since $w0$ is not a irredundant contributor with respect to any of the two validations in which it participates. Additionally since the co-contributors of $w0$ are all strongly justified, $w0$ is weakly justified.

The idea of conditional justification applies to steps that are redundant contributors to every causal link to which they contribute. No strongly justified steps can be removed from the plan without making the plan incorrect (by definition 3). All unjustified steps and weakly justified steps can be removed simultaneously without affecting the correctness of the plan (in the later case, some redundancy in the validation structure is eliminated). Any one conditionally justified step can be removed without affecting the correctness of the plan. Removing more than one conditionally justified step simultaneously can make the plan incorrect. This is because the step could be a redundant contributor of a causal link along with another step. Each contributor by itself can be removed without violating the causal link, but not both at the same time.

Similar justifications can also be developed for ordering constraints and binding constraints:

Definition 12 (Ordering justification) *An ordering relation $(s_1 \prec s_2) \in O$ of a plan $\mathcal{P} : \langle T, O, \Pi\rangle$ is said to be justified with respect to a validation structure $\mathcal{V}$ of the plan, if and only if one of the following conditions is true: (i) $\exists\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ such that $s_1 \in \mathcal{S} \wedge s_2 = w$* **or** *(ii) $\exists\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ such that $s_1 = w$ and $\neg q \in effects(s_2)$ and $\Diamond(q \approx p)$* **or** *(iii) $\exists\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ such that $s_2 \in \mathcal{S}$ and $\neg q \in effects(s_1)$ and $\Diamond(q \approx p)$. Additionally we say that the ordering relation $s_1 \prec s_2$ is strongly justified if it is either justified by one of the last two clauses, or if it justified by the first clause and s_1 is an irredundant contributor of $\langle \mathcal{S}, p, w\rangle$.*

Definition 13 (Codesignation justification)
A codesignation constraint $(q \approx p) \in \Pi$ of a plan $\mathcal{P} : \langle T, O, \Pi\rangle$ is justified with respect to a validation structure $\mathcal{V}$ if and only if there exists a causal link $\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$ such that $\exists s \in \mathcal{S}$ and $q \in effects(s)$ (i.e., the validation corresponding to the causal link $\langle \mathcal{S}, p, w\rangle$ (see Definition 1.1) is $\mathcal{SE}$ such that $\langle s, q\rangle \in \mathcal{SE}$).

Further, if s is an irredundant contributor of the validation $\langle \mathcal{S}, p, w\rangle$, then the codesignation constraint is said to be strongly justified.

Definition 13.1 (Separation justification)
A non-codesignation constraint $(q \not\approx p) \in \Pi$ of a plan $\mathcal{P} : \mathcal{P} : \langle T, O, \Pi\rangle$ is justified with respect to a validation structure $\mathcal{V}$ if and only if there exists a causal link $\langle \mathcal{S}, p, w\rangle \in \mathcal{V}$, and a step u in the plan such that $\neg q \in effects(u)$, and $\forall s \in \mathcal{S}\, \Diamond(s \prec u \prec w)$.

Additionally, every justified non-codesignation constraint is also said to be strongly justified.

Finally, using the justifications for individual constraints, we can now discuss the notion of justifying a plan with respect to a validation structure as follows:

Definition 14 (Justification of a plan w.r.t. to a validation structure) *A plan $\mathcal{P} : \langle T, O, \Pi\rangle$ is said to be justified with respect to a validation structure $\mathcal{V}$ if and only if every causal link in $\mathcal{V}$ is justified, and every step $s \in T$, every ordering constraint $o \in O$ and every binding constraint $c \in \Pi$ is justified with respect to $\mathcal{V}$.*

Additionally, it is said to be strongly justified w.r.t. $\mathcal{V}$ if all the steps, ordering constraints and binding constraints are strongly justified.

Justifications like these can be computed for each individual decision in polynomial time or can be maintained incrementally during planning and plan modification (*cf* [9]). These justifications can be used to *retract* superfluous constraints from the plan while preserving the correctness of the plan. In the following we describe two slightly different justification procedures with differing properties:

Justifying a Plan: Justifying a plan is an iterative process. Given a plan $\mathcal{P}$ and a causal structure $\mathcal{V}$, we construct the justified plan $\mathcal{P}'$ by removing all constraints of $\mathcal{P}$ that are unjustified with respect to $\mathcal{V}$. The resultant plan $\mathcal{P}'$ will still be correct with respect to $\mathcal{V}$. $\mathcal{V}$ may however contain some unjustified causal links with respect to $\mathcal{P}'$ as a result of this retraction. If this is the case, then we construct a new validation structure $\mathcal{V}'$ by removing all the unjustified causal links from $\mathcal{V}$. We then repeat the whole process for $\mathcal{P}'$ and $\mathcal{V}'$ (until $\mathcal{P}'$ is justified w.r.t. $\mathcal{V}'$ and vice versa).

Minimizing a Justified Plan: A justified plan is not necessarily the minimal such plan capable of achieving the goals of the problem. In particular, there may be weakly justified

[12]Note: For the special case of single-contributor validation structures, this definition reduces to that of of e-conditions of a step defined in [9][10].

and conditionally justified constraints in the plan. We can minimize a justified plan further by removing such weakly justified constraints. However, every time a conditionally justified constraint is retracted, we need to update the justifications before retracting another one, since removal of one conditionally justified constraint can make another constraint strongly justified.

When justifying a plan, we attempt to keep the intent of the validation structure intact. If for example, the plan was designed to have redundant contributors for some prerequisite (either to increase robustness or ensure exhaustiveness), then justifying a plan will not thwart this intent. Minimization, on the other hand, cares only about the correctness of the plan. If $\mathcal{V}$ is a relevant and exhaustive validation structure for a plan $\mathcal{P}$, and $\mathcal{P}'$ is the result of strongly justifying $\mathcal{P}$ with respect to $\mathcal{V}$. Then $\mathcal{V}$ is not guaranteed to be a relevant or exhaustive validation structure for $\mathcal{P}'$. Both these notions of justifications become equivalent in single contributor validation structures.

The justification framework described in this section can form the basis for plan modification [9] and plan generalization (cf. [10]) procedures, based on multi-contributor validation structures. It can also be used to support retraction of contributors as a way of resolving interactions during planning (as discussed at the end of Section 4.1).

5 Related Work

To our knowledge, NONLIN [21] and its successors are the only previous planners to have used multi-contributor causal links. NONLIN's GOST table, in conjunction with its Q&A procedure, was capable of maintaining multiple redundant contributors for each prerequisite in the plan. NONLIN's method of maintaining the multiple contributors was not complete, however. It would include multiple contributors only when it was achieving the prerequisite for the first time. In this case, it used its Q&A procedure (which is equivalent to TWEAK truth criterion for ground plans) to check for simple establishment. If Q&A returns more than one possible contributor (the so called critical PV-nodes in NONLIN terminology[13]), then all such nodes are included as contributors of the prerequisite. During subsequent planning, additional contributors of that prerequisite may be introduced into the plan, but NONLIN will not increment the contributor set unless there is negative interaction between the effects of some newly introduced node and one of the contributors of the pre-requisite. Thus, the validations in the GOST are not in general guaranteed to be either *relevant* or *exhaustive*.

During interaction resolution, NONLIN exploited the presence of multiple contributors -- when a particular interaction clobbers the desired effect of one of the contributors for a prerequisite, NONLIN would check to see if there were other contributors that are unaffected. If so, NONLIN would simply delete the affected contributors from GOST and continue. The initial implementations of NONLIN did not attempt to re-justify the plan after such a retraction. This may leave the plan with unjustified constraints (thereby affecting the minimality of the plan and the completeness of the planner). O-PLAN [3], a successor of NONLIN, has some provisions to rectify this [4]. The development of justification framework in Section 4.2 provides a systematic basis for doing this. O-PLAN also had a more generalized notion of protection intervals called ''clouds'' [22], which were designed to manage the contributors and terminators of aggregated sets of dependencies. Clouds also allowed O-PLAN to manage multiple contributors all through the planning, by actively keeping track of the ''last incoming contributor'' wavefront.

There are also some interesting relations between this paper and the recent work on systematic nonlinear planning algorithms. In contrast to traditional planning algorithms like NONLIN [21] and TWEAK [1], the planning algorithms described in [12] and [13][14] maintain exhaustive validation structures. As mentioned in Section 3.3, exhaustiveness property provides a tight correspondence between a nonlinear plan and its completions, which is used in these planners to avoid redundancy in the search space. As we pointed out in Section 3.3, maintaining exhaustive validation structures in general forces the planner to make additional (ordering and binding) commitments on the plan. Our empirical experimentation with a systematic planner ([12]) shows that this increased commitment leads to excessive backtracking on the average and thereby adversely affect the planners performance [11].

What we have here is a tradeoff between redundancy in the search space explored by the planner, and the amount of commitment the planner is making. Planners like TWEAK [1] have very low commitment, but may be searching in highly redundant search spaces. Planners like UA [13] and SNLP [12, 18] guarantee systematicity, but impose higher commitment and thus may lead to more backtracking. The tradeoff between non-redundancy in search space and least-commitment will depend to a large extent on the density of solutions in the domain [6]. In particular, if the domain is such that the planner is forced to go through most of its search space before finding a solution, a planner with low redundancy in its search space can be expected to do better than a fully least-committed planner such as TWEAK [1]. On the other hand, when the solution density is not very low, a planner which provides systematicity property through increased commitment may do worse than TWEAK.

The planning algorithm based on multi-contributor causal structures, described in Figure 3 strikes an interesting balance here. In particular, by using multi-contributor validation structures, our planner reduces amount of commitment, while still maintaining exhaustiveness. Compared to TWEAK [1], which does not maintain any type of causal structures, this algorithm still does more commitment and higher backtracking. However, by maintaining exhaustive and relevant validation structures during planning, it also keeps down the redundancy in its search space.

6 Conclusion and Future Directions

Although widely used in classical planning, single-contributor causal structures have several disadvantages in

[13]Critical PV-nodes are essentially the last nodes on each incoming branch which assert the condition, without it being asserted or deleted subsequently in that branch. Thus, none of the initial contributors are all irrelevant. However, subsequent planning may introduce ordering relations among them, making them irrelevant

[14]Although Minton et al's planner does not explicitly keep track of causal structures, the net effect of the *unambiguity restriction* used in their planner seems to be to provide exhaustiveness of causal structure.

dealing with partially ordered partially instantiated plans, which can be overcome by using multi-contributor causal structures. The primary contribution of this paper is a clear characterization of multi-contributor causal structures for classical planning. We provided a general formulation of multi-contributor causal links, and explored a variety of sub-classes of this formulation with interesting properties. We have also discussed applications of these formulations in plan generation, modification and generalization.

There are several issues that remain to be addressed regarding multi-contributor causal structures. Foremost among these is characterizing their effect on the planning performance. In Section 5, we suggested that the algorithm shown in 3 strikes the middle ground between planners such as TWEAK [1] which have very low commitment but have high redundancy in the search space, and planners such as SNLP [12] and UA [13] which have very high commitment but avoid redundancy in the search space. Our next task will be to implement the planning algorithm shown in Figure 3, and conduct empirical experimentation to test this conjecture.

Acknowledgements:

Thanks are due to Austin Tate, who read a draft of this paper and provided many valuable suggestions and pointers; to Mark Drummond who listened to some of these ideas in the early stages and provided useful feedback; to John Bresina, whose comments about validation structure based generalization of plans lead me to look into multi-contributor validation structures seriously in the first place; and to the anonymous reviewers of AAAI-92 and AIPS-92 for their helpful comments.

References

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

[2] S.A. Chien. An Explanation-Based Learning Approach to Incremental Planning. (Ph.D. Dissertation). Available as Technical Report UIUCDCS-R-90-1646, Dept. of Computer Science, University of Illinois, Urbana, IL, 1990.

[3] K. Currie and A. Tate. O-Plan: The Open Planning Architecture. *Artificial Intelligence*, 52:49-86.

[4] L. Daniels. Planning and operations research. In: *Artificial Intelligence: Tools, Techniques, and Applications* (T. O'Shea and M. Eisenstadt (Ed). Harper & Row, New York, 1984.

[5] S. Hanks and D. Weld. Systematic Adaptation for Case-Based Planning. Technical Report 91-10-03, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 1991.

[6] P. Langley. Systematic and Nonsystematic search strategies. Submitted to AAAI-92.

[7] S. Kambhampati. Mapping and retrieval during plan reuse: A validation-structure based approach. In *Proceedings of 8th National Conference on Artificial Intelligence*, August 1990.

[8] S. Kambhampati. A theory of plan modification. In *Proceedings of 8th National Conference on Artificial Intelligence*, August 1990.

[9] S. Kambhampati and J.A. Hendler. A validation structure based theory of plan modification and reuse. *Artificial Intelligence* (*To appear*). (Available as Technical Report STAN-CS-90-1312, Computer Science Department, Stanford University).

[10] S. Kambhampati and S.T. Kedar. Explanation-Based Generalization of Partially Ordered Plans. In *Proc. 9th AAAI*, 1991.

[11] S. Kambhampati. Characterizing Multi-Contributor Causal Structures for Planning. Technical Report, Dept. of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287.

[12] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proc. 9th AAAI*, 1991.

[13] S. Minton, J. Bresina and M. Drummond. Commitment Strategies in Planning: A Comparative Analysis. In *Proc. 12th IJCAI*, 1991.

[14] M.E. Pollack. A Model of Plan Inference that Distinguishes Between the Beliefs of Actors and Observers. In *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, Morgan Kaufmann, Palo Alto, 1987.

[15] P.S. Rosenbloom, S. Lee and A. Unruh. Bias in Planning and Explanation-Based Learning. In *Machine Learning Methods for Planning and Scheduling*. S. Minton (Ed.). Morgan Kaufmann (*in press*)

[16] E. Sacerdoti. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence*, 5(2), 1975.

[17] R. Simmons. A Theory of Debugging. In *Proceedings of 7th AAAI*, St. Paul, MN, 1988.

[18] S. Soderland A. Barrett and D. Weld. The effect if step-order representations on planning. Technical Report 91-05-06, Department of Computer Science and Engineering, University of Washington, Seattle, WA, June 1991.

[19] G.J. Sussman. *A Computer Model of Skill Acquisition*. American Elsevier, New York, 1975

[20] A. Tate. Interacting Goals and Their Use. In *Proceedings of IJCAI-75*, pages 215-218, Tbilisi, USSR, 1975.

[21] A. Tate. Generating Project Networks. In *Proceedings of IJCAI-77*, pages 888--893, Boston, MA, 1977.

[22] A. Tate. Goal Structure, Holding Periods and ''Clouds.'' In *Proceedings of 1986 Timberline workshop on Reasoning about Actions and Plans*, pages 267-277, Morgan Kaufmann, 1986.

[23] R. Waldinger. Achieving several goals simultaneously. In *Machine Intelligence 8*, Ellis Horwood Limited, Chichester, 1977.

[24] D. Wilkins Domain Independent Planning: Representation and Plan Generation. *Artificial Intelligence*, 22:3, 1984.

[25] Q. Yang and J.D. Tenenberg. ABTWEAK: Abstracting a nonlinear, least-commitment planner. In *Proceedings of 8th AAAI*, 1990.