Multi-Contributor Causal Structures for Planning: A Formalization and Evaluation

Subbarao Kambhampati[*]
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406

Running Head: Multi-contributor Causal Structures for Planning

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. Specifically they are (i) incapable of exploiting redundancy in the plan causal structure and (ii) force premature commitment to individual contributors thereby causing unnecessary backtracking. In this paper, we study 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 then describe two planning algorithms -- MP and $\mathrm{MP\text{-}I}$ -- that use multi-contributor causal links to organize their search for plans. We will describe empirical studies demonstrating the advantages of MP and $\mathrm{MP\text{-}I}$ over planners that use single contributor causal structures, and argue that they strike a more favorable balance in the tradeoff between search space redundancy and premature commitment to contributors. Finally, we will present a framework for justifying plans with respect to multi-contributor causal structures and describe its applications in plan modification and generalization.

[*]A preliminary version of this paper was presented at the 1st Intl. Conference on AI Planning Systems, College Park, MD, 20742, June 1992 [11]. Author's e-mail address is *rao@asuvax.asu.edu*

1 Introduction

Representation and use of causal structure of plans is ubiquitous in classical planning. Some of the specific (though similar) causal structure representations that have been proposed include *Protection intervals* [26, 27, 30, 19], *goal structure* [28], *plan rationale* [31], *causal links* [16], and *validations* [8]. The original motivation for these representations was to keep track of interactions and to organize and systematize the planner's search process [26, 28, 16]. However, they also found wide-spread use in plan recognition to capture the plan rationale [21]; in replanning, plan modification and abstraction planning to justify individual planning decisions and to retract unjustified ones [8, 6, 32]; in plan debugging to characterize the plan failures [26, 24]; and in plan generalization to explain plan correctness and use that explanation as a basis for generalization [9, 2].

Most of the previous work modeled plan causal structures in terms of single-contributor causal links, which maintain causal links as the dependencies between a consumer step requiring a prerequisite, and a *single* producer step which contributes that prerequisite. There are two important disadvantages with such single-contributor causal structures:

- Single contributor causal links are unable to deal with, and exploit redundancy in the plan causal structure. Consider, for example, the prerequisite R of the step fin in the plan MPEX 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 etcitefficient 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, such a planner will be forced to commit to one of $w0$, $s5$ or $w2$ as the contributor.

- Second and perhaps more importantly, single-contributor causal structures force a planner into premature commitment to specific contributors during planning, which in turn may lead to unnecessary backtracking and wasted effort. Consider the example shown in Figure 2.a where a planner using single contributor causal structure has committed to the step $s1$ to provide the condition P to w. Suppose that at a later stage in planning, the planner reaches the plan shown in Figure 2.b where there are two steps $s2$ and $s3$ such that $s1 \prec s2 \prec s3$, $s1$ deletes P, and $s3$ adds P back. This forces the planner to backtrack over its previous premature commitment to $s1$ as the contributor of P to w. Such backtracking becomes especially deleterious in domains where there are high-frequency conditions (such

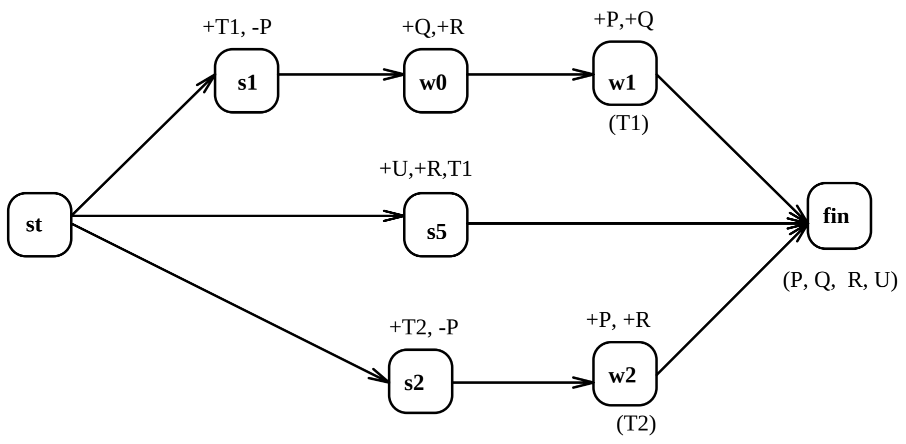


Figure 1: MPEX : A partially ordered plan

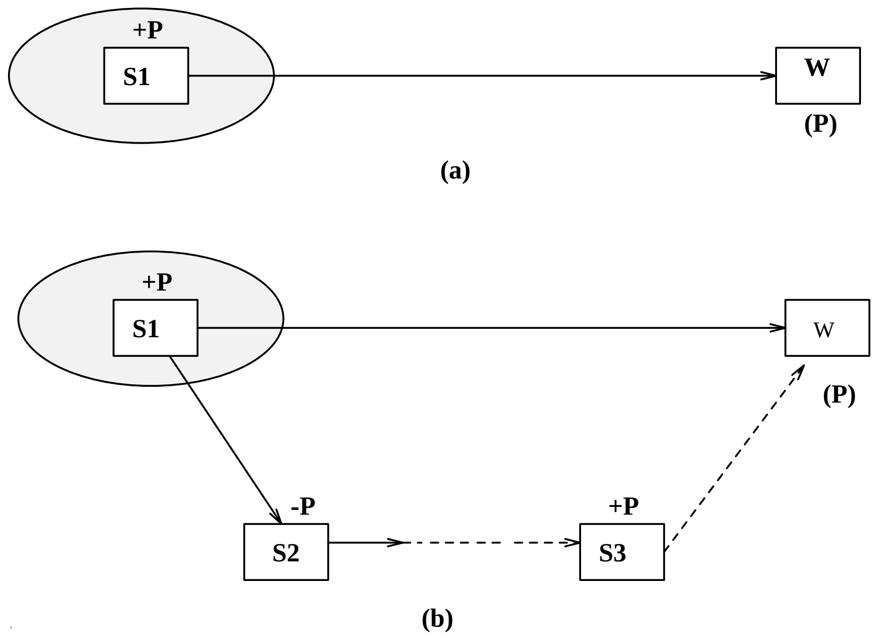


Figure 2: Example showing premature commitment in planners using single contributor causal structures

as *handempty* in the blocks world), that are supplied as well as deleted by multiple actions in the domain.[1]

A way of avoiding this overcommitment, while still organizing the search around causal links and keeping down the redundancy in the planner's search space (See Section 7), 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. Such multi-contributor causal structures can also provide more flexibility during other phases of planning such as generalization and modification.

Although the idea of multi-contributor causal links has been first introduced in Tate's NONLIN [28] (see Section 7), there has neither been a systematic study of their properties nor an evaluation of their effectiveness. The objectives of our research are thus to:

- Provide a systematic formalization of multi-contributor causal structures
- Develop planning algorithms based on multi-contributor causal links and evaluate their effectiveness
- Explore the utility of multi-contributor causal structures in plan modification and reuse

In keeping with these objectives, 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 planning algorithms based on multi-contributor causal links, and empirically evaluate their effectiveness. We will also develop a justification framework based on multi-contributor causal structures that can form the basis for plan modification and generalization.

1.1 Guide to the paper

The rest of the paper is organized as follows: Section 2 describes the action representation and planning terminology that is used throughout the rest of the paper. Section 3 provides the general formulation of multi-contributor causal links, and characterizes the correctness of a plan

[1]Planners such as SNLP [16, 25] and UA [17] 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, are especially troublesome in this respect. (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). For a discussion of the performance tradeoffs involved in redundancy and the extent of over-commitment, see Section 7.1).

with respect to this formulation. Section 4 defines and explores a variety of special classes of multi-contributor validation structures with useful properties. Section 5 describes two planning algorithms, MP and $\mathrm{MP\text{-}I}$ respectively, that use multi-contributor causal structures to generate plans; and discusses their soundness and completeness. Section 6 discusses the properties of these planners, and provides an empirical evaluation of their advantages over planners using single contributor causal structures. Section 7 discusses the relations with past research. Of special focus here will be the recent work on systematic nonlinear planners (c.f. [16, 17]). We will argue that MP and $\mathrm{MP\text{-}I}$ strike a more favorable balance in the tradeoff between the redundancy in the search space and the extent of (over)commitment, than is provided by single contributor systematic planners such as SNLP. Section 8 summarizes the contributions of the paper. In Appendix A, we investigate the application of multi-contributor causal links to plan modification and generalization. Specifically, we develop the notion of justifying individual planning decisions with respect to causal structure. Appendix B contains proofs for all the propositions stated in the paper.

2 Preliminaries

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 4-tuple $\langle T, O, \mathcal{B}, \mathcal{ST}\rangle$, where: T is the set of step names in the plan, and $\mathcal{ST}$, called symbol table, is a mapping from step names to actions in the domain.[2] O is a partial ordering relation over T, and $\mathcal{B}$ is a set of codesignation (binding) and non-codesignation (prohibited bindings) constraints on the variables in $\mathcal{P}$. T contains two distinguished steps 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. An ordering constraint, such as "s is ordered to precede s'", is expressed as $s \prec s'$. The notation $s_1 \parallel s_2$ is used to express that the partial plan does not order s_1 and s_2 (i.e., neither $s_1 \prec s_2$ nor $s_2 \prec s_2$). Similarly the expressions $x \approx y$ and $x \not\approx y$ are used to represent codesignation and non-codesignation constraints between two variables x and y. We shall assume that the actions are represented by instantiated STRIPS-style operators with *Add*, *Delete* and *Precondition* lists, all of which are conjunctions of functionless first order literals (c.f. [1]). For simplicity, we shall omit explicit mention of symbol table in this paper, and write plans as 3-tuples: $\mathcal{P} : \langle T, O, \mathcal{B}\rangle$. When we talk about a step, it is assumed that we are also talking about the action (operator) signified by that step.

By definition, a partially ordered partially instantiated plan $\mathcal{P}$ corresponds to a set of totally

[2]As McAllester [16] points out, this indirection through symbol table is necessitated by the fact that a plan can have multiple instances of a specific domain action serving different roles.

ordered totally instantiated execution sequences, called *completions*. Each completion $\mathcal{CP}$ of $\mathcal{P}$ corresponds to a complete instantiation of variables $\mathcal{B}_{\mathcal{CP}}$ and a total ordering on the steps of the plan, $\prec_{\mathcal{CP}}$ such that $\prec_{\mathcal{CP}}$ is consistent with the ordering constraints in O (i.e., matches a topological sort), and $\mathcal{B}_{\mathcal{CP}}$ is consistent with the binding constraints in $\mathcal{B}$. For example, the following equivalence holds between $\prec$ and $\prec_{\mathcal{CP}}$ relations:

$$s_1 \prec s_2 \;\equiv\; \forall_{\mathcal{CP} \in Completions(\mathcal{P})}\; s_1 \prec_{\mathcal{CP}} s_2$$

Definition 2.1 (Correctness of Partially Ordered Plans) *A partially ordered and partially instantiated plan $\mathcal{P}$ is said to be* correct *if and only if each of its completions is a correct plan for achieving the preconditions of t_G starting from a state of the world where the effects of t_I are true.*

A completion $\mathcal{CP}$ of $\mathcal{P}$ is correct if it is executable, and the execution results in a state of the world where all the preconditions of t_G are satisfied. Under the assumptions of classical planning, this will be the case if and only if every precondition of every step in $\mathcal{CP}$ (including the dummy step t_G) is guaranteed to be true in the state of the world preceding the execution of that step.

For plans involving the class of action representations used in classical planning (including ADL [19], STRIPS [18] and TWEAK [1] representations), the following causality theorem holds (c.f. [19]):

> A condition p is true in the state preceding the execution of a step w in a completion $\mathcal{CP}$, if and only if there exists some step s that precedes w in $\mathcal{CP}$, such that s has an effect p and no step s' coming between s and w delete p.

Since correctness is intertwined with contribution of conditions between plan steps, we will find it useful to talk in terms of particular steps contributing certain conditions to other steps. Specifically, we shall define:

Definition 2.2 (Effective and Feasible Contributors in a completion) *Let $\mathcal{CP}$ be a completion of plan $\mathcal{P}$. A step s is said to be a* **feasible contributor** *of p to w in $\mathcal{CP}$, if s has an effect p, it precedes w in $\mathcal{CP}$, and no step between s and w deletes p. Additionally, s is called an* **effective** *contributor of p to w in $\mathcal{CP}$, if there is no other step between s and w which is a feasible contributor of p. In other words, s is the last step before w which adds p without any intervening step adding or deleting it.*

From the definition of plan correctness above, it is easy to see the following:

Proposition 2.2.1 *Given a correct plan $\mathcal{P}$ and a completion $\mathcal{CP}$ of $\mathcal{P}$, every precondition p of every step w will have at least one feasible contributor, and a unique effective contributor of p to w in $\mathcal{CP}$.*

We will use the modal necessity and possibility operators ''$\Box$'' and ''$\Diamond$'' to relate correctness of certain propositions -- statements such as ''step s precedes step w'', and ''literal p codesignates with literal q'' -- in a partially ordered plan $\mathcal{P}$ and its completions $\mathcal{CP}$. In particular, a proposition f is said to be necessarily (possibly true) in $\mathcal{P}$, if f is true in all (at least one) completion of $\mathcal{P}$. They are denoted, respectively, as $\Box f$ and $\Diamond f$. [3] For example, the following equivalences hold between modal statements about step orderings, the ordering constraints on the plan, and the ordering constraints in the individual plan completions:

$$\Box(s_1 \prec s_2) \equiv \forall_{\mathcal{CP} \in Completions(\mathcal{P})} (s_1 \prec_{\mathcal{CP}} s_2) \equiv s_1 \prec s_2 \ ;$$
$$\Diamond(s_1 \prec s_2) \equiv \exists_{\mathcal{CP} \in Completions(\mathcal{P})} (s_1 \prec_{\mathcal{CP}} s_2) \equiv \neg\Box(s_2 \prec s_1) \equiv (s_1 \parallel s_2) \vee (s_1 \prec s_2)$$

A planner that conducts its search in the space of partially ordered (partially instantiated) plans is called a partial order planner. The completeness of a partial order planner is defined as follows:

Definition 2.3 (Completeness of a Partial Order Planner) *A partial order planner is said to be complete if and only if every minimal ground operator sequence that is a correct solution to a given planning problem is also a completion of at least one of the (partially ordered) plans returned by the planner.*

Note that the completeness of a partial order planner is defined in terms of the ability to find all correct ground totally ordered plans. This is reasonable since the aim of partial order planning is to improve efficiency, rather than to *generate* partially ordered plans.

3 Formulating Multi-contributor causal links

In formulating multi-contributor validation structures we are faced with a choice as to how least-committed our formulation ought to 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 least-committed 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 (see also Section 7.1):

Definition 3.1 (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 [16]) where (i) w is an individual step*

[3]See [11] for a discussion of some important asymmetries in the modal operators, as applied to TWEAK-type plans

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 $\Box(e \approx p) \in \mathcal{B}$.*[4]

Definition 3.1.1 (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: $\{\langle s, e \rangle | s \in \mathcal{S}, e \in \text{effects}(s), \Box(e \approx p)\}$.*

Consider again the plan MPEX 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 3.2 (Violated Causal Links) *A causal link $\langle \mathcal{S}, p, w \rangle$ of a plan $\mathcal{P}$ is said to be violated if there exists a completion of $\mathcal{P}$ where none of the contributors in $\mathcal{S}$ can be feasible contributors of p to w. 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:

Proposition 3.2.1 *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 $s' \prec s$.*

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 MPEX 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

[4]Note that requiring that all the contributors of a causal link must precede the destination node, and provide an effect that will necessarily codesignate with the condition being supported, is in general stronger than the minimal constraints required to satisfy the 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 $P(v)$ at w, which is $[x \approx v \wedge z \approx v] \vee [x \approx v \wedge (y \not\approx v \vee z \approx v)] \vee [z \approx v \wedge (u \not\approx v \vee x \approx v)]$.

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

Figure 3: A multi-contributor validation structure for the plan MP

$w2$ 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 obfuscates 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.

3.1 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 3.3 (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 $(p' \approx p) \in \mathcal{B}$ 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 $\mathcal{P}$.*

Proposition 3.3.1 *If a partially ordered partially instantiated plan $\mathcal{P}$ is correct by the Definition 3.3, then all of its completions are guaranteed to be correct, thus satisfying the definition of correctness given in Section 2.*

Thus, correctness w.r.t. a multi-contributor validation structure provides sufficient conditions for ensuring the correctness of a plan. Figure 3 shows a set of valid (multi-contributor) causal links under which the plan MPEX shown in Figure 1 is correct.[5]

[5]The converse of this 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.

4 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.

4.1 Irredundant Validation Structures

The most stringent restriction on multi-contributor causal links would be to stipulate that every contributor in the causal link is *required* in some strong sense to keep the validation un-violated. This is captured by the notion of irredundant validation structures defined below:

Definition 4.1 (Irredundant Contributors) *A contributor s of an initially un-violated causal link $\langle \mathcal{S}, p, w \rangle$ is said to be irredundant if and only if there exists at least one completion of the plan where s is the only feasible contributor of p to w in $\mathcal{S}$*

From the definition, it follows that removal of an irredundant contributor is guaranteed to cause violation of the validation.

Proposition 4.1.1 *Given an un-violated 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:*

1. $(n \prec s) \wedge (w \parallel n)$ *and*
2. $\neg d \in effects(n)$ *such that* $\Diamond(d \approx p)$ *and*
3. $\forall s_i \in \mathcal{S}$ *if* $(s_i \not\prec s)$ *then* $(n \parallel s_i)$

(In other words, s is the only step in the plan capable of foiling the harmful interaction caused by n.)

Definition 4.1.1 (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.

Proposition 4.1.2 *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 there is no single step $s \in \mathcal{S}$ that is a feasible contributor (see Definition 2.2) of p to w in all completions.*

Thus, irredundant validation structures admit multiple contributors in causal links only when they are absolutely necessary.

4.2 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 3 as it stipulates that each contributor should be an effective contributor in at least one completion of the partially ordered plan.

Definition 4.2 (Irrelevant Contributors) *A step s is said to be an* **irrelevant** *contributor of a validation $\langle \mathcal{S}, p, w\rangle$ in plan $\mathcal{P}$ if and only if there exists no completion of $\mathcal{P}$ in which s is the effective contributor of p to w.*

Proposition 4.2.1 *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 3, $w0$ 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 4.2.1 (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 $\mathcal{P}$.*

From this definition and the property 4.2.1, we have:

Proposition 4.2.2 *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 4.2.1, by selecting the step u from $\mathcal{S}$. The validation structure in Figure 3 is not a relevant validation structure for MP. However, it can be made relevant by removing $w0$ from the validation $\langle \{w0, w1\}, Q, fin\rangle$.

4.3 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 is the effective contributor of that prerequisite in some completion of the plan.

Definition 4.3 (Exhaustive Validation) *An un-violated 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.

Proposition 4.3.1 *A validation $\langle \mathcal{S}, p, w\rangle$ of a plan $\mathcal{P}$ is exhaustive if and only if $\forall n \in \mathcal{P}$, 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** *be the case that $w \prec n$* **or** *it must be the case that $\exists s \in \mathcal{S}$ such that $n \prec s$.*

Definition 4.3.1 (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.*

Consider the validation $\langle\{s1\}, T1, w1\rangle$ in Figure 3. This validation is not exhaustive since $s5$, which can possibly come before $w1$ can also provide $T1$ to $w1$. Steps like $s5$ which threaten the exhaustiveness of a causal link are called **+ve threats** to that causal link. We can make $\langle\{s1\}, T1, w1\rangle$ exhaustive by either promoting $s5$ to come after $w1$ or merging $s5$ into the existing validation (which involves ordering $s5$ to come before $w1$).

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.

It is easy to see that exhaustiveness does not imply irredundance or vice versa. For example, in the plan MPEX in Figure 1, the validation $\langle\{wo, s5, w2\}, R, fin\rangle$ is an exhaustive validation, but it is not an irredundant validation. In particular, since any one of the three contributors can be a feasible contributor of R to fin in all completions of MPEX , including all of them makes at least two contributors non-irredundant.

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 before further refinement, In particular, we have the following property:

Proposition 4.3.2 (Uniqueness of Exhaustive Validation Structures) *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 an equivalence class relation between partial order plans and their completions. In particular, we have the following proposition:

Proposition 4.3.3 *If $\mathcal{P}$ is a partially ordered plan with the single-contributor exhaustive validation structure $\mathcal{V}$, then it is possible to uniquely reconstruct $\mathcal{P}$ given only $\mathcal{V}$ and any one of $\mathcal{P}$'s completions.*

It is worth noting at this point that the ability to maximally generalize orderings/bindings etc. based on a validation structure and a single completion is not limited to exhaustive validations structures alone. In fact this a common technique used in explanation-based generalization (c.f. [9]). What is interesting about single contributor exhaustive validation structures is that they guarantee a unique such maximal generalization. Thus the class of partial order plans with single contributor exhaustive causal links can be used to define a equivalence class relation over the plan completions. Searching in the space of partially ordered plans becomes equivalent to searching in the space of equivalence classes of plan completions.

This property does not hold for multi-contributor exhaustive validation structures. In terms of the proof of the proposition in Appendix B, this is primarily because it is not always possible to uniquely determine a safety ordering, given a threat and a completion. In the construction above, suppose $\langle \mathcal{S}, p, w \rangle$ is the multi-contributor exhaustive validation, and s' is the threat that we are trying to order. From the information in the completion, we can only tell with certainty whether s' should come after w or before one of the steps in $\mathcal{S}$. In the latter case, there is still a choice as to which of the steps of $\mathcal{S}$ should s' precede, which is not uniquely determined by the completion. Although exhaustive causal structures do not guarantee unique maximal generalization in the multi-contributor case, they do reduce the number of possible maximal generalizations (see Figure 11).

5 MP and MP-I: Two Algorithms for Planning with Multi-Contributor Causal Links

In this section, we describe two planning algorithms, MP and MP-I respectively, that utilize the multi-contributor causal links formulated in the previous sections. Both these algorithms generate plans by finding and maintaining *relevant* and *un-violated* multi-contributor causal links for supporting each goal and prerequisite of the plan. They differ mainly in the eagerness with which they accrue additional contributors to a causal link. MP , like SNLP aims to maintain *exhaustive* causal structures for all the partial plans during search. As we discussed in Section 4.3.1, this reduces redundancy in the search space but leads to higher commitment. MP-I on the other hand takes a *lazy* approach to accumulating additional contributors, adding new contributors to a causal link only as a way of resolving a violation to the causal link. This reduces premature commitment (in terms of ordering constraints). (In the lifted version of these algorithms, the eager accumulation also has ramifications on the binding constraints posted by the planner. In particular, MP's eager accumulation of contributors makes it add more binding constraints (between the effects of the contributors and the required condition of the consumer) than MP-I , thus leading to further over-commitment.)

These differences lead to differing notions of conflicts/interactions during planning -- given a validation $\langle \mathcal{S}, p, w\rangle$, MP considers any step v that can possibly intervene between the contributors and the consumer of the validation and either add or delete p, as a potential threat to that validation (if v deletes p, then it violates $\langle \mathcal{S}, p, w\rangle$, while if it adds p, then it undermines the exhaustiveness of the validation). Since MP-I doesn't aim for exhaustiveness of the validation structure, it considers a step v as a threat only in situations where v deletes p. In the following, we will first formalize these two differing notions of threat to a validation, and then use them to formally describe the termination conditions for both the planners.

Definition 5.1 (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 $\exists q \; s.t. \; q \in effects(v) \vee \neg q \in effects(v)$ such that $\Diamond(q \approx p)$. Further v is called a* **-ve threat** *if $\neg q \in effects(v)$ and it is called a* **+ve threat** *if $q \in effects(v)$.*

Definition 5.2 (MP Complete Plans) *A plan $\mathcal{P} : \langle T, O, \mathcal{B}\rangle$ is* MP 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 $(p' \approx p) \in \mathcal{B}$.*
- *If $\mathcal{P}$ contains a causal link $\langle \mathcal{S}, p, w\rangle$, and a step v that is either a* **+ve** *or a* **-ve** *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.*

Definition 5.3 (MP-I Complete Plans) *A plan $\mathcal{P} : \langle T, O, \mathcal{B}\rangle$ is* MP-I 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 $(p' \approx p) \in \mathcal{B}$.*
- *If $\mathcal{P}$ contains a causal link $\langle \mathcal{S}, p, w\rangle$, and a step v that is a* **-ve** *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.*

Definitions 5.2 and 5.3 serve as termination conditions for the search processes of MP and $\mathrm{MP\text{-}I}$ planners respectively. Note that plans complete by either definition are also correct according to Definition 3.3. The main difference is that plans complete by Definition 5.2 also have exhaustive validation structures (by property 4.3.1), while this is not guaranteed for plans complete by Definition 5.3. As discussed earlier, this implies that $\mathrm{MP\text{-}I}$ has more redundancy in its search space than MP.

Figure 4 shows a planning algorithm, MP, which generates plans that are complete by Definition 5.2. The algorithm $\mathrm{MP\text{-}I}$ for generating plans that are complete by Definition 5.3 is obtained by replacing step 3 of the MP algorithm with the alternative way of treating threatened causal links shown in Figure 5. (To simplify discussion, we only show the procedures for generating ground partially ordered plans. The procedures for generating partially instantiated and partially ordered plans can be obtained in a straightforward fashion using the lifting transformation discussed in [16]. The implementations of these planners, discussed in Section 6, do use lifted versions of these algorithms.)

Just as in planners using single-contributor causal links (e.g. SNLP [16]), the important steps in the multi-contributor causal link planners MP and $\mathrm{MP\text{-}I}$ are (i) to handle open goals and prerequisites by adding causal links to support them, and (ii) to handle unsafe causal links by adding additional ordering constraints on the plan. The main differences come in their treatment of unsafe causal links.

In the case of MP (see Figure 4), given that a plan that is complete for it has causal links that are both exhaustive and un-violated, a causal link $\langle \mathcal{S}, p, w\rangle$ may be threatened by both a step that deletes p (-ve threat) and a step that *adds* p (+ve threat). A -ve threat is handled by either promoting it to come after w (3(a)), or by demoting it to come before *one* of the steps in $\mathcal{S}$ (3(b)). A +ve threat is handled by either promoting it to come after w (3(a)) or merging it into the contributor set $\mathcal{S}$ (3(c)).[6] This is what is done in step 3(c).

[6]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 5.2), 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 4: 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 5: Treatment of threatened causal links in MP-I

In contrast, since MP-I does not aim to maintain exhaustiveness of plan validation structure, it is only interested in ensuring that no causal link be threatened by a -ve threat. However, it does exploit the redundancy in the plan causal structure in a *lazy* manner by introducing a constrained form of white-knight declobbering clause into its treatment of -ve threats (see Figure 5): apart from promotion and demotion[7], a -ve threat v can also be neutralized by finding an *existing* step v' such that $v \prec v'$ and v' adds p, and merging v' into the contributor set of the threatened causal link.[8]

For both MP and MP-I , 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]

[7]Note that in the lifted case, a threat can also be resolved by ‘‘separation,’’ i.e., constraining the threat’s effects to not codesignate with the condition being supported by the causal link. Separation falls naturally out of the lifting transformation (see [16])

[8]It is instructive to note the differences between this lazy declobbering strategy and the TWEAK white-knight clause: MP-I uses declobbering only when a white-knight step already exists in the plan and is already constrained to necessarily follow the clobberer (-ve threat). In contrast, a straight forward inversion of TWEAK MTC would allow for introducing white-knights by either adding new steps or introducing additional ordering relations (see Figure 12).

[9]In implementing this procedure, it is possible to reduce some of the 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 [28]).

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 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 in which 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 $\mathcal{S}$. Once a step has been removed from a causal link by **MakeRelevant** procedure, it will *never be* reintroduced into that link in that branch of the search process (this is because any irrelevant contributor is temporally dominated by the rest of the contributors the validation, and since ordering decisions are never retracted, it can never interact with that validation again). Thus there will not be any looping behavior because of removal of irrelevant contributors).[11]

5.1 Soundness and Completeness of MP and MP-I

It is easy to verify that the plans generated by MP and MP-I algorithms will in fact satisfy the respective termination conditions. In particular, both algorithms terminate only after they have explicitly considered each open condition for establishment. This means that every pre-requisite is supported by a causal link (validation). Both the algorithms explicitly keep track of potential -ve threats to the plan validations. Thus, by the time they terminate, all validations are guaranteed to be un-violated (by Proposition 3.2.1). Finally, by Proposition 3.3.1, every completion of a plan returned by MP and MP-I represent a correct solution for the planning problem. In addition, the algorithm MP also considers and resolves +ve threats. This, in conjunction with the MakeRelevant

[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 when a contributor becomes irrelevant for a validation and is thus removed by the **MakeRelevant** procedure, it may also become unjustified or purpose-less. When this happens, the plan in question can be pruned from the search space without loss of completeness. See Section A.

procedure, which keeps contributor set unordered, ensures that the validations maintained by MP are also exhaustive.

In terms of completeness, we note that the algorithms MP and $\mathrm{MP\text{-}I}$ are generalized versions of the SNLP algorithm [16], in the sense that (i) every plan that satisfies the termination conditions of SNLP also satisfies the termination conditions of MP and MP-I and (ii) the set of refinements allowed by MP and MP-I algorithms is a strict superset of the refinements allowed by SNLP. Based on these, and the fact that SNLP is complete, it can be easily shown that MP and MP-I are complete. In particular, we have:

Proposition 5.3.1 *The planning algorithms* MP *and* $\mathrm{MP\text{-}I}$ *described in Figures 4 and 5 are both complete in the technical sense of Definition 2.3.*

Intuitively, MP and $\mathrm{MP\text{-}I}$ can be seen as adding redundant refinement branches to the search tree of SNLP[12]. The redundant branches are there to improve planning performance by reducing the solution depth, and avoiding unnecessary backtracking. Theoretically, failures in redundant branches do not in any way affect the completeness of the overall algorithm. In particular, the presence or absence of the MakeRelevant procedure, used in MP and $\mathrm{MP\text{-}I}$ algorithms, does not affect the completeness of the algorithms. Its only purpose is to improve efficiency by obviating unnecessary choice points and conflict resolutions involving irrelevant contributors. This situation is akin to adding macro-operators to the search space of a problem solver.

The completeness and soundness arguments can also be extended in a straightforward fashion to lifted versions of MP and $\mathrm{MP\text{-}I}$. Here, we discuss some issues relevant to such extension. In the lifted case, if the MakeRelevant procedure removes an irrelevant contributor s from the contributor set of some validation $\langle \mathcal{S}, p, w\rangle$, the binding constraints that were originally added to make some effect of s codesignate with p unnecessary.[13] If the planner keeps such unnecessary bindings, then it may miss a sequence of refinements of the current plan that could end in a plan satisfying the termination conditions. This however does not affect completeness, since all the completions of such a plan can still be found in the single-contributor branches of the search tree (see the discussion above). For efficiency reasons however, it is better to remove the unnecessary bindings, so that the planner can avoid unnecessary backtracking. Removing bindings introduces another complication -- it is possible that some establishment and conflict resolution choices, which were previously infeasible, may become feasible after the removal of bindings. Retrying all those lost alternatives can be computationally expensive. Once again, although we need to retry all

[12]Actually, $\mathrm{MP\text{-}I}$ should be seen as redundant branches to the search tree of a variant of SNLP that does not resolve +ve threats; see discussion of McNonlin in Section 6.

[13]Note that the ordering constraint '$s \prec w$' still holds since MakeRelevant procedure removes only those contributors that are necessarily followed by other contributors (which themselves, by definition, must necessarily precede the consumer step, w).

those previous alternatives to ensure completeness within the current search branch, such retries are not required to guarantee completeness of the overall search.

6 Discussion and Evaluation of MP and MP-I

In this section we will consider the properties of MP and MP-I algorithms from the perspective of plan generation,[14] present some hypotheses about their performance based on these properties, and finally provide an empirical validation of those hypotheses.

To begin with, as discussed in Section 1, MP and MP-I will be able to find less constrained plans than planners using single-contributor causal structures such as SNLP. In particular, unlike SNLP (see [16]), both MP and MP-I can accommodate plans such as the one shown in Figure 6, proving whose correctness requires the white-knight clause of Chapman's Modal Truth Criterion. However, the ability to find less constrained plans does not in itself argue for the use of these planners during plan generation since the objective of planning is almost always to find *a* sequence of actions that can solve the problem. Deferred commitment regarding orderings and bindings used in nonlinear planning is merely a means towards improving efficiency of finding plans rather than an end unto itself. (Even if less committed plans or plans having multi-contributor causal structures are required for execution and generalization purposes, we can get them by postprocessing plans generated by SNLP using EBG techniques (c.f. [9]).)

The expectation regarding the improved efficiency of MP and MP-I comes from the fact that both planners have the ability to avoid premature commitment to specific contributors by flexibly accommodating multiple contributors. This flexibility enables them to avoid unnecessary backtracking that is inevitable with planners using single contributor causal structures. In the example shown in Figure 2 (discussed in Section 1), where planners using single-contributor causal links will be forced to backtrack over their commitments, both MP and MP-I can avoid the backtracking by simply merging $s3$ into the contributor set.[15] Such an ability to avoid premature commitment provides MP more stability with respect to the planning order (i.e., the order in which the open prerequisites are addressed by the planner), where as the planning order can have

[14]The ability of MP and MP-I to take advantage of redundancy in the plan causal structure by maintaining multiple contributors for prerequisites, can also help in latter generalization, reuse and modification of plans. In Section A, we present a framework for justifying planning decisions with respect to a multi-contributor validation structure, which can be used to guide modification and generalization of plans. In this section, we restrict our attention to evaluating the advantages of using MP and MP-I for plan generation.

[15]Note that when s_3 is merged into the contributor set of the validation supporting the condition PP at w, the original contributor s_1 becomes irrelevant and eventually gets removed by the **MakeRelevant** procedure, making s_3 the sole new contributor! The effect of first merging and then making the validations relevant is thus to provide an indirect way of allowing for change of contributors **without** backtracking.

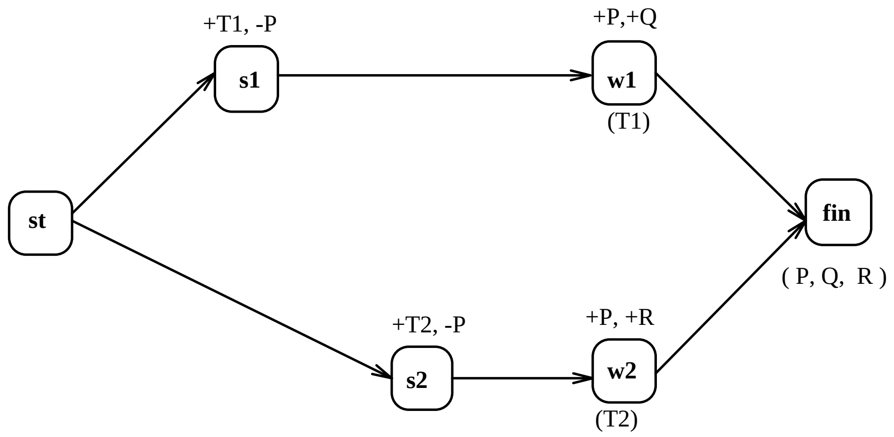


Figure 6: A Plan that is complete for MP and MP-I but not for SNLP

a more drastic impact on the efficiency of planners using single-contributor causal links.

The above discussion leads us to the following plausible hypothesis regarding the advantages of MP and MP-I over planners using single contributor causal structures:

> **Hypothesis:** MP and MP-I should perform better in domains containing preconditions that are added and deleted by many actions in the domain. In all other cases their performance should be comparable to that of planners using single-contributor validation structures.

Unfortunately, however, since both MP and MP-I reduce commitment to contributors at the expense of increased redundancy in the search space, as well as slightly increased cost of maintaining multi-contributor causal links, the validity of the above hypothesis is by no means obvious at the outset. There are no *a priori* reasons to believe that the tradeoff between redundancy and commitment made by MP and MP-I are better compared to that made by planners using single contributor causal structures such as SNLP. Empirical evaluation of this hypothesis is thus needed.

Consequently, we implemented lifted versions of MP and MP-I over Weld et. al.'s implementation of SNLP [25][16], and compared their performance to SNLP and a NONLIN-like variant of SNLP, called McNONLIN, (which ignores **+ve** threats and thus sacrifices exhaustiveness of validation structure for partial plans) in a variety of domains. All planners used A^* search regime with identical evaluation functions.[17] Since all the planners used the same basic routines,

[16]Contact rao@asuvax.asu.edu for obtaining the code for our implementation

[17]The g component of the basic evaluation function is given by the number steps while the h component is given by the sum of the number of open conditions and the number of unsafe validations. In the case of MP and SNLP, which

ART-MD	$(Defstep\ A_i\ \underline{precond}:\ I_i\ \underline{add}:\ G_i\ \underline{delete}:\ \{I_j \mid j < i\})$
ART-1D	$(Defstep\ A_i\ \underline{precond}:\ I_i\ \underline{add}:\ G_i\ \underline{delete}:\ I_{i-i})$
ART-MD-NS	$(Defstep\ A_i^1\ \underline{precond}:\ I_i\ \underline{add}:\ P_i\ \underline{delete}:\ \{I_j \mid j < i\})$
	$(Defstep\ A_i^2\ \underline{precond}:\ P_i\ \underline{add}:\ G_i\ \underline{delete}:\ \{I_j \mid \forall j\} \cup \{P_j \mid j < i\})$

Figure 7: The specification of Weld et. al.'s Synthetic Domains

search strategies, and heuristics, and differ only in the way they maintain their causal links, the comparisons are very fair.

Our test domains included the classical toy-worlds such as blocks world, as well as the synthetic domains used in Weld et. al.'s work [25]. In this paper, we will concentrate on the results from Weld et. al.'s synthetic domains and our variants of them, as they provide for a more controlled testing of our hypotheses. Weld et. al.'s original domains include ART-MD, ART-1D and ART-MD-NS (also referred to, in their later papers, as $D^m S^1$, $D^1 S^1$ and $D^m S^2$ respectively), which are designed to contain easily serializable, laboriously serializable and non-serializable sub-goals respectively. To these, we also added our own variants of Weld et. al.'s domains: ART-MD-RD, ART-1D-RD, and ART-MD-NS-RD, which introduce preconditions achieved and deleted by multiple operators. These variants are motivated by the observation that premature commitment to contributors is especially harmful when there are preconditions with multiple adders and deleters since a wrong commitment can cause extensive backtracking (as discussed earlier).

The domain specifications of ART-MD, ART-1D and ART-MD-NS domains are given in Figure 7. The variants ART-MD-RD, ART-1D-RD and ART-MD-NS-RD are produced by making every even numbered action require an additional precondition *he*, delete that precondition and add an additional postcondition *hf*.[18] The odd numbered actions similarly require and delete *hf* and add *he*. The specification of ART-MD-RD is shown as follows:

$$\begin{array}{rl} \text{For even } i & (Defstep\ A_i\ \underline{precond}:\ I_i, he\ \underline{add}:\ G_i, hf\ \underline{delete}:\ \{I_j \mid j < i\} \cup \{he\}) \\ \text{For odd } i & (Defstep\ A_i\ \underline{precond}:\ I_i, hf\ \underline{add}:\ G_i, he\ \underline{delete}:\ \{I_j \mid j < i\} \cup \{hf\}) \end{array}$$

ART-1D-RD is produced similarly from ART-1D. In the case of ART-MD-NS, we can introduce the *hf* and *he* preconditions into either A_i^1 or A_i^2. Accordingly, we made two variants from this domain: ART-MD-NS-RD-1 where A_i^1's are modified to require and delete new preconditions

consider both **+ve** and **-ve** threats, the planners also use the additional heuristic strategy of postponing +ve threats -- i.e., they address all the -ve threats for a validations before they address the positive threats.

[18] *he* and *hf* are supposed to be mnemonics for $handempty$ and $handfull$ conditions in the traditional blocksworld domain [18], which are achieved or deleted by many of the actions in the domain.

and ART-MD-NS-RD-2 where A_i^2's are modified. In each of these seven domains, we compared all four planners, MP , MP-I , SNLP and McNONLIN over *solvable* problems with 1 to 8 goals from the set $\{G_1 \cdots G_8\}$. Since the commitment to contributors depends largely upon the order in which the various goals and subgoals are addressed by the planner (this is typically referred to as the *planning order*), we experimented with two types of goal order strategies: strategy L which is a LIFO strategy where a goal and all its recursive subgoals are addressed before the next higher level goal is addressed; and strategy GbyG, which is a FIFO strategy where all the top level goals are addressed before their subgoals are considered by the planner (strategy L corresponds to a depth-first traversal of the goal-subgoal tree, while strategy GbyG corresponds to a breadth-first traversal).

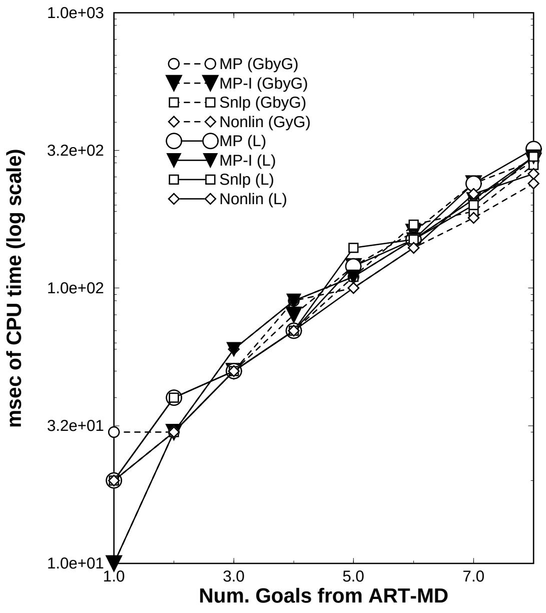
If our hypotheses about MP and MP-I hold, then their performance should be comparable to that of SNLP and McNONLIN in ART-MD, ART-1D, and ART-MD-NS domains, while the performance should be superior in ART-MD-RD, ART-1D-RD, ART-MD-NS-RD-1 and ART-MD-NS-RD-2.

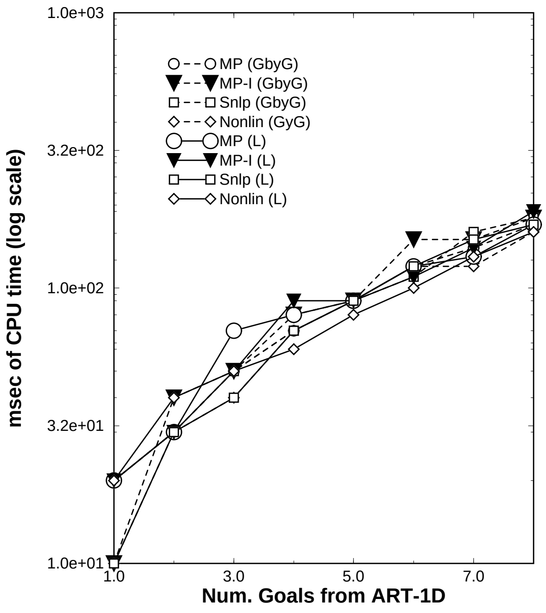
The plots in Figure 8 compare the cpu time (in m.sec. on a SUN SPARC-II work station running compiled LUCID common lisp) taken by the three different planners for each of the goal ordering strategies in ART-MD, ART-1D and ART-MD-NS[19] domains. They demonstrate that there is no appreciable difference in time taken by the planners for solving problems in these domains. A comparison of the number of partial plans expanded showed that all planners explore identical parts of the search space (that is, given the same heuristic and goal-ordering strategy, all of them expand the same number of nodes to reach the solution). This is in line with our hypotheses since in these domains each precondition is ultimately provided by a single action in the plan, and there is thus no penalty for committing prematurely to that action as the contributor. The plots also suggest that the multi-contributor causal link maintenance routines do not significantly add to the cost of planning.

The plots in Figure 9 compare the performance of the planners in ART-MD-RD, ART-1D-RD and ART-MD-NS-RD domains which contain the easily achieved and deleted conditions *hf* and *he*. A comparison of the number of partial plans expanded by each of the planners yielded very similar patterns.[20] Here, as expected, MP and MP-I 's ability to avoid premature commitment

[19]Note that the problems in the MD-NS domains are inherently ''*harder*'' than those in MD and 1D domains. In particular, the length of the minimal-length plan for an n-goal problem is n in ART-1D, and ART-MD while it is $2n$ in ART-MD-NS. This is the reason why we were able to solve problems with only up to 6 goals in ART-MD-NS, and up to 5 goals in ART-MD-NS-RD domains, before exhausting resources.

[20]To see if these performance profiles are dependent on the particular heuristics used in the search, we also experimented with two other heuristics, one in which the weight for the path cost to the state is increased and one in which the path cost (i.e., the g component of the A^* evaluation function $f = g + h$) is given weight 0 in computing the heuristic. We found very similar performance patterns in both these cases.





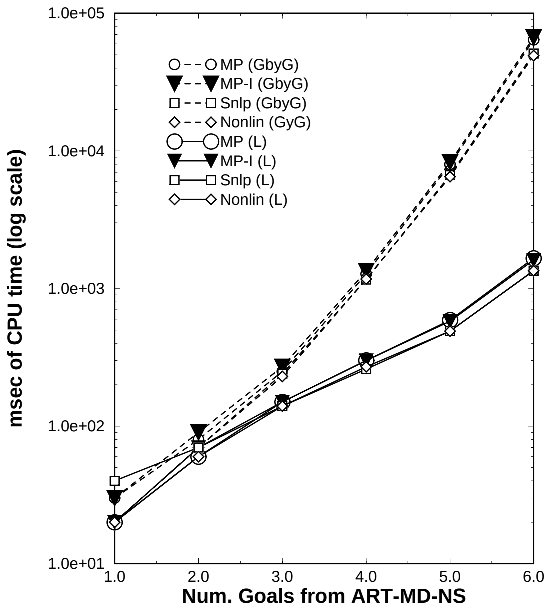


Figure 8: Performance of MP , SNLP and McNONLIN in ART-MD,ART-1D and ART-MD-NS Domains (all planners expand equal number of nodes)

to single contributors makes them less sensitive to the order in which the goals are expanded, compared to SNLP and McNONLIN. In particular, for the LIFO goal ordering strategy L, SNLP and McNONLIN perform exponentially worse than MP and MP-I (note that all plots are in *logarithmic* scale). Furthermore, MP-I 's lazy approach to accruing multiple contributors turns out to be better than the eager approach used by MP .

Once again, this behavior is in line with our hypotheses -- in the LIFO ordering strategy L, SNLP and McNONLIN are forced to commit to a specific contributor for the *hf* and *he* subgoals (preconditions) of a top level goal G_i, before the other top level goals are expanded. Since both *hf* and *he* are easily added and deleted by many actions in the domains, such premature commitment has a high probability of being wrong. Since both SNLP and McNONLIN protect causal links, the only way they can get rid of a wrong causal link is to backtrack over it (i.e., go over to another branch of the search space)[21] This turns out to be very costly in terms of performance. MP and MP-I avoid this problematic backtracking as they can deal with their initial wrong commitment by merging additional contributors into the contributor list as and when they become available.

Premature commitment turns out to be less of a problem when the FIFO ordering strategy GbyG is used, since in this case G_i and I_i are addressed before hf and he, and each action A_i is capable of giving only one of the goals G_i. Since only the initial state is capable of giving all I_i, the orderings imposed to deal with the deletions of I_i's by individual actions of the plans constrain the plan enough so that by the time hf and he are addressed, the only contributor choice also happens to be the correct choice. Although premature commitment is not a problem, the maintenance of exhaustiveness validation structures forces SNLP and MP to consider +ve as well as -ve threats, leading to increased solution depth.[22] This makes both of them inferior to MP-I and McNONLIN for the GbyG goal ordering strategy.

6.1 On the generality of the Experiments

A few words are in order regarding the generality of the experiments reported in this paper. First, it must be emphasized that the performance of a planner does not depend solely on the degree of commitment to contributors, but rather on a combination of factors including the order in which the goals are addressed and the particular termination conditions used (c.f. [13]). In our experiments, we kept these factors constant while varying only the goal protection strategies. Varying solutions to these other factors could affect the relative performance of planners based on single and multi-contributor causal links. For example, we found that using more adaptive goal-selection

[21]Unlike McNONLIN, Tate's NONLIN [28] in fact could non-monotonically violate a protection, and re-satisfy it in some alternate way. Nonlin allowed typed preconditions to indicate circumstances in which it could do this.

[22]Note that plans that are complete for MP-I and McNONLIN may still need to be refined further to ensure exhaustiveness of their validation structure and thereby make them complete with respect to MP and SNLP.

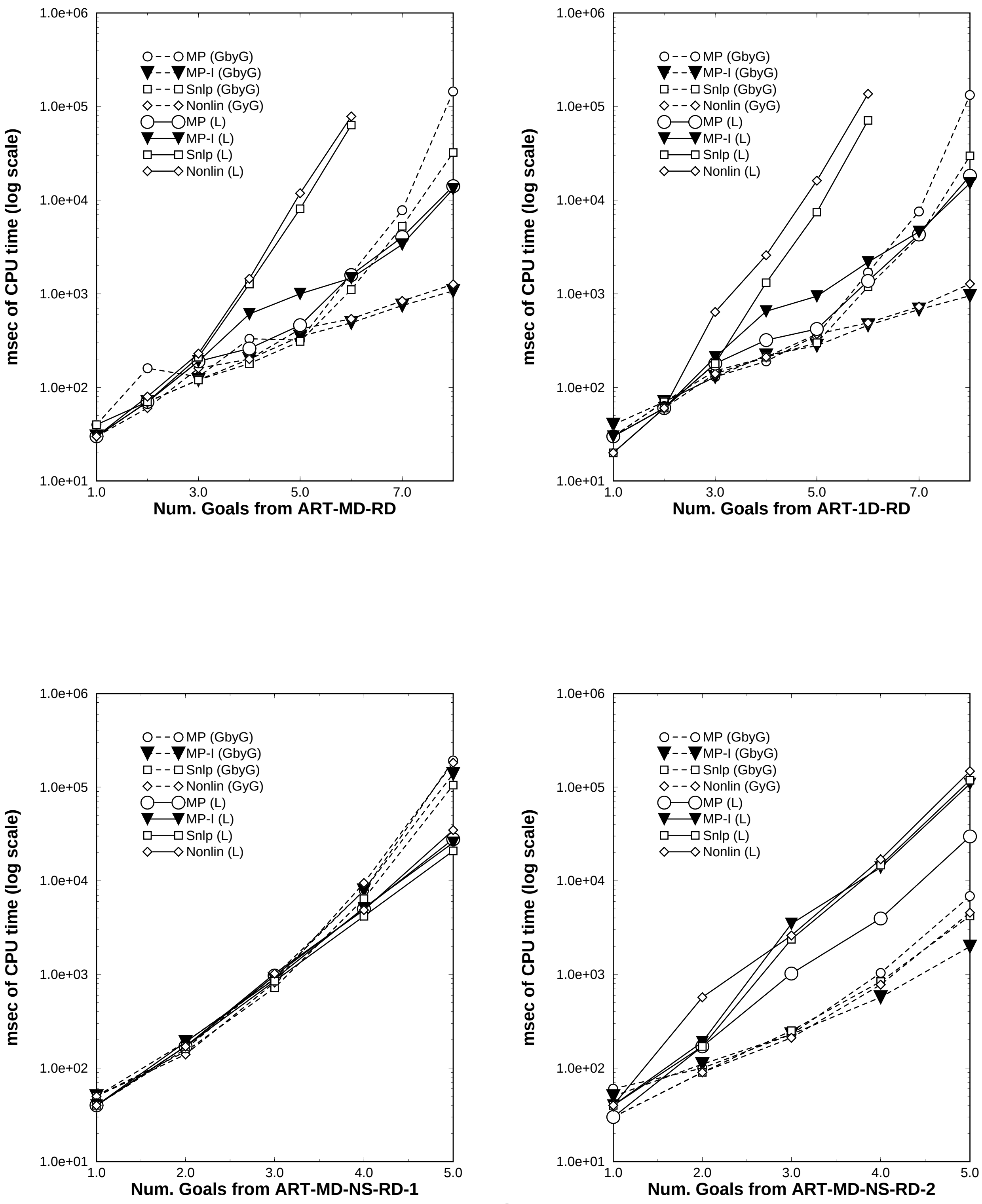


Figure 9: Performance of MP , SNLP and McNONLIN in ART-MD-RD,ART-1D-RD and ART-MD-NS-RD Domains

strategies could reduce the differential between single-contributor and multi-contributor planners in our experiments. Techniques for doing this include precondition abstraction strategies which put ‘‘hf’’ and ‘‘he’’ at a lower level of abstraction than the rest of the preconditions; as well as goal selection strategies that use modal truth criteria to select and work on only those goals that are not necessarily true.[23] This is not surprising since, as discussed earlier, the performance penalty incurred by premature commitment is a function of the order in which goals are addressed (LIFO vs. FIFO in our experiments).

The fact that the performance penalties of single-contributor causal structures could be reduced with the help of appropriate goal selection strategies does not necessarily cast shadow on the utility of multi-contributor causal structures. This is because adaptive goal selection strategies have their own set of performance tradeoffs. For example, MTC based goal selection strategies become NP-hard for action representations that are even moderately more expressive than that sanctioned by TWEAK representation (e.g., have conditional effects or finite domain variables). Similarly, there are no tractable automated precondition abstraction mechanisms in existence which can induce the goal ordering of the type described above. Most existing tractable abstraction strategies are based solely on analyses of potential interactions [14, 32], and are of limited utility. In particular, according to such analyses ‘‘hf’’ and ‘‘he’’ will be of the same level of importance as other preconditions. The predicate-relaxation approach advocated by Christensen [3] will be able to separate ‘‘hf’’ and ‘‘he’’ from other preconditions, but it is not in general tractable.

Finally, although most of the experiments reported in this section were done in ground domains, we did test our planners in domains with variables. Figure 10 shows the performance of the four planners in a variant of ART-MD-RD domain with variables in the operator descriptions[24]. From these plots, we note that although the overall complexity of the problems increases in the variablized domains, the relative performance of the planners remains largely unaffected.

7 Related Work

To our knowledge, NONLIN [28] 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,

[23]In the case of ART-X-RD domains, such strategies could reduce the number of times that ‘‘hf’’ and ‘‘he’’ are picked up for establishment. This allows them to offset the disadvantages of single-contributor causal structures, since attempts to establish hf and he explicitly invariably lead to wrong contributor commitments.

[24]Specifically, the hf and he conditions are parameterized as (hf ?x) and (he ?x) respectively.

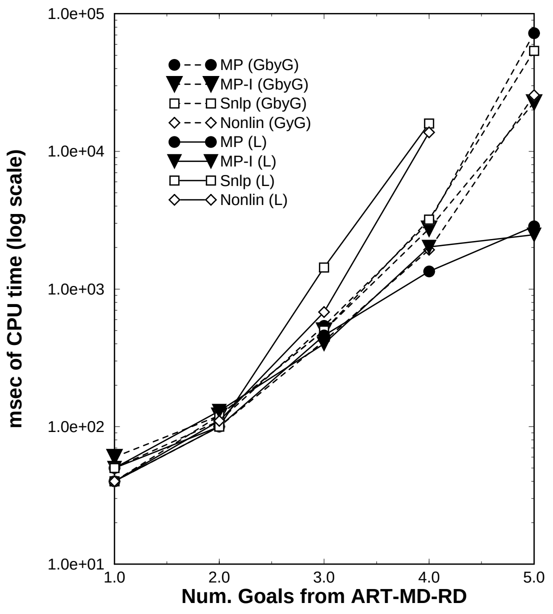


Figure 10: Comparison of MP , MP-I , SNLP and McNONLIN for problems in lifted ART-MD-RD domain

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[25]), 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 are other contributors that are affected. 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). Latter work on NONLIN provided some techniques to rectify this [5]. The development of

[25]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 irrelevant. However, subsequent planning may introduce ordering relations among them, making them irrelevant

justification framework in Appendix A provides a systematic basis for doing such retraction in the context of multi-contributor causal structures. O-PLAN, a successor of NONLIN, also has a more generalized notion of protection intervals called ‘‘clouds’’ [29], which were designed to manage the contributors and terminators of aggregated sets of dependencies. Clouds allowed O-PLAN to manage multiple contributors all through the planning, by actively keeping track of the ‘‘last incoming contributor’’ wavefront.

In [7], Hertzberg and Horz discuss a formalization of plan causal dependencies that allows more than one dependency to support a pre-requisite, thus allowing for redundancy in the plan casual structure. They however do not discuss or evaluate the utility of such dependencies in planning. The idea of justifying planning decisions with respect to the underlying causal (goal) structure of the plan has been first introduced in Daniels’ work to augment NONLIN [5], and is formalized in our work on PRIAR plan modification framework [8]. The framework discussed in Section A can be seen as an extension of that formalization to the case of multi-contributor validation structures.

In their recent work on extending refinement planning to more expressive action representations [20], Weld and Penberthy discuss the need for generalizing the notion of causal links to allow for multiple contributors. Despite some surface similarity, their motivation for allowing multiple contributors is quite distinct from ours. In particular, they intend to use the multiple contributors to model synergistic effects. For example, if a step s_1 has an effect ‘$x = 3$’ and another step s_2 has an effect ‘$y > 4$’, then in their framework, the two steps can together be used to satisfy the condition ‘$x < y$’ at some other step s' using a causal link $\langle \{s_1, s_2\}, x < y, s' \rangle$. In contrast, the semantics of causal links discussed in this paper do not allow for modeling of synergistic effects. As mentioned, in our approach, multiple contributors are intended to capture redundancy in the plan causal structure, and to reduce the commitment to specific contributors. In principle, however, it should be possible to integrate both these functionalities of multi-contributor casual structures.

Finally, the work described in this paper has some important relations to the recent work on systematic nonlinear planning algorithms (c.f. [16, 17]). The next section explores these in depth.

7.1 Tradeoffs between redundancy in search space vs. extent of commitment

Systematicity is the ability to eliminate redundancy in the search space by ensuring that no two partial plans in the search space will have safe overlapping completions. Systematicity puts a strict upper bound on the search space of a partial ordering planner by ensuring that the number of completions explored by the partial order planner cannot be more than that explored by a corresponding total ordering planner. As observed in Section 4.3, systematicity is achieved by maintaining single contributor exhaustive validation structures for all partial plans during the search, which in turn leads to increased commitment.

Although theoretically appealing, systematicity is not guaranteed to have any direct correlation with the efficiency of planning. Indeed, as our experiments show, the additional commitment incurred to ensure systematicity may sometimes lead to reduced, rather than increased, efficiency in planning. This should not be surprising. The size of the overall search space will have a significant bearing on the average cost of planning only when the solution density is so low as to force the planner to search a significant part of its search space. In particular, in cases where there is no plan for a given problem, we expect a systematic planner like SNLP to give up much faster than a non-systematic one like MP and McNONLIN. In situations involving solvable problems, however, if the solution density is not too low, the average case performance is influenced more by the amount of commitment and backtracking done by the planner, than by the overall size of its search space.

What we have here is a tradeoff between redundancy in the search space explored by the planner, and the amount of commitment it is making. Planners like TWEAK [1] have very low commitment, but may be searching in highly redundant search spaces. Planners like UA [17] and SNLP [16, 25] guarantee systematicity, but impose higher commitment and thus may lead to more backtracking. In Section 4.3 we related this increased commitment to exhaustiveness of validation structures. 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 (c.f. [15]).

Consider for example the empirical comparison between MP , MP-I , SNLP and McNONLIN, discussed in Section 6. Figure 11 compares the total search space sizes of these four planners for problems in ART-MD-RD.[26] When we contrast these with the performance profiles in Figure 9, we observe that:

- MP and MP-I perform better than SNLP and McNONLIN in the case of the LIFO goal ordering strategy L, even though the latter two search in exponentially smaller search spaces (with SNLP in particular having the systematicity property).

- MP-I and McNONLIN perform better than SNLP and MP in the case of GbyG strategy, even though the former two don't maintain exhaustive causal structures and thus search in more redundant spaces.

The behavior in both cases can be plausibly explained by the fact that the solution density is such that the extent of commitment forced by a planner, rather than the overall size of its search space, wind up dictating the performance. Specifically, the behavior in the former case can be explained by the excessive backtracking caused by the premature commitment to specific contributors in SNLP and McNONLIN. The behavior in the latter case can be explained by observing that the

[26]The size of the overall search space is found by setting the termination condition for each planner to be uniformly false, thus forcing the planners to visit every node in the search space before giving up.

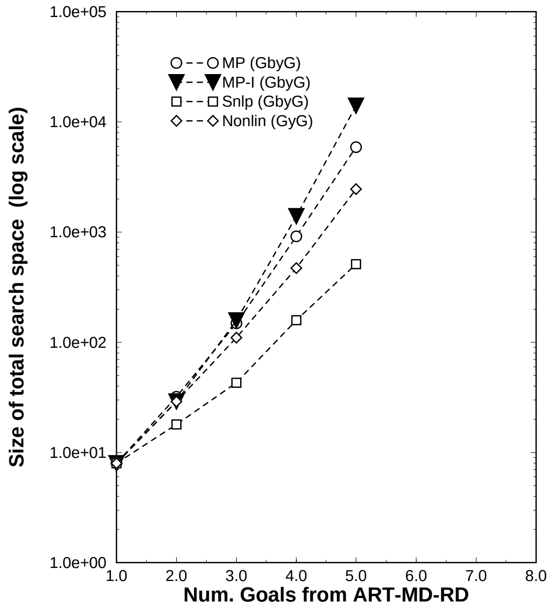


Figure 11: Comparison of total search space sizes of MP , MP-I , SNLP and McNONLIN for problems in ART-MD-RD

additional commitment necessary to ensure exhaustiveness (and thereby reduce redundancy in the over all search space) leads to an increase in the depth of the solutions for MP and SNLP (as compared to MP-I and McNONLIN).

Note also that the solution density and the ability to avoid premature commitment are more predictive of performance than is the the presence or absence of +ve threats. In particular, the plots in Figure 9 show that McNONLIN performs worse than SNLP in the LIFO strategy. This can be explained by noting that the redundancy that McNONLIN introduces into its search space by ignoring +ve threats, adversely affects the performance exactly when the planner's initial commitments all turn out to be wrong, forcing it to look at a significant portion of its search space.

Spectrum of Tradeoffs: More generally, there is a spectrum of tradeoffs between the redundancy in the search space and the extent of premature commitment forced by the planner. At one end of this spectrum we have planners that are generated by blindly inverting Chapman's MTC [1]. A representative search cycle for such planners is shown in Figure 12. Notice that such planners not only defer decisions about bindings and orderings, but also completely avoid committing to specific contributors for prerequisites. However, they have very high redundancy in the search space [17] (that is, they may consider the same potential solution in more than one search branch). Part of this redundancy comes from the fact that many of the planning decisions faced by TWEAK-like

```
Initialize Search queue with the null plan.
Begin Loop
    1. Pick a plan $\mathcal{P}$ from the search queue. If it is already correct according to MTC, terminate.
    2. Pick a precondition $p$ of $\mathcal{P}$ that is not necessarily true
    3. Using MTC, generate refinements of $\mathcal{P}$, one for each way of making $p$ necessarily true.
       This involves considering all possible ways of establishing $\mathcal{P}$ (simple establishment or step
       addition), and for each possible establisher, considering all possible ways of declobbering
       any clobberers (promotion, demotion, separation, white-knight declobbering).
    4. Add all the refinements to the search queue
End Loop
```

Figure 12: Planner obtained by inverting TWEAK MTC

planners are mutually redundant[27] leading to a dense graph-structured rather than a tree structured search space.[28] While some of this of redundancy can be avoided by ignoring (or ‘‘cutting’’) goal ordering choices and conflict resolution ordering choices from the search space, there is another more subtle form of redundancy -- that of plans having overlapping linearizations [16] -- which still poses problems. Visiting plans having overlapping linerizations amounts to visiting some solutions (correct completions) more than once. The effect of such redundancy is to increase worst case size of the search space.

Part of the problem with TWEAK is that because it does not maintain a history of the goals that it had already worked on before, it does a poor job of keeping track of its own progress. This leads to indiscriminate undoing of previously established goals, or redoing of previously failed establishment structures. The original motivation for causal links/protection intervals in planning was exactly to systematize this search process so that the planner can do a better job of keeping track of its progress.

[27]for example, any planner that backtracks on operator choices, ordering choices and binding choices can safely ignore goal ordering choices without losing completeness [19, 16]

[28]Graph-structured search space in itself would not have been a problem if it were practical to maintain a *closed* list of all previously visited plans and checking for duplicates during search. Unfortunately such a strategy poses prohibitive time (checking of equivalence of partially ordered partially instantiated plans takes $O(n^3)$ time) as well as space requirements.

While TWEAK suffers from high redundancy in the search space, on the other extreme from it are planners such as SNLP [16] which organize the search space in such a way as to avoid all types of redundancy in the search space. The main technique they use to avoid visiting plans with overlapping linearizations is to organize the search around the causal structure of the plans, and using the ‘‘exhaustiveness’’ property to enforce a tight correspondence between the causal structure of a plan and its completions.[29] However, as we discussed in Section 4.3, this non-redundancy in search space is achieved at the expense of increased commitment to particular causal structures, which in turn can have significant negative effects on the performance (see Section 6).

McNonlin, which commits to specific contributors, but does not insist on exhaustiveness of validation structure (thus admitting some amount of redundancy in the search space) would fall in the middle of this spectrum. As our empirical study shows, even McNONLIN’s commitment to single contributors can be disastrous when the domain has conditions that are added and deleted by multiple actions in the domain. Our experiments also show the advantages of regulating search through multi-contributor causal links, as is done for example in MP and MP-I . In particular, they strike a more favorable balance between the high redundancy of the TWEAK search space, and the high commitment of the planners using single contributor causal links. Unlike the former, they safe-guard against the repeated clobbering and achievement of the same preconditions, and unlike the latter they avoid the premature commitment to contributors.

8 Summary and Conclusion

Although widely used in classical planning, single-contributor causal structures have several disadvantages in dealing with partially ordered partially instantiated plans, which can be overcome by using multi-contributor causal structures. The primary contribution of this paper is to provide the first systematic characterization and evaluation 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 then described two planning algorithms based on multi-contributor causal structures and empirically established their advantages over planners using single contributor causal links. We have also argued that planners that use multi-contributor causal links to organize their search spaces strike a more favorable balance in the tradeoff between redundancy and commitment. In Appendix A, we will describe a framework for justifying individual planning decisions with respect to multi-contributor causal structures, which can be used to support more flexible modification and reuse of plans.

[29]They avoid cycles in the search space by ‘‘cutting’’ (i.e., not backtracking on) certain planning decisions, including planning order and the order of conflict resolution from the search process.

A Supporting Modification and Reuse with Multi-contributor Causal Structures

Causal structure representations have been shown to be very valuable in guiding plan modification [8, 6], and generalization [9]. In this section, we look at the support provided by multi-contributor causal structures for plan modification. 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. Supporting the retraction process is thus a central requirement for modification and reuse of plans.

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 overall goals of the plan, and then justify the other constraints in the plan in terms of the causal links they support [8, 32, 5]. Such a justification structure allows the planner to locate parts of the plan that become superfluous whenever a particular retraction occurs. When a decision is retracted, then all that is needed to rejustify the plan would be to locate the inconsistencies and superfluous steps. For example, we can say that a step in the plan is justified as long as it is supporting at least one causal link of the plan. The idea here being that removal of such a step will lead to violation of some validation, and consequently cause the plan to fail. In a similar fashion, we can also justify ordering constraints and binding constraints [8].

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 A.1 (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, there exists q in $prerequisites(t_G)$ such that $(q \approx p)$ in $\mathcal{P}$ and either $w = t_G$ or there exists another justified causal link $\langle \mathcal{S}', p', w' \rangle$ in $\mathcal{V}$ such that $w \in \mathcal{S}'$.*

Definition A.2 (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.[30]

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 3 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 3.3). 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 may make the plan incorrect. This is because the step could be a redundant contributor of a causal link only in the presence of another contributor. 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 A.3 (Ordering justification) *Let O^* be the transitive closure of the ordering relations O among the steps of the plan $\mathcal{P}$. An ordering relation $(s_1 \prec s_2) \in O^*$ of a plan $\mathcal{P} : \langle T, O, \mathcal{B} \rangle$ 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:*

1. *$\exists \langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ such that $s_1 \in \mathcal{S} \wedge s_2 = w$* **or**
2. *$\exists \langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ such that $s_1 = w$ and $\neg q \in effects(s_2)$ and $\Diamond(q \approx p)$* **or**
3. *$\exists \langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ such that $s_2 \in \mathcal{S}$ and $\neg q \in effects(s_1)$ and $\Diamond(q \approx p)$.*

[30]Note: For the special case of single-contributor validation structures, this definition reduces to that of of e-conditions of a step defined in [8][9].

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 A.4 (Codesignation justification) *A codesignation constraint $(q \approx p) \in \mathcal{B}$ of a plan $\mathcal{P} : \langle T, O, \mathcal{B} \rangle$ 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 3.1.1) 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 A.4.1 (Separation justification) *A non-codesignation constraint $(q \not\approx p) \in \mathcal{B}$ of a plan $\mathcal{P} : \langle T, O, \mathcal{B} \rangle$ 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 A.5 (Justification of a plan w.r.t. to a validation structure) *A plan $\mathcal{P} : \langle T, O, \mathcal{B} \rangle$ 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 $c \in \mathcal{B}$ 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* [8]). These justifications can be used to *retract* superfluous constraints from the plan while preserving the correctness of the plan. For any solvable problem P, there exists a plan $\mathcal{P}$ and a validation structure $\mathcal{V}$ such that $\mathcal{P}$ is justified with respect to $\mathcal{V}$ and $\mathcal{P}$ is a complete plan for solving the problem. Thus justifying plans during planning/plan modification processes does not lead to loss of completeness. 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 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.

It is instructive to note the differences between justification and minimization. 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 [8] and plan generalization (cf. [9]) procedures, based on multi-contributor validation structures. Our current re-implementation of PRIAR plan-modification system (c.f. [8]) provides support for this.

B Proofs

Before proceeding with the proofs of the propositions in the paper, we will state and prove a useful lemma about the limitations of the partial plan representation described in Section 2.

Lemma A: *Given a partial plan $\mathcal{P}$ in the representation described in Section 2, the weakest conditions that are needed on $\mathcal{P}$ to guarantee that a step s' in $\mathcal{P}$ will precede at least one of a set of steps $\mathcal{S}$ in each of the completions of $\mathcal{P}$ is to order s' to necessarily precede one of the steps of $\mathcal{S}$.*

Proof: We want to guarantee that:

$$\forall_{\mathcal{CP} \in Completions(\mathcal{P})} \; \exists_{s \in \mathcal{S}} \; (s' \prec_{\mathcal{CP}} s) \tag{1}$$

and the lemma states that the weakest conditions required to guarantee this are:

$$\exists_{s \in \mathcal{S}} \; s' \prec s \tag{2}$$

From Section 2, we know that 2 is equivalent to

$$\exists_{s \in \mathcal{S}} \forall_{\mathcal{CP} \in Completions(\mathcal{P})} \; (s' \prec_{\mathcal{CP}} s) \tag{3}$$

It is easy to see that 3 (and thereby 2) is stronger than 1, and thus posting the constraints of 2 on the plan guarantees 1. We will now show that it is not possible to post conditions weaker than 3 to guarantee 1 in our plan representation.

To represent the constraints in 1 in terms of the ordering relation O on $\mathcal{P}$, we need the ability to post ordering constraints between sets of steps (with the semantics that a set of steps precedes another if and only if at least one of the steps of the first step precedes at least one of the steps of second step in each of the completions).

However, our partial plan representation (which is also the representation used by most of the classical planning systems), allows only ordering constraints between individual steps. Further, any ordering relation between two steps has to be obeyed by all the completions. Within these representational constraints, the weakest ordering constraints that can be posted on $\mathcal{P}$ to guarantee that s' precedes at least one element of $\mathcal{S}$ in each completion, is to ensure that s is ordered to necessarily precede at least one of the steps of $\mathcal{S}$ in $\mathcal{P}$. In other words, 2 represents the weakest ordering constraints on the plan required to guarantee 1. ■

Proposition 2.2.1 *Given a correct plan $\mathcal{P}$ and a completion $\mathcal{CP}$ of $\mathcal{P}$, every precondition p of every step w will have at least one feasible contributor, and a unique effective contributor of p to w in $\mathcal{CP}$.*

Proof: By definition, if $\mathcal{P}$ is a correct plan, then every completion $\mathcal{CP}$ must be a correct (totally ordered and totally instantiated) plan. This means that for every precondition p of step w in $\mathcal{CP}$, there must be at least one step s that precedes w and has an effect p such that, no step between s and w deletes p. Thus, there is at least one feasible contributor (in this case s) for each precondition p of each step w in $\mathcal{CP}$.

To show that there is a unique effective contributor of p to w in $\mathcal{CP}$, we provide a method for finding it: Starting from w go backward over $\mathcal{CP}$ until we find a step s' that has an effect w. The first such step s' is the unique effective contributor of p to w. To see that this is unique, we recall that $\mathcal{CP}$ is totally ordered and fully instantiated sequence of actions. To see that s' is an effective contributor, we need to see that there is no step s'' that comes between s' and w and either adds or deletes p. The first part follows from the construction--s' is the first step before w in $\mathcal{CP}$ that adds p. To see the second part -- that there is no s'' between s' and w that deletes p -- we observe that if such a step exists, then there can be no feasible contributor of p to w, violating the assumption of correctness of the plan. ■

Proposition 3.2.1 *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 $s' \prec s$.*

Proof: We will first show that the conditions stated in the property are sufficient to guarantee that $\langle \mathcal{S}, p, w\rangle$ is not violated. Let us assume that the conditions are satisfied, but that $\langle \mathcal{S}, p, w\rangle$ is violated. Then it must be the case that there exists some completion $\mathcal{CP}$ of the plan $\mathcal{P}$ such that none of the contributors in $\mathcal{S}$ can be a feasible contributor of p to w in $\mathcal{CP}$. For this to be true, it must be the case that there exists some step s' in the plan such that s' comes before w, and after all the contributors in $\mathcal{S}$, and deletes p. However, this contradicts our assumption that every deleter possibly preceding w must also necessarily precede at least one contributor $s \in \mathcal{S}$.

To show that the conditions are also necessary, we make use of the Lemma A above. Specifically, to ensure that $\langle \mathcal{S}, p, w\rangle$ is un-violated, we must ensure that at least one of the contributors in $\mathcal{S}$ can be a feasible contributor of p to w in every completion. This in turn means that every step s' which deletes p (or has an effect $\neg q$ which codesignates with p) must either be after w or before at least one contributor $s \in \mathcal{S}$ in *every* completion. That is,

$$\forall_{\mathcal{CP} \in Completions(\mathcal{P})} \Box \left(w \prec_{\mathcal{CP}} s' \vee \exists_{s \in \mathcal{S}}\ s' \prec_{\mathcal{CP}} s\right)$$

By Lemma A, the weakest conditions on $\mathcal{P}$ required to satisfy the above constraint is: $(w \prec s') \vee \exists_{s \in \mathcal{S}}\ (s' \prec s)$ ■

Proposition 3.3.1 *If a partially ordered partially instantiated plan $\mathcal{P}$ is correct by the Definition 3.3, then all of its completions are guaranteed to be correct, thus satisfying the definition of correctness given in Section 2.*

Proof: From Section 2, we know that for a completion to be correct, there should be a feasible contributor for each prerequisite of each step in the completion. We will now show that this is guaranteed for each completion of $\mathcal{P}$, if $\mathcal{P}$ is correct according to a validation structure $\mathcal{V}$. To see this, consider a completion $\mathcal{CP}$ of the plan. Let v be the last step before w in $\mathcal{CP}$ such that v deletes p. By definition 3.3, there must be a causal link $\langle \mathcal{S}, p', w\rangle$ in the plan such that $p' \approx p$.

Further, since $\langle \mathcal{S}, p', w \rangle$ is not violated, by definition 3.2 there must be a step $s \in \mathcal{S}$ such that s is a feasible contributor of p to w. ■

Proposition 4.1.1 *Given an un-violated validation $\langle \mathcal{S}, p, w \rangle$, a contributor $s \in \mathcal{S}$ is irredundant if* **either** *$\mathcal{S}$ is a singleton set,* **or** *there exists some step n in the plan such that:*

1. *$(n \prec s) \wedge (w \parallel n)$ and*
2. *$\neg d \in effects(n)$ such that $\Diamond(d \approx p)$ and*
3. *$\forall s_i \in \mathcal{S}$ if $(s_i \not= s)$ then $(n \parallel s_i)$*

(In other words, s is the only step in the plan capable of foiling the interaction caused by n.)

Proof: If $\mathcal{S}$ is a singleton, then clearly, $s \in \mathcal{S}$ is irredundant, since without it there will be no contributors in $\mathcal{S}$ which can be feasible contributors of p to w in any completion. Suppose $\mathcal{S}$ is not a singleton, but the conditions 1,2 and 3 above are true. Then there exists at least one completion in which n comes before s and after all the other contributors in $\mathcal{S}$. In that completion, s is the only feasible contributor of p to w, thus proving that s is an irredundant contributor. ■

Proposition 4.1.2 *If $\mathcal{V}$ is an un-violated 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 there is no single step $s \in \mathcal{S}$ that is a feasible contributor (see Definition 2.2) of p to w in all completions.*

Proof: To see the ''if'' portion, suppose there is a validation $\langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ such that there is no step $s \in \mathcal{S}$ that is a feasible contributor of p to w in all completions. In particular, let $\mathcal{CP}$ be a completion in which s is not a feasible contributor of p to w. In this case, since $\langle \mathcal{S}, p, w \rangle$ is supposed to be un-violated, there must exist at least one step other than s in $\mathcal{S}$ which can serve as a feasible contributor of p to w in this completion. Thus $\mathcal{S}$ cannot be a singleton.

To see the ''only if'' part, let us suppose that there is a validation $\langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ such that $\mathcal{S}$ is a non-singleton set. By our claim, this should imply that there is no step $s \in \mathcal{S}$ such that s is a feasible contributor of w to p in all completions. Suppose such an s exists. Then, we can remove every step other than s from $\mathcal{S}$ and still leave the validation un-violated. Thus, all these other contributors in $\mathcal{S}$ would be non-irredundant contributors of $\langle \mathcal{S}, p, w \rangle$. This in turn violates the assumption that $\mathcal{V}$ is an irredundant validation structure of $\mathcal{P}$. ■

Proposition 4.2.1 *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).*

Proof: If step u which satisfies the constraints of the property exists, then clearly s can never be the effective contributor of p to w in any completion since u will always follow it and add or delete p before w. ■

Proposition 4.2.2 *All the contributors of a relevant validation are necessarily unordered with respect to each other.*

Proof: Suppose there is an relevant validation $\langle \mathcal{S}, p, w\rangle$ which does not obey this property. In particular, suppose that there exist two steps $s, s' \in \mathcal{S}$ such that $s \prec s'$. Thus s' will follow s in every completion, and since s' has an effect p, s can never be an effective contributor of p to w in any completion. Thus s is an irrelevant contributor, violating the assumption that $\langle \mathcal{S}, p, w\rangle$ is a relevant validation. (This property also follows as a corollary of property 4.2.1, by selecting the step u from $\mathcal{S}$.) ■

Proposition 4.3.1 *A validation $\langle \mathcal{S}, p, w\rangle$ of a plan $\mathcal{P}$ is exhaustive if and only if $\forall n \in \mathcal{P}$, 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** *be the case that $w \prec n$* **or** *it must be the case that $\exists s \in \mathcal{S}$ such that $n \prec s$.*

Proof: To see the ''if'' part, consider a validation $\langle \mathcal{S}, p, w\rangle$ such that it satisfies the constraints of the property. Suppose $\langle \mathcal{S}, p, w\rangle$ is not exhaustive. That is, there exists at least one completion of the plan such that the effective contributor of p to w in that completion, call it n, does not belong to $\mathcal{S}$. However, according to the antecedent of the property, any step n which does not belong to $\mathcal{S}$ and has an effect codesignating with p must must either necessarily come after w, or must necessarily come before some step $s \in \mathcal{S}$. In both cases n cannot be an effective contributor of p to w in *any* completion (in the former case because it is coming after w and in the latter case because some step $s \in \mathcal{S}$, having an effect codesignating with p, comes after n). This contradicts our assumption that n does not belong to $\mathcal{S}$ and thus proves that $\langle \mathcal{S}, p, w\rangle$ is exhaustive.

To see the ''only if'' part, suppose there exists an exhaustive validation that doesn't satisfy the conditions stipulated in the property. That is, there is a step n which has an effect that codesignates with p, such that (i) n is not a contributor of the validation, (ii) it does not necessarily come after w and (iii) there is no step $s \in \mathcal{S}$ which necessarily follows n. In such a case, there exists at least one completion such that n comes before w and after all the steps of $\mathcal{S}$. Thus, in that completion, none of the steps in $\mathcal{S}$ serve as the effective contributor of p to w, violating the assumption that $\langle \mathcal{S}, p, w \rangle$ is an exhaustive validation. (Note that it is possible for step n to not necessarily precede any single step of $\mathcal{S}$, and yet be blocked by $\mathcal{S}$, such that some step of $\mathcal{S}$ will always come after n in every completion. However, by Lemma A, we know that the weakest conditions required to guarantee the latter is to make n necessarily precede at least one step of $\mathcal{S}$). ■

Proposition 4.3.2 Uniqueness of Exhaustive Validation Structures: *If $\mathcal{V}$ and $\mathcal{V}'$ are two exhaustive validation structures for a plan $\mathcal{P}$, then it must be the case that $\mathcal{V} = \mathcal{V}'$.*

Proof: Suppose $\mathcal{V}$ and $\mathcal{V}'$ are different. Then they must differ in at least one validation. Let the differing validations be $\langle \mathcal{S}, p, w \rangle \in \mathcal{V}$ and $\langle \mathcal{S}', p, w \rangle \in \mathcal{V}'$. Clearly, $\mathcal{S}$ and $\mathcal{S}'$ must differ in at least one contributor. Suppose s is such a contributor, such that $s \in \mathcal{S}$ and $s \notin \mathcal{S}'$. Since $\langle \mathcal{S}, p, w \rangle$ is also relevant, it follows that there exists at least one completion $\mathcal{CP}$, such that s is the (unique) effective contributor of p to w in that completion. Since $s \notin \mathcal{S}'$, this also means that $\langle \mathcal{S}', p, w \rangle$ does not account for the effective contributor p of w in $\mathcal{CP}$. This contradicts the assumption that $\langle \mathcal{S}', p, w \rangle$ is an exhaustive validation, and that $\mathcal{V}'$ is an exhaustive validation structure different from $\mathcal{V}$. ■

Proposition 4.3.3 *If $\mathcal{P}$ is a partially ordered plan with the single-contributor exhaustive validation structure $\mathcal{V}$, then it is possible to uniquely reconstruct $\mathcal{P}$ given only $\mathcal{V}$ and any one of $\mathcal{P}$'s completions.*

Proof: For the sake of simplicity, we shall only consider the case where $\mathcal{P}$ is a ground partially ordered plan. (The proof can be extended to partially instantiated plans in a straightforward fashion.) Suppose we are given $\mathcal{V}$, the single-contributor exhaustive validation structure, and $\mathcal{CP}$ which is a completion of $\mathcal{P}$. By definition, the set of steps in $\mathcal{P}$ is identical to the set of steps in $\mathcal{CP}$ (since $\mathcal{CP}$ is a completion of $\mathcal{P}$). Thus to reconstruct $\mathcal{P}$, we only need to reconstruct the orderings O among the steps of $\mathcal{P}$. This involves adding just enough orderings so as to make $\mathcal{V}$ the exhaustive causal structure of $\mathcal{P}$. We will split the orderings into two sets: causal orderings, and safety orderings. Causal orderings simply ensure that the contributor step of each validation precedes consumer step, and can thus be computed in a straightforward fashion. For every validation $\langle \{s\}, p, w \rangle \in \mathcal{V}$, we add the ordering $s \prec w$ to O. Safety orderings ensure that none of the validations will have threats -- i.e., steps which can intervene between the contributor and consumer of the validation, and either *add* or *delete* the condition being supported. For each validation $\langle \{s\}, p, w \rangle \in \mathcal{V}$, we can compute the requisite safety orderings as follows: Let $s' \in \mathcal{P}$

be a step such that s' has either an add or delete list literal codesignating with p. To make the validation safe, for each such s', we need to decide whether s' should come after w or before s. The choice can be uniquely determined from the completion. If s' comes before s in $\mathcal{CP}$, then we add the ordering $s' \prec s$ to O. If s' comes after w in $\mathcal{CP}$, then we add the ordering $w \prec s'$.[31]

It is easy to see that at the end of this construction $\mathcal{V}$ is the un-violated exhaustive validation structure for $\mathcal{P}$, and $\mathcal{CP}$ is one of its completions. That there is a unique such $\mathcal{P}$ is proved by the fact that orderings are all uniquely determined during construction. Adding any further orderings than are warranted by the above construction will lead to loss of relevance of $\mathcal{V}$. Similarly, removing any orderings will lead to violation of some validation belonging to $\mathcal{V}$. ■

Proposition 5.3.1 *The planning algorithms* MP *and* $\mathrm{MP\text{-}I}$ *described in Figures 4 and 5 are both complete in the technical sense of Definition 2.3.*

Proof Sketch: Our proof uses the fact that SNLP algorithm, described in [16] is provably complete (see [16] for a proof sketch). We will presently consider the completeness of MP . The only difference between the termination conditions of SNLP and MP is that the former maintains single-contributor validation structures. In otherwords, adding the additional constraint that ''each validation is a single contributor validation'' makes the termination condition of MP equivalent to that of SNLP. Thus, MP terminates for any plan for which SNLP terminates. Next, we note that the set of refinements used by MP is a *superset* of the refinements used by SNLP. In particular, with the removal of step 3.c, the algorithm MP becomes identical to the algorithm SNLP given in [16]. (Note that 3.c is the only step of the algorithm that introduces multi-contributor validations, and MakeRelevant procedure becomes an identity operation for single contributor causal structures). It follows from the above that every partial plan generated by SNLP will also be generated by MP , and that MP is guaranteed to terminate whenever SNLP does. Since SNLP is complete, this also implies that MP is complete.

In the case of $\mathrm{MP\text{-}I}$, it is still the case that the termination conditions of $\mathrm{MP\text{-}I}$ are subsumed by the termination conditions of SNLP. In particular, adding the twin restrictions that the validations should all be single-contributor and exhaustive, would make the termination criterion of $\mathrm{MP\text{-}I}$ identical to that of SNLP. Although the set of refinements used by $\mathrm{MP\text{-}I}$ are not a strict superset of those used by SNLP, they are however a superset of the refinements used by a version of SNLP that does not resolve +ve threats. It is possible to show that SNLP algorithm without +ve threat resolution (referred to as McNonlin in Section 6) is still sound and complete (c.f. [16]).

In particular, the only difference between the refinement operations of McNonlin and SNLP

[31]Note that from the definition of exhaustive validation structure and completions, s' could not have come between s and w in $\mathcal{CP}$. (Because, if it does, then $\mathcal{V}$ will not be a exhaustive validation structure of $\mathcal{P}$).

is that the former does not use +ve threat resolution. We recall that ignoring +ve threats does not affect soundness, since the plan correctness is guaranteed as long as all the -ve threats are resolved (see Proposition 3.3.1). Further, if there is a refinement branch terminating in a complete plan in SNLP's search space, then it is easy to see that ignoring all the +ve threat resolutions (i.e., ignoring the ordering and binding constraints posted by such resolutions) in that refinement branch will still leave us with a plan that satisfies the termination conditions of McNonlin (and is thus correct). In other words, McNonlin terminates for all successful refinement branches of SNLP search tree, and thus is complete.

Since McNonlin is complete, and since MP-I uses refinements that are a superset of those used by McNonlin, based on arguments similar to those used for MP , we conclude that MP-I is also complete. ■

Acknowledgements:

I would like to thank Austin Tate, who read a draft of this paper and provided many valuable suggestions and pointers; Mark Drummond who listened to some of these ideas in the early stages and provided useful feedback; 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 Dan Weld, David McAllester, Jengchin Chen, Bulusu Gopi Kumar, and the anonymous reviewers of AAAI-92 and AIPS-92 for helpful comments and criticisms.

Support for this research has been provided in part by the National Science Foundation research initiation award IRI-9210997 and the ARPA/Rome Laboratory planning initiative under grant F30602-93-C-0039 (to University of Maryland and Arizona State University).

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] J. Christensen. A hierarchical planner that generates its own hierarchies. In *Proc. Eighth Natl. Conf. on Artificial Intell (AAAI-90)*, 1990.

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

[5] 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.

[6] S. Hanks and D. Weld. Systematic Adaptation for Case-Based Planning. In *Proceedings of 1st Intl. Conf. on AI Planning Systems*, College Park, MD, 1992.

[7] J. Hertzberg and A. Horz. Towards a Theory of Conflict Detection and Resolution in Nonlinear Plans In *Proceedings of 11th IJCAI*, August 1989.

[8] S. Kambhampati and J.A. Hendler. A validation structure based theory of plan modification and reuse. *Artificial Intelligence*, 55(2-3):193-258, 1992.

[9] S. Kambhampati and S.T. Kedar. A unified framework for explanation based generalization of partially ordered partially instantiated plans. *Artificial Intelligence*, To appear in Spring 94. (A preliminary version appears in Proc. of AAAI-91).

[10] S. Kambhampati. Characterizing Multi-Contributor Causal Structures for Planning. In *Proceedings of First Intl. Conference on AI Planning Systems*, June 1992.

[11] S. Kambhampati and D.S. Nau. On the Nature and Role of Modal Truth Criteria in Planning Tech. Report. ISR-TR-93-30, Inst. for Systems Research, University of Maryland, March, 1993.

[12] S. Kambhampati. On the utility of systematicity: Understanding tradeoffs between redundancy and commitment in partial order planning. In *Proc. 13th IJCAI*, 1993.

[13] S. Kambhampati. Planning as Refinement Search: A unified framework for comparative analysis of search space size and performance. ASU-CS-TR 93-004, Dept. of Computer Sci. & Engg., Arizona State University, Tempe, AZ 85287, 1993.

[14] C. Knoblock, J. Tenenberg and Q. Yang. Characterizing abstraction hierarchies for planning. In *Proceedings of 9th AAAI*, pp. 692-697, 1991.

[15] P. Langley. Systematic and Nonsystematic search strategies. In *Proceedings of 1st Intl. Conference on AI Planning Systems*, June 1992.

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

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

[18] N.J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishers, Palo Alto, CA, 1980.

[19] E.P.D. Pednault. Synthesizing Plans that contain actions with Context-Dependent Effects *Computational Intelligence*, Vol. 4, 356-372 (1988).

[20] J.S. Penberthy and D.S. Weld. Temporal Planning with Constraints. Working Notes of AAAI Spring Symposium on Foundations of Automatic Planning: The Classical Approach and Beyond. 1993. Available as a AAAI Technical Report.

[21] 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.

[22] 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.

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

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

[25] A. Barrett and D. Weld. Partial order planning: Evaluating possible efficiency gains. Technical Report 92-05-01, Department of Computer Science and Engineering, University of Washington, Seattle, WA, June 1992.

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

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

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

[29] 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.

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

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

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

Contents

List of Figures