On the Nature and Role of Modal Truth Criteria in Planning

Subbarao Kambhampati [a,1] and Dana S. Nau [b]

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

[b] *Department of Computer Science, Institute for Systems Research, and Institute for Advanced Computer Studies. University of Maryland, College Park, Maryland 20742*

Abstract

Chapman's paper, ''Planning for Conjunctive Goals,'' has been widely acknowledged for its contribution toward understanding the nature of partial-order planning, and it has been one of the bases of later work by others---but it is not free of problems. This paper addresses some problems involving modal truth and the Modal Truth Criterion (MTC). Our results are as follows:

(i) Even though modal duality is a fundamental property of classical modal logics, it does not hold for modal truth in Chapman's plans; i.e., ''necessarily p'' is not equivalent to ''not possibly $\neg p$.''

(ii) Although the MTC for necessary truth is correct, the MTC for possible truth is incorrect: it provides necessary but *insufficient* conditions for ensuring possible truth. Furthermore, even though necessary truth can be determined in polynomial time, possible truth is NP-hard.

(iii) If we rewrite the MTC to talk about modal *conditional* truth (i.e., modal truth conditional on executability) rather than modal truth, then both the MTC for necessary conditional truth and the MTC for possible conditional truth are correct; and both can be computed in polynomial time.

(iv) The MTC plays a different role in plan generation than it does in checking the correctness of plans, and this has led to several misconceptions about the MTC. Several researchers have mistakenly attempted to simplify the MTC by eliminating the white-knight declobbering clause from it; and others have used Chapman's results to conjecture that partial-order planning will not scale up to more expressive action representations. We point out that these ideas are misconceptions, and explain why.

[1] Corresponding author's Fax: (602) 965-2751, E-mail: rao@asu.edu.

1 Introduction

Chapman's paper, ''Planning for Conjunctive Goals,'' [2] has been widely acknowledged as an important step towards formalizing partial-order planning, [2] and it has been one of the bases of later work by others (for example, [7,10,8,12,17,31,34]). Unfortunately, however, Chapman's work is not free of problems, and this has led to confusion about the meaning of his results. Previous papers [7,8,16,34] have pointed out several of these problems.

One of the fundamental concepts used by Chapman is the idea of modal truth in plans. We will discuss the details of this concept later---but a simple version of it is that if P is a partially ordered, partially instantiated plan and p is a ground literal, then p is *necessarily* (or *possibly*) true in P's final situation if for every (or some) totally ordered ground instance P' of P, p is true after executing P'. Chapman's Modal Truth Criterion (MTC) purports to give necessary and sufficient conditions for ensuring that p is necessarily or possibly true. As we describe below, this paper addresses several problems with modal truth and the MTC.

Modal Duality and the MTC. Chapman explicitly states and proves the MTC for necessary truth, and claims that by modal duality (i.e., the equivalence of ''necessarily p'' and ''not possibly $\neg p$''), the MTC for possible truth is obtained via a simple rewording of the MTC for necessary truth. But in this paper, we show that although modal duality is a fundamental property of classical modal logics, it does *not* hold for modal truth in Chapman's plans. This has several consequences:

(i) The MTC for possible truth is not completely correct: it provides necessary but insufficient conditions for ensuring possible truth. Furthermore, although necessary truth in plans can be computed in polynomial time as pointed out by Chapman, [3] the same is not true for possible truth. Instead, the problem of computing possible truth in plans is NP-hard. [4]

(ii) We can define a concept called *modal conditional truth*, which is similar to modal truth but does not require that a plan be executable as modal truth does. Necessary conditional truth and possible conditional truth *are* duals of each other, and both can be computed in polynomial time. Furthermore, if we rewrite the MTC to talk about modal conditional truth rather than modal truth,

[2] *Partial-order planning* is planning by searching in the space of partially ordered, partially instantiated plans. We prefer not to use the more common term ''nonlinear planning,'' because it appears to mean different things to different people.

[3] There are some difficulties with Chapman's proof of this, but these difficulties have been cleared up by Nebel and Backstrom [25].

[4] If modal duality held, then both necessary truth and possible would be at similar levels of complexity: either both would be polynomial, or one would be NP-hard and the other co-NP-hard. Section 3.2.2 discusses some formulations of planning in which this occurs.

then both the MTC for necessary conditional truth *and* the MTC for possible conditional truth are correct.

The Role of the MTC in Plan Generation. The MTC plays a different role in plan generation than it does in checking the correctness of plans. In particular, the MTC provides both necessary and sufficient conditions for necessary truth---but it is possible to write sound and complete partial-order planners which use only *sufficient* but not *necessary* conditions for necessary truth.

This has led to a number of misconceptions about the MTC. Several researchers have mistakenly attempted to simplify the MTC by eliminating the white-knight declobbering clause [5] from it. Others (including Chapman) (c.f. [2,23,35]) have used Chapman's results to conjecture that partial-order planning will not scale up to more expressive action representations. In this paper, we explain why both of these notions are incorrect---and observe that at the root of the confusion lies the peculiar predicament of partial-order planners, which search in the space of partially ordered partially instantiated plans, but need completeness only in the space of totally ordered ground plans.

This paper is organized as follows. Section 2 contains basic definitions, and clarifications and corrections of some of Chapman's terminology. Section 3 presents results about modal duality, the complexity of modal truth, and the modal truth criterion, and compares and contrasts these results with Chapman's claims, as well as with other related work. Section 4 discusses and clarifies the misconceptions regarding the role of modal truth criterion in plan generation vs. verification of plan correctness. Section 5 contains concluding remarks. Complicated proofs appear in the appendix; simpler proofs are in the body of the paper.

2 Definitions

Below, we have tried to be as compatible as possible with Chapman's ''TWEAK-style'' plans, situations, and modal truth. However, at certain points, technical problems have forced us to adopt a different approach. At those points, we explain how our approach differs and why.

[5] For the benefit of those unfamiliar with this clause, our presentation of the MTC in Section 3.2.1 points it out explicitly.

2.1 Basics

The planning language $\mathcal{L}$ is any function-free first-order language.[6] Since $\mathcal{L}$ is function-free, every term is either a variable symbol or a constant symbol, and thus every ground term is a constant symbol. We follow the usual convention of defining an *atom* to be a predicate symbol followed a list of terms, a *literal* to be an atom or its negation, and a *proposition* to be a 0-ary atom. *Thus, what Chapman calls a proposition, we call a literal.*

A *state* is any finite collection of ground atoms of $\mathcal{L}$. If a state s contains a ground atom p, then p is true in s and $\neg p$ is false in s; otherwise p is false in s and $\neg p$ is true in s. Thus, a state is simply an Herbrand interpretation for the language $\mathcal{L}$, and hence each formula of first-order logic is either satisfied or not satisfied in s according to the usual first-order logic definition of satisfaction.

If T is a finite set of terms, then a *codesignation constraint* on T is a syntactic expression of the form '$t \approx u$' or '$t \not\approx u$', where $t, u \in T$. Let D be a set of codesignation constraints on T, and θ be a *ground substitution* over T (i.e., a substitution that assigns a ground term to each variable in T). Then θ *satisfies* D if $t\theta = u\theta$ for every syntactic expression '$t \approx u$' in D, and $t\theta \neq u\theta$ for every syntactic expression '$t \not\approx u$' in D. D is *consistent* if there is at least one ground substitution θ that satisfies D. If $t\theta = u\theta$ for every θ that satisfies D, then t *codesignates with* u.[7]

A *step* is a triple $a = (\text{name}(a), \text{pre}(a), \text{post}(a))$, where name($a$) is a constant symbol called a's *name*, and pre(a) and post(a) are collections of literals called a's *preconditions* and *postconditions*.[8] If A is a set of steps, then an *ordering constraint* on A is a syntactic expression of the form '$a \prec b$' (read as ''a *precedes* b''), where $a, b \in A$. If O is a set of ordering constraints on A and $\prec$ is a total ordering on A, then $\prec$ *satisfies* O if for every syntactic expression '$a \prec b$' in O, $a \prec b$.

A *partially ordered, partially instantiated plan* (or more succinctly, a *plan*) is

[6] Conventional first-order languages contain only finitely many constant symbols, but Chapman requires his planning language to contain an infinite number of constant symbols. For compatibility with Chapman's work, we will assume that $\mathcal{L}$ contains infinitely many constant symbols---but in Section 3.2.2 we will discuss what happens if the number of constant symbols is finite.

[7] If D is consistent, then this is equivalent to saying that t codesignates with u iff '$t \approx u$' is in D's transitive closure.

[8] Informally, we will use the terms a and name(a) interchangeably. Chapman's definition of a step actually omits name(a) completely---but as pointed out by McAllester and Rosenblitt [22], unless we give unique names to steps, it is impossible for a plan to contain two distinct steps that have the same preconditions and postconditions.

a 4-tuple $P = (s_0, A, D, O)$, where s_0 is a state called P's *initial* state, A is a set of steps, D is a set of codesignation constraints on the terms of P (i.e., the terms in s_0 and A), and O is a set of ordering constraints on the steps of A. P is *complete* if it is totally ordered and ground, i.e., if there is a unique total ordering $a_1 \prec a_2 \prec \ldots \prec a_n$ over A that satisfies O, and a unique ground substitution θ over the terms of P that satisfies D. If P is not complete, then it is *incomplete*. If P is complete, we will often write P informally as $a_1 \prec a_2 \prec \ldots \prec a_n$.

Suppose that P is a complete plan, and let k be the largest integer $\leq n$ for which there are states $s_1, s_2, \ldots, s_k$ such that the following properties are satisfied for $1 \leq i \leq k$:

(i) s_{i-1} satisfies a_i's preconditions; i.e., $p\theta$ is true in s_{i-1} for every literal $p \in \text{pre}(a_i)$.
(ii) s_i is the state produced by performing the step a_i in the state s_{i-1}; i.e., $s_i = (s_{i-1} - f_i) \cup t_i$, where t_i is the set of all ground atoms $p\theta$ such that $p \in \text{post}(a_i)$, and f_i is the set of all negated ground atoms $p\theta$ such that $p \in \text{post}(a_i)$.

Then for $1 \leq i \leq k$, a_i is *executable* in the *input* state s_{i-1}, *producing* the *output* state s_i. If $k = n$, then P is *executable*, and it *produces* the *final* state s_n.

A plan $P' = (s'_0, A', D', O')$ is a *constrainment* of a plan $P = (s_0, A, D, O)$ if $s'_0 = s_0$, $A' = A$, $O \subseteq O'$, and $D \subseteq D'$. A *completion* of P is any constrainment of P that is complete.[9] P is *consistent* if it has at least one completion; otherwise P is *inconsistent*.

2.2 Situations, Truth, and Modality

Chapman defines a *situation* to be a collection of literals.[10] Given a literal p and a situation s, he defines p to be true if it codesignates with a literal in s, and false if it codesignates with the negation of a literal in s. Chapman also makes the following definitions [2, p. 338]:

> A plan has an *initial situation*, which is a set of [literals] describing the world at the time that the plan is to be executed, and a *final situation*, which describes the state of the world after the whole plan has been executed. Associated with

[9] Chapman's definition of a completion does not make it entirely clear whether a completion of P should include only the steps in P, or allow other steps to be added. However, other statements in his paper make it clear that he means for a completion to include only the steps in P, so this is how we and most others (e.g., [17,34]) use the term.
[10] He calls them propositions---but as mentioned at the beginning of Section 2.1, we call them literals instead.

each step in a plan its *input situation*, which is the set of [literals] that are true in the world just before it is executed, and its *output situation*, which is the set of [literals] that are true in the world just after it is executed. In a complete plan, the input situation of each step is the same as the output situation of the previous step. The final situation of a complete plan has the same set of [literals] in it as the output situation of the last step. ... A [literal] is denied in a situation if its negation is asserted there. It is illegal for a [literal] to be both denied and asserted in a situation.

This approach leads to several difficulties:

(i) As pointed out by Yang and Tenenberg [34], if a plan P is not complete, then its situations are ill-defined. For example, suppose P consists of two unordered steps a and b, such that a asserts p and denies q, and b asserts q and denies p. Then P's final situation is either $\{p\}$ or $\{q\}$, depending on which completion of P we choose.

(ii) If a situation contains literals that are not completely ground, then what those literals mean is problematic. For example, suppose a plan's initial situation contains the literal $p(x)$, where x is a variable symbol. This cannot mean $(\forall x)p(x)$, because Chapman's TWEAK planner may later constrain $x \not\approx y$ for some constant or variable y. It cannot mean $(\exists x)p(x)$, because TWEAK may later constrain $x \approx y$. Apparently, it means $p(x)$ for some undetermined x, and TWEAK gets to choose what x is. In other words, if the initial situation contains any variables, then TWEAK changes the meaning of the initial situation as it goes along.

Thus, rather than using Chapman's approach, we define plans using STRIPS-style states of the world, and then define situations in terms of states. The intent of our definitions is that if a plan is complete and can be executed at least far enough to reach the situation s, then s corresponds to some state t that arises while executing the plan; and what is true and false in s is precisely what is true and false in t.[11] Otherwise, *nothing* is true or false in s although certain things may be *conditionally* true or false (as defined below). These ideas are formalized below.

If P is a plan, then associated with each step a of P are two symbols $\mathsf{in}(a)$ and $\mathsf{out}(a)$, called a's *input* and *output* situations. Associated with P are symbols init and fin called the *initial* and *final* situations of P. All of these symbols must be distinct. Whenever $a \prec b$, we will also say that $x \prec y$, where x may be a or $\mathsf{in}(a)$ or $\mathsf{out}(a)$, and y may be b or $\mathsf{in}(b)$ or $\mathsf{out}(b)$. [12]

[11] From our definition of a state earlier, the truth value of every literal p is known in the state t, and hence in the situation s. This differs from Chapman's formulation of a situation, in which the truth value of p is unknown in s unless s explicitly contains something codesignating with p or $\neg p$.

[12] According to this definition, $\mathsf{out}(a)$ and $\mathsf{in}(b)$ are always distinct; hence we would say $\mathsf{out}(a) \prec \mathsf{in}(b)$ in some cases where Chapman would say $\mathsf{out}(a) = \mathsf{in}(b)$. However, this

We now define what is *true* and *false* in a situation of a complete plan. Let P be a complete plan, and p be a ground literal. Then p is true in init if p is true in P's initial state, and p is true in fin if p is true in P's final state. If a is an executable step of P, then p is true in in(a_i) (or out(a_i)) if p is true in a's input state (or output state, respectively). A ground literal p is false in a situation s iff $\neg p$ is true in s. Note that if P is not executable, then the law of the excluded middle does not apply, for p will be neither true nor false in P's final situation.

As a consequence of the above definitions, it follows that p is true in s (which we write as $\mathcal{M}(p, s)$) iff the following three conditions are satisfied:

Establishment: either p codesignates with a postcondition of some step a such that $a \prec s$, or else $p \in s_0$.
Nondeletion: for all steps b between a (or s_0) and s, no postcondition of b codesignates with $\neg p$. [13]
Executability: every step that precedes s is executable.

A closely related concept is *conditional truth*, which is like ordinary truth except that it does not require executability: p is *conditionally true* in s (which we write symbolically as $\mathcal{C}(p, s)$) iff the establishment and nondeletion conditions hold.

We defined truth and conditional truth only for complete plans, because for incomplete plans, what is true or conditionally true will vary depending on which completion we choose. In incomplete plans, we instead need to talk about *modal* truth, which Chapman defines as follows [2, p. 336]:

> I will say ''*necessarily p*'' if p is true of all completions of an incomplete plan, and ''*possibly p*'' if p is true of some completion.

Above, Chapman apparently means p to be nearly any statement about a plan: examples in his paper include not only statements about specific literals and situations in the plan, but also statements about the entire plan (e.g., the statement [2, p. 341] that a plan ''necessarily solves the problem''). However, unless we place some restrictions on the nature of p, this has some dubious results---for example, if P is an incomplete plan, then all completions of P are complete, and therefore P itself is necessarily complete. Therefore, for the formal results in the paper, we will use ''necessarily'' and ''possibly'' only in the following cases (although we will sometimes use them informally in a broader sense). If p is an atom, P is a plan, and s is a situation in P, then:

-- p is *necessarily* (or *possibly*) true in s (written $\Box\mathcal{M}(p, s)$ and $\Diamond\mathcal{M}(p, s)$, respectively) iff $\mathcal{M}(p, s)$ in every (or some) completion of P;

---

makes no significant difference in any of the results.

[13] This is basically the white-knight declobbering clause of the MTC (see Section 3.2.1), simplified to handle the special case where the plan is complete.

Table 1
Relationships among truth, modal truth and modal conditional truth of a literal p in the final situation of a plan P.

Complete plans	**Incomplete plans**
Truth $\mathcal{M}(p, \mathsf{fin})$: P is executable, *and* produces p in fin. This requires that the establishment, non-deletion and executability conditions hold.	**Nec. Truth** $\Box\mathcal{M}(p, \mathsf{fin})$: *every* completion of P is executable, *and* produces p in fin. **Poss. Truth** $\Diamond\mathcal{M}(p, \mathsf{fin})$: *some* completion of P is executable, *and* produces p in fin.
Conditional Truth $\mathcal{C}(p, \mathsf{fin})$: if P's steps did not have preconditions, then P would have produced p in fin. This requires the establishment and non-deletion conditions, but *not* the executability condition.	**Nec. Cond. Truth** $\Box\mathcal{C}(p, \mathsf{fin})$: if P's steps did not have preconditions, then *every* completion of P *would* have produced p in fin. **Poss. Cond. Truth** $\Diamond\mathcal{C}(p, \mathsf{fin})$: if P's steps did not have preconditions, then *some* completion of P would have produced p in fin.

-- p is *necessarily* (or *possibly*) conditionally true in s (written $\Box\mathcal{C}(p, s)$ and $\Diamond\mathcal{C}(p, s)$, respectively) iff $\mathcal{C}(p, s)$ in every (or some) completion of P.

Table 1 summarizes the relationships among truth, modal truth and modal conditional truth.

We now define the following decision problems:

NECESSARY TRUTH: given a ground literal p and a plan P, is p necessarily true in P's final situation fin? Or equivalently, does every completion of P produce a final state in which p is true?

POSSIBLE TRUTH: given a ground literal p and a plan P, is p possibly true in P's final situation fin? Or equivalently, is there a completion of P that produces a final state in which p is true?

NECESSARY CONDITIONAL TRUTH: given a ground literal p and a plan P, is p necessarily conditionally true in P's final situation fin?

POSSIBLE CONDITIONAL TRUTH: given a ground literal p and a plan P, is p possibly conditionally true in P's final situation fin?

3 Modal Duality and the Complexity of Modal Truth

3.1 Our Results

From the definitions of truth and conditional truth in the previous section, it is easy to see that in each completion of a plan P, a literal p is true in the final situation fin if and only if (1) p is conditionally true in fin, and (2) the completion itself is executable. The completion is executable if and only if for every action a and every precondition p_a of a, p_a is conditionally true in the situation $\mathsf{in}(a)$. Thus,

$$\mathcal{M}(p, \mathsf{fin}) \equiv \left[\mathcal{C}(p, \mathsf{fin}) \wedge \underbrace{\left[\bigwedge_{\forall a \in P, \forall p_a \in \mathrm{pre}(a)} \mathcal{C}(p_a, \mathsf{in}(a)) \right]}_{\text{Executability of the completion}} \right], \tag{1}$$

Since ensuring that p is necessarily true in P's final situation is equivalent to ensuring that p is true in the final situation of every completion of P, we have:

$$\Box\mathcal{M}(p, \mathsf{fin}) \equiv \Box \left[\mathcal{C}(p, \mathsf{fin}) \wedge \left[\bigwedge_{\forall a \in P, \forall p_a \in \mathrm{pre}(a)} \mathcal{C}(p_a, \mathsf{in}(a)) \right] \right]. \tag{2}$$

Now, since modal necessity distributes over conjunctions (i.e., $\Box(p \wedge q) \equiv \Box(p) \wedge \Box(q)$), we can rewrite Eq. 2 as

$$\Box\mathcal{M}(p, \mathsf{fin}) \equiv \left[\Box\mathcal{C}(p, \mathsf{fin}) \wedge \underbrace{\left[\bigwedge_{\forall a \in P, \forall p_a \in \mathrm{pre}(a)} \Box\mathcal{C}(p_a, \mathsf{in}(a)) \right]}_{\text{Executability of all completions}} \right]. \tag{3}$$

Thus we can determine whether p is necessarily true by checking to see whether it is necessarily conditionally true, and whether the preconditions of each step of the plan are necessarily conditionally true. This can be done in polynomial time using the same technique Chapman suggests for computing the MTC in [2]. In particular, computing the necessary conditional truth of a literal in a situation (which involves checking whether the MTC's establishment and declobbering clauses are consistent with the plan's ordering and codesignation/non-codesignation constraints) can be done in time polynomial ($O(n^3)$) in the plan length. Thus, since the total number of preconditions in a plan is of the order of number of actions in the plan, computing whether p is necessarily true can also be done in polynomial time.

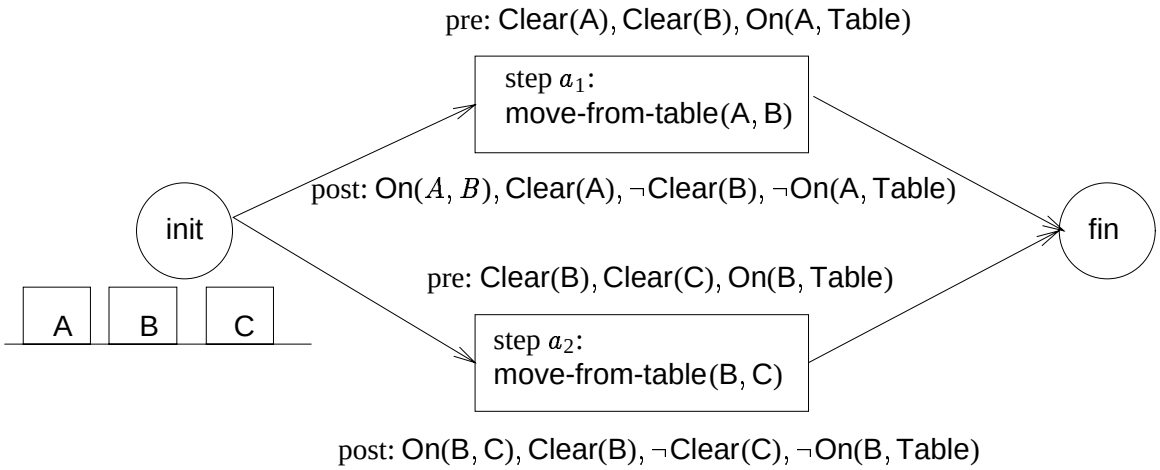


Fig. 1. A blocks-world plan illustrating the non-duality of necessary truth and possible truth in TWEAK-style plans. As shown in the drawing, the following atoms are true in the initial situation: On(A, Table), On(B, Table), On(C, Table), Clear(A), Clear(B), Clear(C). The operator move-from-table(x, y) moves block x from the table to block y. The preconditions and postconditions of each step are shown above and below the box that represents the step.

Coming to the case of possible truth, by similar arguments we get:

$$\Diamond\mathcal{M}(p, \mathsf{fin}) \equiv \Diamond\left[\mathcal{C}(p, \mathsf{fin}) \wedge \left[\bigwedge_{\forall a \in P, \forall p_a \in \mathrm{pre}(a)} \mathcal{C}(p_a, \mathsf{in}(a))\right]\right]. \tag{4}$$

But possible truth does not distribute over conjunctions (i.e., in general, $\Diamond(p \wedge q) \not\equiv \Diamond(p) \wedge \Diamond(q)$), so there is no way to simplify Eq. 4 into component tests of computing possible conditional truth of individual literals. Thus, even though possible conditional truth in fin and necessary conditional truth in fin are duals of each other (i.e., $\Diamond\mathcal{C}(p, \mathsf{fin}) \equiv \neg\Box\neg\mathcal{C}(p, \mathsf{fin})$), possible truth in fin and necessary truth in fin are *not* duals of each other. More specifically:

Theorem 1 *There exist a ground literal p and a plan P such that in P's final situation* fin, $\Box\mathcal{M}(p, \mathsf{fin}) \not\equiv \neg\Diamond\neg\mathcal{M}(p, \mathsf{fin})$.

Proof. Consider the blocks-world plan shown in Fig. 1. This plan has only one executable completion, namely $a_2 \prec a_1$. This completion produces a final state in which A is on B and B is on C. Thus, no executable completion produces a final state where ¬On(A, B) is true, so ¬On(A, B) is not possibly true in the final situation fin. If possible truth and necessary truth were duals, then this would mean that On(A, B) is necessarily true in fin. However, On(A, B) is not necessarily true in fin, because the plan contains an *unexecutable* completion, namely $a_1 \prec a_2$. Thus possible truth and necessary truth are not duals. □

Thus, unlike necessary conditional truth and possible conditional truth, necessary truth and possible truth do not obey the modal duality that is obeyed by all classical modal logics [4, p. 62], and thus do not define a well-formed modal logic. It is easy to understand why this is so. The semantics of modal logics are based on Kripke structures (a.k.a. possible worlds). In this formulation, if p is a ground literal, then for every possible world, p must either be true or false in that world. For partially ordered plans, one might expect that each completion of the plan would give rise to a possible world. However, the modal truth of p in a situation of a plan requires that the plan's actions be executable in order to produce that situation. Thus, if a completion is not executable, then the truth of p is not defined in the corresponding possible world. [14]

Given a ground literal p and a plan P, p is possibly true in P's final situation if and only if there is an executable completion of P that produces a final state in which p is true, and this happens iff it is not the case that every executable completion of P produces a final state in which $\neg p$ is true. Thus, POSSIBLE TRUTH is the dual of the following problem:

PARTIAL TRUTH: given a ground literal p and a plan P, does every executable completion of P produce a final state in which p is true? [15]

Lemma 1 of the appendix shows that PARTIAL TRUTH is NP-hard.

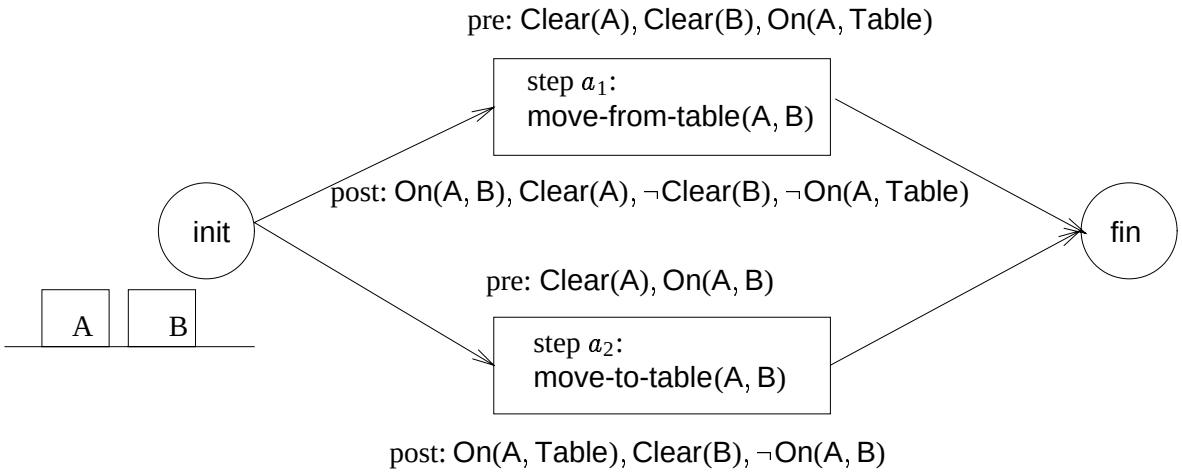
PARTIAL TRUTH is a weaker condition than both NECESSARY TRUTH and NECESSARY CONDITIONAL TRUTH. [16] The example in Fig. 2 illustrates this. There are some cases in which every executable completion of P produces a final state in which p is true, but p is neither necessarily true nor necessarily conditionally true in P's final situation.

Another way of understanding the problem with simplifying Eq. 4 is to note that if p is possibly conditionally true and that all the preconditions of the preceding actions are possibly conditionally true, this only implies that each of them is *individually*

[14] Although TWEAK plans cannot be modeled using the semantics of classical modal logics, they can be modeled in a variant of modal logics, called *first order dynamic logic* [32]. Dynamic logic, which has been used to provide semantics for programs and plans, provides a clean way to separate executability/termination conditions from goal satisfaction conditions. More about this in Section 3.2.2.

[15] PARTIAL TRUTH corresponds closely to the notion of partial correctness, which was studied in connection with dynamic-logic-based modeling of computer programming languages [28,32].

[16] It may seem at first glance that partial truth rather than necessary truth should form the basis of planning, since we are only interested in the executable completions of the plan. However, if a planner returned a plan whose conditions were all partially true rather than necessarily true, this would require the user of the plan to spend time trying to figure out which of the possibly exponential number of completions is actually executable.



Property	**Atoms with that property**
Necessarily true (produced by all completions)	*none*
Necessarily conditionally true (produced by all completions, if preconditions are stripped off)	Clear(A), On(B, Table)
Partially true (produced by all executable completions)	Clear(A), On(A, Table), Clear(B), On(B, Table)

Fig. 2. Example showing that partial truth is weaker than both necessary truth and necessary conditional truth. The table shows which atoms satisfy various truth conditions in fin.

true in at least one completion---and this condition is *necessary* but *insufficient* for ensuring possible truth. We could check possible truth by checking to see whether all these conditions are *collectively* true in at least one completion of the plan, but since the number of completions of a plan is exponential in the number of actions of the plan, this would take exponential time. Furthermore, the following theorem (proved in the appendix) shows that unless P=NP, there is no polynomial-time approach for solving this problem.

Theorem 2 POSSIBLE TRUTH *is NP-hard.*

Thus, NECESSARY TRUTH and POSSIBLE TRUTH have different levels of complexity. If modal duality held, then this would not be so, for each would be reducible to the other's complement via an equivalence of the form $\Diamond\mathcal{M}(p,s) \equiv \neg\Box\neg\mathcal{M}(p,s)$. Thus it would follow [9, p. 29] that either POSSIBLE TRUTH would be polynomial like NECESSARY TRUTH, or else NECESSARY TRUTH would be co-NP-hard. In Section 3.2.2, we discuss some planning situations where this occurs.

3.2 Comparison with Other Work

3.2.1 The Modal Truth Criterion

Chapman states the MTC as follows [2, p. 340]:

> **Modal Truth Criterion.** A [literal] p is necessarily true in a situation s iff two conditions hold:[17] there is a situation t equal or necessarily previous to s in which p is necessarily asserted; and for every step C possibly before s and every [literal] q possibly codesignating with p which C denies, there is a step W necessarily between C and s which asserts r, a [literal] such that r and p codesignate whenever p and q codesignate. The criterion for possible truth is exactly analogous, with all the modalities switched (read ''necessary'' for ''possible'' and vice versa).

If we take these words literally, then the definition of modal truth tells us that the plan must be modally executable. This is consistent with Chapman's definition of a situation (quoted in Section 2.2), from which it follows that a step's output situation (and hence what is true in that situation) is only defined if the step can be executed. However, a careful look at Chapman's proof of necessity and sufficiency of his MTC reveals that his proof deals with necessary *conditional* truth rather than necessary truth.[18] In proving that any literal with an establisher and no clobberer must be necessarily true, Chapman's proof refers to white-knight steps for every potential clobberer, [2, p. 370], without checking that the white knights are in fact executable.[19]

For the ''necessary truth'' version of the MTC, this does not affect the validity of Chapman's proof, since executability occurs naturally as a consequence of applying necessary conditional truth recursively to prerequisites of all preceding steps. The same, however, cannot be guaranteed for possible truth, since modal possibility does not commute over conjunctions---and thus Chapman's proof cannot be extended to possible truth. In particular, the following theorem shows that the ''possible truth'' version of the MTC sometimes fails:

Theorem 3 *There is a plan P and a ground literal p such that in P's final situation, p is not possibly true but the MTC concludes otherwise.*

---

[17] The second of these conditions is the ''white-knight declobbering clause'' that we refer to elsewhere. The steps C and W are often called a *clobberer* and a *white knight*, respectively. The situation t can be either the initial situation or the output situation of some step e, and in the latter case e is often called an *establisher*.

[18] Had Chapman explicitly noted this use of modal conditional truth in his proof, we believe he would have noticed the non-duality of necessary and possible truths.

[19] Note that in Chapman's terminology, the establisher is a *situation*, while clobberers and white knights are *steps*.

Proof. In the blocks-world plan in Fig. 2, consider the condition On(A, B), which holds in out(a_1). a_2 deletes On(A, B), but a_1 adds On(A, B), and a_1 can possibly come after a_2. Thus, the MTC would conclude that On(A, B) is possibly true in fin. In this case the MTC is incorrect, since there is no executable completion of the plan for which On(A, B) is true in the final state. [20] □

The MTC and modal conditional truth. The above discussion suggests an alternative interpretation of the MTC that sidesteps the problem: drop the executability requirement, and interpret the MTC as a statement about modal *conditional* truth rather than modal truth. This alternative interpretation is not as far-fetched as it might sound. To see this, note that Chapman defines the notion of truth of a literal in a situation as follows [2, p. 338]:

> A [literal] is true in a situation if it codesignates with a [literal] that is a member of the situation. A step asserts a [literal] in its output situation if the [literal] codesignates with a postcondition of the step.

Here, there is no explicit requirement that the step be executable. This suggests that the MTC does not require that P be modally executable, and thus suggests that Chapman was talking about modal conditional truth. This interpretation is also consistent with his ''nondeterministic achievement procedure'' [2, Fig. 7], where to make a literal necessarily true in a situation, he only ensures establishment and declobbering without explicitly stating that the establisher needs to be executable. (As explained above, for the case of necessary truth, executability follows from making every prerequisite of every action necessarily conditionally true.)

The ''conditional truth'' interpretation of the MTC gives a quasi-local flavor to planning, by separating the process of ensuring local establishment and declobbering from the process of ensuring executability, with the understanding that if all preconditions are necessarily established and declobbered, then the whole plan itself will be executable and correct. In fact, some latter rewrites of the MTC (e.g. [34,17]) use this interpretation to eliminate the notion of situations entirely, and state the MTC solely in terms of steps (operators) and their preconditions and postconditions.

Although a truth criterion for modal conditional truth does have utility in plan generation, it is of limited utility in projecting plans or partially ordered events. As mentioned in Section 2.2, the latter are more naturally related to modal truth.

[20] Note, however, that On(A, B) is possibly conditionally true in fin, because if the steps had no preconditions, then one of the completions would have produced On(A, B).

3.2.2 Modal Duality and Universal Executability

In Section 3.1, we observed that the main reason why necessary truth and possible truth are not duals in TWEAK-style plans is that such plans can contain unexecutable completions. Thus, one way to achieve duality between necessary truth and possible truth is to restrict our attention to plans whose completions are always executable. One way to guarantee that plans will always be executable is to restrict the actions to have no preconditions, i.e., to consider only those plans P such that pre(a) = $\emptyset$ for every step a of P. In this case, it is easy to see that Equations 2 and 4 in Section 3.1 will simplify respectively to:

$$\Box\mathcal{M}(p, s) \equiv \Box\mathcal{C}(p, s); \tag{5}$$
$$\Diamond\mathcal{M}(p, s) \equiv \Diamond\mathcal{C}(p, s). \tag{6}$$

In other words, for the set of plans composed entirely of precondition-less steps, modal truth and modal conditional truth are identical, necessary truth and possible truth are duals, and all are computable in polynomial time.

The above approach to achieving universal executability is clearly too restrictive, since it precludes modeling actions with any form of preconditions. But if we relax the restrictions of TWEAK-style action representation, there is a more reasonable way to guarantee universal executability: let an action a be executable even if its preconditions are not satisfied. If the preconditions are satisfied, then a will produce its postconditions; otherwise, a will simply have no effects. For plans that contain only this type of action, possible truth and necessary truth are duals of one another, computation of possible truth is NP-hard, and computation of necessary truth is co-NP-hard. Fig. 3 summarizes the complexity relations among the various decision problems. As discussed below, this approach has been used in different forms by several different researchers.

Propositional Dynamic Logic. To our knowledge, the above approach was first used in Rosenchein's work [32] on providing semantics to plans based on first-order propositional dynamic logic. Propositional Dynamic Logic (PDL) is a variant of modal logic, which was originally designed to provide semantics to computer programs [28]. In PDL, the semantics of a program are described in terms of what will be necessarily and possibly true after the execution of that program. A program is said to be totally correct if (a) it halts, and (b) whenever it halts, certain goal propositions will be true in the resulting world. Programs that only satisfy condition b are said to be partially correct. (Note the similarity between partial correctness and PARTIAL TRUTH). In using PDL to provide semantics to plans, Rosenchein guarantees universal executability of plans by starting with a loop-free subset of PDL, and restricting it further to allow only the so-called C-programs. C-programs restrict the use of conditionals in PDL to guarantee that the plan terminates irrespective of which branch of the conditional it takes.

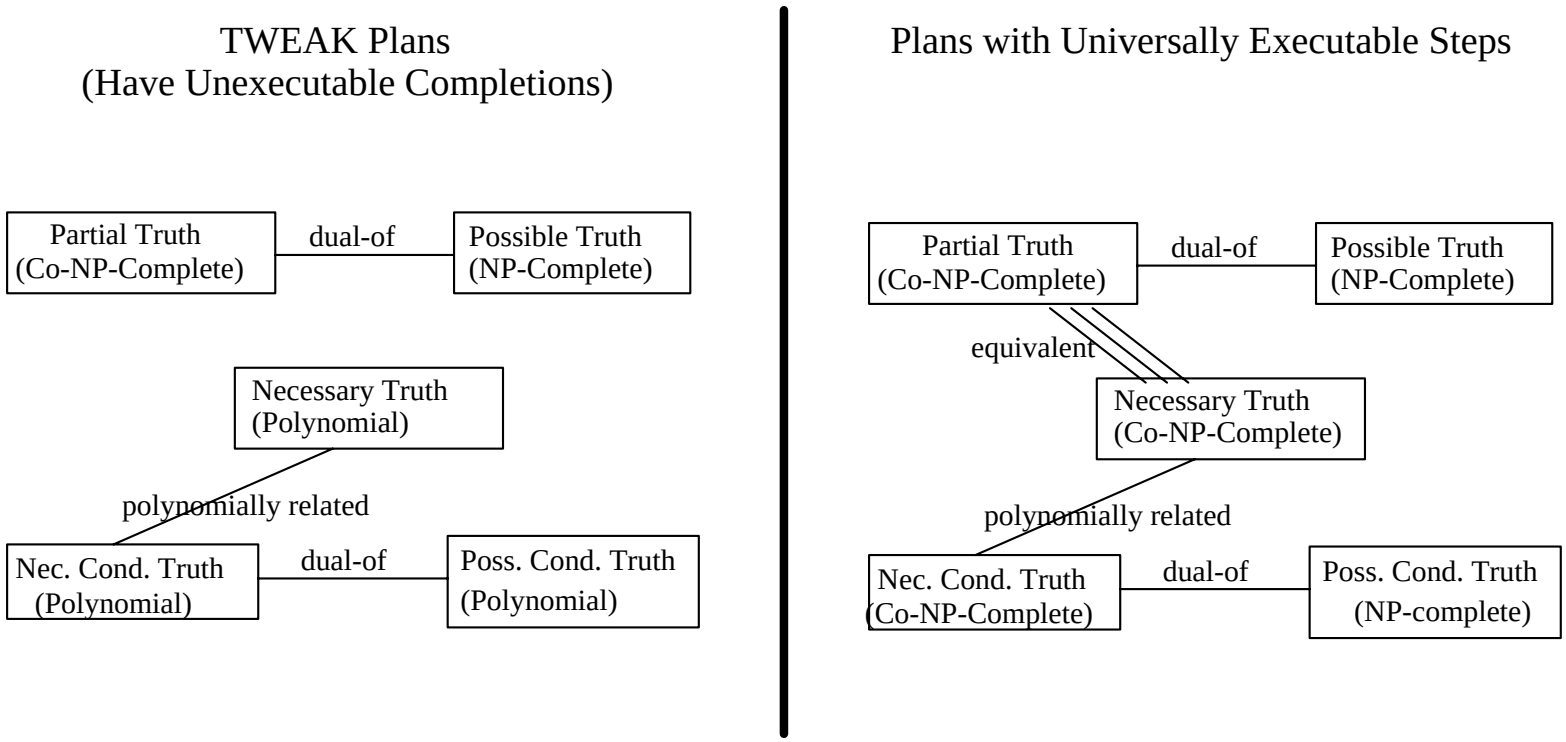


Fig. 3. Complexity relations among decision problems for plans containing actions that have executability preconditions, and plans containing universally executable actions.

Temporal Projection. A very similar idea is used in Dean and Boddy's work on temporal projection [5]. Specifically, they use actions that have ground preconditions and effects. The effects of the actions are defined in terms of projection rules, which are of the form $\langle e, \phi, \alpha, \delta \rangle$, where e is an event with which the rule is associated, and ϕ is a set of antecedent conditions, which if true before e, will cause the α conditions to be added and the δ conditions to be deleted. Dean and Boddy are concerned with the following decision problem: given a partially ordered set of events A, does a condition C belong to Possible(e), where the latter is the set of conditions that hold immediately following the event e in some totally ordered ground instance (i.e., completion) of A.

In Dean and Boddy's formulation, A is executable even when a rule's preconditions don't hold (in which case the rule simply has no effect). Thus as discussed earlier, possible truth is equivalent to possible conditional truth, necessary truth is equivalent to necessary conditional truth, and possible truth and necessary truth are duals. Hence they are able to prove that in their formalism, determining possible truth is NP-hard and determining necessary truth is co-NP-hard.

Conditional Steps. Chapman uses universally executable actions (he calls them conditional steps) in proving his intractability theorem for actions containing conditional effects. Specifically, Chapman defines a conditional step as follows [2, p. 371]:

> A conditional step is always applicable, but has two sets of postconditions, the if-true and the if-false postconditions. The if-true postconditions hold in the output situation if all the preconditions were satisfied in the input situation; otherwise the if-false postconditions hold.

Since these conditional steps are always applicable, a plan composed entirely of such steps will always be executable. Thus, just as in Dean and Boddy's formalism, determining necessary truth is co-NP-hard, as shown by Chapman in the proof of his Intractability Theorem.

Since the Intractability Theorem is based on planning operators that have conditional effects, it has been natural for planning researchers to interpret it to mean that the conditionality of these operators is what causes necessary truth to be intractable. However, this interpretation is misleading. The intractability result depends just as much on the universal executability of Chapman's conditional steps as it does on their conditionality. Below we explain why.

Consider an incomplete plan P composed of ordinary ''unconditional'' steps as defined in Section 2, and let a be a step of P such that pre(a) and post(a) contain an unconstrained variable x. Then for the purposes of both planning and temporal projection, the effects of a are to some extent conditional. In particular, depending

on what value we give to x, a will have different effects in different completions of P. However, computing necessary truth in such plans is still polynomial. Since Chapman's planning language has an infinite number of constant symbols, it follows that in the plan P we can *always* find a binding for x that makes a unexecutable. As a consequence, P will always have at least one unexecutable completion. Hence, determining necessary truth is trivial: nothing will be necessarily true in P's final situation.

Now, suppose we restrict our planning language $\mathcal{L}$ to contain only finitely many constant symbols (and thus only finitely many ground terms, since $\mathcal{L}$ is function-free). Then there will be some plans in which a is executable for every binding of x. In this case, as the following theorem shows, checking necessary truth will be co-NP-hard, even with unconditional steps.

Theorem 4 *If the language $\mathcal{L}$ contains only finitely many constant symbols, then* NECESSARY TRUTH *is co-NP-hard.*

Notice that this result is related to Chapman's observation [2, p. 356] that restricting the range of a variable to a finite set will defeat the MTC, and make constraint computations NP-complete.

Coherent Plans. Nebel and Backstrom [25] have recently studied the computational complexity of plan validation and temporal projection. While our investigation was initially motivated by the apparent lack of modal duality in Chapman's MTC, Nebel and Backstrom's work is motivated by the apparent asymmetry between the complexity of plan validation as studied by Chapman, and temporal projection as studied by Dean and Boddy [5].

Although Nebel and Backstrom's results are related to ours, there are several significant differences. Rather than interpret the MTC in terms of modal conditional truth and use that to explain the asymmetry in the possible and necessary truth, as we have done in this paper, Nebel and Backstrom instead chose to restrict the MTC to apply only to plans whose completions are all executable (they call this property *coherence*). Furthermore, they restricted their plans to be ground (i.e., to contain no variables), and our plans do not have this restriction. We believe that the results in this paper complement theirs and together provide a coherent interpretation of the role of modal truth criteria in planning.

4 The Role of the MTC in Partial-Order Planning

Chapman's original motivation for the formulation of the MTC was to provide a formal basis for partial-order planning. Intuitively, since the MTC accounts for all

the scenarios in which a proposition p is necessarily true in a situation, we can make p necessarily true by simply adding constraints to the plan to *make* one of those scenarios true. While this use of the MTC provides a *sufficient* formal basis for partial-order planning, it turns out not to be *necessary*. More specifically, sound and complete partial-order planners:

-- *do not need to* reason about the correctness of arbitrary partially ordered plans.
-- *do not need to* consider only those goals that are not necessarily true for achievement.
-- *do not need to* base their goal achievement procedure on a necessary and sufficient truth criterion for partially ordered plans.

These seemingly counter-intuitive facts are a result of the somewhat peculiar predicament of partial-order planners: they search in a space of plans that are partially ordered and partially instantiated, to find a totally ordered ground plan that solves the problem. This has lead to several confusions about the role of the MTC in planning. In this section, we will clarify the role played by the MTC in partial-order planning, and then address some misconceptions that resulted from misunderstandings in this regard.

4.1 Unnecessity of the MTC for Partial-Order Planning

Many planning algorithms can be thought of as repeated iterations of the following steps: take a plan, evaluate it to see if it is a solution, and if it is not, then refine it further using a goal achievement procedure. For example, Chapman describes TWEAK, a planner based on his MTC, as follows[2]:

> ''[The planner] enters a loop in which some goal *not yet achieved* is chosen and the [goal achievement] procedure is applied'' (p. 344; emphasis ours)
>
> ''The goal achievement procedure is derived by interpreting the necessary truth criterion as a nondeterministic procedure. The criterion tells us all the ways a proposition could be necessarily true; the procedure chooses one of them and modifies the plan accordingly.'' (p. 341)

In the above, the MTC plays three separate roles: as a termination criterion (the planning stops when all the goals are necessarily correct), as a goal selection criterion (only conditions that are not necessarily true are selected for goal achievement), and as the basis for the nondeterministic goal achievement procedure.

Although the MTC is sufficient for serving these roles, it is not *required* for either of them---it is possible to provide a formal basis for partial-order planning without recourse to the MTC (c.f. [27]). To see this, we must start with a clear understanding of the objectives of partial-order planning. For both partial-order and total-order planning, the objective is to find a ground operator sequence which when executed

in the initial state produces a desired goal state. In the case of partial-order planning, the search is conducted in the space of partially ordered plans for efficiency of plan generation, and each partially ordered plan is simply a shorthand representation for the set of all ground operator sequences that are consistent with the plan's constraints.[21] This means that one can do partial-order planning without having to reason about the ''correctness'' of partially ordered plans. Let us now re-examine how crucial the MTC is for termination, goal-selection and goal achievement procedures of a partial-order planner.

4.1.1 Termination

Since the objective of partial-order planning is to find plans that are ground and totally ordered, it follows that for sound and complete partial-order planning, it is sufficient for the termination condition to be capable of checking the correctness of totally ordered ground plans. Rather than using Chapman's MTC for this purpose, a termination condition such as the one given below can be used instead:

> **Eager Termination:** Randomly generate a completion (ground linearization) of the current plan. If the completion solves the problem the planner is trying to solve, then terminate the planner and return the completion.

The eager termination criterion is always tractable, since a completion can be enumerated in polynomial time, and can be checked for correctness in polynomial time. Furthermore, a planner using this termination criterion will terminate on any incomplete plan on which a planner using MTC terminates. This is because whenever a plan satisfies the MTC, all of its completions (including the randomly generated one) are guaranteed to solve the problem the planner is trying to solve. Finally, the eager termination criterion may allow the planner to terminate earlier than the MTC would, because the MTC is not satisfied unless all completions of the current plan solve the problem.

4.1.2 Goal Selection

Since the order in which goals are selected does not affect the completeness of partial-order planning [22,18], MTC-based goal selection is just one of many possible goal selection strategies. Goal selection based on the MTC essentially boils down to preferring to work on preconditions of the plan that are not necessarily true, with the rationale that such a strategy may allow the planner to exploit any serendipitously satisfied goals by terminating without having to explicitly work on achieving them.[22] Replacing this strategy or complementing it with other

[21] See [18,19] for an elaboration of this view.

[22] The occurrence of such a serendipitously satisfied goal has been referred to in [11] as an enabling-condition interaction.

goal selection strategies does not affect the soundness and completeness of the underlying planner.

4.1.3 Goal Achievement

The last and perhaps most important role of the MTC in plan generation in TWEAK is as the basis for the goal achievement procedure. Although Chapman's interpretation of the MTC as a nondeterministic program provides a sufficient basis for goal achievement, it is once again not *necessary*. In fact, many partial-order planners, such as McAllester's SNLP [22,15,24], use a goal achievement procedure that corresponds to a modified MTC in which the white-knight declobbering clause is replaced with a much simpler demotion clause:

> A literal p is necessarily true in a situation s if p is necessarily asserted in a situation s' which necessarily precedes s, and p is necessarily not deleted by any step a that possibly comes between s' and s.

With this modification, the truth criterion is sufficient but unnecessary for ensuring the truth of a literal (see Section 4.2 for an example and related discussion). A planner using this modified truth criterion can still be complete, because for the special case of totally ordered plans, this criterion is equivalent to the MTC, and provides sufficient as well as necessary conditions for determining whether the plan solves the problem. To put it another way, for every plan P that is correct according to Chapman's MTC, there will be a constrainment P' of P that will be correct according to this modified truth criterion such that whenever TWEAK terminates with P, the planner using the modified truth criterion will terminate with the constrainment P'.

4.2 The Role of White-Knight Declobbering

4.2.1 White-Knight Declobbering in Checking Plan Correctness

Starting with the fact that one does not need the full power of Chapman's MTC in order to do sound and complete partial-order planning, some researchers have attempted to simplify the MTC by eliminating the white-knight clause from it. Unfortunately, such a simplification is erroneous. As Chapman's proof shows, something similar to the white-knight clause is still required if we want to state the necessary and sufficient conditions for the necessary truth of a literal in a given partially ordered plan (or equivalently, recognize the correctness of a given partially ordered plan) in *polynomial time*.

If we don't care about polynomial time, then we can simply enumerate all the completions of the plan, and verify that each completion is a correct totally

ordered ground plan for solving the problem. But since the number of completions of a partially ordered plan is exponential in the size of the plan, this is very inefficient unless the plan is already totally ordered---in which case one can use a ''nondeletion'' condition similar to that we discussed in Section 2.2.

Although the necessity of the white-knight declobbering clause does depend on whether or not the plan is totally ordered, it does not depend on whether or not the plan is ground. (Chapman's use of a partially instantiated plan [2, Fig. 5, p. 339] to motivate white-knight declobbering seems to have caused this misimpression.) The following example, due to Mark Drummond [6], illustrates this point. Consider the ground partially ordered plan in Fig. 4, in which the literal p is required in the final situation fin, the steps b_1 and b_2 add p, the steps a_1 and a_2 delete p, and the steps a_1 and b_1 are unordered with respect to a_2 and b_2. We can see that p is true in the situation preceding fin in every completion of this plan. However, without the white-knight clause, the modal truth criterion would not be able to recognize this fact. [23]

Since the MTC can be used to determine efficiently (in polynomial time) whether all the completions of an arbitrary partially ordered plan are correct, it can also be used as a basis for removing any unnecessary orderings in a given plan in polynomial time [16,1]: repeatedly remove some (non-transitive) ordering relation from the plan, and check if all its completions are still correct. [24] Such ''order generalization'' could be useful if one wants to execute steps of the plan in parallel in order to improve the execution time. It could also be useful when one wants to separate independent subparts of the plan to facilitate storage compactions in case-based approaches (c.f. [20]).

[23] *Historical Note:* Although the term ''white knight'' became popular after Chapman's work on TWEAK [2], Tate's Nonlin was the first planner to use a white-knight clause to specify weakest conditions for establishment and declobbering. Nonlin's Q&A procedure [33] says that a literal p is true at a step s in a partially ordered plan, if and only if (1) there exists a step n' such that $n' \prec n$, and n' asserts p, and (2) there does not exist a step n'' such that n'' deletes p, and (2.1) either n'' is unordered with respect to n or (2.2) there does not exist any step $n''' \prec n$ which deletes p without a subsequent node $w \succ n'''$ asserting it back again. According to this criterion, the plan in Fig. 4 is found to be correct. In Nonlin, this check is done by ensuring that (1) for every branch of the plan that is coming into s, the last node in the branch that gives a value to p must be asserting p and (2) no branch parallel to s contains a node that deletes p. Unlike TWEAK, Nonlin did not deal with partially instantiated plans (however, O-PLAN [3], a successor of Nonlin, does deal with such plans).

[24] Note that this involves removing existing orderings, without adding any new orderings. Backstrom [1] shows that if we also allow arbitrary addition and deletion of orderings, then the problem of finding the least constrained plan is NP-hard.

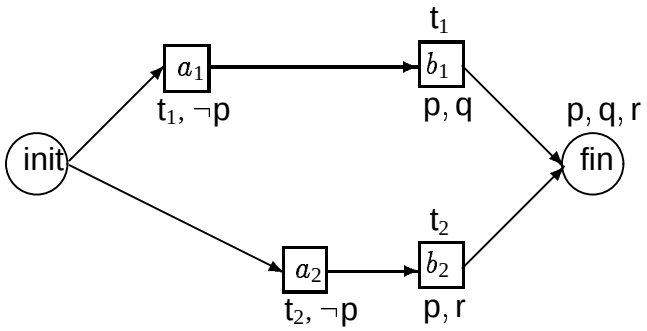


Fig. 4. A ground partially ordered plan for which the white-knight clause is needed to verify plan correctness (example due to Mark Drummond). Each step's name is in a box, with its preconditions and postconditions above and below the box. fin is the final situation, and init is the initial situation.

4.2.2 White-Knight Declobbering in Goal Achievement

Although the white-knight declobbering clause is needed in the MTC, we pointed out in Section 4.1.3 that white-knight declobbering is not required to guide plan generation. In fact, Chapman's own implementation of TWEAK [2, p. 361], as well as many later partial order planners such as SNLP [22] do not use white-knight declobbering clause in the goal-achievement procedure. However, as Chapman remarks [2, p. 359], avoiding the white-knight declobbering clause during planning means that the planner may terminate with somewhat more constrained plans. For the example in Fig. 4, a planner such as SNLP that does not use the white-knight declobbering clause will find one of two alternate plans:

$$P_w^1 : \ a_1 \prec b_1 \prec a_2 \prec b_2;$$
$$P_w^2 : \ a_2 \prec b_2 \prec a_1 \prec b_1.$$

Note that both of these plans are constrainments of the original plan. However, since the objective of partial-order planning is only to find *a* ground operator sequence, terminating with these constrainments in itself is not a problem, unless there is a concomitant loss of efficiency in planning. [25]

[25] A related question is whether the MTC, and in particular the white-knight declobbering clause, would be necessary in the goal achievement procedure if one wants to find optimal partially ordered plans (for example, to ensure optimal execution time, c.f. [21]). At first glance, it might seem that we must search in the space of all partially ordered plans to find the optimal partially ordered plan, and thus the white-knight declobbering clause would be necessary. However, this is not strictly required, since the unnecessary ordering constraints can be removed from plans in polynomial time (Section 4.2.1). In particular, suppose we defined the cost of each plan P to be the execution time that would be needed for P if all unnecessary ordering constraints were removed. If this cost were used as part of an admissible search strategy in a planner like SNLP [22] or even a total-order planner, then the planner would terminate with some plan P_c such that P_c is a constrainment of an optimal partially ordered plan P_s. We could then derive P_s from P_c by once again removing all unnecessary ordering constraints from P_c.

This gives rise to the question of whether the use of white-knight declobbering will improve or reduce the efficiency of plan generation of a partial-order planner. The use of white-knight declobbering as part of the goal-achievement procedure does tend to increase the redundancy in the search space (in particular, the same ground operator sequence may be considered in more than one search branch of the planner)---but as we discuss in [14,15], whether or not such redundancy leads to inefficiency depends on the tradeoffs between the search space redundancy vs. level of commitment made by the planner. Thus using the white-knight declobbering clause does not *ipso facto* make planning inefficient, as has been conjectured by some researchers (c.f. [13]). An important issue is whether the planner implements white-knight declobbering only through steps that already exist in the plan, or whether it also allows new steps to be introduced as white-knights into the plan. In [15], we describe a planner called MP-I, which allows white-knight declobbering only via already existing steps. Our experiments show that this opportunistic declobbering leads to significant performance improvements in certain domains.

4.3 Plan Generation with More Expressive Action Representations

One unfortunate result of the misinterpretation of the role of the MTC in partial-order planning is the misconception that partial-order planning has more difficulty in scaling up to a more expressive action representation than does total-order planning. Not only did this belief slow down progress on planning with expressive action representations, it also inhibited some learning researchers from basing their work on partial-order planning frameworks [35,23,20]

As we noted earlier, planning can be seen as an iterative process, such that during each iteration the planner takes a plan, evaluates it to see if it is a solution, and refines it further (using the goal achievement procedure) if it is not. The argument about the disadvantages of partial order planning for expressive action representation is based on the complexity of each of these iterations. It starts with Chapman's Intractability Theorem, which shows that if conditional steps are allowed, then necessary truth is NP-hard. [26] The argument goes that since a planner must compute necessary truth each time it evaluates and refines a plan, the amount of time taken per iteration will increase drastically in partial order planning.

This reasoning is fallacious, since, as we noted earlier, determining necessary truth is *not required* in order to do partial-order planning. In particular, as we discussed in Section 4.1, a sound and complete partial-order planner will not have to compute necessary truth either for termination or for goal selection. Thus, the NP-hardness result is clearly irrelevant. Indeed, Pednault [26,27] provides a formal theory of partial-order planning in the presence of actions with conditional and quantified

[26] However, as we discussed in Section 3.2.2, the NP-hardness depends as much on the universal executability of these steps as it does on the conditionality of their effects.

effects,[27] and his theory has served as the basis for a popular implementation called UCPOP [30], that takes only a polynomial amount of time per iteration.

Moreover, even if one were to compute necessary truth during goal selection and termination, it is possible to devise partial-order planners in which each plan that is generated is constrained in such a way that necessary truth can be evaluated in polynomial time. One extreme example of this would be a planner that generates only ground linear plans, but there are however other types of constrainments in which the plans are partially ordered---for example, unambiguous constrainments (c.f. [24]), and safe constrainments (c.f. [22,18])---which avoid the extreme of searching with totally ordered plans. All of these attempt to reduce the cost of plan evaluation and refinement by possibly increasing the search space size.

In order to determine the overall time complexity of the planner, what really counts not the time per iteration, but the tradeoff between the time per iteration and the size of the space searched (i.e., the number of iterations), since the time complexity is the product of these two factors. In [18,19], we systematically classify the types of operations (called ''tractability refinements'') used by various planners to ensure tractable plan evaluation and analyze the tradeoffs offered by them.

Finally, it is also wrong to believe that planning itself is more difficult if conditional operators are allowed. Erol *et al.* [7] have analyzed how the complexity of planning varies under a wide variety of conditions, including whether or not function symbols, negative preconditions, or delete lists (i.e., negative postconditions) are allowed, whether or not the predicates are propositional (i.e., 0-ary), and whether the planning operators are part of the input or fixed in advance. In all of these cases, the presence or absence of conditional operators made no difference in the complexity or decidability of planning.

5 Concluding Remarks

In this paper, we have discussed several misconceptions regarding the role of modal truth and the Modal Truth Criterion (MTC) in planning. Along the way, we have also clarified and corrected several problems with Chapman's terminology.

First, we have presented the following results about modal truth and the modal truth criterion:

(i) Contrary to Chapman's statement, the principle of modal duality that is obeyed by all classical modal logics is not obeyed in TWEAK-style plans. The lack of

[27] Instead of checking for necessary truth, Pednault's theory of planning concentrates on adding a sufficient number of constraints (including steps, orderings, bindings, and secondary preconditions) to ensure necessary truth in the resulting plan.

duality between necessary truth and possible truth is related to the fact that modal truth of a literal in a situation of a plan requires that the plan's actions be executable in order to produce that situation. To achieve modal duality, one needs universally executable plans.

(ii) Even though necessary truth in plans can be determined in polynomial time as stated by Chapman, the same statement does not hold for possible truth. Instead, the problem of determining possible truth in plans is NP-hard. This is important because checking possible truth has several applications in plan projection [5] as well as plan generalization [16].

(iii) As stated by Chapman, the MTC is correct only as a criterion for necessary truth (not as a criterion for possible truth). However, if we reinterpret it as a criterion for modal *conditional* truth (i.e., modal truth conditional on plan executability), then it is correct as a criterion for both necessary conditional truth and possible conditional truth.

Second, we clarified the role of the MTC in plan generation vs. checking the correctness of a given plan, by emphasizing the peculiar predicament of partial-order planners: they search in the space of partially ordered partially instantiated plans, but need completeness only in the space of totally ordered and totally instantiated plans. We showed that misunderstandings in this regard have been the root of several of the confusions regarding the role of the MTC:

(i) Sound and complete partial-order planning is possible as long as the goal achievement procedure is based on a truth criterion that is consistent with necessary and sufficient truth criterion for totally orderered plans.

(ii) Although the MTC provides a sufficient basis for partial-order planning, it is *not necessary* for sound and complete partial-order planning. Specifically, it is possible to devise sound and complete partial-order planners whose termination, goal selection and goal achievement procedures do not depend upon Chapman's MTC.

(iii) Although the white-knight declobbering clause of the MTC is needed in order to provide both necessary and sufficient conditions for ensuring truth of a literal in a partially ordered plan, white-knight declobbering is not required for partial-order planning. Several sound and complete partial-order planners use demotion instead.

(iv) Although Chapman proved that is NP-hard to verify necessary truth in plans whose steps have conditional effects, this does not necessarily imply (as has been conjectured elsewhere) that partial-order planners are any worse off than total-order planners in dealing with actions that have conditional effects. There are two reasons for this: (1) partial-order planners do not have to compute necessary truth in order to be sound and complete, and (2) even if a partial-order planner does compute necessary truth, it can be sound and complete while only generating plans for which necessary truth can be computed in polynomial time.

Because of the wide impact of Chapman's paper, it is important to correct any misimpressions that may result from it. We hope readers will find this paper useful for that purpose.

Acknowledgements:

We wish to thank Christer Backstrom, John Bresina, Mark Drummond, Kutluhan Erol, Jim Hendler, Craig Knoblock, Ed Pednault, Arunabha Sen, Austin Tate and V. S. Subrahmanian for their helpful criticisms and comments.

Kambhampati's research is supported in part by an NSF Research Initiation Award IRI-9210997, an NSF Young Investigator Award IRI-9457634, and ARPA/Rome Laboratory planning initiative under grant F30602-93-C-0039. Nau's research is supported in part by NSF grants IRI-8907890, NSFD CDR 88003012, and NSF EEC 94-02384. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or ARPA.

References

[1] C. Backstrom. Finding least-constrained plans and optimal parallel executions is harder than we thought. In *Proc. 2nd European Workshop on Planning*, 1993.n

[2] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333--379, 1987.

[3] K. Currie and A. Tate. O-Plan: The Open Planning Architecture. *Artificial Intelligence*, 51(1), 1991.

[4] E. Davis. *Representations of Commonsense Knowledge* Morgan Kaufmann Publishers, Inc. San Mateo, California, USA, 94403.

[5] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36:375-399, 1988.

[6] M. Drummond. Private Communication. 1991 .

[7] K. Erol, D. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 1994. To appear.

[8] K. Erol, D. Nau, and V. S. Subrahmanian. When is planning decidable? In *Proc. First Internat. Conf. AI Planning Systems*, pp. 222--227, June 1992.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness* W.H. Freeman and Company, New York, 1979.

[10] M. L. Ginsberg. What is a modal truth criterion? Unpublished manuscript, November 1990.

[11] N. Gupta and D.S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence*, 56(2-3):223--254, August 1992.

[12] S. Hanks and D. S. Weld. Systematic adaptation for case-based planning. In *Proc. First Internat. Conf. AI Planning Systems*, pp. 96--105, June 1992.

[13] E. Jacopin and C. Le Pape and J.F. Puget. A Theoretical Analysis of the ''uselessness'' of white-knights. Intitut Blaise Pascal, Technical Report 92/27.

[14] S. Kambhampati. On the utility of systematicity: Understanding tradeoffs between redundancy and commitment in partial ordering planning. In *Proc. 13th Intl. Joint Conf. on Artificial Intelligence*, August 1993.

[15] S. Kambhampati. Multi-Contributor causal structures for Planning: A Formalization and Evaluation. *Artificial Intelligence*, Vol. 69, 1994. pp. 235-278.

[16] S. Kambhampati and S. Kedar. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence*, Vol. 67, No. 1, pp. 29-70, 1994.

[17] S. Kambhampati and S. Kedar. Explanation-based generalization of partially ordered plans. In *AAAI-91*, pp. 679--685, July 1991.

[18] S. Kambhampati. Refinement search as a unifying framework for analyzing planning algorithms. In *Proc. 4th Intl. Conf. on Ppls. of KR & R (KR-94)*, May 1994.

[19] S. Kambhampati, C. Knoblock and Q. Yang. Planning as Refinement Search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence*, Special Issue on Planning and Scheduling. 1995. To appear.

[20] S. Kambhampati and J. Chen. Relative utility of EBG based plan reuse in total ordering vs. partial ordering planning. In *Proc. AAAI-93*, July 1993.

[21] C.A. Knoblock. Generating parallel execution plans with a partial-order planner. In *Proc. AIPS-94*, June 1994.

[22] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *AAAI-91*, pp. 634--639, July 1991.

[23] D. McDermott. Regression Planning. *Internat. Jour. Intelligent Systems*, 6:357-416, 1991.

[24] S. Minton, M. Drummond, J. Bresina, and A. Philips. Total order vs. partial order planning: Factors influencing performance. In *Proc. KR-92*, 1992.

[25] B. Nebel and C. Backstrom. On the computational complexity of temporal projection, planning and plan validation. *Artificial Intelligence*, Vol. 56, No. 1, 1994.

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

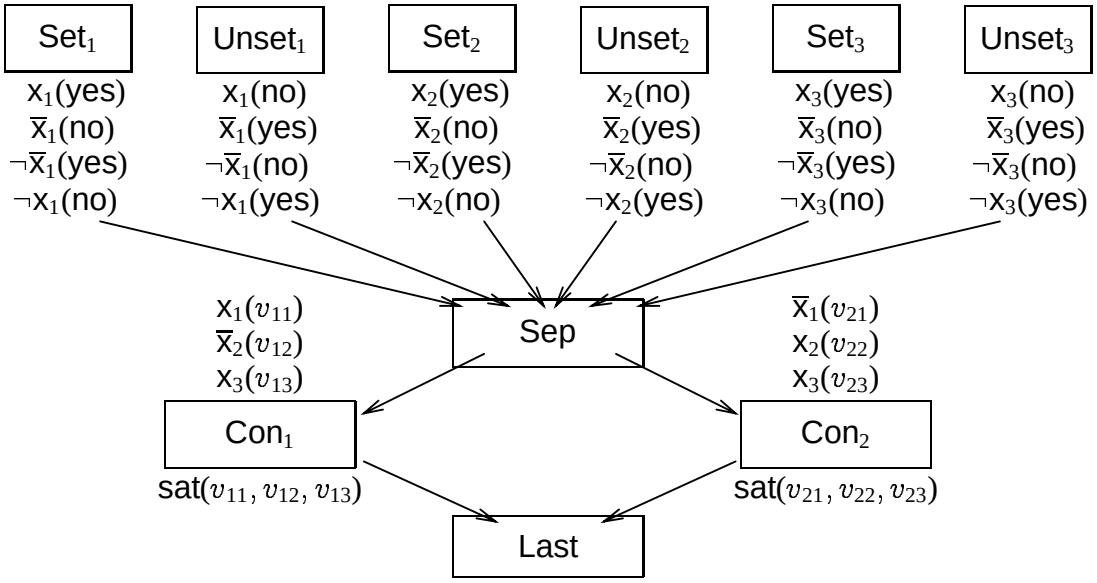


Fig. 5. An example of the plan P_X^* in the case where $X = x_1\overline{x_2}x_3 + \overline{x_1}x_2x_3$. Each step's name is in a box, with its preconditions and postconditions above and below the box.

[27] E.P.D. Pednault. Generalizing nonlinear planning to handle complex goals and actions with context-dependent effects. In *Proc. IJCAI-91*, 1991.

[28] V. Pratt. Semantical considerations on Floyd-Hoare Logic. In *Proc. 17th FOCS*, 109-121.

[29] D. Nau. On the complexity of possible truth. In *AAAI Spring Symposium*, April 1993.

[30] J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. KR-92*, 1992.

[31] M. A. Peot. Conditional nonlinear planning. In *Proc. First International Conference on AI Planning Systems*, pp. 189--197, 1992.

[32] S. Rosenchein. Plan Synthesis: A logical perspective. In *Proc. IJCAI-81*, pp. 331-337, 1981.

[33] A. Tate. Generating project networks. In *Proc. 5th International Joint Conference on Artificial Intelligence*, pp. 888--893, 1977.

[34] Q. Yang and J. D. Tenenberg. Abtweak: Abstracting a nonlinear, least commitment planner. In *AAAI-90*, pp. 204--209, 1990.

[35] M. M. Veloso, M. A. Perez, and J. G. Carbonell. Nonlinear planning with parallel resource allocation. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 207--212, November 1990.

[36] M. M. Veloso. *Learning by Analogical Reasoning in General Problem Solving*. PhD thesis, Carnegie-Mellon University, 1992. (CMU-CS-92-174).

Appendix: Proofs

Lemma 1 PARTIAL TRUTH is co-NP-hard.

Proof. The proof is by reduction from the complement of 3SAT (satisfiability with three literals per clause). In particular, let $X = c_1 + c_2 + \ldots + c_m$ be a DNF formula over the Boolean variables $x_1, x_2, \ldots, x_n$, where each c_i is a conjunct of three literals $c_i = l_{i1}l_{i2}l_{i3}$. We encode X as a plan P_X^* and a ground atom $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$, such that every executable completion of P_X^* produces a final situation containing $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff X is a tautology. P_X^* is the following plan (see Fig. 5):

Initial state. P_X^*'s initial state s_0 is the empty set.

Steps. For each Boolean variable x_i, there are two steps, Set_i and Unset_i.

Set_i has no preconditions, and has the following postconditions:

$$\neg\overline{\mathsf{x}}_i(\mathsf{yes}), \overline{\mathsf{x}}_i(\mathsf{no}), \neg\mathsf{x}_i(\mathsf{no}), \mathsf{x}_i(\mathsf{yes}).$$

Unset_i has no preconditions, and has the following postconditions:

$$\overline{\mathsf{x}}_i(\mathsf{yes}), \neg\overline{\mathsf{x}}_i(\mathsf{no}), \mathsf{x}_i(\mathsf{no}), \neg\mathsf{x}_i(\mathsf{yes}).$$

Here, yes and no are constant symbols; the interpretations of $\mathsf{x}_i(\mathsf{yes})$, $\overline{\mathsf{x}}_i(\mathsf{no})$, $\overline{\mathsf{x}}_i(\mathsf{yes})$, and $\mathsf{x}_i(\mathsf{no})$ are that the Boolean variable x_i is true, not false, false, and not true, respectively. Thus, the interpretations of Set_i and Unset_i are that they make x_i true and false, respectively.

There is a step Sep, which has no preconditions nor postconditions.[28] Its only purpose is to separate the steps Set_i and Unset_i (defined above) from the steps Con_i defined below.

For each conjunct $c_i = l_{i1}l_{i2}l_{i3}$ in X, there is a step Con_i. Corresponding to the literals in c_i, Con_i has preconditions L_{i1}, L_{i2}, L_{i3}, as follows. Each l_{ij} is either x_k or $\overline{x_k}$ for some x_k. If $l_{ij} = x_k$, then L_{ij} is $\mathsf{x}_k(v_{ij})$, where v_{ij} is a variable; if $l_{ij} = \overline{x_k}$, then L_{ij} is $\overline{\mathsf{x}}_k(v_{ij})$. Con_i has one postcondition: $\mathsf{sat}(v_{i1}, v_{i2}, v_{i3})$.

The interpretation of $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is that X is satisfied. For any other constant symbols u, v, w, $\mathsf{sat}(u, v, w)$ has no particular interpretation. Thus, the interpretation of Con_i is that if $c_i = l_{i1}l_{i2}l_{i3}$ is satisfied, then Con_i asserts that X is satisfied.

[28] To prove his Intractability Theorem, Chapman also uses steps that have no preconditions and postconditions. However, this raises the question of whether TWEAK can ever create a plan such as P_X^*. It is easy to modify P_X^* so that TWEAK will construct it; here's how. For each step a of P_X^*, add a new postcondition $\mathsf{done}(\mathrm{name}(a))$ (recall that name(a) is a constant symbol). For each ordering constraint '$a \prec b$' of P_X^*, give b a new precondition $\mathsf{done}(\mathrm{name}(a))$.

There is one other step, Last, which has no preconditions and no postconditions. Last's purpose is to provide a final step in the plan.

Constraints. O contains an ordering constraint '$\mathsf{Set}_i \prec \mathsf{Sep}$' for every Set_i, an ordering constraint '$\mathsf{Unset}_i \prec \mathsf{Sep}$' for every Unset_i, and ordering constraints '$\mathsf{Sep} \prec \mathsf{Con}_i$' and '$\mathsf{Con}_i \prec \mathsf{Last}$' for every Con_i. There are no other ordering constraints. There are no codesignation constraints; i.e., $D = \emptyset$.

Let P be any executable completion of P_X^*, and θ be the unique ground substitution that satisfies P's codesignation constraints. In P, Sep's input and output states are a set s of ground atoms corresponding to truth values for all the x_i's. More specifically, $s = s_1 \cup s_2 \cup \ldots \cup s_n$, where each s_k is either $\{\overline{\mathsf{x}}_k(\mathsf{yes}), \mathsf{x}_k(\mathsf{no})\}$ (meaning x_k is false), or $\{\overline{\mathsf{x}}_k(\mathsf{yes}), \mathsf{x}_k(\mathsf{no})\}$ (meaning x_k is true).

The input state for each Con_i consists of some ground atoms of the form $\mathsf{sat}(u, v, w)$, plus the set s described above. Since Con_i is executable, each precondition L_{ij} of Con_i codesignates with an atom in Con_i's input state. In particular, since each L_{ij} is either $\mathsf{x}_k(v_{ij})$ or $\overline{\mathsf{x}}_k(v_{ij})$ for some k, it follows that $L_{ij}\theta \in s_k$. Thus, either $v_{ij}\theta = \mathsf{yes}$ or $v_{ij}\theta = \mathsf{no}$, depending on whether s_k corresponds to a truth value for x_k that makes l_{ij} true, or one that makes l_{ij} false. Con_i asserts $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff s corresponds to a set of truth values that make l_{i1}, l_{i2}, and l_{i3} all true.

Thus, P produces a final state containing $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff s corresponds to a set of truth values that makes at least one of the conjuncts $c_i = l_{i1}l_{i2}l_{i3}$ true. Since s may correspond to any assignment of truth values to $x_1, x_2, \ldots, x_n$, this means that all executable completions of P_X^* produce final states containing $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff $X = c_1 + c_2 + \ldots + c_m$ is true for all assignments of truth values to $x_1, x_2, \ldots, x_n$. □

Theorem 2 POSSIBLE TRUTH is NP-hard.

Proof. Let $Y = c_1c_2 \ldots c_m$ be a CNF formula over the Boolean variables $y_1, y_2, \ldots, y_n$, with three literals in each disjunctive clause c_i. Let $X = \neg Y$. Using de Morgan's laws, in linear time we can express X as a DNF formula over $y_1, y_2, \ldots, y_n$, with three literals in each conjunct.

Suppose Y is unsatisfiable. Then X is a tautology, so from the proof of Lemma 1, every executable completion of P_X^* produces a final state containing $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$. Thus, no executable completion of P_X^* produces a final state in which $\neg\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is true, so $\neg\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is not possibly true in P_X^*'s final situation.

Suppose Y is satisfiable. Then X is not a tautology, so from the proof of Lemma 1, there is an executable completion P of P_X^* that produces a final state that does

not contain sat(yes, yes, yes). Thus ¬sat(yes, yes, yes) is true in P's final state, so it is possibly true in P_X^*'s final situation. □

Remark. The above proof makes use of the duality between satisfiability checking and tautology checking. However, it is also quite straightforward to prove the theorem without using this duality, by constructing a plan Q_Y^* and a ground atom sat(yes, ..., yes) such that that some completion of Q_Y^* produces sat(yes, ..., yes) iff Y is satisfiable. Such a proof appears in [29].

Theorem 4 If the language $\mathcal{L}$ contains only finitely many constant symbols, then NECESSARY TRUTH is co-NP-hard.

Proof. In the proof of Lemma 1, suppose we specify that the only constant symbols in the language $\mathcal{L}$ are yes and no. Then every completion of P_X^* is executable, and thus sat(yes, yes, yes) is necessarily true in fin iff the formula X is a tautology.

Even if $\mathcal{L}$ contains finitely many additional constant symbols, we can still make sat(yes, yes, yes) necessarily true in fin iff the formula X is a tautology, by adding codesignation constraints to P_X^* of the form $v \not\equiv c$ for each constant symbol c other than yes or no, and each variable v appearing in the steps Con_1 and Con_2. Thus Lemma 1 shows that if $\mathcal{L}$ contains only finitely many constant symbols, then NECESSARY TRUTH is co-NP-hard even with ordinary ''unconditional'' steps. □