On the Nature and Role of Modal Truth Criteria in Planning

by S. Kambhampati and D.S. Nau

The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

TR 93-30

On the Nature and Role of Modal Truth Criteria in Planning[*]

Subbarao Kambhampati[†]
Arizona State University
Tempe, AZ 85287-5406

Dana S. Nau[‡]
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 nonlinear (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:

1. 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 $\neg p$."
2. 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.
3. 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.
4. 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.

[*]The authors' names are given in alphabetical order.

[†]Department of Computer Science and Engineering. Email rao@asuvax.asu.edu. This work was supported in part by NSF Grant IRI-9210997.

[‡]Department of Computer Science, Institute for Systems Research, and Institute for Advanced Computer Studies. Email nau@cs.umd.edu. This work was supported in part by NSF Grants IRI-8907890 and NSFD CDR-88003012.

1 Introduction

Chapman's paper, "Planning for Conjunctive Goals," [1] has been widely acknowledged as an important step towards formalizing partial-order planning, and it has been one of the bases of later work by others (for example, [4, 7, 5, 8, 14, 23, 26]). Unfortunately, however, Chapman's work is not free of problems, and this has led to confusion about the meaning of his results. Previous papers [4, 5, 13, 26] 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:

1. 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, the same is not true for possible truth. Instead, the problem of computing possible truth in plans is NP-hard.[1]

2. 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, 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[2] from it. Others (including Chapman) (c.f. [1, 17, 27]) 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 nonlinear (partial-order) planners, which search in the space of

[1]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.

[2]For the benefit of those unfamiliar with this clause, we point it out explicitly in Footnote 15.

partially ordered partially instantiated plans, but need completeness only in the space of totally ordered and totally instantiated plans.

This paper is organized as follows. Section 2 contains basic definitions, and clarifications/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. All proofs appear in the appendix.

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.

2.1 Basics

The planning language $\mathcal{L}$ is any function-free first-order language.[3] 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.[4]

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*.[5] 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 *plan* is 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

[3]In his paper, 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.

[4]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.

[5]Chapman's definition of a step is basically similar to this, but without name(a). However, as pointed out by McAllester and Rosenblitt [16], 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.

is a set of ordering constraints on the steps of A. P is *complete* 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. (Note that a complete plan need not necessarily be executable). Suppose that P is complete, 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 $1 \leq i \leq k$:

1. s_{i-1} satisfies a_i's preconditions; i.e., $p\theta$ is true in s_{i-1} for every literal $p \in \text{pre}(a_i)$.
2. 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.[6] 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.[7] 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 [1, 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 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:

1. As pointed out by Yang and Tenenberg [26], if a plan P is incomplete, 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.
2. 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

[6]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., [14, 26]) use the term.

[7]He calls them propositions—but as mentioned at the beginning of Section 2.1, we call them literals instead.

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.[8] 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 $\mathsf{out}(b)$.[9]

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 $\mathsf{in}(a_i)$ (or $\mathsf{out}(a_i)$) 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 symbolically as $\mathcal{M}(p, s)$) iff the following three conditions are satisfied:

Establishment: either p codesignates with a postcondition of some step $a \prec s$, or else $p \in s_0$.

Nondeletion: for all steps b between a (or s_0) and s, no postcondition of b codesignates with $\neg p$.[10]

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 [1, 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 [1, p. 341] that a plan "necessarily solves the

[8] 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 $\neg p$.

[9] 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 makes no significant difference in any of the results.

[10] This is basically the white-knight declobbering clause of the MTC (see Footnote 15), simplified to handle the special case where the plan is complete.

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;
- 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.

We now define the following decision problems:[11]

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

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

3 Modal Duality, and Complexity of Modal Truth

3.1 Our Results

Given the definitions of modal truth and modal conditional truth above, it is easy to see that a literal p is necessarily true in a situation s if and only if (1) p is necessarily conditionally true in s, and (2) for every action a that precedes s in at least one completion of the plan and every precondition p_a of a, p_a is necessarily conditionally true in the situation $\mathsf{in}(a)$. Thus,

$$\Box\mathcal{M}(p,s) \equiv \Box\left[\mathcal{C}(p,s) \wedge \left[\bigwedge_{\forall a\in S, \forall p_a \in \mathsf{pre}(a)} \mathcal{C}(p_a, \mathsf{in}(a))\right]\right], \tag{1}$$

where S is the set of all actions that precede s in at least one completion of the plan. Now, since modal necessity commutes over conjunctions (i.e., $\Box(p \wedge q) \equiv \Box(p) \wedge \Box(q)$), we can write Eq. 1 as

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

Thus computing whether p is necessarily true in s involves computing whether p is necessarily conditionally true in s, as well as computing the necessary conditional truth of all preconditions of

[11]Although we talk about final situation of the plan, it is easy to see that these problems are equivalent to finding truth in any given situation of the plan.

all steps preceding s. As noted in Chapman, 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. Coming to the case of possible truth, we have

$$\Diamond\mathcal{M}(p,s) \equiv \Diamond\left[\mathcal{C}(p,s) \wedge \left[\bigwedge_{\forall a\in S, \forall p_a \in \mathrm{pre}(a)} \mathcal{C}(p_a, \mathsf{in}(a))\right]\right]. \tag{3}$$

But possible truth does not commute 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. 3 into component tests of computing possible conditional truth of individual literals. Thus, even though possible conditional truth in s and necessary conditional truth in s are duals of each other (i.e., $\Diamond\mathcal{C}(p,s) \equiv \neg\Box\neg\mathcal{C}(p,s)$),[12] possible truth in s and necessary truth in s are *not* duals of each other. More specifically:

Theorem 1 *There is a ground literal p, a plan P, and a situation s of P such that $\Box\mathcal{M}(p,s) \not\equiv \neg\Diamond\neg\mathcal{M}(p,s)$.*

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 [2, 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 truth of p is not defined in the corresponding possible world.[13]

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:

CONDITIONAL 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?[14]

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

CONDITIONAL TRUTH is a weaker condition than both NECESSARY TRUTH and NECESSARY CONDITIONAL TRUTH. There are some cases (one occurs in the proof of Lemma 1) in which every

[12]To see this, let P be a plan, s be a situation of P, and p be a literal. Then $\Diamond\mathcal{C}(p,s)$ iff there is a completion of P such that the establishment and nondeletion conditions are satisfied for p. This is true iff it is *not* the case that one or both conditions fail for p in every completion of P, i.e., iff $\neg\Box\neg\mathcal{C}(p,s)$. Thus $\Diamond\mathcal{C}(p,s) \equiv \neg\Box\neg\mathcal{C}(p,s)$.

[13]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* [24]. 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.

[14]CONDITIONAL TRUTH corresponds closely to the notion of partial correctness, which was studied in connection with dynamic-logic-based modeling of computer programming languages [20, 24].

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. 3 is to note that if p is conditionally possibly true and that all the preconditions of the preceding actions are possibly conditionally true, this only implies that each of them is *individually* 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 [6, 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 [1, p. 340]:

> **Modal Truth Criterion.** A [literal] p is necessarily true in a situation s iff two conditions hold:[15] 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.[16] 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, [1, p. 370], without checking that the white knights are in fact executable.[17]

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

[15]The second of these conditions is the "white-knight declobbering clause" that we refer to elsewhere.

[16]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.

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

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.*

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 [1, 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" [1, 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 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. [26, 14]) use this interpretation to eliminate the notion of situations entirely, and state 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.

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 $\text{pre}(a) = \emptyset$ for every step a of P. In this case, it is easy to see that Equations 1 and 3 in Section 3.1 will simplify respectively to:

$$\Box\mathcal{M}(p,s) \equiv \Box\mathcal{C}(p,s); \tag{4}$$
$$\Diamond\mathcal{M}(p,s) \equiv \Diamond\mathcal{C}(p,s). \tag{5}$$

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. 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 [24] 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 [20]. 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 CONDITIONAL 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.

Temporal Projection. A very similar idea is used in Dean and Boddy's work on temporal projection [3]. 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 extension (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.[18]

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 [1, 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

[18]Actually, their result is misstated: they say the problem is NP-hard, but prove it is co-NP-hard.

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 proved by Chapman in his Intractability Theorem.[19]

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. Here's 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) post(a) contain an unbound variable x. Then for the purposes of both planning and temporal projection, a has conditional effects: its effects will be different in different completions of P, depending on what we bind x to. 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 [1, p. 356] that restricting the range of a variable to a finite set will defeat the MTC, and make constraint computations NP-complete.

4 The MTC in Plan Generation vs. Correctness Checking

4.1 Plan Generation

In his paper, Chapman suggests that a sound and complete partial-order planner can be generated by "inverting" his modal truth criterion. This has lead to some misconceptions regarding the exact role of MTC in plan generation. Specifically, several researchers have (mistakenly) attempted to simplify MTC by eliminating the white-knight clause from it. Still others (including Chapman) (c.f. [27]) have used Chapman's proof regarding the NP-hardness of checking truth in plans having actions with conditional effects, to conjecture that nonlinear (partial-order) planning will not scale up to more expressive action representations. In this section we attempt to resolve these confusions by clarifying the role of MTC in plan generation vs. checking correctness of the plans.

Although inverting Chapman's modal truth criterion provides sufficient basis for doing sound and complete nonlinear (partial-order) planning, it is stricter than is actually needed for partial-order planning. The reason for this has to do with the peculiar predicament of nonlinear (partial-order) planners, which search in the space of partially ordered plans, but require completeness

[19]He makes the same misstatement as Dean and Boddy: he says the problem is NP-hard, but proves it co-NP-hard.

only in the space of totally ordered plans. To put it another way, the objective of planning, be it total-order or partial-order planning, is to find a sequence of actions (not necessarily the least-constrained) which when executed from the initial state, get us to a state in which goal conditions are true. Thus the aim of the least-commitment techniques of partial-order planning is only to improve the efficiency of partial-order planning, not to find the least-constrained partially ordered (nonlinear) plan for the problem.[20] What this means is that it is possible to write a *sound and complete* partial-order planner that does uses a truth criterion that provides only sufficient rather than necessary and sufficient conditions for ensuring truth. For example, many planners (c.f. [16, 12, 18]) modify the MTC by replacing its white-knight declobbering clause with a much simpler demotion clause as a way of resolving a conflict during planning:

> 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 in any situation s'' that possibly comes between s' and s.

With this modification, the truth criterion is sufficient but unnecessary for ensuring truth of a literal. However, a planner using this modified truth criterion can still be complete, because for every plan P that is correct according to Chapman's truth criterion, there will be a *constrainment* of P (see Section 2.1) that will be correct according to this modified truth criterion.

4.2 Checking Correctness of Plans

The discussion above should not be interpreted as a license to simplify MTC by eliminating the white-knight clause. On the contrary, as Chapman's proof shows, the white-knight clause is still required if we want to state the necessary and sufficient conditions for the correctness 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 and totally instantiated 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.)

Furthermore, the necessity of the white-knight declobbering clause has nothing to do with whether the plan is totally instantiated or not[21] (Chapman's use of a partially instantiated plan [1, Fig. 5, p. 339] to motivate white-knight declobbering seems to have caused this misimpression). The following example illustrates this point. Consider the ground partially ordered plan in Fig. 1, in which the literal p is required in the final situation **fin**, the steps **w1** and **w2** add p, the steps **s1** and **s2** delete p, and the steps **s1** and **w1** are unordered with respect to **s2** and **w2**. We can see that p is true in the situation preceding **fin** in every completion of this plan. However, a modal truth criterion without the white-knight clause will not be able to recognize this fact.[22]

In practice, avoiding the white-knight declobbering clause during planning means that the planner may wind up finding somewhat more constrained plans. In particular, a planner not using

[20] "Least-commitment" planning is a bad name for partial-order planning, since it gives the misleading impression that the aim is to find the least-committed plan. The idea instead is to defer commitment to the extent it is computationally advantageous to do so. Thus, a better name might be deferred-commitment planning.

[21] In contrast, the need for the white-knight declobbering clause does depend on whether or not the plan is totally ordered. For example, in lieu of white-knight declobbering, Pednault's causality theorem [19] uses a condition similar to the "nondeletion" condition that we discussed in Section 2.2. Any variant of the causality theorem applicable to partially ordered plans would instead need a declobbering clause, if it were to provide provide necessary and sufficient conditions to test the correctness of a given nonlinear plan (other than by enumerating all its completions and checking their correctness individually, as we discussed above). However, as explained earlier, this does not affect

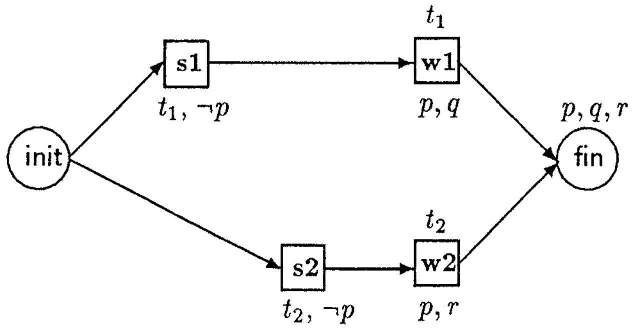

Figure 1: A correct ground partially ordered plan that requires the white-knight clause (example due to Mark Drummond). Each step's name is in a box, with its preconditions and postconditions above and below the box. fin is the final situation, and init is the initial situation.

the white-knight clause will find one of two alternate nonlinear plans: one in which s_1 and w_1 precede s_2 and w_2, and the other in which the order is reversed. Both these plans are more constrained than the original plan. Generating over-constrained plans in itself is not a problem, unless there is a concomitant loss of efficiency in planning.[23] In particular, if we need least-constrained plans as a result of planning, we can always start from the over-constrained plans *after the planning*, and use the full MTC to do a polynomial-time EBG-type analysis and eliminate additional constraints [14, 13]). In fact, there can be a spectrum of complete and sound partial-order planners each using truth criteria less general than the modal truth criterion (see [12, 10]).[24]

4.3 Plan Generation with More Expressive Action Represention

The misconception that a complete and sound partial-order planner has to, of necessity, interpret a necessary and sufficient truth criterion during planning, has also lead to a mistaken belief that partial-order planning has more difficulty in scaling up to more expressive action representations

the completeness of nonlinear (partial-order) planning based on the causality theorem.

[22] *Historical Note:* Nonlin was the first planner to use a white-knight clause to specify weakest conditions for establishment and declobbering. Nonlin's Q&A procedure [25] 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. 1 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.

[23] Although a serious discussion of practical utility of white-knight declobbering clause in planning is beyond the scope of this paper, we would like to observe that using a white-knight declobbering clause does not *ipso facto* make planning inefficient, as has been conjectured by some researchers (c.f. [9]). An important issue is whether the planner implements white-knight declobbering only through steps that are already existing in the plan, or whether it also allows new steps to be introduced as white-knights into the plan. In [11, 10], 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.

[24] A related but orthogonal question is whether a planner using a more general truth criterion will be more or less efficient than one using a less general one. In [10], we argue that this will depend more on the tradeoffs between the search space redundancy vs. level of commitment made by the planner.

than does total-order planning (e.g., [27, 17]).[25] The usual justification for this belief is 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, of necessity, interpret the complete MTC in each iteration, even the per-node cost of a partial-order planner goes up drastically.

The discussion above shows the fallacy in this reasoning.[27] In the extreme case, as discussed above, a partial-order planner can use a truth criterion for totally ordered totally instantiated plans and still be complete, and the NP-hardness result clearly won't affect such a partial-order planner. Recent work [16, 22, 18, 19] shows that it is possible to design partial-order planners without going to this extreme: they constrain plans more than necessary to ensure that correctness check can be done efficiently, but still avoid degnerating into total-order planning. They can thus retain their relative advantages over total-order planners even with more expressive action representations. At the very least, there is no theoretical reason to think that partial-order planners are worse off than total-order planners in scaling up to more expressive action representations.

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:

1. 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 duality between necessary truth and possible truth is related to (a) 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, and (b) the asymmetry in the way necessary conditional truth and possible conditional truth commute over conjunctions: $\Box(p \wedge q) \equiv \Box(p) \wedge \Box(q)$ while $\Diamond(p \wedge q) \not\equiv \Diamond(p) \wedge \Diamond(q)$. To achieve modal duality, one needs universally executable plans.

2. 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.

3. 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.

Checking possible truth has several applications in plan projection [3] as well as plan generalization [13].

[25]This belief has inhibited some researchers from basing their work on partial-order planning (see [15]).

[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.

[27]Furthermore, it is wrong to believe that planning itself is more difficult if conditional operators are allowed. Erol *et al.* [4] 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.

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 MTC:

1. The white-knight declobbering clause of the MTC is needed in order to provide both necesary and sufficient conditions for ensuring truth of a literal in a partially ordered plan. However, if we are interested in plan generation rather than checking the correctness of plans, this clause is stricter than is actually needed. It is possible to develop sound and complete nonlinear planners which use truth criteria that provide only sufficient, but not necessary, conditions for ensuring truth. Specifically, several sound and complete nonlinear planners replace the white-knight declobbering clause of MTC with a simpler demotion clause.

2. Since sound and complete nonlinear (partial-order) planners are not required to use a necessary and sufficient truth criterion during planning, Chapman's proof regarding the NP-hardness of verifying truth of a literal in a plan containing actions with conditional effects, does not necessarily imply that partial-order planners are any worse off than total-order planners in dealing with richer action representations (as has been conjectured elsewhere).

   In particular, as recent work (c.f. [18] shows), it is possible to device polynomial-time sufficiency conditions for ensuring truth, even for plans containing actions with conditional effects.

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.

Acknowledgement

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

References

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

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

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

[4] K. Erol, D. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. 1992. Submitted for publication.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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 (in press).

[11] S. Kambhampati. Multi-Contributor causal structures for Planning: A Formalization and Evaluation. Arizona State University Technical Report, CS TR-92-019, July 1992.

[12] S. Kambhampati. Characterizing multi-contributor causal structures for planning. In *Proc. First Intl. Conf. on AI Planning Systems*, 1992.

[13] S. Kambhampati and S. Kedar. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. Tech. Report TR-92-008, Dept. of Computer Science and Engineering, Arizona State University, April 1992. Submitted for publication.

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

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

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

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

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

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

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

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

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

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

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

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

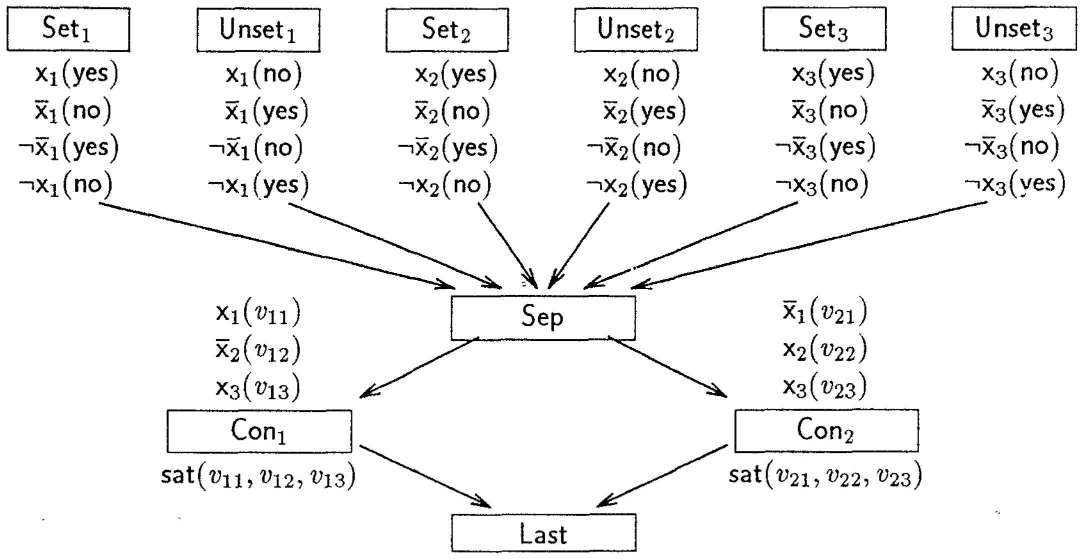

Figure 2: An example of the plan P_X^* in the case where $X = x_1\overline{x_2}x_3 + \overline{x_1}x_2x_3$. Each step's name is in a box, with its preconditions and postconditions above and below the box.

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

[27] 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.

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

Appendix

Lemma 1 CONDITIONAL 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 sat(yes, yes, yes), such that every executable completion of P_X^* produces a final situation containing sat(yes, yes, yes) iff X is a tautology. P_X^* is the following plan (see Fig. 2):

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 postconditions $\neg\overline{\mathrm{x}}_i$(yes), $\overline{\mathrm{x}}_i$(no), $\neg\mathrm{x}_i$(no), x_i(yes).
Unset$_i$ has no preconditions, and has the postconditions $\overline{\mathrm{x}}_i$(yes), $\neg\overline{\mathrm{x}}_i$(no), x_i(no), $\neg\mathrm{x}_i$(yes).
Here, yes and no are constant symbols; the interpretations of x_i(yes), $\overline{\mathrm{x}}_i$(no), $\overline{\mathrm{x}}_i$(yes), and x_i(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 $\mathsf{sat(yes, yes, yes)}$ 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.

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 $\mathsf{sat(yes, yes, yes)}$ 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 $\mathsf{sat(yes, yes, yes)}$ 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 $\mathsf{sat(yes, yes, yes)}$ 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 1 There is a ground literal p, a plan P, and a situation s of P such that $\Box\mathcal{M}(p, s) \not\equiv \neg\Diamond\neg\mathcal{M}(p, s)$.

Proof. In the proof of Lemma 1, let X be a tautology; and let $p = \mathsf{sat(yes, yes, yes)}$. From the proof of Lemma 1, it follows that every executable completion of P_X^* produces a final state containing p. Therefore no completion of P_X^* produces a final state in which $\neg p$ is true, so $\neg\Diamond\neg\mathcal{M}(p, \mathsf{fin})$ holds.

[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 done(name(a)) (recall that name(a) is a constant symbol). For each ordering constraint '$x \prec y$' of P_X^*, give y a new precondition done(name(a)).

However, $\Box\mathcal{M}(p, \mathsf{fin})$ does not hold, because there are some (non-executable) completions in which Last's postcondition $\mathsf{sat}(v_{21}, v_{22}, v_{23})$ does not codesignate with p (for example, this happens if we constrain $v_{21} \approx \mathsf{foo}$, $v_{22} \approx \mathsf{bar}$, and $v_{23} \approx \mathsf{baz}$, where $\mathsf{foo}, \mathsf{bar}, \mathsf{baz}$ are constant symbols). ■

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 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 $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$. Thus $\neg\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ 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 $\mathsf{sat}(\mathsf{yes}, \ldots, \mathsf{yes})$ such that that some completion of Q^*_Y produces $\mathsf{sat}(\mathsf{yes}, \ldots, \mathsf{yes})$ iff Y is satisfiable. Such a proof appears in [21].

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

Proof. First, we state the "possible truth" version of the MTC explicitly:

> A literal p is possibly true in a situation s iff two conditions hold: there is a situation t equal or possibly previous to s in which p is possibly asserted; and for every step C necessarily before s and every literal q necessarily codesignating with p which C denies, there is a step W possibly between C and s which asserts r, a literal such that r and p codesignate whenever p and q codesignate.

Consider the plan P^*_X of Lemma 1, in the case where X is a tautology. Every executable completion of P^*_X will produce a final state in which $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is true. Thus, no executable completion of $P^\dagger_X$ will produce 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 $P^\dagger_X$'s final situation.

If we apply the criterion for possible truth, we will reach a different conclusion. $\neg\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is true in s_0 and thus in P's initial situation; and P's initial situation precedes its final situation. Thus the first condition of the criterion is satisfied. In every executable completion of P, there is *some* step that denies $\neg\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$. But in general, there is no step that denies $\neg\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ in *every* completion of P. Thus, the criterion concludes, incorrectly, that $\mathsf{unsat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is possibly true in $P^\dagger_X$'s final situation. ■

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 $v \not\approx c$ 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. ■