Polynomial-Length Planning Spans the Polynomial Hierarchy (Incomplete Draft)

Hudson Turner
Computer Science Department
University of Minnesota, Duluth
hudson@d.umn.edu

Abstract

This paper presents a family of results on the computational complexity of the decision problem associated with planning: Does a valid plan exist? Several classes of planning problems are considered: classical planning, conformant planning, conditional planning with full or partial observability. Throughout, attention is restricted to plans of polynomially-bounded length. For conditional planning, restriction to plans of polynomial size is also considered. Throughout, the analysis employs a simple, general action representation framework in which a planning domain is described by a transition relation encoded in classical propositional logic. Given the widespread use of satisfiability-based planning methods, this is a natural choice for action representation. Moreover, this choice makes it possible to develop a unified representation—in second-order propositional logic—of the range of planning problems considered. In turn, the QBF representations simplify many of the complexity proofs.

The paper title is meant to highlight the following new results.

- If the current state is fully observable during plan execution, conditional planning for plans of length k is Π^P_{2k+1}-complete. It remains Π^P_{2k+1}-hard even if actions are always executable.
- If the current state is partially observable during plan execution, conditional planning for plans of length k is Π^P_{2k+2}-complete.
- If the current state is partially observable during plan execution, and the executability of actions is polynomial-time decidable, conditional planning for plans of length k is Π^P_{2k+1}-complete. It remains Π^P_{2k+1}-hard even if actions are deterministic and always executable.

On the other hand, many results reported here closely resemble results previously reported elsewhere, obtained using a variety of action representation frameworks. In surveying these previous results, it is not always clear which assumptions are essential in obtaining a particular result. By

describing a wide range of results within a single, fairly general framework, the current paper sheds new light on how planning complexity is affected by common assumptions such as determinism and polynomial-time decidability of executability of actions.

1 Introduction

This paper presents a family of results on the computational complexity of planning problems in which a plan of polynomially-bounded length is sought. A planning problem traditionally consists of three elements: a description of the planning domain (specifying how actions can affect the state of the world), a description of the initial state of the world, and a description of the goal. A solution to a planning problem is a *valid* plan—one whose execution is guaranteed to achieve the goal, and whose executability is also guaranteed. In investigating the computational complexity of planning, it is convenient to focus on the associated decision problem: Does a valid plan exist?

In classical planning, the initial state is completely specified, the action domain is deterministic, and a plan is simply a finite sequence of actions to be performed. These assumptions can be relaxed to obtain larger classes of planning problems. For instance, in so-called conformant planning, one no longer assumes that the initial state is completely specified, and actions are allowed to have nondeterministic effects, but a plan is still a sequence of actions. Another class of planning problems is obtained by allowing conditional plans, in which the execution of a plan at each step may depend on the previous execution history. In characterizing this dependence, we may assume that the current state is fully observable or only partially observable during plan execution. This paper considers the complexity of each of these classes of planning problems.

This is certainly not a new line of research. Many papers with results on planning complexity have been published in the last decade, among them [Bylander, 1994, Backstrum and Nebel, 1995, Erol *et al.*, 1995, Littman, 1997, De Giacomo and Vardi, 1999, Haslum and Jonsson, 1999, Rintanen, 1999, Baral *et al.*, 2000, Eiter *et al.*, 2000]. In general terms, the work presented here is unique in employing a single framework to study the complexity of such a wide range of polynomial-length planning problems.

In fact, one of the contributions of this paper is the development of a unified representation—in second-order propositional logic—of the decision problems associated with classical, conformant and conditional planning (with full or partial observability). In many cases, membership of a planning problem in a given complexity class can be determined by the form of its QBF representation, and several of the hardness results are especially easy to obtain in this setting.

The analysis employs a simple, general action representation framework in which a planning domain is described by a transition relation encoded in classical propositional logic. Of course this is technically convenient, given our interest in QBF representations. It is also a natural choice in light of the widespread use of satisfiability methods in automated planning, which depends on the fact that

such descriptions can be obtained by polynomial-time translation from many other action description languages. Thus, the results obtained here have fairly wide applicability.

Moreover, this action representation framework allows for such complications as nondeterminism and concurrent execution of actions. This expressiveness makes it convenient for us to consider the incremental effect of imposing common planning assumptions such as determinism and nonconcurrency. Consequently, our analysis sheds new light on the role such assumptions play in determining the complexity of each of the families of planning problems considered here.

For instance, while the possibility of concurrent actions may complicate the task of describing an action domain, it plays only a minor role in planning complexity. We will see that a number of *hardness* results that hold for fixed-length plans with concurrent actions require plans of linearly-bounded length if no concurrency is allowed. More significantly, the complexity of fixed-length conformant planning actually drops slightly (because without concurrent actions the number of possible plans is polynomial).

Many accounts of planning are based on action representations with another significant property: the executability of any given action(s) in any given state can be determined in polynomial time. This assumption does not hold for the action representation framework used here. Let us consider why it can be convenient to abandon it. In general, this assumption will not hold for action representations that allow for *implied* action preconditions. For example, a domain description may describe the fact that two blocks cannot occupy the same position, and from this general fact many action preconditions may follow. For instance, it can be inferred that block B_1 cannot be moved to the location of block B_2 (unless block B_2 is moved somewhere else). It also follows that blocks B_1 and B_2 cannot be moved concurrently to the same location.

Before continuing, it is important to emphasize that a number of results reported here are similar to (although distinct from) results in [Littman, 1997, Rintanen, 1999, Baral *et al.*, 2000, Eiter *et al.*, 2000]. In each of these cases, the similarities and differences will be noted when the results are encountered in the text.

We proceed as follows.

Section 2 specifies the action representation framework employed in this paper, describes the basic elements with which planning problems are formulated in this framework, and provides a precise characterization of some of the assumptions we will investigate: namely, completely specified initial states, deterministic action domains, always executable actions, polynomial-time determination of executability of actions in a state, and nonconcurrency. Section 2 also includes a convenient characterization of the complexity classes that will interest us.

Section 3 discusses the complexity of classical planning in this framework, a question that is interesting in itself since the framework accomodates concurrent actions and dynamic worlds, and does not assume that executability of actions in a state is decidable in polynomial time.

Section 4 investigates conformant planning, showing that the associated de-

cision problem is Σ_3^P-complete for plans of polynomially-bounded length. In fact, it is Σ_3^P-hard even for plans of length 1. We also see that conformant planning drops to Σ_2^P if we assume that the executability of actions in a state is polynomial-time decidable. Remarkably, the problem remains Σ_2^P-hard (for plans of length 1) even if actions are deterministic and always executable.

Section 5 considers the intuitively more difficult problem of conditional planning (with full observability of the current state during plan execution). It turns out that the decision problem for conditional planning for plans of length k is Π_{2k+1}^P-complete. Consequently, conditional planning is **PSPACE**-complete for plans of polynomially-bounded length. (Note that we have **PSPACE** membership even though the executability of actions in a state is not polynomial-time decidable.) The problem remains Π_{2k+1}^P-hard (for plans of length k) even if we assume that actions are always executable. So nondeterminism is the key to hardness in this case; the complexity drops to Π_2^P if we assume that actions are deterministic (even though the executability of actions is not p-time decidable).

Section 6 considers conditional planning with partial observability—an elaboration of conditional planning in which some fluents may not be observable in the current state during plan execution. (Hence their values cannot play a direct role in the specification or execution of a conditional plan.) In this case the decision problem for plans of length k is Π_{2k+2}^P-complete. Interestingly, if we assume that the executability of actions is polynomial-time decidable, the complexity drops only slightly: then the decision problem is Π_{2k+1}^P-complete for plans of length k. If we assume in addition that actions are deterministic and always executable, the problem remains Π_{2k+1}^P-hard.

Section 7 discusses the fact that eliminating concurrent actions has an interesting, subtle effect on hardness results: a number of hardness results that hold for fixed-length plans with concurrent actions require instead plans of linearly-bounded length if no concurrency is allowed. We also observe that the complexity of conformant planning drops slightly if we assume *both* nonconcurrency and fixed plan length.

Section 8 discusses ramifications of the observation that polynomial-length conditional plans do not have polynomially-bounded size. This has led researchers [Littman *et al.*, 1998, Rintanen, 1999] to consider the possibility of directly imposing a polynomial-size restriction on conditional plans themselves. In practice, this requires choosing a particular form for representing the set of candidate plans. Typically, such a representation should guarantee that the next actions to be performed at each step can be computed in polynomial time. It should also be possible to obtain this representation in polynomial time, given the action description and a plan length. One family of such plan representations is briefly considered. The limited form of conditional planning allowed by such a plan representation is essentially an extension of conformant planning, and has the same complexity. Hence, as with conformant planning, the critical factor is whether or not the executability of actions is polynomial-time decidable; this assumption drops the complexity from Σ_3^P to Σ_2^P, even when actions are nondeterministic.

Section 9 briefly discusses related work and offers a few concluding remarks.

Many results are proved when stated. The remaining proofs (or sketches of them) appear in an appendix.

2 Preliminaries

2.1 Action Representation Framework

The analysis in this paper employs a simple, general domain representation framework—the action domain is described by a transition relation encoded in classical propositional logic.

More precisely, begin with a set A of action symbols and a disjoint set F of fluent symbols. A transition relation is a labeled directed graph. Each node is a subset of F, standing for the state in which the fluents corresponding to elements of F are true, and the fluents corresponding to fluent symbols not in F are false. The edges are labeled with subsets of A, with the meaning that concurrently executing the corresponding actions (and no others) in the start state can take the world to the end state.

We assume that such a transition relation is encoded in classical propositional logic as follows. Let $state(F)$ be a formula in which the only nonlogical symbols are elements of F. It encodes the set of states that correspond to its models. Let $act(F, A, F')$ be a formula with nonlogical symbols taken from $F \cup A \cup F'$, where F' is obtained from F by priming each element of F. (Assume F' is disjoint from $F \cup A$. From now on similar disjointness assumptions will be left unstated.) Let $state(F')$ be the formula obtained from $state(F)$ by substituting for each occurrence of each fluent symbol its primed counterpart. (Throughout, this sort of notation will be used to generate formulas by substitution.) Of course each model of $state(F')$ corresponds to a state in the obvious way. Then the formula

$$state(F) \wedge act(F, A, F') \wedge state(F') \tag{1}$$

encodes the set of labeled edges that correspond to its models: that is, the start state corresponds to the interpretation of the symbols in F, the label corresponds to the interpretation of the symbols in A, and the end state corresponds to the interpretation of the symbols in F'. It is convenient to let

$$tr(F, A, F')$$

stand for (1).

This domain representation framework accomodates nondeterminism, concurrent actions and dynamic worlds. For an example illustrating all three features, take

$$F = \{Heads, JustWon\}$$

and

$$A = \{Toss, CallHeads\}$$

and let

$$state(F) = JustWon \supset Heads\,.$$

For $act(F, A, F')$ take the conjunction of the following two formulas.

$$\neg Toss \supset (Heads' \equiv Heads)$$
$$JustWon' \equiv Heads' \wedge Toss \wedge CallHeads$$

Executing *Toss* and *CallHeads* (concurrently) in any state results in either state $\{Heads, JustWon\}$ or state $\emptyset$. Executing no action in state $\{Heads, JustWon\}$ results in state $\{Heads\}$.

This paper does not address how such action representations are to be obtained. There are many possibilities. For instance, domain descriptions in STRIPS-based languages [Fikes and Nilsson, 1971] are easily translated (in polynomial time) into domain descriptions in the format used in this paper. (This observation has played a major role in recent planning research.) Such descriptions can also be obtained by translation from more expressive action languages, such as the high-level action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998], for which "definite" descriptions yield such classical propositional theories in polynomial time by the syntactic process of "literal completion" introduced in [McCain and Turner, 1997]. For example, the domain description above can be obtained from the following $\mathcal{C}$ description.

never $JustWon \wedge \neg Heads$
default $\neg JustWon$
inertial $Heads, \neg Heads$
possibly caused $Heads$ **after** $Toss$
possibly caused $\neg Heads$ **after** $Toss$
caused $JustWon$ **if** $Heads$ **after** $Toss \wedge CallHeads$

Giunchiglia [2000] describes an implemented conformant planning system based on such a translation from $\mathcal{C}$ to propositional logic.

2.2 Planning Framework

In the remainder of the paper, we use this action representation framework to investigate the computational complexity of several classes of planning problems. That is, we consider the complexity of the associated decision problem—existence of a valid plan. The input size parameter for our complexity results is (any reasonable measure of) the size of (1) plus the size of the formula

$$init(F)$$

describing the initial state and the formula

$$goal(F)$$

describing the goal. (As before, we assume that the nonlogical symbols in these formulas are taken from F.) Henceforth, we will find it convenient to assume that

$$\mathit{init}(F) \models \mathit{state}(F)\,.$$

We will be interested in plans whose length is bounded by a polynomial in the input size.

Precise formulations (in second-order propositional logic) for the existence of a valid plan, for various types of plans—namely, unconditional, conditional, and conditional with partial observability—will be developed in Sections 3–6.

2.3 The Polynomial Hierarchy and PSPACE

For our purposes, it is convenient to characterize complexity classes in terms of second-order propositional logic. (See, for instance, [Papadimitriou, 1994].) Consider quantified boolean formulas (QBFs) of the form

$$Q_1\, x_1\ Q_2\, x_2\ \cdots\ Q_k\, x_k\ \Phi(x_1, x_2, \ldots, x_k) \tag{2}$$

where (i) each of $x_1, x_2, \ldots, x_k$ stands for a tuple of propositional variables, (ii) each of $Q_1, Q_2, \ldots, Q_k$ stands for a quantifier, with each Q_{i+1} different from Q_i, and (iii) $\Phi(x_1, x_2, \ldots, x_n)$ is a formula without quantifiers and without nonlogical constants, whose variables are taken from $x_1, x_2, \ldots, x_n$. For a formula of this kind, the corresponding decision problem is this: Is it satisfiable? Equivalently, we can ask if it is logically valid, or, since no nonlogical constants occur in it, we may simply ask if it is true.

For each natural number k, the complexity class Σ_k^P corresponds to the family of formulas of form (2) with $Q_1 = \exists$, and the complexity class Π_k^P corresponds to the family of such formulas with $Q_1 = \forall$. That is, for each of these families of QBFs, the decision problem is complete for the corresponding complexity class. Some important special cases are $\Sigma_1^P = \mathbf{NP}$, $\Pi_1^P = \mathbf{coNP}$, and $\Sigma_0^P = \Pi_0^P = \mathbf{P}$.

We'll also have occasion to consider the complexity class D^P, which consists of problems that can be solved in polynomial-time given the answer to one **NP** problem and one **coNP** problem.

Finally, the complexity class **PSPACE** corresponds to the decision problem for all QBFs of form (2).

Let D be the decision problem associated with a particular class of planning problems. Typically, in this paper, to show that D belongs to a complexity class C, we show how, given an arbitrary instance of D, we can obtain in polynomial-time a QBF F such that

- F is true iff a valid plan exists, and
- F has the form (2) corresponding to complexity class C.

Having shown that D belongs to complexity class C, we then show that D is as hard as any problem in C—and therefore complete for C—by specifying a polynomial-time translation that takes any QBF F corresponding to complexity

class C and turns it into an instance of D such that F is true iff a valid plan exists.

In some cases, it will be convenient to utilize the standard characterization of the polynomial hierarchy in terms of oracle machines. (See, for instance, [Papadimitriou, 1994].)

2.4 Common Assumptions in Planning

We are now ready to characterize in our framework a number of common planning assumptions—completely specified initial state, deterministic action domain, always executable actions, polynomial-time determination of executability of actions, and nonconcurrency.

2.4.1 Completely Specified Initial State

The assumption that the initial state is completely described can be expressed in second-order propositional logic by the QBF

$$\exists f \ \ (\mathit{init}(f) \wedge \forall f' \ (\mathit{init}(f') \supset f = f')) \tag{3}$$

where f (respectively f') is a tuple of distinct propositional variables corresponding to the propositional constants in F (respectively F'), and $f = f'$ is shorthand for the conjunction of formulas $p \equiv p'$ for each pair of similar variables $p \in f$ and $p' \in f'$. (From now on we'll employ such notation without remarking on it.) Notice that (3) is true iff $\mathit{init}(F)$ has a unique model. Since $\mathit{init}(F)$ is an arbitrary formula (with signature F), determining that the initial state is completely described is exactly the well-known decision problem "unique sat," which belongs to complexity class D^P, but is not known to be complete for that class [Papadimitriou, 1994].

2.4.2 Determinism

An action domain is deterministic iff the formula

$$\mathit{tr}(F, A, F') \wedge \mathit{tr}(F, A, F'') \supset F' = F'' \tag{4}$$

is logically valid, which is to say that the QBF

$$\forall f \ \forall a \ \forall f' \ \forall f'' \ (\mathit{tr}(f, a, f') \wedge \mathit{tr}(f, a, f'') \supset f' = f'')$$

is true. So determining that an action domain is deterministic is a problem in **coNP**. Moreover, we can show that it is **coNP**-hard.

Proposition 1 *The problem of determining that an action domain is deterministic is* **coNP***-complete.*

Proof. We have already observed that the problem belongs to **coNP**; it remains only to show hardness. The complexity class **coNP** is expressed by QBFs of the form

$$\forall x_1 \cdots x_n \ \Phi(x_1, \ldots, x_n) \tag{5}$$

where $\Phi(x_1, \ldots, x_n)$ is a formula without quantifiers and without nonlogical constants, whose variables are taken from $x_1, \ldots, x_n$. We need an action domain description obtainable from (5) in polynomial time, such that (4) is logically valid iff (5) is. Let $F = \{P, X(1), \ldots, X(n)\}$. (So in addition to fluent symbol P, we introduce a fluent symbol corresponding to each of the propositional variables.) Let $A = \emptyset$. Take $\mathit{state}(F) = \top$ and $\mathit{act}(F, A, F') = \neg\Phi(X(1), \ldots, X(n))$. Assume that (5) is true. Then $\mathit{act}(F, A, F')$ is unsatisfiable, and consequently (4) is logically valid. For the other direction, assume (5) is false. Then $\mathit{act}(F, A, F')$ is satisfiable. Moreover, it is satisfied by a pair of interpretations that differ on the truth value assigned to P'. We can conclude that (4) is not logically valid. □

2.4.3 Actions Always Executable

Actions are always executable iff

$$\forall f\ \forall a\ \exists f'\ (\mathit{state}(f) \supset \mathit{tr}(f, a, f')) \tag{6}$$

is true. We can conclude that determining that actions are always executable is a Π_2^P problem. Again, hardness is not difficult to show.

Proposition 2 *The problem of determining that actions are always executable is Π_2^P-complete.*

Proof. We know the problem belongs to Π_2^P, which is expressed by QBFs of the form

$$\forall x_1 \cdots x_m \exists y_1 \cdots y_n\ \Phi(x_1, \ldots, x_m, y_1, \ldots, y_n) \tag{7}$$

where $\Phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is a formula without quantifiers and without nonlogical constants, whose variables are taken from $x_1, \ldots, x_m, y_1, \ldots, y_n$. To show hardness, we need an action domain description obtainable from (7) in polynomial time, such that (6) is true iff (7) is.

Let $F = \{X(1), \ldots, X(m), Y(1), \ldots, Y(n)\}$. (So there is a fluent symbol corresponding to each variable in (7).) Let $A = \emptyset$. Take $\mathit{state}(F) = \top$ and

$$\mathit{act}(F, A, F') = \Phi(X(1), \ldots, X(m), Y(1)', \ldots, Y(n)')\,.$$

Given our choices for $\mathit{state}(F)$ and A, (6) is logically equivalent to

$$\forall\, x(1) \cdots x(m)\ \exists\, y(1)' \cdots y(n)'\ \Phi(x(1), \ldots, x(m), y(1)', \ldots, y(n)')$$

which is in turn clearly equivalent to (7). □

The previous proof construction appears to use nondeterminism in an essential way. We will encounter several other such cases, where the contribution of executability to hardness seems to rely on nondeterminism. In these cases, roughly speaking, determinism would yield an instance of the so-called "unambiguous sat" problem: Is a propositional theory with at most one model satisfiable? This problem is thought to be beyond **P** without being **NP**-hard [Papadimitriou, 1994].

2.4.4 Executability of Actions Polynomial-Time Decidable

In many accounts of planning, it would be unusual for action descriptions to make actions always executable. Nonetheless, action description languages used for planning often have a similarly useful property: not only is there a polynomial-time algorithm for obtaining formulas $state(F)$ and $act(F, A, F')$, but there is also a polynomial-time algorithm for obtaining a classical propositional formula $executable(F, A)$ (in which only atoms from $F \cup A$ occur) that is logically equivalent to

$$state(F) \supset \exists f'\, tr(F, A, f')\,.$$

We will say of such an action description language that it "makes it easy to determine whether actions are executable in a state."

Whenever such a characterization of executability is available, there is an easy transformation that makes actions always executable, as follows. Add a fluent symbol *FailedExecution*. Let $state(F \cup \{FailedExecution\}) = state(F)$, and let $act(F \cup \{FailedExecution\}, A, F' \cup \{FailedExecution'\})$ be the conjunction of the following three formulas.

$$FailedExecution' \equiv FailedExecution \vee \neg executable(F, A)$$
$$\neg FailedExecution' \supset act(F, A, F')$$
$$FailedExecution' \supset F = F'$$

Now a planning problem for the original domain description can be equivalently posed for the transformed description by conjoining $\neg FailedExecution$ with each of $init(F)$ and $goal(F)$. (By the way, notice that this transformation does not introduce additional concurrency or nondeterminism.)

Thus, for instance, a planning problem posed using a STRIPS-based action representation language can be translated in polynomial time into our planning framework, with the additional guarantee that actions will be always executable in the resulting domain description.

We say that the *executability of actions is polynomial-time computable* if the executability of given actions in a given state can always be decided in polynomial time, which is simply to say that, for every interpretation I of $F \cup A$ that satisfies $state(F)$, whether

$$I \models \exists f'\, tr(F, A, f')$$

can be decided in polynomial time.

Of course in our framework it is in general not the case that the executability of given actions in a given state can be determined in polynomial time (assuming $\mathbf{P} \neq \mathbf{NP}$).

Proposition 3 *Determining that given actions are executable in a given state is* **NP***-complete.*

2.4.5 No Concurrent Actions

For much of this paper, we will allow concurrent actions in action domains. In Section 7 we will consider the effects of assuming that actions do not occur concurrently. An action domain is *nonconcurrent* iff

$$tr(F, A, F') \supset \bigwedge_{\substack{A(1),\, A(2) \in A \\ A(1) \neq A(2)}} \neg(A(1) \wedge A(2))$$

is logically valid.

Proposition 4 *Determining that an action domain is nonconcurrent is* **coNP***-complete.*

3 Classical Planning

In our planning framework, classical planning is essentially what is widely known as satisfiability planning [Kautz and Selman, 1992]. In satisfiability planning, an action domain is (also) described by a finite theory of propositional logic. Typically, the language will include action atoms (each representing the proposition that a certain action occurs at a certain time) and fluent atoms (each representing the proposition that a certain fluent holds at a certain time). Assume that action atoms have the form Act_t, where Act is an action symbol and subscript t is a natural number—called the "time stamp" of Act_t. Similarly, assume that fluent atoms have the form P_t, with P a fluent symbol and t a time stamp.

A plan of length k is obtained by finding a model of the formula

$$init(F_0) \wedge \bigwedge_{t=0}^{k-1} tr(F_t, A_t, F_{t+1}) \wedge goal(F_k) \tag{8}$$

where each F_i (respectively A_i) is the set of fluent atoms (respectively action atoms) obtained by adding time stamp i to each fluent symbol (respectively action symbol). The plan is the sequence of (possibly concurrent) action occurrences corresponding to the interpretation of the action atoms.

In order to guarantee that the plan extracted from a model of (8) is valid—that is, always executable and sure to achieve the goal under all possible executions—the initial state must be completely described, and the domain must be deterministic. If these assumptions are indeed satisfied, the existence of a valid plan is expressed by the QBF

$$\exists f_0 \overbrace{\exists a_0\ \exists f_1} \cdots \overbrace{\exists a_{k-1}\ \exists f_k} \left(init(f_0) \wedge \bigwedge_{t=0}^{k-1} tr(f_t, a_t, f_{t+1}) \wedge goal(f_k) \right) \tag{9}$$

where each f_i (respectively a_i) is a tuple of distinct propositional variables corresponding to the propositional constants in F_i (respectively A_i). This shows

once again what is widely known: classical planning, for plans of polynomially-bounded length, belongs to **NP**. (Of course k is polynomially bounded so that the formula $\bigwedge_{t=0}^{k-1} tr(f_t, a_t, f_{t+1})$ can be obtained in polynomially-bounded time from $tr(F, A, F')$.)

Of course the **NP** membership of polynomially-bounded classical planning has been clear at least since Kautz and Selman ([Kautz and Selman, 1992]) pointed out that it can be formulated as satisfiability problem involving a classical propositional theory obtained in polynomial-time. Recall though that, in the current framework, classical planning includes the possibility of concurrent actions. Moreover, it may not be possible to determine in polynomial time whether given actions are executable in a given state. These complications do not push the problem beyond **NP**. On the other hand, hardness can be shown even for one-step plans in action domains in which actions are always executable.

Theorem 1 *Classical planning for plans of length* 1 *is* **NP***-hard, even if actions are always executable.*

Proof. The class **NP** is expressed by QBFs of the form

$$\exists x_1 \cdots x_n \Phi(x_1, \ldots, x_n) \tag{10}$$

where $\Phi(x_1, \ldots, x_n)$ is a formula without quantifiers and without nonlogical constants, whose variables are taken from $x_1, \ldots, x_n$. We need a deterministic action domain with actions always executable, and a planning problem with a completely specified initial state, obtainable from (10) in polynomial time, such that

$$\exists f_0 \ \exists a_0 \ \exists f_1 \ \ (\mathit{init}(f_0) \wedge tr(f_0, a_1, f_1) \wedge \mathit{goal}(f_1)) \tag{11}$$

is true iff (10) is.

Let $F = \{P\}$. Let $A = \{X(1), \ldots, X(n)\}$. (So we introduce an action symbol corresponding to each of the propositional variables.) Let $\mathit{state}(F) = \top$ and $\mathit{act}(F, A, F') = P' \equiv \Phi(A)$. It is clear that this action domain is deterministic. Notice also that actions are always executable. Take $\mathit{init}(F) = P$. So the initial state is completely specified. Take $\mathit{goal}(F) = P$ also. In this case, (11) is

$$\exists\, p_0 \ \exists\, x(1)_0 \cdots x(n)_0 \ \exists\, p_1 \ \ (p_0 \wedge \top \wedge (p_1 \equiv \Phi(x(1)_0, \ldots, x(n)_0)) \wedge \top \wedge p_1)$$

which is equivalent to

$$\exists\, x(1)_0 \cdots x(n)_0 \ \Phi(x(1)_0, \ldots, x(n)_0)$$

which is clearly equivalent to (10). □

Corollary 1 *Classical planning is* **NP***-complete, for plans of polynomially-bounded length.*

Notice that (9) is true iff there is a plan of exactly length k. This is significant because our framework allows for dynamic worlds in which the state may change

even when no actions occur. It is not difficult though to modify a planning problem so that we can instead decide directly the existence of a plan of length *at most* k. Add a fluent symbol *Ended* and an action symbol *End*. Let

$$state(F \cup \{Ended\}) = state(F)$$

and let $act(F \cup \{Ended\}, A \cup \{End\}, F' \cup \{Ended'\})$ be the conjunction of the following three formulas.

$$\begin{gathered} \neg Ended' \supset act(F, A, F') \\ Ended \supset F = F' \\ Ended' \equiv End \vee Ended \end{gathered}$$

Then take

$$init(F \cup Ended) = init(F) \wedge \neg Ended\,.$$

Notice that this transformation does not add nondeterminism, nor does it affect executability.

Finally, notice that the proof of Theorem 1 does not make use of dynamic worlds. That is, the construction guarantees that

$$\bigwedge_{Act \in A} \neg Act \wedge tr(F, A, F') \supset F = F'$$

is logically valid. This fact is of little significance though. It is easy to modify any $tr(F, A, F')$ to satisfy this property. Simply add an action symbol *DummyAction* and add let $tr(F, A \cup \{DummyAction\}, F')$ be the conjunction of $tr(F, A, F')$ and $\bigwedge_{Act \in A} \neg Act \supset DummyAction$. Notice that this transformation does not add nondeterminism. It does have a slight effect on executability—it is no longer possible to perform no action—but if the executability of actions was polynomial-time computable before, it remains so. Therefore, we will typically leave unmentioned the fact that subsequent hardness results also do not require dynamic worlds.

4 Conformant Planning

In conformant planning, a plan is still a sequence of (possibly concurrent) actions, but the action domain may be nondeterministic, and the initial state may be incompletely specified. As usual, a plan is valid if it is guaranteed to be executable, and every possible execution achieves the goal. The key issues can be illustrated by considering plans of length 1. There is at least one possible initial state:

$$\exists f_0 \; init(f_0)\,. \tag{12}$$

There is an unconditional one-step plan that is executable starting from any of the possible initial states:

$$\exists a_0 \; \forall f_0 \; (init(f_0) \supset \exists f_1 \; tr(f_0, a_0, f_1))\,. \tag{13}$$

There is an unconditional one-step plan that achieves the goal in every possible execution starting from any of the possible initial states:

$$\exists a_0 \; \forall f_0 \; (\mathit{init}(f_0) \supset \forall f_1 \; (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))) \, .$$

So to say there is a one-step unconditional plan that is executable (starting in any possible initial state) and sufficient to achieve the goal (when executed starting in any possible initial state), we can write

$$\exists a_0 \; \forall f_0 \; (\mathit{init}(f_0) \supset \exists f_1 \; \mathit{tr}(f_0, a_0, f_1) \wedge \; \forall f_1 \; (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))) \, . \quad (14)$$

Consequently, the decision problem for one-step unconditional planning is expressed by the conjunction of (12) and (14).

We can put (14) in prenex form as follows.

$$\exists a_0 \; \forall f_0 \; \forall f_1 \; \exists f_1' \; (\mathit{init}(f_0) \supset \mathit{tr}(f_0, a_0, f_1') \wedge (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1)))$$

This shows that the decision problem for one-step conformant planning belongs to the complexity class Σ_3^P. Hardness can also be shown.

Theorem 2 *Conformant planning for plans of length 1 is Σ_3^P-hard.*

Proof. We need to obtain from

$$\exists \, x_1 \cdots x_l \; \forall \, y_1 \cdots y_m \; \exists \, z_1 \cdots z_n \; \Phi(x_1, \ldots, x_l, y_1, \ldots, y_m, z_1, \ldots, z_n) \quad (15)$$

in polynomial-time a set F of fluent symbols, a set A of action symbols, and formulas $\mathit{tr}(F, A, F')$, $\mathit{init}(F)$ and $\mathit{goal}(F)$ for which the conjunction of (12) and (14) is logically equivalent to (15).

Take $F = \{Y(1), \ldots, Y(m), Z(1), \ldots, Z(n)\}$, corresponding to propositional variables $y_1, \ldots, y_m, z_1, \ldots, z_n$, and take $A = \{X(1), \ldots, X(l)\}$, corresponding to propositional variables $x_1, \ldots, x_l$. Let $\mathit{state}(F) = \top$, and let

$$\mathit{act}(F, A, F') = \Phi(X(1), \ldots, X(l), Y(1), \ldots, Y(m), Z(1)', \ldots, Z(n)') \, .$$

Take $\mathit{init}(F) = \mathit{goal}(F) = \top$. Since $\mathit{init}(F) = \top$, (12) is true. So we need to show that (14) is equivalent to (15). Given the choices of $\mathit{init}(F)$, $\mathit{goal}(F)$, and $\mathit{state}(F)$, we can simplify (14) to obtain

$$\exists a_0 \; \forall f_0 \; \exists f_1 \; \mathit{act}(f_0, a_0, f_1) \, . \quad (16)$$

The formula $\mathit{act}(f_0, a_0, f_1)$ in this case can be written more fully as

$$\Phi(x(1)_0, \ldots, x(l)_0, y(1)_0, \ldots, y(m)_0, z(1)_1, \ldots, z(n)_1) \, ,$$

in which variables $y(1)_1, \ldots, y(m)_1, z(1)_0, \ldots, z(n)_0)$ do not appear. Consequently, (16) is logically equivalent to

$$\exists \, x(1)_0 \cdots x(l)_0 \; \forall \, y(1)_0 \cdots y(m)_0 \; \exists \, z(1)_1 \cdots z(n)_1 \\ \Phi(x(1)_0, \ldots, x(l)_0, y(1)_0, \ldots, y(m)_0, z(1)_1, \ldots, z(n)_1)$$

which is in turn equivalent to (15). □

So far we have considered conformant planning only for plans of length 1. As a first step toward the general formulation, let us consider plans of length 2. We need not alter the formula (12) expressing the existence of at least one possible initial state. To say that there is an unconditional plan whose first step is executable in any possible initial state, we can (again) write (13). To say that there is an unconditional plan such that the second step is executable in any state reached after executing the first step in a possible initial state, we can write

$$\exists a_0\ \exists a_1\ \forall f_0\ \forall f_1 (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \exists f_2\ \mathit{tr}(f_1, a_1, f_2))\,.$$

To say that there is an unconditional two-step plan whose execution is sufficient to achieve the goal (starting in any possible initial state), we can write

$$\exists a_0\ \exists a_1\ \forall f_0\ \forall f_1\ \forall f_2\ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \wedge \mathit{tr}(f_1, a_1, f_2) \supset \mathit{goal}(f_2))\,.$$

Combining (13) and these last two formulas appropriately in in prenex form, we can obtain

$$\begin{array}{l}\exists a_0\ \exists a_1\ \forall f_0\ \forall f_1\ \forall f_2\ \exists f_1'\ \exists f_2' \\ \qquad ((\mathit{init}(f_0) \supset \mathit{tr}(f_0, a_0, f_1')) \\ \qquad\ \wedge\ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \mathit{tr}(f_1, a_1, f_2')) \\ \qquad\ \wedge\ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \wedge \mathit{tr}(f_1, a_1, f_2) \supset \mathit{goal}(f_2)))\ .\end{array} \tag{17}$$

So conformant planning for plans of length 2 is expressed by the conjunction of (12) and (17).

The general form is similar. For conformant planning for plans of length k, take the conjunction of (12) and

$$\begin{array}{l}\exists a_0 \cdots \exists a_{k-1}\ \forall f_0 \cdots \forall f_k\ \exists f_1' \cdots \exists f_k' \\ \qquad \left(\bigwedge_{t=0}^{k-1} \left(\mathit{init}(f_0) \wedge \bigwedge_{u=0}^{t-1} \mathit{tr}(f_u, a_u, f_{u+1}) \supset \mathit{tr}(f_t, a_t, f_{t+1}')\right)\right. \\ \qquad\ \left.\wedge \left(\mathit{init}(f_0) \wedge \bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \supset \mathit{goal}(f_k)\right)\right)\,.\end{array} \tag{18}$$

The form of (18) shows that conformant planning for plans of polynomially-bounded length is in Σ_3^P.

Corollary 2 *Conformant planning is Σ_3^P-complete, for plans of polynomially-bounded length.*

In a comment following their formally stated complexity results, Eiter *et al.* [2000] mention a similar Σ_3^P-completeness result for conformant planning. Their result is restricted to plans of a fixed length, but this can probably be relaxed. They employ a high-level action description language whose salient feature in this context is that determining executability of actions in a state is **NP**-complete. The semantics of their action language is closely modeled on logic programming under the answer set (or stable model) semantics [Gelfond

and Lifschitz, 1991], and it appears to match up well in terms of complexity with the action representation framework used in this paper.[1]

If we assume that actions are always executable, we can eliminate the check for plan executability from (18), which can then be reduced to

$$\exists a_0 \cdots \exists a_{k-1}\ \forall f_0 \cdots \forall f_k\ \left(\mathit{init}(f_0) \wedge \bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \supset \mathit{goal}(f_k) \right) . \qquad (19)$$

So the complexity of conformant planning falls to Σ_2^P if actions are always executable. We can show hardness for plans of length 1 even if we assume in addition that the action domain is deterministic.

Theorem 3 *Conformant planning for plans of length 1 is Σ_2^P-hard for deterministic planning domains in which actions are always executable.*

Proof. For plans of length 1, (19) becomes

$$\exists a_0\ \forall f_0\ \forall f_1\ \ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1)) . \qquad (20)$$

We need to obtain from

$$\exists x_1 \cdots x_m \forall y_1 \cdots y_n\ \Phi(x_1, \ldots, x_m, y_1, \ldots, y_n) \qquad (21)$$

in polynomial time a set F of fluent symbols, a set A of action symbols, and formulas $\mathit{tr}(F, A, F')$, $\mathit{init}(F)$ and $\mathit{goal}(F)$ for which the conjunction of (12) and (20) is logically equivalent to (21). Moreover, the domain should be deterministic with actions always executable. Let $F = \{P, Y(1), \ldots, Y(n)\}$ and $A = \{X(1), \ldots, X(m)\}$. Let $\mathit{state}(F) = \top$, and let $\mathit{act}(F, A, F')$ be the conjunction of

$$P' \equiv \Phi(X(1), \ldots, X(m), Y(1), \ldots, Y(n))$$

and

$$\bigwedge_{i=1}^{n} (Y(i)' \equiv Y(i)) .$$

Notice that this action domain is deterministic, with actions always executable. Take $\mathit{init}(F) = \top$ and $\mathit{goal}(F) = P$. By choice of $\mathit{init}(F)$, (12) is logically valid, so it remains to show that (20) is logically equivalent to (21). Since $\mathit{init}(F) = \mathit{state}(F) = \top$, we can simplify (20) to obtain

$$\exists\, x(1)_0 \cdots x(m)_0\ \forall\, p_0\ y(1)_0 \cdots y(n)_0\ \forall\, p_1\ y(1)_1 \cdots y(n)_1$$
$$\left((p_1 \equiv \Phi(x(1)_0, \ldots, x(m)_0, y(1)_0, \ldots, y(n)_0)) \wedge \bigwedge_{i=1}^{n} (y(i)_1 \equiv y(i)_0) \supset p_1 \right)$$

[1] On March 27, 2000, commenting via email on a preliminary draft of [Eiter *et al.*, 2000] that did not contain the Σ_3^P result, I suggested to Nicola Leone that I expected them to discover that an assumption of p-time decidability of executability of actions would be necessary for their planning system system, which is based on a Σ_2^P variant of logic programming. They subsequently found this to be the case.

which is logically equivalent to

$$\exists\, x(1)_0 \cdots x(m)_0\ \forall\, y(1)_0 \cdots y(n)_0\ \Phi(x(1)_0, \ldots, x(m)_0, y(1)_0, \ldots, y(n)_0)$$

which is clearly equivalent to (21). □

Although we obtain Σ_2^P-hardness for one-step conformant planning with actions always executable, we can show that conformant planning falls to Σ_2^P even under the weaker assumption that the executability of actions in a state can be determined in polynomial time.

Theorem 4 *Conformant planning is Σ_2^P-complete, for plans of polynomially-bounded length, if the executability of actions is polynomial-time computable.*

A similar result (Theorem 2) in [Baral *et al.*, 2000] is obtained using an action representation language that guarantees determinism and nonconcurrency, and makes it easy to determine whether actions are executable in a state (as discussed in Section 2.4). Our analysis suggests that only the last property is crucial for placing the problem within Σ_2^P. (For a more closely related hardness result, see Theorem 10(iii) in Section 7.)

On the other hand, as mentioned previously, [Eiter *et al.*, 2000] employs a relatively expressive action description language, and a related Σ_2^P result reported there (Theorem 3) is also obtained by directly imposing polynomial-time decidability of executability of given actions in a given state. The latter result again assumes that plan length is fixed (not just polynomially-bounded).

We are now in position to verify that our formulation of conformant planning reduces to classical planning if we assume determinism and a completely specified initial state. Under the assumption of a completely specified initial state, (12) can be dropped, and (19) can be replaced by

$$\exists a_0 \cdots \exists a_{k-1}\, \exists f_0\, \exists f_1 \cdots \exists f_k \left(\mathit{init}(f_0) \wedge \bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \wedge \mathit{goal}(f_k) \right)$$

which is equivalent to (9).

5 Conditional Planning

Conditional planning is an extension of conformant planning in which the actions to be performed in executing a plan may depend on the previous execution history. In this setting, interesting questions arise concerning the nature of this dependence. For the analysis in this section, we simply assume that each step in the execution of a conditional plan may depend arbitrarily on the previous execution history. Thus, in a rather general sense, we address here the fundamental question of existence of a valid conditional plan. In the following section, we go on to consider what happens if the current state is only partially observable during plan execution.

As with conformant planning, it is convenient to begin by considering plans of length 1. Of course the existence of at least one possible initial state is again expressed by (12). To say that there is a one-step conditional plan that is executable in any possible initial state, we can write

$$\forall f_0 \ (\mathit{init}(f_0) \supset \exists a_0 \ \exists f_1 \ \mathit{tr}(f_0, a_0, f_1)) \,. \tag{22}$$

To say that there is a one-step conditional plan that is sufficient to achieve the goal if executed starting in any possible initial state, we can write

$$\forall f_0 \ (\mathit{init}(f_0) \supset \exists a_0 \ \forall f_1 \ (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))) \,.$$

Consequently, a formula for the decision problem corresponding to one-step conditional planning is the conjunction of (12) and

$$\forall f_0 \ \exists a_0 \ \exists f_1' \ \forall f_1 (\mathit{init}(f_0) \supset \mathit{tr}(f_0, a_0, f_1') \wedge (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))) \,. \tag{23}$$

This shows that one-step conditional planning belongs to complexity class Π_3^P.

Assuming that actions are always executable doesn't help much here. Under this assumption, we can simplify (23) to obtain

$$\forall f_0 \ \exists a_0 \ \forall f_1 \ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1)) \,. \tag{24}$$

As a first step toward obtaining the general form, let's consider conditional plans of length 2. We need not alter the formula (12) expressing the existence of at least one possible initial state. To say that there is a conditional plan whose first step is executable in any possible initial state, we (again) write (22). To say that there is a conditional plan such that the second step is executable in any state reached after executing the first step in any possible initial state, we can write

$$\forall f_0 \ (\mathit{init}(f_0) \supset \exists a_0 \ \forall f_1 \ (\mathit{tr}(f_0, a_0, f_1) \supset \exists a_1 \ \exists f_2 \ \mathit{tr}(f_1, a_1, f_2))) \,.$$

To say that there is a two-step conditional plan whose execution is sufficient to achieve the goal (starting in any possible initial state), we can write

$$\forall f_0 \ (\mathit{init}(f_0) \supset \exists a_0 \ \forall f_1 \ (\mathit{tr}(f_0, a_0, f_1) \supset \exists a_1 \ \forall f_2 \ (\mathit{tr}(f_1, a_1, f_2) \supset \mathit{goal}(f_2)))) \,.$$

Combining (22) and these last two formulas appropriately in prenex form, we can obtain

$$\begin{aligned} \forall f_0 \ \exists a_0 \ \forall f_1 \ \exists a_1 \ \exists f_1' \ \exists f_2' \ \forall f_2 \\ ((\mathit{init}(f_0) \supset \mathit{tr}(f_0, a_0, f_1')) \\ \wedge \ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \mathit{tr}(f_1, a_1, f_2')) \\ \wedge \ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \wedge \mathit{tr}(f_1, a_1, f_2) \supset \mathit{goal}(f_2))) \,. \end{aligned}$$

Notice that, written in this form, it is clear that the "choice" of actions to be performed at time 0 depends only on the values of fluents at time 0. Similarly, the choice of actions to be performed at time 1 depends only on the values of

fluents at time 0, the actions performed at time 0, and the values of fluents at time 1.

Here's the general form. For conditional planning for plans of length k, take the conjunction of (12) and the following.

$$\begin{array}{l}\overbrace{\forall f_0\,\exists a_0}\ \cdots\ \overbrace{\forall f_{k-1}\,\exists a_{k-1}}\ \exists f'_1 \cdots \exists f'_k\ \forall f_k \\ \quad \left(\bigwedge_{t=0}^{k-1}\left(\mathit{init}(f_0) \wedge \bigwedge_{u=0}^{t-1} \mathit{tr}(f_u, a_u, f_{u+1}) \supset \mathit{tr}(f_t, a_t, f'_{t+1})\right)\right. \\ \quad \left.\wedge\left(\mathit{init}(f_0) \wedge \bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \supset \mathit{goal}(f_k)\right)\right)\end{array} \tag{25}$$

So the decision problem for conditional planning for plans of length k ($k > 0$) belongs to complexity class Π^P_{2k+1}.

If actions are always executable, (25) can be simplified:

$$\forall f_0\ \exists a_0\ \forall f_1 \cdots \exists a_{k-1}\ \forall f_k\ \left(\mathit{init}(f_0) \wedge \bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \supset \mathit{goal}(f_k)\right) \tag{26}$$

Theorem 5 *Conditional planning for plans of length k is Π^P_{2k+1}-hard, even if actions are always executable.*

Proof (*for $k = 1$ only*). We need to obtain from

$$\forall x_1 \cdots x_l \exists y_1 \cdots y_m \forall z_1 \cdots z_n \Phi(x_1, \ldots, x_l, y_1, \ldots, y_m, z_1, \ldots, z_n) \tag{27}$$

in polynomial-time a set F of fluent symbols, a set A of action symbols, and formulas $\mathit{tr}(F, A, F')$, $\mathit{init}(F)$ and $\mathit{goal}(F)$ for which the conjunction of (12) and (24) is logically equivalent to (27). Moreover, we require that actions be always executable. Let $F = \{P, X(1), \ldots, X(l), Z(1), \ldots, Z(n)\}$ and $A = \{Y(1), \ldots, Y(m)\}$. Take $\mathit{state}(F) = \top$. Let $\mathit{act}(F, A, F')$ be the formula

$$P' \equiv \Phi(X(1), \ldots, X(l), Y(1), \ldots, Y(m), Z(1)', \ldots, Z(n)')\,.$$

Notice that actions are always executable. Let $\mathit{init}(F) = \top$ and $\mathit{goal}(F) = P$. Since $\mathit{init}(F) = \top$, we need to show that (24) is equivalent to (27). Given the choices of $\mathit{init}(F)$ and $\mathit{state}(F)$, we can simplify (24) to obtain

$$\begin{array}{l}\forall p_0\, x(1)_0 \cdots x(l)_0\, z(1)_0 \cdots z(n)_0\ \ \exists y(1)_0 \cdots y(m)_0 \\ \quad \forall p_1\, x(1)_1 \cdots x(l)_1\, z(1)_1 \cdots z(n)_1 \\ \quad\quad ((p_1 \equiv \Phi(x(1)_0, \ldots, x(l)_0, y(1)_0, \ldots, y(m)_0, z(1)_1, \ldots, z(n)_1)) \supset p_1)\end{array}$$

which is logically equivalent to

$$\begin{array}{l}\forall x(1)_0 \cdots x(l)_0\ \exists y(1)_0 \cdots y(m)_0\ \forall z(1)_1 \cdots z(n)_1 \\ \quad\quad \Phi(x(1)_0, \ldots, x(l)_0, y(1)_0, \ldots, y(m)_0, z(1)_1, \ldots, z(n)_1)\end{array}$$

which is in turn equivalent to (27). □

Corollary 3 *Conditional planning for plans of length k ($k > 0$) is Π^P_{2k+1}-complete.*

Corollary 4 *Conditional planning is* **PSPACE***-complete, for plans of polynomially-bounded length.*

A related result, obtained for probabilistic planning, appears in [Littman, 1997] (Theorem 2). Littman studies the complexity of deciding plan existence when the action domain is represented by a (fully-observable) Markov Decision Process (MDP). He shows that when the MDP is given a concise ("factored") representation, deciding existence of a polynomial-length plan is **PSPACE**-complete. It is interesting that Littman's proof of **PSPACE**-hardness uses only nondeterminism—the probabilities of different outcomes do not play a role. Moreover, in his setting, actions are always executable. Hence his hardness result is very similar to the hardness in Corollary 4, which is obtained from Theorem 5. In fact, since the MDP setting does not allow concurrent actions, Littman's result is even more similar to Theorem 10(iv) in Section 7, and can be understood to imply it. On the other hand, the previous discussion makes it clear that Littman's **PSPACE** membership result does not imply the membership result in Corollary 4, and, of course, Corollary 4 does not imply Littman's result.

What if we assume that the action domain is deterministic? Then (25) can be equivalently replaced with

$$\forall f_0 \overbrace{\exists a_0\ \exists f_1} \cdots \overbrace{\exists a_{k-1}\ \exists f_k} \left(\mathit{init}(f_0) \supset \bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \wedge \mathit{goal}(f_k) \right) \quad (28)$$

which drops the complexity to Π^P_2! Moreover, we can show hardness even if actions are always executable.

Theorem 6 *For deterministic planning domains in which actions are always executable, conditional planning for plans of length 1 is Π^P_2-hard.*

Proof. For plans of length 1, (28) is

$$\forall f_0\ \exists a_0\ \exists f_1\ \ (\mathit{init}(f_0) \supset \mathit{tr}(f_0, a_1, f_1) \wedge \mathit{goal}(f_1)) \quad (29)$$

and we need to obtain from (7) in polynomial time a set F of fluent symbols, a set A of action symbols, and formulas $\mathit{tr}(F, A, F')$, $\mathit{init}(F)$ and $\mathit{goal}(F)$ for which the conjunction of (12) and (29) is logically equivalent to (7). Moreover, the domain should be deterministic, with actions always executable. Let $F = \{P, X(1), \ldots, X(m)\}$ and $A = \{Y(1), \ldots, Y(n)\}$. Let $\mathit{state}(F) = \top$, and let $\mathit{act}(F, A, F')$ be the conjunction of the following two formulas.

$$\bigwedge_{i=1}^{m} (X(i)' \equiv X(i))$$

$$P' \equiv \Phi(X(1), \ldots, X(m), Y(1), \ldots, Y(n))$$

Notice that the domain is deterministic and actions are always executable. Take $\mathit{init}(F) = \top$ and $\mathit{goal}(F) = P$. By choice of $\mathit{init}(F)$, (12) is logically valid, so it remains to show that (29) is logically equivalent to (7). Since $\mathit{init}(F) = \mathit{state}(F) = \top$, we can simplify (29) to obtain

$$\forall p_0\, x(1)_0 \cdots x(m)_0\ \exists y(1)_0 \cdots y(n)_0\ \exists p_1\, x(1)_1 \cdots x(m)_1 \\ (\textstyle\bigwedge_{i=1}^{m}(x(i)_1 \equiv x(i)_0) \wedge (p_1 \equiv \Phi(x(1)_0, \ldots, x(m)_0, y(1)_0, \ldots, y(n)_0)) \supset p_1)$$

which is logically equivalent to

$$\forall x(1)_0 \cdots x(m)_0\ \exists y(1)_0 \cdots y(n)_0\ \Phi(x(1)_0, \ldots, x(m)_0, y(1)_0, \ldots, y(n)_0)$$

which is clearly equivalent to (7). □

A similar hardness result is obtained in [Rintanen, 1999] (Theorem 2), using a deterministic action language, without concurrency, for which it is easy to determine executability of actions in a state.

Corollary 5 *Conditional planning in deterministic planning domains is* Π_2^P*-complete, for plans of polynomially-bounded length.*

A similar completeness result appears in [Baral *et al.*, 2000] (Theorem 8). As mentioned previously, they use deterministic action domains for which it is easy to determine executability of actions in a state. Our analysis suggests that the key is determinism.

In fact, since the approaches of Rintanen and Baral *et al.* assume nonconcurrency, their hardness results are more closely related to Theorem 10(v) in Section 7.

6 Conditional Planning with Partial Observability

Our analysis up to this point in the paper cannot account for an intriguing result obtained by Baral *et al.* [2000]. Despite the fact that they consider only deterministic action domains, their Theorem 4 identifies a variant of polynomial-length conditional planning that is **PSPACE**-hard. The key is that their action description language incorporates a rudimentary account of sensing actions. This feature allows them to model planning domains in which knowledge of the current state during plan execution is incomplete because the current state is only partially observable. Intuitively it is not surprising that in such settings conditional planning may turn out to be harder. The key to their hardness proof is that they can construct action domains in which knowledge of the current value of a given fluent is available only at a particular step in the execution of a plan. Hence, for them an incompletely known initial state can play, essentially, the role that nondeterminism plays in the proof construction for our Theorem 5 (which in turn yields the **PSPACE** result of Corollary 4).

The fundamental mechanism used in [Baral *et al.*, 2000] for restricting access to knowledge of the current state during plan exection can be modeled in our setting as follows. Partition the set F of fluent symbols into two sets F^O, F^N intuitively corresponding to observable and nonobservable fluents, respectively.

Let's start again with plans of length 1. As usual, the existence of at least one possible initial state is expressed by (12). The existence of a conditional one-step plan that can be executed in any possible initial state is expressed by

$$\forall f_0^O \ \exists a_0 \ \forall f_0^N \ (\mathit{init}(f_0) \supset \exists f_1 \ \mathit{tr}(f_0, a_0, f_1)) \,.$$

Notice that the choice of actions to be performed at time 0 depends only on the values of *observable* fluents at time 0. The existence of a conditional one-step plan sufficient to achieve the goal (when executed starting in any possible initial state) is expressed by

$$\forall f_0^O \ \exists a_0 \ \forall f_0^N \ (\mathit{init}(f_0) \supset \forall f_1 \ (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))) \,.$$

Again, the choice of actions to be performed at time 0 depends only on the values of observable fluents at time 0. As we did previously for simpler forms of planning, we combine formulas expressing executability and sufficiency, obtaining

$$\begin{aligned} \forall f_0^O \ \exists a_0 \ \forall f_0^N \ \forall f_1 \ \exists f_1' \ (\mathit{init}(f_0) \supset \ & \mathit{tr}(f_0, a_0, f_1') \\ & \wedge (\mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))) \,, \end{aligned} \tag{30}$$

which, conjoined with (12), expresses the existence of a valid conditional plan under conditions of partial observability of the current state during plan execution. This shows that one-step conditional planning with partial observability belongs to complexity class Π_4^P.

To extend this to two-step conditional planning, we need to express the existence of a two-step conditional plan such that the second step is executable in any state reached by executing the first step in any possible initial state. In doing so, we must capture the fact that (i) the choice of actions to be performed at time 0 depends only on the observable fluents at time 0, and (ii) the choice of actions to be performed at time 1 depends only on the observable fluents at time 0, the actions performed at time 0, and the observable fluents at time 1.

$$\forall f_0^O \ \exists a_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \ \forall f_1^N \ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \exists f_2 \ \mathit{tr}(f_1, a_2, f_2))$$

We similarly need to express that there exists such a plan that is sufficient to achieve the goal when executed starting in any possible initial state.

$$\begin{aligned} \forall f_0^O \ \exists a_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \ \forall f_1^N \\ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \forall f_2 \ (\mathit{tr}(f_1, a_2, f_2) \supset \mathit{goal}(f_2))) \end{aligned}$$

These observations suggest that two-step conditional planning with partial observability can be expressed by the conjunction of (12) and

$$\begin{aligned} \forall f_0^O \ \exists a_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \ \forall f_1^N \ \forall f_2 \ \exists f_1' \ \exists f_2' \\ ((\mathit{init}(f_0) \supset \mathit{tr}(f_0, a_0, f_1')) \\ \wedge (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \mathit{tr}(f_1, a_1, f_2')) \\ \wedge (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \wedge \mathit{tr}(f_1, a_1, f_2) \supset \mathit{goal}(f_2))) \,. \end{aligned} \tag{31}$$

Here's the general form. For conditional planning with partial observability for plans of length k ($k > 0$), take the conjunction of (12) and the following.

$$\begin{array}{l}\overbrace{\forall f_0^O\,\exists a_0}\cdots\overbrace{\forall f_{k-1}^O\,\exists a_{k-1}}\;\forall f_0^N\cdots\forall f_{k-1}^N\,\forall f_k\,\exists f_1'\cdots\exists f_k' \\ \quad\Big(\bigwedge_{t=0}^{k-1}\Big(\mathit{init}(f_0)\wedge\bigwedge_{u=0}^{t-1}\mathit{tr}(f_u,a_u,f_{u+1})\supset\mathit{tr}(f_t,a_t,f_{t+1}')\Big) \\ \quad\wedge\;\Big(\mathit{init}(f_0)\wedge\bigwedge_{t=0}^{k-1}\mathit{tr}(f_t,a_t,f_{t+1})\supset\mathit{goal}(f_k)\Big)\Big)\end{array} \tag{32}$$

So for plans of length k, conditional planning with partial observability belongs to complexity class Π_{2k+2}^P. We can immediately conclude, based on the hardness results established in the previous section (with "complete" observability), that the problem is **PSPACE**-complete.

Corollary 6 *Conditional planning with partial observability is* **PSPACE***-complete, for plans of polynomially-bounded length.*

We can refine this result.

Theorem 7 *Conditional planning with partial observability for plans of length k ($k > 0$) is Π_{2k+2}^P-complete.*

Proof (*for $k = 1$ only*). We have already established membership by observation. For hardness, we need to obtain from

$$\begin{array}{l}\forall x_1\cdots x_l\exists y_1\cdots y_m\forall z_1\cdots z_n\exists w_1\cdots w_r \\ \qquad\Phi(x_1,\ldots,x_l,y_1,\ldots,y_m,z_1,\ldots,z_n,w_1,\ldots,w_r)\end{array} \tag{33}$$

in polynomial-time disjoint sets F^O and F^N of fluent symbols, a set A of action symbols and formulas $\mathit{tr}(F,A,F')$, $\mathit{init}(F)$ and $\mathit{goal}(F)$ for which the conjunction of (12) and (30) is logically equivalent to (33). For simplicity in displaying the proof, we will show only a single variable for each level of quantification. As in previous proof constructions, all variables within a given level of quantification are handled symmetrically, so this merely simplifies the presentation.

Let $F^O = \{X\}$, $F^N = \{Z, W\}$ and $A = \{Y\}$. Let $\mathit{state}(F) = \top$ and

$$\mathit{act}(F,A,F') = \Phi(X,Y,Z,W')\,.$$

Take $\mathit{init}(F) = \top$ and $\mathit{goal}(F) = \top$. Since $\mathit{init}(F) = \top$, (12) is true, and we need to show that (30) is equivalent to (33). Given the choices of $\mathit{init}(F)$, $\mathit{state}(F)$, and $\mathit{goal}(F)$ we can simplify (30) to obtain

$$\forall x_0\;\exists y_0\;\forall z_0\,w_0\;\exists x_1\,z_1\,w_1\;\Phi(x_0,y_0,z_0,w_1)$$

which is equivalent to

$$\forall x_0\;\exists y_0\;\forall z_0\;\exists w_1\;\Phi(x_0,y_0,z_0,w_1)$$

which is clearly what we're after. □

If actions are always executable, take the conjunction of (12) and the following.

$$\overbrace{\forall f_0^O \, \exists a_0} \cdots \overbrace{\forall f_{k-1}^O \, \exists a_{k-1}} \; \forall f_0^N \cdots \forall f_{k-1}^N \; \forall f_k \\ \Big(\mathit{init}(f_0) \wedge \textstyle\bigwedge_{t=0}^{k-1} \mathit{tr}(f_t, a_t, f_{t+1}) \supset \mathit{goal}(f_k) \Big) \tag{34}$$

So the problem falls to Π^P_{2k+1}, and we've already established hardness for the special case when all fluents are observable (Theorem 5). Here's a stronger hardness result, in which we assume also that the action domain is deterministic.

Theorem 8 *For deterministic planning domains with actions always executable, conditional planning with partial observability for plans of length k is Π^P_{2k+1}-hard.*

In conjunction with Corollary 3, this result provides evidence that nondeterminism and partial observability are to some degree interchangeable in conditional planning. In fact, it has often been observed that nondeterminism in an action domain can be eliminated by adding fluents sufficient to make the domain deterministic. Of course this is only true in general if we restrict consideration to k steps: each state must include within it all information necessary to make k time steps deterministic. It is clear that no more than $k \cdot |F| \cdot 2^{|A|}$ additional fluents are necessary (or $k \cdot |F| \cdot (|A| + 1)$ if we don't allow concurrency), but it is not clear whether significantly fewer are sufficient in general.

If nondeterminism is elminated in this fashion, and the auxiliary fluents are declared nonobservable, as indeed they should be, then it is essentially as if we were doing conditional planning in the original nondeterministic domain. But actually doing this—eliminating nondeterminism for k time steps by adding auxiliary fluents—appears to be a hard problem in general. In practice, it seems that this approach is typically entertained in contexts where there is clearly a polynomial-time syntactic method available.

Corollary 7 *For deterministic planning domains with actions always executable, conditional planning with partial observability is* **PSPACE***-hard, for plans of polynomially-bounded length.*

Corollary 7 is closely related to Theorem 4 of [Baral *et al.*, 2000], which established **PSPACE**-completeness for conditional planning of essentially this kind, but using a language without concurrent actions. (Hence their result is even more closely related to Theorem 10(vi) in the next section.)

Theorem 9 *Conditional planning with partial observability for plans of length k ($k > 0$) is Π^P_{2k+1}-complete, if the executability of actions is polynomial-time computable.*

We are now in position to verify that all forms of planning considered in this paper can be seen as special cases of conditional planning with partial observability. If all fluents are observable, (32) reduces to a formula equivalent to (25), so, as one would expect, conditional planning with partial observability

includes conditional planning with "complete" observability as a special case. If on the other hand no fluents are observable, (32) reduces to (18), showing that conformant planning too is a special case of conditional planning with partial observability. Of course we observed previously that classical planning is a special case of conformant planning.

7 Hardness without Concurrent Actions

The action description framework used in this paper allows concurrent actions, and all results so far in this paper concern action domains in which concurrent execution of actions is allowed. We will now verify that this has little effect on complexity.

Recall that Section 2 included a definition of nonconcurrent action domains and a complexity result (Proposition 4) showing that determining nonconcurrency is **coNP**-complete. It is convenient to introduce a special notion of "always executable" suitable for nonconcurrent domains.

Nonconcurrent actions are always executable iff the following formula is logically valid.

$$state(F) \ \wedge \bigwedge_{\substack{A(1),\, A(2) \in A \\ A(1) \neq A(2)}} \neg(A(1) \wedge A(2)) \ \supset \ \exists f' \ tr(F, A, f')$$

Of course a nonconcurrency assumption does not cause any problem we consider to lose membership in a complexity class, but we must consider whether hardness results still hold.

The proofs of hardness for Propositions 1–3 do not utilize concurrency, and so those results hold even for nonconcurrent domains.

Theorems 1–3 and 5–8 establish hardness results for plans of fixed length, but the proof constructions rely on concurrency. Corresponding hardness results for nonconcurrent domains can be established by allowing instead plans of length $|A|$ (so plan length can be linearly-bounded in the size of input). In each case, there is a corresponding proof construction in which A is the set of action symbols used in the original (concurrent) construction and actions previously performed concurrently are performed sequentially instead. The corresponding results are collected in the following theorem.

Theorem 10 *For nonconcurrent domains:*

(i) Classical planning for plans of length $|A|$ is **NP***-hard, even when actions are always executable.*

(ii) Conformant planning for plans of length $|A|$ is Σ_3^P-hard.

(iii) Conformant planning for plans of length $|A|$ is Σ_2^P-hard for deterministic planning domains in which actions are always executable.

(iv) Conditional planning for plans of length $|A|$ is **PSPACE***-hard, even if actions are always executable.*

(v) Conditional planning for plans of length $|A|$ is Π_2^P-hard for deterministic action domains in which actions are always executable.

(vi) Conditional planning with partial observability for plans of length $|A|$ is **PSPACE***-hard for deterministic action domains in which actions are always executable.*

Of course the more refined results for conditional planning cannot be conveniently expressed here. (For instance, we do not say that conditional planning for plans of length $|A| \cdot k$ is Π_{2k+1}^P-hard.) Instead, we settle for the less specific **PSPACE** results.

We have noted that the complexity results from [Eiter *et al.*, 2000] most closely related to those of this paper employ an assumption of fixed plan length (rather than polynomially-bounded plan length). It may be interesting to consider the effect of assuming both nonconcurrency and fixed plan length.

For classical and conformant planning, a fixed plan length guarantees that the number of possible plans with no concurrent actions is polynomially-bounded: there are $(|A| + 1)^k$ nonconcurrent plans of length k. Consequently, the existential quantifiers over actions in formula (9) for classical planning and formula (18) for conformant planning can be eliminated in polynomial time.

It is not clear that this possibility reduces the complexity of classical planning. Roughly speaking, after quantifiers over actions are eliminated from (9), we are left with a disjunction of a polynomial number of "unambiguous sat" problems (mentioned previously, Section 2.4.3).

On the other hand, when plan length is fixed the complexity of conformant planning drops for nonconcurrent domains.

Theorem 11 *For nonconcurrent domains:*

(i) Conformant planning for plans of length 1 is Π_2^P-hard.

(ii) Conformant planning for plans of fixed length is Π_2^P-complete.

(iii) For deterministic planning domains in which actions are always executable, conformant planning for plans of length 1 is D^P-hard.

(iv) For deterministic planning domains in which the executability of actions is polynomial-time computable, conformant planning for plans of fixed length is D^P-complete.

Parts (ii) and (iv) of Theorem 11 are very similar to results in [Eiter *et al.*, 2000] (Theorem 4 and subsequent remarks).

What about the effect of fixed plan length on the complexity of conditional planning without concurrency? It is clear that we do not obtain a polynomial bound on the number of possible plans. In fact, we do not even obtain a

polynomial bound on the space needed to represent a conditional plan, since a conditional plan is essentially a function that maps the observable portion of a partial plan execution to the next action to be performed. On this view, for plan length k, a plan function has a domain of size $\Sigma_{i=1}^{k}(2^{|F^O|})^i$.

8 Conditional Planning with Polynomially-Bounded Plan Representations

Since even a fixed plan length does not yield a polynomial bound on the size of conditional plans, we may wish to consider imposing such a bound directly. In fact this possibility is crucial in the computational approach to conditional planning described by Rintanen [1999]. As mentioned previously, Rintanen's paper includes a proof of Π_2^P-hardness for conditional planning. But his main concern is with computational approaches based on problem encodings that directly restrict the size of possible plans to be considered. More precisely, he introduces several different planning problem encodings (in second-order propositional logic), each requiring only polynomial time to construct (given a domain description in his action representation language). Each such encoding restricts consideration to the family of conditional plans that are representable *in that particular polynomial-size encoding*.

Rintanen claims for such computational approaches a complexity of Σ_2^P. Notice though that we can understand conformant planning as one of the possible manifestations of such an approach, and *it* is Σ_3^P-hard. It appears that Rintanen's informal argument depends on an implicit assumption that executability of actions in a state is polynomial-time computable.

Rintanen's approach resembles that taken by [Littman *et al.*, 1998] in the setting of probabilistic planning (also with the implicit assumption that actions are always executable). Littman *et al.* consider several plan encodings of polynomially-bounded size and obtain complexity results intuitively related to the Σ_2^P result for propositional conditional planning, but instead of $\mathbf{NP^{NP}}$ the complexity becomes $\mathbf{NP^{PP}}$, which is still within **PSPACE**, but contains all of the polynomial hierarchy!

At any rate, despite their practical importance, it appears that such results are necessarily fragmentary—that is, each depends on a particular choice of (polynomial-size) plan representation. As mentioned previously, we can view a conditional plan as a function that maps the observable portion of a partial execution history to the set of next actions to take. Presumably, not just any polynomial-size representation of such a plan function will do. In particular, it seems reasonable to choose a representation guaranteed to allow polynomial-time computation of the plan function.

Here's a sketch of another family of such results. Define (as a ptime-computable function of the set $F^O \cup A$ of atoms and the plan length k) a set P of new "plan" atoms, and define a polynomial-time construction that takes P

and $tr(F, A, F')$ to a classical propositional formula

$$plan(P, F^O, A, P')$$

in which each atom X of $A \cup P'$ appears only as the left-hand side of an equivalence $X \equiv F_X(P, F^O)$. So each atom of $A \cup P'$ is explicitly defined in terms of atoms from $P \cup F^O$. (Consequently, the value of each atom in $A \cup P'$ can be computed in polynomial-time given the values of the atoms in $P \cup F^O$.)

In this formulation, the conditional plan will be determined by the choice of values for the plan atoms at time 0.

Let us consider how the associated planning problem can be expressed. For one-step planning, we can write:

$$\begin{aligned}\exists p_0\ \forall p_1\ \forall a_0\ \forall f_0\ f_1\ \exists f_1' \\ (init(f_0) \wedge plan(p_0, f_0^O, a_0, p_1) \supset tr(f_0, a_0, f_1') \\ \wedge\ (tr(f_0, a_0, f_1) \supset goal(f_1)))\,.\end{aligned}$$

Let $step(P, F, A, P', F')$ stand for $plan(P, F^O, A, P') \wedge tr(F, A, F')$. Then for two-step planning, we can write:

$$\begin{aligned}&\exists p_0\ \forall p_1\ p_2\ \forall a_0\ a_1\ \forall f_0\ f_1\ f_2\ \exists f_1'\ f_2' \\ &\quad((init(f_0) \wedge plan(p_0, f_0^O, a_0, p_1) \supset tr(f_0, a_0, f_1')) \\ &\quad\ \wedge\ (init(f_0) \wedge step(p_0, f_0, a_0, p_1, f_1) \wedge plan(p_1, f_1^O, a_1, p_2) \supset tr(f_1, a_1, f_2')) \\ &\quad\ \wedge\ (init(f_0) \wedge step(p_0, f_0, a_0, p_1, f_1) \wedge step(p_1, f_1, a_1, p_2, f_2) \supset goal(f_2)))\,.\end{aligned}$$

The general formulation is:

$$\begin{aligned}&\exists p_0\ \forall p_1 \cdots p_k\ \forall a_0 \cdots a_{k-1}\ \forall f_0 \cdots f_k\ \exists f_1' \cdots f_k' \\ &\qquad\Big(\textstyle\bigwedge_{t=0}^{k-1}(init(f_0) \wedge \bigwedge_{u=0}^{t-1} step(p_u, f_u, a_u, p_{u+1}, f_{u+1}) \\ &\qquad\qquad\wedge\ plan(p_t, f_t^O, a_t, p_{t+1}) \supset tr(f_t, a_t, f_{t+1}')) \\ &\qquad\ \wedge\ \ (init(f_0) \wedge \textstyle\bigwedge_{t=0}^{k-1} step(p_t, f_t, a_t, p_{t+1}, f_{t+1}) \supset goal(f_k))\Big)\,.\end{aligned}$$

It is clear that this problem belongs to Σ_3^P, for any choice of polynomial-time construction for plan formula $plan(P, F^O, A, P')$. It is also clear that we can devise a polynomial-time construction for $plan(P, F^O, A, P')$ that captures conformant planning for plans of length 1. Hence, for some such families of polynomial-size conditional plans, the plan existence problem is Σ_3^P-hard.

As with conformant planning, the plan existence problem will drop to Σ_2^P if the executability of actions is polynomial-time computable.

9 Discussion and Related Work

As mentioned in the introduction, many papers with results on planning complexity have been published in the last decade. No exhaustive survey is attempted here. Instead, a range of representative results are mentioned, and contrasted with the results presented in this paper.

Bylander [1994] showed that classical planning is **PSPACE**-complete when propositional STRIPS is used to represent action domains. The key observation is that in some cases the shortest valid plan may be exponentially long, because there are exponentially many states. Bylander considered the effect of a variety of syntactic restrictions on the form of STRIPS operators, and identified classes of problems complete for **P**, **NP**, and **PSPACE**. He did not directly consider the possibility of allowing only plans of polynomially-bounded length, but his **NP** results are obtained by imposing syntactic restrictions that indirectly guarantee such a bound.

Bäckström and Nebel [1995] explored similar questions, but their setting allows for incomplete information about the initial state, and so may be understood to overlap with conformant planning. The correspondence is weak though. Their actions are nonconcurrent and deterministic, and executability of actions is easy to determine. They did not (directly) consider a restriction to polynomial-length plans. Most significantly though, they represented incomplete knowledge of the initial state by allowing atoms to take a distinguished value u (unknown), rather than considering all ("complete") initial states consistent with what is known. It is interesting to note that, under these restrictions, classical and conformant planning enjoy the same complexity.

Erol, Nau and Subrahmanian [1995] also investigated STRIPS-based planning complexity. Much of their interest was in establishing when classical planning is decidable when the action domain is described in *nonpropositional* STRIPS. They also presented results for propositional STRIPS planning, some involving restrictions on syntactic form of operators.

As mentioned previously, Littman [1997] investigated the complexity of probabilistic planning, showing that polynomial-length conditional planning in the fully-observable Markov decision process (MDP) setting is **PSPACE**-complete when a concise ("factored") representation of the action domain is used. The probabilistic nature of the problem plays no role in the hardness result—only nondeterminism is needed—and in the MDP setting actions are nonconcurrent and always executable, so Littman's **PSPACE**-hardness result implies the hardness result of Theorem 10(iv). On the other hand, since MDP actions are always executable, Littman's results do not imply the **PSPACE** membership of conditional planning as formulated in the current paper. (There is another potential source of complication here—although it appears easy to devise a polynomial-time mapping from the various concise MDP representations considered in [Littman, 1997] into the action representation used in the current paper, it is not so clear what a polynomial-time mapping in the other direction would be.)

Littman, Goldsmith, and Mundhenk [1998] reported that polynomial-length conditional planning in the partially-observable Markov decision process (POMDP) setting is **EXPSPACE**-complete when a concise representation of the action domain is used. The large gap between polynomial-length conditional planning for MDP's and POMDP's is striking. By comparison, in the setting explored in the current paper, polynomial-length conditional planning remains in **PSPACE** when we move from full to partial observability (even with concurrent actions

and **NP**-hardness of determining the executability of actions in a state). As mentioned in the previous section, Littman *et al.* showed that polynomial-length conditional planning for both MDP's and POMDP's drops to $\mathbf{NP^{PP}}$-complete when any of a variety of polynomial-size plan representations are employed. Here again we seem to pay a price for probabilistic action effects, since the corresponding result in the current paper is only $\mathbf{\Sigma_2^P}$ (or $\mathbf{\Sigma_3^P}$ if we don't assume that the deciding the executability of actions in a state belongs to **P**).

De Giacomo and Vardi [1999] also investigated conformant and conditional planning (in fully and partially observable domains). They considered deterministic actions domains with nonconcurrent actions that are always executable, but their results don't overlap with those of the current paper because they consider temporally extended goals with infinite time horizons, rather than plans of polynomially-bounded length.

Haslum and Jonsson [1999] studied conformant planning with nonconcurrent, nondeterministic actions that are always executable. Their primary interest was in plans of unbounded length, but they did show that polynomial-length conformant planning belongs to **PSPACE**. They left open the question of hardness. In their setting it appears the problem is in fact Σ_2^P-complete.

Continue with [Rintanen, 1999, Baral *et al.*, 2000, Eiter *et al.*, 2000]...

Acknowledgements

Thanks to Vladimir Lifschitz and Jussi Rintanen for helpful comments. This work partially supported by NSF Career Grant #0091773.

References

[Backstrum and Nebel, 1995] Christer Backstrum and Bernhard Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Baral *et al.*, 2000] Chitta Baral, Vladik Kreinovich, and Raul Trejo. Computational complexity of planning and approximate planning in presence of incompleteness. *Artificial Intelligence*, 2000. To appear.

[Bylander, 1994] Tom Bylander. The computational complexity of propostional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[De Giacomo and Vardi, 1999] Giuseppe De Giacomo and Moshe Vardi. Automata-theoretic approach to planning for temporally extended goals. In *Proc. of 5th European Conf. on Planning*, 1999.

[Eiter *et al.*, 2000] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. Planning under incomplete information. In *Proc. Computational Logic 2000*, 2000.

[Erol *et al.*, 1995] Kutluhan Erol, Dana S. Nau, and V.S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76(1–1):75–88, 1995.

[Fikes and Nilsson, 1971] Richard Fikes and Nils Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630, 1998.

[Giunchiglia, 2000] Enrico Giunchiglia. Planning as satisfiability with expressive action languages: Concurrency, constraints and nondeterminism. In *Principles of Knowledge Representation and Reasoning: Proc. of the Seventh Int'l Conf.*, pages 657–666, 2000.

[Haslum and Jonsson, 1999] Patrik Haslum and Peter Jonsson. Some results on the complexity of planning with incomplete information. In *Proc. of 5th European Conf. on Planning*, 1999.

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In J. Lloyd, editor, *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 92)*, pages 359–379, Vienna, Austria, 1992.

[Littman *et al.*, 1998] Michael Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[Littman, 1997] Michael Littman. Probabilistic propositional planning: representations and complexity. In *Proc. of AAAI-97*, pages 748–754, 1997.

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. of AAAI-97*, pages 460–465, 1997.

[Papadimitriou, 1994] Christos Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[Rintanen, 1999] Jussi Rintanen. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

Appendix: Additional Proofs and Proof Sketches

A number of proofs were included in the main body of the paper. The remaining proofs (or proof sketches) are presented here.

Proposition 3 *Determining that given actions are executable in a given state is* **NP***-complete.*

Proof. To see that the problem belongs to **NP**, take any action domain, and consider any interpretation I_F of the set F of fluent symbols that satisfies $state(F)$, along with any interpretation I_A of the set A of action symbols. For convenience, identify each of this interpretations with the set of literals true in it. The question then is whether

$$I_F \cup I_A \models \exists f'(act(F, A, f') \wedge state(f')) \,. \tag{35}$$

We can substitute (in polynomial time) each occurrence of any symbol P from $F \cup A$ in the rhs with $\top$ if $I_F \cup I_A \models P$ and $\bot$ otherwise. The resulting formula has the form $\exists f' \Phi(f')$, and the question is reduced to logical validity of this formula.

To show hardness, it is sufficient to obtain from (10) in polynomial time a set of fluent symbols F, a set of action symbols A, formulas $state(F)$ and $act(F, A, F')$, an interpretation I_F of the set F of fluent symbols that satisfies $state(F)$, and an interpretation I_A of the set A of action symbols such that (35) holds iff (10) is logically valid. Take $F = \{X(1), \ldots, X(n)\}$ and $A = \emptyset$. Let $state(F) = \top$ and $act(F, A, F') = \Phi(F')$. Let I_F be any interpretation of F, and let I_A be the only interpretation of A. By the choice of $state(F)$ and $act(F, A, F')$, we have (35) iff

$$I_F \cup I_A \models \exists f' \Phi(f') \,.$$

The latter can be reduced to

$$\models \exists f' \Phi(f')$$

which is clearly what we're after. □

Proposition 4 *Determining nonconcurrency is* **coNP***-complete.*

Proof. Clearly the problem belongs to **coNP**. For hardness, we need to obtain from (5) in polynomial-time an action domain that is nonconcurrent iff (5) is logically valid. Take $F = \{X(1), \ldots, X(n)\}$ and $A = \{B, C\}$. Take $state(F) = \neg\Phi(F)$ and $act(F, A, F') = \top$. This domain is nonconcurrent iff

$$\neg\Phi(F) \wedge \top \wedge \neg\Phi(F') \supset \neg B \vee \neg C$$

is logically valid, which is the case iff $\neg\Phi(F) \supset \bot$ is logically valid, which is the case iff (5) is. □

Theorem 4 *Conformant planning is Σ_2^P-complete, for plans of polynomially-bounded length, if the executability of actions is polynomial-time computable.*

Proof. Hardness follows from Theorem 3. To show that conformant planning in fact falls to Σ_2^P if the executabililty of actions is polynomial-time computable, it is convenient to utilize the standard characterization of the polynomial heirarchy in terms of oracle machines. (See, for instance, [Papadimitriou, 1994].) So we need to show that the problem can be solved in nondeterministic polynomial time given an **NP** oracle. Clearly there is no problem determining that there is at least one possible initial state. For the rest of the problem, for all $t = 0, \ldots, k-1$, guess an interpretation I_{A_t} of A_t, and then use the oracle to determine that the conformant plan corresponding to $I_A = \bigcup_{t=0}^{k-1} I_{A_t}$ is both (i) sufficient and (ii) executable.

We will show that sufficiency is a **coNP** problem, and so can be handled by the **NP** oracle. So we need to show that determining that the given plan, represented by I_A, is *not* sufficient is an **NP** problem. For all $t = 0, \ldots, k$, guess an interpretation I_{F_t} of F_t, and let $I_F = \bigcup_{t=0}^{k} I_{F_t}$. Only polynomial time is needed to check that

$$I_F \cup I_A \models \mathit{init}(F_0) \wedge \bigwedge_{t=0}^{k-1} \mathit{tr}(F_t, A_t, F_{t+1}) \wedge \neg \mathit{goal}(F_k)\,.$$

It remains only to show that under the assumption that executability of actions is polynomial-time computable, executability of the given plan is also a **coNP** problem. That is, we need to show that determining that the given plan is not executable is a problem in **NP**. For all $t = 0, \ldots, k$, guess an interpretation I_{F_t} of F_t, and guess a time n $(0 \leq n < k)$. Only polynomial time is need to check that

$$I_{F_0} \models \mathit{init}(F_0)$$

and for all t $(0 \leq t < n)$

$$I_{F_t} \cup I_{A_t} \cup I_{F_{t+1}} \models \mathit{tr}(F_t, A_t, F_{t+1})\,.$$

And since the executability of actions is polynomial-time computable, we can also check that

$$I_{F_n} \cup I_{A_n} \not\models \exists f_{n+1}\ \mathit{tr}(F_n, A_n, f_{n+1})$$

in polynomial time. □

Theorem 5 *Conditional planning for plans of length k is Π_{2k+1}^P-hard, even if actions are always executable.*

Proof. The result is easy for $k = 0$. We have already proved it for $k = 1$. We need a more complex construction for $k > 1$.

We need to obtain from

$$\overbrace{\forall x_1^0 \cdots x_{m_0}^0 \exists y_1^1 \cdots y_{n_1}^1} \ \cdots\ \overbrace{\forall x_1^{k-1} \cdots x_{m_{k-1}}^{k-1} \exists y_1^k \cdots y_{n_k}^k}\ \forall x_1^k \cdots x_{m_k}^k$$
$$\Phi(x_1^0, \ldots, x_{m_0}^0, y_1^1, \ldots, y_{n_1}^1, \ \cdots\ , x_1^{k-1}, \ldots, x_{m_{k-1}}^{k-1}, y_1^k, \ldots, y_{n_k}^k, x_1^k, \ldots, x_{m_k}^k) \tag{36}$$

(for each choice of k) in polynomial-time a set F of fluent symbols, a set A of action symbols, and formulas $tr(F, A, F')$, $init(F)$ and $goal(F)$ for which the conjunction of (12) and (26) is logically equivalent to (36). Moreover, we require that actions be always executable.

We simplify the presentation, as in the proof of the $k = 1$ case of Theorem 7 in Section 6, by showing only one fluent corresponding to each "family of quantified variables" in (36). That is, for each i $(0 \leq i \leq k)$, the family of variables x_j^i will correspond here to a single fluent $X(i)$, and, for each i $(0 < i \leq k)$, the family of variables y_j^i will correspond to a single fluent $Y(i)$. Since all variables in each family are to be handled symmetrically, this convention just simplifies the presentation.

Let $F = \{T(0), \ldots, T(k), X(0), \ldots, X(k), Y(1), \ldots, Y(k)\}$. Fluents $T(i)$ will be used, roughly speaking, to identify the current "time" (number of steps already taken) during plan execution. Let $A = \{A(1), \ldots, A(k)\}$. Intuitively, action $A(i)$ corresponds to fluent $Y(i)$. Take

$$state(F) = \bigvee_{i=0}^{k} T(i) \wedge \bigwedge_{0 \leq i < j \leq k} \neg(T(i) \wedge T(j)) \,.$$

So, intuitively, each state has a unique associated "time." Let $act(F, A, F')$ be the conjunction of the following formulas.

$$\bigwedge_{i=0}^{k-1} (T(i) \supset T(i+1)')$$

$$\bigwedge_{i=0}^{k} \left(T(i) \supset \bigwedge_{j=0}^{i} (X(j)' \equiv X(j)) \right)$$

$$\bigwedge_{i=0}^{k-1} (Y(i+1)' \equiv Y(i+1) \vee (A(i+1) \wedge T(i)))$$

Notice that actions are indeed always executable.

Intuitively, the first conjunct of $act(F, A, F')$ advances the "time" by one at each step of plan execution (the time will be 0 in the initial state), and the second conjunct "fixes" the value of each fluent $X(i)$ starting at time i. The third conjunct of $act(F, A, F')$ allows the value of $Y(i)$ to change from false to true iff action $A(i)$ is performed at time $i - 1$. (All fluents $Y(i)$ will be false in the initial state.)

Take

$$init(F) = T(0) \wedge state(F) \wedge \bigwedge_{i=1}^{k} \neg Y(i).$$

So the "time" indeed starts at 0. (The conjunct $state(F)$ is included for a technical reason; we previously assumed, for convenience, that $init(F) \models state(F)$

in planning problems.) Each fluent $Y(i)$ starts out false (and the values of fluents $X(i)$ are arbitrary).

Let

$$goal(F) = \Phi(\overbrace{X(0), Y(1)}, \ldots, \overbrace{X(k-1), Y(k)}, X(k)) .$$

Here's a rough description of how the "effects" of the alternating quantifiers in (36) are captured in this construction. At time 0 (during plan execution), the value of $X(0)$ cannot change during subsequent steps, so the decision whether to make $Y(1)$ true (for all subsequent times) by performing action $A(1)$ can (only reasonably) depend on the value of $X(0)$. At time 1, the values of $X(0)$ and $Y(1)$ are already fixed. Moreover, the value of $X(1)$ will not change subsequently. So the decision whether to make $Y(2)$ true (for all subsequent times) by performing action $A(2)$ can (only reasonably) depend on the values of $X(0), Y(1), X(1)$. And so forth. Thus, at each time i ($i < k$), the decision on fluent $Y(i)$ is based on the values of $X(j)$ and $Y(j)$ ($j < i$). After the decision at time $k-1$, the values of all fluents except $X(k)$ are decided, and $X(k)$ still varies freely.

Now let's prove that this construction works as described.

Observe that (26) is equivalent to

$$\forall f_0 (init(f_0) \supset goal_0(f_0)) , \tag{37}$$

where

$$goal_k(F) = goal(F) \tag{38}$$

and for all j ($0 \leq j < k$)

$$goal_j(F) = \exists a_j \forall f_{j+1} (tr(F, a_j, f_{j+1}) \supset goal_j(f_{j+1})) . \tag{39}$$

Let

$$init_i(F) = T(i) \wedge state(F) \wedge \bigwedge_{j=i+1}^{k} \neg Y(j).$$

Notice that $init_0(F) = init(F)$.

Here's what we will show (for any $k > 1$). For all i ($0 \leq i \leq k$), the assumption $init_i(F_i)$ entails the equivalence of formulas $goal_i(F_i)$ and

$$\begin{aligned} &\overbrace{\exists a(i+1)_i \forall x(i+1)_{i+1}} \cdots \overbrace{\exists a(k)_{k-1} \forall x(k)_k} \\ &\qquad \Phi(\overbrace{X(0)_i, Y(1)_i}, \ldots, \overbrace{X(i-1)_i, Y(i)_i}, X(i)_i, \\ &\qquad\quad \overbrace{a(i+1)_i, x(i+1)_{i+1}}, \ldots, \overbrace{a(k)_{k-1}, x(k)_k}) . \end{aligned} \tag{40}$$

Proof will be by induction on i.

In particular, we are interested in proving this for $i = 0$, since if the assumption $init_0(F_0)$ entails the equivalence of $goal_0(F_0)$ and

$$\overbrace{\exists a(1)_0 \forall x(1)_1} \cdots \overbrace{\exists a(k)_{k-1} \forall x(k)_k} \Phi(X(0)_0, \overbrace{a(1)_0, x(1)_1}, \ldots \overbrace{a(k)_{k-1}, x(k)_k}) ,$$

then (37) is logically equivalent to

$$\forall f_0(\mathit{init}(f_0) \supset \overbrace{\exists a(1)_0 \forall x(1)_1}\cdots\overbrace{\exists a(k)_{k-1}\forall x(k)_k}\,\Phi(x(0)_0,\overbrace{a(1)_0,x(1)_1},\ldots\overbrace{a(k)_{k-1},x(k)_k}))\,,$$

which is in turn equivalent to

$$\forall x(1)_0\overbrace{\exists a(1)_0 \forall x(1)_1}\cdots\overbrace{\exists a(k)_{k-1}\forall x(k)_k}\,\Phi(x(0)_0,\overbrace{a(1)_0,x(1)_1},\ldots\overbrace{a(k)_{k-1},x(k)_k}))\,,$$

which is clearly what we're after.

Base case: $i = k$. By (38), $\mathit{goal}_k(F_k) = \mathit{goal}(F_k)$, which is exactly (40).

Ind. step: By IH, $\mathit{init}_{i+1}(F_{i+1})$ $(0 < i < k)$ entails the equivalence of $\mathit{goal}_{i+1}(F_{i+1})$ and

$$\begin{aligned}&\overbrace{\exists a(i+2)_{i+1}\forall x(i+2)_{i+2}}\cdots\overbrace{\exists a(k)_{k-1}\forall x(k)_k}\\ &\qquad\Phi(\overbrace{X(0)_{i+1},Y(1)_{i+1}},\ldots,\overbrace{X(i)_{i+1},Y(i+1)_{i+1}},X(i+1)_{i+1},\\ &\qquad\overbrace{a(i+2)_{i+1},x(i+2)_{i+2}},\ldots,\overbrace{a(k)_{k-1},x(k)_k})\,.\end{aligned}\tag{41}$$

We need to show that $\mathit{init}_i(F_i)$ entails the equivalence of $\mathit{goal}_i(F_i)$ and (40).

Assume $\mathit{init}_i(F_i)$. By (39), $\mathit{goal}_i(F_i)$ is

$$\exists a_i \forall f_{i+1}(\mathit{tr}(F_i,a_i,f_{i+1}) \supset \mathit{goal}_{i+1}(f_{i+1}))\,.\tag{42}$$

Consider the formula

$$\mathit{tr}(F_i,A_i,F_{i+1}) \supset \mathit{goal}_{i+1}(F_{i+1})\,.\tag{43}$$

One easily verifies that $\mathit{init}_i(F_i)$ and $\mathit{tr}(F,A_i,F_{i+1})$ together entail $\mathit{init}_{i+1}(F_{i+1})$. Consequently, by the IH, $\mathit{init}_i(F_i)$ and $\mathit{tr}(F,A_i,F_{i+1})$ together also entail that $\mathit{goal}_{i+1}(F_{i+1})$ is equivalent to (41). We conclude (under assumption $\mathit{init}_i(F_i)$ only) that (43) is equivalent to

$$\begin{aligned}&\mathit{tr}(F_i,A_i,F_{i+1}) \supset\\ &\qquad\overbrace{\exists a(i+2)_{i+1}\forall x(i+2)_{i+2}}\cdots\overbrace{\exists a(k)_{k-1}\forall x(k)_k}\\ &\qquad\qquad\Phi(\overbrace{X(0)_{i+1},Y(1)_{i+1}},\ldots,\overbrace{X(i-1)_{i+1},Y(i)_{i+1}},\\ &\qquad\qquad\quad X(i)_{i+1},Y(i+1)_{i+1},X(i+1)_{i+1},\\ &\qquad\qquad\quad\overbrace{a(i+2)_{i+1},x(i+2)_{i+2}},\ldots,\overbrace{a(k)_{k-1},x(k)_k})\,,\end{aligned}$$

which is in turn equivalent to

$$\begin{aligned}&\mathit{tr}(F_i,A_i,F_{i+1}) \supset\\ &\qquad\overbrace{\exists a(i+2)_{i+1}\forall x(i+2)_{i+2}}\cdots\overbrace{\exists a(k)_{k-1}\forall x(k)_k}\\ &\qquad\qquad\Phi(\overbrace{X(0)_i,Y(1)_i},\ldots,\overbrace{X(i-1)_i,Y(i)_i},\\ &\qquad\qquad\quad X(i)_i,A(i+1)_i,X(i+1)_{i+1},\\ &\qquad\qquad\quad\overbrace{a(i+2)_{i+1},x(i+2)_{i+2}},\ldots,\overbrace{a(k)_{k-1},x(k)_k})\,,\end{aligned}\tag{44}$$

since $\mathit{init}_i(F_i) \wedge \mathit{tr}(F_i,A_i,F_{i+1})$ entails

- $X(j)_i \equiv X(j)_{i+1}$ for $0 \leq j \leq i$, and
- $Y(j)_i \equiv Y(j)_{i+1}$ for $0 < j \leq i$, and
- $Y(i+1)_{i+1} \equiv A(i+1)_i$.

Since (43) is equivalent to (44), (42) is equivalent to

$$\begin{aligned}\exists a_i \forall f_{i+1}(tr(F_i, a_i, f_{i+1}) \supset \\ \overbrace{\exists a(i+2)_{i+1} \forall x(i+2)_{i+2}} \cdots \overbrace{\exists a(k)_{k-1} \forall x(k)_k} \\ \Phi(\overbrace{X(0)_i, Y(1)_i}, \ldots, \overbrace{X(i-1)_i, Y(i)_i}, \\ X(i)_i, a(i+1)_i, x(i+1)_{i+1}, \\ \overbrace{a(i+2)_{i+1}, x(i+2)_{i+2}}, \ldots, \overbrace{a(k)_{k-1}, x(k)_k})) ,\end{aligned}$$

which is in turn equivalent to (40). □

For subsequent theorems whose proof requires similarly complex constructions, we specify the construction but do not provide a complete proof.

Theorem 7 *Conditional planning with partial observability for plans of length k ($k > 0$) is Π^P_{2k+2}-complete.*

Proof Idea. We have already observed that the problem falls within the specified complexity class, and we proved hardness for $k = 1$. We need a more complex construction for $k > 1$.

As in previous proofs, we simplify the presentation by showing only one fluent corresponding to each "family of quantified variables."

We need to obtain from

$$\begin{aligned}&\overbrace{\forall x^1_1 \cdots x^1_{l_1} \exists y^1_1 \cdots y^1_{m_1}} \ \cdots \ \overbrace{\forall x^k_1 \cdots x^k_{l_k} \exists y^k_1 \cdots y^k_{m_k}} \forall z_1 \cdots z_{n_1} \exists w_1 \cdots w_{n_2} \\ &\Phi(x^1_1, \ldots, x^1_{l_1}, y^1_1, \ldots, y^1_{m_1}, \ \ldots \ , x^k_1, \ldots, x^k_{l_k}, y^k_1, \ldots, y^k_{m_k}, z_1, \ldots, z_{n_1}, w_1, \ldots, w_{n_2})\end{aligned} \tag{45}$$

(for each choice of k) in polynomial-time disjoint sets F^O and F^N of fluent symbols, a set A of action symbols, and formulas $tr(F, A, F')$, $init(F)$ and $goal(F)$ for which the conjunction of (32) and (12) is logically equivalent to (45).

Take $F^0 = \{X(1), \ldots, X(k)\}$,

$$F^N = \{T(0), T(1), \ldots, T(k), Y(1), \ldots, Y(k), Z, W\}$$

and $A = \{A(1), \ldots, A(k)\}$. Let $state(F)$ be a formula guaranteeing that exactly one of the fluents $T(i)$ is true in a state. Let $act(F, A, F')$ be the conjunction of the following.

$$\bigwedge_{i=1}^{k} (T(i-1) \supset T(i)')$$

$$\bigwedge_{i=1}^{k} (A(i) \supset T(i-1))$$

$$\bigwedge_{i=1}^{k}\left(T(i-1)\supset\bigwedge_{j=1}^{i}(X(j)'\equiv X(j))\right)$$

$$\bigwedge_{i=1}^{k}(Y(i)'\equiv Y(i)\vee A(i))$$

$$T(k)'\supset\Phi(\overbrace{X(1)',Y(1)'},\ldots,\overbrace{X(k)',Y(k)'},Z,W')$$

Take $\mathit{init}(F)=T(0)\wedge\mathit{state}(F)\wedge\bigwedge_{i=1}^{k}\neg Y(i)$ and $\mathit{goal}(F)=\top$.

Our choice of $\mathit{init}(F)$ makes (12) true, so it remains to show that (32) is logically equivalent to (45).

To see how this works, consider $k=2$. In this case, given our choice of $\mathit{goal}(F)$, (31) is equivalent to

$$\begin{aligned}\forall f_0^O\ \exists a_0\ \forall f_1^O\ \exists a_1\ \forall f_0^N\ \forall f_1^N\ \exists f_2'\\ ((\mathit{init}(f_0)\supset\exists f_1'\ \mathit{tr}(f_0,a_0,f_1'))\\ \wedge\ (\mathit{init}(f_0)\wedge\mathit{tr}(f_0,a_0,f_1)\supset\mathit{tr}(f_1,a_1,f_2')))\end{aligned}\tag{46}$$

We can begin by noticing that

$$\mathit{init}(F_0)\models\exists f_1'\ \mathit{tr}(F_0,A_0,f_1')\equiv\neg A(2)_0\,.$$

Moreover, by choice of $\mathit{init}(F_0)$,

$$\forall f_0^O\ \exists a_0\ \forall f_0^N\ (\mathit{init}(f_0)\supset\neg a(2)_0)\equiv\exists a_0\ \neg a(2)_0\,.$$

Consequently, we can simplify (46) as

$$\begin{aligned}\forall f_0^O\ \exists a_0\ \forall f_1^O\ \exists a_1\ \forall f_0^N\ \forall f_1^N\ \exists f_2'\\ (\neg a(2)_0\wedge(\mathit{init}(f_0)\wedge\mathit{tr}(f_0,a_0,f_1)\supset\mathit{tr}(f_1,a_1,f_2')))\end{aligned}\tag{47}$$

Next, notice that

$$\mathit{init}(F_0)\models\mathit{tr}(F_0,A_0,F_1)\equiv\mathit{ante}(F_0,A_0,F_1)$$

where $\mathit{ante}(F_0,A_0,F_1)$ is the conjunction of

$$T(1)_1\wedge\neg T(0)_1\wedge\neg T(2)_1$$

$$\neg A(2)_0$$

$$X(1)_1\equiv X(1)_0$$

$$Y(1)_1\equiv A(1)_0$$

$$\neg Y(2)_1$$

Consequently, we can simplify (47) as

$$\begin{aligned}\forall f_0^O\ \exists a_0\ \forall f_1^O\ \exists a_1\ \forall f_0^N\ \forall f_1^N\ \exists f_2'\\ (\neg a(2)_0\wedge(\mathit{init}(f_0)\wedge\mathit{ante}(f_0,a_0,f_1)\supset\mathit{tr}(f_1,a_1,f_2')))\end{aligned}\tag{48}$$

Next, notice that $init(F_0) \wedge ante(F_0, A_0, F_1)$ entails

$$tr(F_1, A_1, F_2) \equiv cons(F_1, A_1, F_2) \wedge \Phi(X(1)_0, A(1)_0, X(2)_1, A(2)_1, Z_1, W_2)$$

where $cons(F_1, A_1, F_2)$ is the conjunction of

$$T(2)_2 \wedge \neg T(0)_2 \wedge \neg T(1)_2$$

$$\neg A(1)_1$$

$$X(1)_2 \equiv X(1)_1$$

$$X(2)_2 \equiv X(2)_1$$

$$Y(1)_2 \equiv Y(1)_1$$

$$Y(2)_2 \equiv A(2)_1$$

Then observe that

$$\exists f_2 \ (cons(F_1, A_1, f_2) \wedge \Phi(X(1)_0, A(1)_0, X(2)_1, A(2)_1, Z_1, w_2))$$

is logically equivalent to

$$\exists w_2 \ (\neg A(1)_1 \wedge \Phi(X(1)_0, A(1)_0, X(2)_1, A(2)_1, Z_1, w_2)) \, .$$

We can conclude that (48) is equivalent to

$$\begin{array}{l} \forall f_0^O \ \exists a_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \ \forall f_1^N \ \exists w_2 \ (\neg a(2)_0 \\ \quad \wedge \ (init(f_0) \wedge ante(f_0, a_0, f_1) \supset \\ \qquad (\neg a(1)_1 \wedge \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2)))) \, . \end{array}$$

By moving quantifiers inside, we can obtain

$$\begin{array}{l} \forall f_0^O \ \exists a_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \ (\neg a(2)_0 \\ \quad \wedge \ (init(f_0) \wedge \exists t(0)_1 t(1)_1 t(2)_1 y(1)_1 y(2)_1 ante(f_0, a_0, f_1) \supset \\ \qquad (\neg a(1)_1 \wedge \forall z_1 \ \exists w_2 \ \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2)))) \, . \end{array} \tag{49}$$

Notice that

$$\exists t(0)_1 t(1)_1 t(2)_1 y(1)_1 y(2)_1 ante(F_0, A_0, f_1) \equiv \neg A(2)_0 \wedge (X(1)_1 \equiv X(1)_0) \, ,$$

so (49) can be simplified as

$$\begin{array}{l} \forall f_0^O \ \exists a_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \ (\neg a(2)_0 \\ \quad \wedge \ (init(f_0) \wedge \neg a(2)_0 \wedge (x(1)_1 \equiv x(1)_0) \supset \\ \qquad (\neg a(1)_1 \wedge \forall z_1 \ \exists w_2 \ \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2)))) \, . \end{array}$$

Variable $a(2)_0$ can be eliminated as follows.

$$\begin{array}{l} \forall f_0^O \ \exists a(1)_0 \ \forall f_1^O \ \exists a_1 \ \forall f_0^N \\ \quad (init(f_0) \wedge (x(1)_1 \equiv x(1)_0) \supset \\ \qquad (\neg a(1)_1 \wedge \forall z_1 \ \exists w_2 \ \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2))) \end{array}$$

Again by moving quantifiers inside (and dropping clearly irrelevant ones), we can obtain

$$\begin{array}{l}\forall f_0^O\ \exists a(1)_0\ (\exists f_0^N\ \mathit{init}(f_0) \wedge \exists x(1)_1\ (x(1)_1 \equiv x(1)_0) \supset \\ \quad (\exists a(1)_1\ \neg a(1)_1 \\ \quad \wedge \forall x(2)_1\ \exists a(2)_1\ \forall z_1\ \exists w_2\ \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2)))) \,.\end{array} \tag{50}$$

Since each of

$$\forall f_0^O \exists f_0^N \mathit{init}(f_0)\,,$$
$$\forall f_0^O \exists x(1)_1\ (x(1)_1 \equiv x(1)_0)\,,$$
$$\exists a(1)_1\ \neg a(1)_1\,,$$

is logically valid, (50) is equivalent to

$$\forall f_0^O\ \exists a(1)_0\ \forall x(2)_1\ \exists a(2)_1\ \forall z_1\ \exists w_2\ \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2)))\,,$$

which is equivalent to

$$\forall x(1)_0\ \exists a(1)_0\ \forall x(2)_1\ \exists a(2)_1\ \forall z_1\ \exists w_2\ \Phi(x(1)_0, a(1)_0, x(2)_1, a(2)_1, z_1, w_2)\,,$$

which is what we're after. □

Theorem 8 *For deterministic planning domains with actions always executable, conditional planning with partial observability for plans of length k is Π^P_{2k+1}-hard.*

The proof construction used here is very similar to the one used for the related **PSPACE** result (Theorem 4) in [Baral *et al.*, 2000].

Proof Idea. We have already observed that the problem falls within the specified complexity class. It remains to show hardness.

We need to obtain from

$$\begin{array}{l}\overbrace{\forall x_1^0 \cdots x_{m_0}^0 \exists y_1^1 \cdots y_{n_1}^1} \ \cdots\ \overbrace{\forall x_1^{k-1} \cdots x_{m_{k-1}}^{k-1} \exists y_1^k \cdots y_{n_k}^k}\ \forall x_1^k \cdots x_{m_k}^k \\ \quad \Phi(x_1^0, \ldots, x_{m_0}^0, y_1^1, \ldots, y_{n_1}^1, \ \cdots\ , x_1^{k-1}, \ldots, x_{m_{k-1}}^{k-1}, y_1^k, \ldots, y_{n_k}^k, x_1^k, \ldots, x_{m_k}^k)\end{array} \tag{51}$$

(for each choice of k) in polynomial-time disjoint sets F^O and F^N of fluent symbols, a set A of action symbols, and formulas $tr(F, A, F')$, $\mathit{init}(F)$ and $\mathit{goal}(F)$ for which the conjunction of (12) and (34) is logically equivalent to (51). Moreover, the action domain should be deterministic, with actions always executable.

Take $F^0 = \{S\}$,

$$F^N = \{T(0), T(1), \ldots, T(k), X(0), X(1), \ldots, X(k), Y(1), \ldots, Y(k)\}$$

and $A = \{A(1), \ldots, A(k)\}$. Again we are showing only one fluent corresponding to each "family of quantified variables." Here an additional complication is that there should be as many observable variables as there are members of the largest "family" corresponding to an $X(i)$. That is, the observable variable S should be understood to stand for a family of $\max(\{m_0, \ldots, m_k\})$ observable variables.

Let *state*(F) be a formula guaranteeing that exactly one of the fluents $T(i)$ is true in a state conjoined with

$$\bigwedge_{i=0}^{k} (T(i) \supset (S \equiv X(i)) \,.$$

This latter formula will guarantee that at each step i during plan execution, the observable fluent S has the value of fluent $X(i)$. (Or again recalling that S and $X(i)$ each stand for families of variables, each member of the $X(i)$ family will determine the value of a corresponding member of the S family.) Let *act*(F, A, F') be the conjunction of the following.

$$\bigwedge_{i=1}^{k} (T(i-1) \supset T(i)')$$

$$\bigwedge_{i=0}^{k} (X(i)' \equiv X(i))$$

$$\bigwedge_{i=1}^{k} (Y(i)' \equiv Y(i) \vee (A(i) \wedge T(i-1)))$$

Notice that actions are always executable. Notice also that the action domain is deterministic. Fluents $X(i)$ never change value. Each fluent $Y(i)$ can only be changed from false to true by performing action $A(i)$ at time $i-1$.

Take

$$\mathit{init}(F) = T(0) \wedge \mathit{state}(F) \wedge \bigwedge_{i=1}^{k} \neg Y(i)$$

and

$$\mathit{goal}(F) = \Phi(X(0), \overbrace{Y(1), X(1)}, \ldots, \overbrace{Y(k), X(k)}) \,.$$

Now let's work it through for $k = 0$ and then $k = 1$.

If $k = 0$, (34) is simply

$$\forall f_0 (\mathit{init}(f_0) \supset \mathit{goal}(f_0)) \,, \tag{52}$$

with $F = \{S, T(0), X(0)\}$, $A = \emptyset$, $\mathit{state}(F) = T(0) \wedge (T(0) \supset (S \equiv X(0)))$, $\mathit{init}(F) = T(0) \wedge \mathit{state}(F)$, and $\mathit{goal}(F) = \Phi(X(0))$. We can simplify (52) to obtain

$$\forall\, s_0\, t(0)_0\, x(0)_0\ (t(0)_0 \wedge (s_0 \equiv x(0)_0) \supset \Phi(x(0)_0))$$

which is equivalent to $\forall x(0)_0 \Phi(x(0)_0)$ which is what we're after in this case.

If $k = 1$, (34) is

$$\forall f_0^O\, \exists a_0\, \forall f_0^N\, \forall f_1\ (\mathit{init}(f_0) \wedge \mathit{tr}(f_0, a_0, f_1) \supset \mathit{goal}(f_0)) \,, \tag{53}$$

with $F = \{S, T(0), T(1), X(0), X(1), Y(1)\}$, $A = \{A(1)\}$, and

$$\mathit{state}(F) = (T(0) \not\equiv T(1)) \wedge (T(0) \supset (S \equiv X(0))) \wedge (T(1) \supset (S \equiv X(1)))\,.$$

Formula $\mathit{act}(F, A, F')$ is the conjunction of the following.

$$T(0) \supset T(1)'$$

$$(X(0)' \equiv X(0)) \wedge (X(1)' \equiv X(1))$$

$$(Y(1)' \equiv Y(1) \vee (A(1) \wedge T(0)))$$

Also $\mathit{init}(F) = T(0) \wedge \neg Y(1) \wedge \mathit{state}(F)$, and $\mathit{goal}(F) = \Phi(X(0), Y(1), X(1))$. We can simplify (53) to obtain

$$\begin{array}{l}\forall\, s_0\ \exists\, a(1)_0\ \forall\, x(0)_0\, x(1)_0\, y(1)_0\ \forall\, s_1\, x(0)_1\, x(1)_1\, y(1)_1 \\ \quad ((s_0 \equiv x(0)_0) \wedge \neg y(1)_0 \wedge (x(0)_0 \equiv x(0)_1) \wedge (x(1)_0 \equiv x(1)_1) \\ \quad \wedge (y(1)_1 \equiv y(1)_0 \vee a(1)_0) \wedge (s_1 \equiv x(1)_1) \supset \Phi(x(0)_1, y(1)_1, x(1)_1))\end{array}$$

which is equivalent to

$$\forall\, s_0\ \exists\, a(1)_0\ \forall\, s_1\ \Phi(s_0, a(1)_0, s_1)$$

which is what we're after in this case. □

Theorem 9 *Conditional planning with partial observability for plans of length k ($k > 0$) is Π^P_{2k+1}-complete, if the executability of actions is polynomial-time computable.*

Proof Idea. Hardness follows from the previous theorem.

To show that for each k the problem falls to Π^P_{2k+1}, consider oracles, as in the proof of Theorem 4. Roughly, observe that (without the executability assumption) the "innermost" **NP** oracle is needed to solve a problem related to executability. (For each of times 0 to $k-1$ there will be a state and actions already "guessed"—for each of these, guess and check a possible next state.) Under the assumption that executability of actions is polynomial-time computable, this problem falls to **P**, so the "innermost" **NP** oracle can be dropped.

For example, if $k = 1$ we need to show that our problem belongs to Π^P_3. First, it is clear that we can check that there is at least one possible initial state. For the rest, we need to show that we have the complement of a problem that can be solved in nondeterministic polynomial time given a Σ^P_2 oracle. We wish to determine that there exists a conditional one-step plan, in which the actions to be performed depend only on the observable fluents at time 0, that is both sufficient and executable. That is, for every interpretation I^O_0 of observable fluents at time 0 that can be extended to a possible initial state, there is a one-step plan that is both sufficient and executable starting in any possible initial state consistent with I^O_0. So the complement is: there is an interpretation I^O_0

of F_0^O such that (i) $I_0^O \not\models \neg init(F_0)$, and (ii) for any interpretation I_0^A of A_0 there is an interpretation I_0^N of F_0^N such that

$$I_0^O \cup I_0^N \models init(F_0)$$

and either

$$I_0^O \cup I_0^N \cup I_0^A \not\models \exists f_1 tr(F_0, A_0, f_1)$$

or

$$I_0^O \cup I_0^N \cup I_0^A \not\models \forall f_1 tr(F_0, A_0, f_1) \supset goal(f_1)\,.$$

So we guess I_0^O, and use the Σ_2^P oracle to solve this problem. Part (i) is easily done. We'll want to solve the complement of part (ii). So the Σ_2^P oracle guesses an interpretation I_0^A of A_0 and then uses an **NP** oracle to check that for any interpretation I_0^N of F_0^N such that

$$I_0^O \cup I_0^N \models init(F_0)$$

both

$$I_0^O \cup I_0^N \cup I_0^A \models \exists f_1 tr(F_0, A_0, f_1)$$

and

$$I_0^O \cup I_0^N \cup I_0^A \models \forall f_1 tr(F_0, A_0, f_1) \supset goal(f_1)\,.$$

In order to solve this with an **NP** oracle, we'll move the universal quantifier in the last equation "outside": that is, given our earlier guesses of I_0^O and I_0^A, we use an **NP** oracle to check that for any interpretation I_0^N of F_0^N and any interpretation I_1^F of F_1, if

$$I_0^O \cup I_0^N \models init(F_0)$$

then both

$$I_0^O \cup I_0^N \cup I_0^A \models \exists f_1 tr(F_0, A_0, f_1)$$

and

$$I_0^O \cup I_0^N \cup I_0^A \cup I_1^F \models tr(F_0, A_0, F_1) \supset goal(F_1)\,.$$

So guess I_0^N and I_1^F, and do the checks in polynomial time. Clearly the first and third checks are in **P**, and since we've assumed that executability of actions is polynomial-time computable, the second check is also in **P**. □

Theorem 10 *For nonconcurrent domains, the following hold.*

(i) Classical planning for plans of length $|A|$ is **NP***-hard, even when actions are always executable.*

(ii) Conformant planning for plans of length $|A|$ is Σ_3^P-hard.

(iii) Conformant planning for plans of length $|A|$ is Σ_2^P-hard for deterministic planning domains in which actions are always executable.

(iv) Conditional planning for plans of length $|A|$ is **PSPACE***-hard, even if actions are always executable.*

(v) Conditional planning for plans of length $|A|$ is Π_2^P-hard for deterministic action domains in which actions are always executable.

(vi) Conditional planning with partial observability for plans of length $|A|$ is **PSPACE***-hard for deterministic action domains in which actions are always executable.*

Proof Idea. Here we can adapt the proofs of the corresponding hardness results for action domains with concurrency. Essentially, we add a family of "step-counting" fluents that either restrict the executability of actions to particular time points, or, if actions should be always executable, restrict the effects of actions to particular time points. We consider one example of each.

For part (i): We need a nonconcurrent, deterministic action domain with actions (nonconcurrently) always executable, and a planning problem with a completely specified initial state, obtainable from (10) in polynomial time, such that (9) is true (for $k = |A|$) iff (10) is.

Take $F = \{T(0), T(1), \ldots, T(n), X(1), \ldots, X(n)\}$, and $A = \{A(1), \ldots, A(n)\}$. Let *state*$(F)$ be a formula saying that exactly one of the fluents $T(i)$ holds in a state. Take

$$\mathit{init}(F) = T(0) \wedge \mathit{state}(F) \wedge \bigwedge_{i=1}^{n} \neg X(i)\,,$$

and $\mathit{goal}(F) = \Phi(X(1), \ldots, X(n))$. Let $\mathit{act}(F, A, F')$ be the conjunction of a formula guaranteeing nonconcurrency and the following two formulas.

$$\bigwedge_{i=1}^{n} (T(i-1) \supset T(i)')$$

$$\bigwedge_{i=1}^{n} (X(i)' \equiv X(i) \vee (A(i) \wedge T(i-1)))$$

So for each time t, $T(t)$ holds and we also have $\neg T(i)$ for each $i \neq t$. All fluents $X(i)$ are initially false. Their eventual value (at last time n) is determined by whether or not action $A(i)$ occurs in the course of the plan. Each action $A(i)$ can occur only at time $i-1$. So, essentially, a plan is determined by deciding, at each time t $(0 \leq t < n)$, whether to make $X(t+1)$ true or leave it false. These possibilities correspond to the existential quantifier in (10). So a plan is valid iff the choices it makes for values of fluents $X(i)$ satisfies $\Phi(X(1), \ldots, X(n))$.

For part (ii): Let F be

$$\{T(0), T(1), \ldots, T(l), X(1), \ldots, X(l), Y(1), \ldots, Y(m), Z(1), \ldots, Z(n)\}\,.$$

Let $A = \{A(1), \ldots, A(l)\}$. Let *state*$(F)$ be the conjunction of a formula saying that exactly one of the fluents $T(i)$ holds in a state and the formula

$$T(l) \supset \Phi(X(1), \ldots, X(l), Y(1), \ldots, Y(m), Z(1), \ldots, Z(n))\,.$$

Take

$$init(F) = T(0) \wedge state(F) \wedge \bigwedge_{i=1}^{l} \neg X(i)\,,$$

and $goal(F) = \top$. Let $act(F, A, F')$ be the conjunction of a formula guaranteeing nonconcurrency and the following four formulas.

$$\bigwedge_{i=1}^{l} (T(i-1) \supset T(i)')$$

$$\bigwedge_{i=1}^{l} (A(i) \supset T(i-1))$$

$$\bigwedge_{i=1}^{l} (X(i)' \equiv X(i) \vee A(i))$$

$$\bigwedge_{i=1}^{m} (Y(i)' \equiv Y(i))$$

So for each time t, $T(t)$ holds and we also have $\neg T(i)$ for each $i \neq t$. All fluents $X(i)$ are initially false. Their eventual value (at last time l) is determined by whether or not action $A(i)$ occurs in the course of the plan. Each action $A(i)$ can occur only at time $i-1$. So, essentially, a plan is determined by deciding, at each time t ($0 \leq t < l$), whether to make $X(t+1)$ true or leave it false. These possibilities correspond to the outer existential quantifiers in (15). As in the previous construction, the possible initial states account for all possible truth assignments to fluents $Y(i)$, and the values of these fluents do not change over time. This corresponds to the universal quantifiers in (15). The values of fluents $Z(i)$ are not constrained. The state at time l (after the last step in the plan) must satisfy $T(l)$, and hence must also satisfy $\Phi(X(1), \ldots, X(l), Y(1), \ldots, Y(m), Z(1), \ldots, Z(n))$. So a plan is valid iff the choices it makes for values of fluents $X(i)$ along with any possible combination of values of fluents $Y(i)$ can be combined with some valuation for fluents $Z(i)$ to satisfy $\Phi(X(1), \ldots, X(l), Y(1), \ldots, Y(m), Z(1), \ldots, Z(n))$. □

Theorem 11 *For nonconcurrent domains, the following hold.*

(i) Conformant planning for plans of length 1 is Π_2^P-hard.

(ii) Conformant planning for plans of fixed length is Π_2^P-complete.

(iii) For deterministic planning domains in which actions are always executable, conformant planning for plans of length 1 is D^P-hard.

(iv) For deterministic planning domains in which executability of an action in a state is polynomial-time computable, conformant planning for plans of fixed length is D^P-complete.

Proof. Parts (i) and (ii): The fact that the problem belongs to Π_2^P is clear from the remark preceding the statement of the theorem. Hardness can be shown by the construction used in the proof of Proposition 2. So we have hardness for plans of length 1 (with only one possible plan!).

Part (iii): The complexity class D^P is characterized by the family of QBFs of the form with

$$\exists x_1, \ldots, x_m \Phi(x_1, \ldots, x_m) \wedge \forall y_1, \ldots, y_n \Psi(y_1, \ldots, y_n)\,. \tag{54}$$

We must obtain in polynomial time a conformant planning problem, with a nonconcurrent, deterministic planning domain in which actions are always executable, for which there is a valid plan iff (54) is true.

Take $F = \{X(1), \ldots, X(m), Y(1), \ldots, Y(n)\}$ and $A = \emptyset$. Let $\mathit{state}(F) = \top$ and

$$\mathit{act}(F, A, F') = \bigwedge_{i=1}^{m} (X(i)' \equiv X(i)) \wedge \bigwedge_{i=1}^{n} (Y(i)' \equiv Y(i))\,.$$

Take $\mathit{init}(F) = \Phi(X(1), \ldots, X(m))$ and $\mathit{goal}(F) = \Psi(Y(1), \ldots, Y(n))$. Notice that this is a nonconcurrent, deterministic domain with actions always executable. It remains only to show that the conjunction of (12) and (19) is equivalent to (54).

In this case, (12) can be written

$$\exists\, x(1)_0 \cdots x(m)_0\ y(1)_0 \cdots y(n)_0\ \Phi(x(1)_0, \ldots, x(m)_0)$$

which is equivalent to

$$\exists\, x(1)_0 \cdots x(m)_0\ \Phi(x(1)_0, \ldots, x(m)_0)\,. \tag{55}$$

Clearly (55) is equivalent to the first conjunct of (54). It remains only to show that, under the assumption that (55) is true, (19) is equivalent to the second conjunct of (54).

Given the choices of A and $\mathit{state}(F)$, for plan length 1, (19) is equivalent to

$$\forall f_0\ \forall f_1\ \ (\mathit{init}(f_0) \wedge \mathit{act}(f_0, a_0, f_1) \supset \mathit{goal}(f_1))$$

which in this case is the following.

$$\begin{array}{l} \forall\, x(1)_0 \cdots x(m)_0\ y(1)_0 \cdots y(n)_0\ \forall\, x(1)_1 \cdots x(m)_1\ y(1)_1 \cdots y(n)_1 \\ \quad (\Phi(x(1)_0, \ldots, x(m)_0) \wedge \textstyle\bigwedge_{i=1}^{m} (x(i)_1 \equiv x(i)_0) \wedge \bigwedge_{i=1}^{n} (y(i)_1 \equiv y(i)_0) \\ \qquad\qquad \supset \Psi(y(1)_1, \ldots, y(n)_1)) \end{array} \tag{56}$$

Formula (56) is logically equivalent to

$$\begin{array}{l} \forall\, x(1)_0 \cdots x(m)_0\ y(1)_0 \cdots y(n)_0 \\ \qquad\qquad (\Phi(x(1)_0, \ldots, x(m)_0) \supset \Psi(y(1)_0, \ldots, y(n)_0))\,. \end{array} \tag{57}$$

Under the assumption that (55) is true, (57) is equivalent to

$$\forall\, y(1)_0 \cdots y(n)_0\ \Psi(y(1)_0, \ldots, y(n)_0))\,,$$

and we're done.

Part (iv): A problem belongs to D^P if it can be solved in polynomial time given the answer to one **NP** problem and one **coNP** problem. Here the **NP** problem is expressed by (12). The **coNP** problem is more complex. We need to show that at least one of the polynomially-many conformant plans of the fixed length k is valid. That is, we need to show that the following problem is **NP**: each of the conformant plans of length k is either (i) not sufficient or (ii) not executable. To see that for any one plan, each of (i) and (ii) is **NP**, consult the proof of Theorem 4. It remains only to observe that the two problems, for all of the conformant plans of length k, can be solved at once in nondeterministic polynomial time. □