On the Complexity of Domain-Independent Planning*

Kutluhan Erol kutluhan@cs.umd.edu
Dana S. Nau† nau@cs.umd.edu
V. S. Subrahmanian‡ vs@cs.umd.edu

Computer Science Department
University of Maryland
College Park, MD 20742

Abstract

In this paper, we examine how the complexity of domain-independent planning with STRIPS-style operators depends on the nature of the planning operators. We show how the time complexity varies depending on a wide variety of conditions:

- whether or not delete lists are allowed;
- whether or not negative preconditions are allowed;
- whether or not the predicates are restricted to be propositions (i.e., 0-ary);
- whether the planning operators are given as part of the input to the planning problem, or instead are fixed in advance.

Introduction

Despite the acknowledged difficulty of planning, it is only recently that researchers have begun to examine the computational complexity of planning problems and the reasons for that complexity (Chapman, 1987; Bylander, 1991; Gupta & Nau, 1991; Gupta & Nau, 1992; Minton *et al.*, 1991; McAllester and Rosenblitt, 1991). Here, we examine how the complexity of domain-independent planning depends on the nature of the planning operators. We consider planning problems in which the current state is a set of ground atoms, and each planning operator is a STRIPS-style operator consisting of three lists of atoms: a precondition list, an add list, and a delete list. So that it will be decidable whether a plan exists (Erol *et al.*, 1992; Erol *et al.*, 1991), we make the "datalog" restriction that no function symbols are allowed and only finitely many constant symbols are allowed. Our results are summarized in Table 1. Examination of this table reveals several interesting properties:

1. For PLAN EXISTENCE,[1] comparing the propositional case (in which all predicates are restricted to be 0-ary) with the datalog case (in which the predicates may have constants or variables as arguments) reveals a regular pattern. In most cases, the complexity in the datalog case is exactly one level harder than the complexity in the corresponding propositional case. We have EXPSPACE-complete versus PSPACE-complete, NEXPTIME-complete versus NP-complete, EXPTIME-complete versus polynomial.
2. If delete lists are allowed, then PLAN EXISTENCE is EXPSPACE-complete but PLAN LENGTH is only NEXPTIME-complete. Normally, one would not expect PLAN LENGTH to be easier than PLAN EXISTENCE. In this case, it happens because the length of a plan can sometimes be doubly exponential in the length of the input. In PLAN LENGTH we are given a bound k, encoded in binary, which confines us to plans of length at most exponential in terms of the input. Hence in the worst case, PLAN LENGTH is easier than PLAN EXISTENCE.

   We do not observe the same anomaly in the propositional case, because the lengths of the plans are at most exponential in the length of the input. Hence, giving an exponential bound on the length of the plan does not reduce the complexity of PLAN LENGTH. As a result, in the propositional case, both PLAN EXISTENCE and PLAN LENGTH are PSPACE-complete.
3. When the operator set is fixed in advance, any operator whose predicates are not all propositions can be mapped into a set of operators whose predicates are all propositions. Thus, planning with a fixed set of datalog operators has basically the same complexity as planning with propositional operators that are given as part of the input.
4. PLAN LENGTH has the same complexity regardless of whether or not negated preconditions are allowed.

*This work was supported in part by NSF Grant NSFD CDR-88003012 to the University of Maryland Systems Research Center, as well as NSF grants IRI-8907890 and IRI-9109755.

†Also in the Systems Research Center and the Institute for Advanced Computer Studies.

‡Also in the Institute for Advanced Computer Studies.

[1]Informally, PLAN EXISTENCE is the problem of determining whether a plan exists, and PLAN LENGTH is the problem of determining whether there is a plan of length $\leq k$. Formal definitions appear in the next section.

Table 1: Complexity of domain-independent planning.

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Telling if a plan exists	Telling if there's a plan of length $\leq k$
datalog (no function symbols, and only finitely many constant symbols)	given in the input	yes	yes/no	EXPSPACE-complete	NEXPTIME-complete
		no	yes	NEXPTIME-complete	NEXPTIME-complete
			no	EXPTIME-complete	NEXPTIME-complete
			no[α]	PSPACE-complete	PSPACE-complete
	fixed	yes	yes/no	in PSPACE [γ]	in PSPACE [γ]
		no	yes	in NP [γ]	in NP [γ]
			no	in P	in NP [γ]
			no[α]	in NLOGSPACE	in NP
propositional (all predicates are 0-ary)	given in the input	yes	yes/no	PSPACE-complete[δ]	PSPACE-complete
		no	yes	NP-complete [δ]	NP-complete
			no	in P [δ]	NP-complete
			no[α]/no[β]	NLOGSPACE-complete	NP-complete
	fixed	yes/no	yes/no	constant time	constant time

[α] No operator has more than one precondition.
[β] Every operator with more than one precondition is a composition of other operators.
[γ] With PSPACE- or NP-completeness for some sets of operators.
[δ] These results are due to Bylander (1991). All other results are new.

This is because what makes the problem hard is the task of choosing operators that achieve several subgoals in order to minimize the overall length of the plan, and this task remains equally hard regardless of whether negated preconditions are allowed.

5. Delete lists are more powerful than negated preconditions. Thus, if the operators are allowed to have delete lists, then whether or not they have negated preconditions has no effect on the complexity.

Definitions

Let $\mathcal{L}$ be any datalog (i.e., function-free first-order) language. Then a *state* is any nonempty set of ground atoms in $\mathcal{L}$. Intuitively, a state tells us which ground atoms are currently true. Thus, if a ground atom A is in state S, then A is true in state S; if $B \notin S$, then B is false in state S. Thus, a state is simply an Herbrand interpretation for the language $\mathcal{L}$, and hence each formula of first-order logic is either satisfied or not satisfied in S according to the usual first-order logic definition of satisfaction.

Let $\mathcal{L}$ be any datalog language. A *planning operator* α is a 4-tuple (Name(α), Pre(α), Add(α), Del(α)), where

1. Name(α) is a syntactic expression of the form $\alpha(X_1, \ldots, X_n)$ where each X_i is a variable symbol of $\mathcal{L}$;
2. Pre(α) is a finite set of literals, called the *precondition list* of α, whose variables are all from the set $\{X_1, \ldots, X_n\}$;
3. Add(α) and Del(α) are both finite sets of atoms (possibly non-ground) whose variables are taken from the set $\{X_1, \ldots, X_n\}$. Add(α) is called the *add list* of α, and Del(α) is called the *delete list* of α.

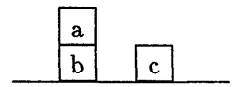

Fig. 1: Initial configuration

Fig. 2: Goal configuration

Observe that atoms and negated atoms may occur in the precondition list, but negated atoms may not occur in either the add list or the delete list.

A *planning domain* is a pair $\mathbf{P} = (S_0, \mathcal{O})$, where S_0 is the *initial state* and $\mathcal{O}$ is a finite set of operators. A *planning problem* is a triple $\mathbf{P} = (S_0, \mathcal{O}, G)$, where $(S_0, \mathcal{O})$ is a planning domain and G is a *goal* (an existentially closed conjunction of atoms). In both cases, the *language* of $\mathbf{P}$ is the datalog language $\mathcal{L}$ generated by the constant, predicate, and variable symbols appearing in $\mathbf{P}$, along with an infinite number of variable symbols.

Example 1 (Blocks World) Consider a blocks-world domain containing three blocks a, b, c, along with Nilsson's "stack", "unstack", "pickup", and "put-down" operators (Nilsson, 1980). Suppose the initial and goal configurations are as shown in Figs. 1 and 2. This domain can be represented as follows:

- *Language.* The language $\mathcal{L}$ contains a supply of variable symbols $X_1, X_2, \ldots$, and three constant symbols a, b, c to represent the three blocks. $\mathcal{L}$ contains a binary predicate symbol "on", unary predicate symbols "ontable", "clear", and "holding", and a 0-ary

predicate symbol "handempty". Operator names, such as "stack", "unstack", etc., are not part of $\mathcal{L}$.

- *Operators.* The "unstack" operator is the following 4-tuple (the "stack", "pickup", and "putdown" operators are defined analogously):

$$\begin{aligned} \text{Name}: &\quad \text{unstack}(X_1, X_2) \\ \text{Pre}: &\quad \{\text{on}(X_1, X_2), \text{clear}(X_1), \text{handempty}()\} \\ \text{Del}: &\quad \{\text{on}(X_1, X_2), \text{clear}(X_1), \text{handempty}()\} \\ \text{Add}: &\quad \{\text{clear}(X_2), \text{holding}(X_1)\} \end{aligned}$$

- *Planning Domain.* The planning domain is $(S_0, \mathcal{O})$, where S_0 and $\mathcal{O}$ are as follows:

$$\begin{aligned} S_0 &= \{\text{clear}(a), \text{on}(a,b), \text{ontable}(b), \\ &\quad \text{clear}(c), \text{ontable}(c), \text{handempty}()\}; \\ \mathcal{O} &= \{\text{stack}, \text{unstack}, \text{pickup}, \text{putdown}\}. \end{aligned}$$

- *Planning Problem.* The planning problem is $(S_0, \mathcal{O}, G)$, where $G = \{\text{on}(b,c)\}$.

Let $\mathbf{P} = (S_0, \mathcal{O})$ be a planning domain, α be an operator in $\mathcal{O}$ whose name is $\alpha(X_1, \ldots, X_n)$, and θ be a substitution that assigns ground terms to each $X_i, 1 \leq i \leq n$. Suppose that the following conditions hold:

$$\{A\theta | A \text{ is an atom in } \text{Pre}(\alpha)\} \subseteq S;$$
$$\{B\theta | \neg B \text{ is a negated literal in } \text{Pre}(\alpha)\} \cap S = \emptyset;$$
$$S' = (S - (\text{Del}(\alpha)\theta)) \cup (\text{Add}(\alpha))\,\theta.$$

Then we say that α is *θ-executable* in state S, *resulting* in state S'. This is denoted symbolically as $S \stackrel{\alpha,\theta}{\Longrightarrow} S'$.

Suppose $\mathbf{P} = (S_0, \mathcal{O}, G)$ is a planning problem. A *plan in* $\mathbf{P}$ *that achieves* G is a sequence

$$S_0 \stackrel{\alpha_0,\theta_0}{\Longrightarrow} S_1 \stackrel{\alpha_1,\theta_1}{\Longrightarrow} S_2 \cdots \stackrel{\alpha_{n-1},\theta_{n-1}}{\Longrightarrow} S_n$$

such that G is satisfied by S_n, i.e. there exists a ground instance of G that is true in S_n. The *length* of the above plan is n.

Let $\mathbf{P} = (S_0, \mathcal{O})$ be a planning domain or $\mathbf{P} = (S_0, \mathcal{O}, G)$ be a planning problem; and let $\mathcal{L}$ be the language of $\mathbf{P}$. Then:

1. $\mathcal{O}$ and $\mathbf{P}$ are *positive* if for all $\alpha \in \mathcal{O}$, $\text{Pre}(\alpha)$ is a finite set of *atoms* (i.e. negations are not present in $\text{Pre}(\alpha)$).
2. $\mathcal{O}$ and $\mathbf{P}$ are *deletion-free* if for all $\alpha \in \mathcal{O}$, $\text{Del}(\alpha) = \emptyset$.
3. $\mathcal{O}$ and $\mathbf{P}$ are *context-free* if for all $\alpha \in \mathcal{O}$, $|\text{Pre}(\alpha)| \leq 1$, i.e., $\text{Pre}(\alpha)$ contains at most one atom.
4. $\mathcal{O}$ and $\mathbf{P}$ are *side-effect-free* if for all $\alpha \in \mathcal{O}$, $|\text{Add}(\alpha) \cup \text{Del}(\alpha)| \leq 1$, i.e., α has at most one post-condition.
5. $\mathcal{O}$ and $\mathbf{P}$ are *propositional* if every predicate P in $\mathcal{L}$ is a propositional symbol (i.e. a 0-ary predicate symbol).

PLAN EXISTENCE is the following problem: "given a planning problem $\mathbf{P} = (S_0, \mathcal{O}, G)$, is there a plan in $\mathbf{P}$ that achieves G?" PLAN LENGTH is the following problem:[2] "given a planning problem $\mathbf{P} = (S_0, \mathcal{O}, G)$ and an integer k encoded in binary, is there a plan in $\mathbf{P}$ of length k or less that achieves G?"

Eliminating Negated Preconditions

We show below that if delete lists are allowed, then we can remove negations from preconditions of operators in polynomial time. Thus, if delete lists are allowed, negated preconditions do not affect the complexity of planning.

Theorem 1 (Eliminating Negated Preconditions) In polynomial time, given any planning domain $\mathbf{P} = (S_0, \mathcal{O})$ we can produce a positive planning domain $\mathbf{P}' = (S_0', \mathcal{O}')$ having the following properties:

1. For every goal G, a plan exists for G in $\mathbf{P}$ if and only if a plan exists for G in $\mathbf{P}'$.
2. For every goal G and non-negative integer l, there exists a plan of length l for G in $\mathbf{P}$ if and only if there exists a plan of length $l + 2^{kv}$ for G in $\mathbf{P}'$, where k is the maximum arity among the predicates of $\mathbf{P}$ and $v = \lceil \lg c \rceil$, where c is the number of constants in $\mathbf{P}$ (i.e., v is the number of bits necessary to encode the constants in binary).

Here is a sketch of $\mathbf{P}'$. For each predicate in $\mathbf{P}$, we introduce a complementary predicate in $\mathbf{P}'$, and modify the operators such that whenever one is added, the other is deleted and vica versa. Hence negated preconditions can be replaced by their complementary predicates. We use a chain of operators to assert the instances of complementary predicates whose corresponding instances are not in the initial state of $\mathbf{P}$. We set the preconditions such that any plan in $\mathbf{P}'$ has to start with this chain. This guarantees that property 2 is satisfied.

Note that $\mathbf{P}'$ will not be deletion-free, even if $\mathbf{P}$ is.

Varying Sets of Operators

In this section, we consider the complexity of planning in the "domain-independent" case, in which the operators are part of the input and thus different problem instances may have different operator sets.

[2] In this definition, we follow the standard procedure for converting optimization problems into yes/no decision problems. What really interests us, of course, is the problem of finding the shortest plan that achieves G. This problem is at least as difficult as PLAN LENGTH, and in some cases harder. For example, in the Towers of Hanoi problem (Aho *et al.*, 1976) and certain generalizations of it (Graham *et al.*, 1989), the length of the shortest plan can be found in low-order polynomial time—but actually producing this plan requires exponential time and space, since the plan has exponential length. For further discussion of the relation between the complexity of optimization problems and the corresponding decision problems, the reader is referred to pp. 115–117 of Garey & Johnson (1979).

Case 1: Propositional Operators

The following theorems deal with the special case in which all predicates are propositions (i.e., 0-ary).

Theorem 2 (due to Bylander (1991))

1. If we restrict **P** to be propositional, then PLAN EXISTENCE is PSPACE-complete.
2. If we restrict **P** to be propositional and positive, then PLAN EXISTENCE is PSPACE-complete.
3. If we restrict **P** to be propositional and deletion-free, then PLAN EXISTENCE is NP-complete.
4. If we restrict **P** to be propositional, deletion-free, and positive then PLAN EXISTENCE is in P.
5. If we restrict **P** to be propositional, positive, and side-effect-free, then PLAN EXISTENCE is in P.

Theorem 3 If we restrict **P** to be propositional, positive, context-free, and deletion-free, then PLAN EXISTENCE is NLOGSPACE-complete.

Theorem 4 If we restrict **P** to be propositional, positive, context-free and deletion-free, then PLAN LENGTH is NP-complete.

Corollary 1 If we restrict **P** to be propositional, positive and deletion-free, then PLAN LENGTH is NP-complete.

Corollary 2 If we restrict **P** to be propositional and deletion-free, PLAN LENGTH is NP-complete.

Theorem 5 PLAN LENGTH is PSPACE-complete if we restrict **P** to be propositional. It is still PSPACE-complete if we restrict **P** to be propositional and positive.

Both Theorem 3 and Clause 5 of Theorem 2 require restrictions on the number of clauses in the preconditions and/or postconditions of the planning operators. These restrictions can easily be weakened by allowing the operators to be composed, as described below.

An operator α is *composable* with another operator β if the positive preconditions of β and del(α) are disjoint, and the negative preconditions of β and add(α) are disjoint.

If α and β are composable, then the *composition* of α with β is

$$\begin{aligned} \text{Pre}: &\quad \text{Pre}(\alpha) \cup (P_1 - \text{Add}(\alpha)) \cup (P_2 - del(\alpha)) \\ \text{Add}: &\quad \text{Add}(\beta) \cup (\text{Add}(\alpha) - \text{Del}(\beta)) \\ \text{Del}: &\quad \text{Del}(\beta) \cup (\text{Del}(\alpha) - \text{Add}(\beta)) \end{aligned}$$

where P_1 and P_2, respectively, are β's positive and negative preconditions.

Theorem 6 (Composition Theorem) Let $\mathbf{P} = (S_0, \mathcal{O})$ be a planning domain, and $\mathcal{O}'$ be a set of operators such that each operator in $\mathcal{O}'$ is the composition of operators in $\mathcal{O}$. Then for any goal G, there is a plan to achieve G in **P** iff there is a plan to achieve G in $\mathbf{P}'$, where $\mathbf{P}' = (S_0, \mathcal{O} \cup \mathcal{O}')$.

The above lets us extend the scope of several previous theorems:

Corollary 3 Suppose we restrict $\mathbf{P} = (S_0, \mathcal{O}, G)$ to be such that $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$, where $\mathcal{O}_1$ is propositional, deletion-free, positive and context-free, and every operator in $\mathcal{O}_2$ is the composition of operators in $\mathcal{O}_1$. Then PLAN EXISTENCE is NLOGSPACE-complete.

Corollary 4 Suppose we restrict $\mathbf{P} = (S_0, \mathcal{O}, G)$ to be such that $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$, where $\mathcal{O}_1$ is propositional, positive, and side-effect-free, and every operator in $\mathcal{O}_2$ is the composition of operators in $\mathcal{O}_1$. Then PLAN EXISTENCE is in P.

Example 2 (Reformulation of Blocks World) Bylander (1991) reformulates the blocks world so that each operator is restricted to positive preconditions and one postcondition. Instead of the usual "on" and "clear" predicates, he uses proposition off_{ij} to denote that block i is not on block j. For each pair of blocks i and j, he has two operators: one that moves block i from the top of block j to the table, and one that moves block i from the table to the top of block j. These operators are defined as follows:

$$\begin{aligned} \text{Name}: &\quad \text{totable}_{ij} \\ \text{Pre}: &\quad \{\text{off}_{1,i}, \text{off}_{2,i}, \ldots, \text{off}_{n,i}, \text{off}_{1,j}, \text{off}_{2,j}, \\ &\quad \ldots, \text{off}_{i-1,j}, \text{off}_{i+1,j}, \ldots, \text{off}_{n,j}\} \\ \text{Del}: &\quad \emptyset \\ \text{Add}: &\quad \{\text{off}_{i,j}\} \end{aligned}$$

$$\begin{aligned} \text{Name}: &\quad \text{toblock}_{ij} \\ \text{Pre}: &\quad \{\text{off}_{1,i}, \text{off}_{2,i}, \ldots, \text{off}_{n,i}, \text{off}_{1,j}, \text{off}_{2,j}, \\ &\quad \ldots, \text{off}_{n,j}, \text{off}_{i,1}, \text{off}_{i,2}, \ldots, \text{off}_{i,n}\} \\ \text{Del}: &\quad \{\text{off}_{i,j}\} \\ \text{Add}: &\quad \emptyset \end{aligned}$$

In Bylander's formulation of blocks world, **P** is positive and side-effect-free. Thus as a consequence of Clause 5 of Theorem 2, in Bylander's formulation of blocks world PLAN EXISTENCE can be solved in polynomial time.

In Bylander's formulation of the blocks world, it is not possible for blocks to be moved directly from one stack to another. This has two consequences, as described below.

The first consequence is that in Bylander's formulation of blocks world, PLAN LENGTH can be solved in polynomial time. To show this, below we describe how to compute how many times each block b must be moved in the optimal plan. Thus, to see whether or not there is a plan of length k or less, all that is needed is to compare k with

$$\sum_b \text{how many times } b \text{ must be moved.}$$

Let S be the current state, and b be any block. If the stack of blocks from b down to the table is consistent with the goal conditions (whether or not this is so

can be determined in polynomial time (Gupta & Nau, 1992)), then b need not be moved. Otherwise, there are three possibilities:

1. If b is on the table and the goal requires that b be on some other block c, then in the shortest plan, b must be moved exactly once: from the table to c.
2. If b is on some block c and the goal requires that b be on the table, then in the shortest plan, b must be moved once: from c to the table.
3. If b on some block c and the goal requires that b be on some block d (which may be the same as c), then in the shortest plan, b must be moved exactly twice: from c to the table, and from the table to d.

The second consequence is that translating an ordinary blocks-world problem into Bylander's formulation will not always preserve the length of the optimal plan. The reason for this is that in the ordinary formulation of blocks world, the optimal plan will often involve moving blocks directly from one stack to another without first moving them to the table, and this cannot be done in Bylander's formulation. It appears that Bylander's formulation cannot be extended to allow this kind of move another without violating the restriction that each has only positive preconditions and one postcondition.

We can easily overcome the above problem by augmenting Bylander's formulation to include all possible compositions of pairs of his operators. Theorem 2 does not apply to this formulation, but Corollary 4 does apply, and gives the same result as before: PLAN EXISTENCE can be solved in polynomial time.

Since this extension to Bylander's formulation allows stack-to-stack moves, there is a one-to-one correspondence between plans in this formulation and the more usual formulations of the blocks world, such as those given in (Charniak & McDermott, 1985; Warren, 1990; Nilsson, 1980; Waldinger, 1990; Gupta & Nau, 1991; Gupta & Nau, 1992). Thus, from results proved in (Gupta & Nau, 1992), it follows that in this extension of Bylander's formulation, PLAN LENGTH is NP-complete.

Case 2: Datalog Operators

Below, we no longer restrict the predicates to be propositions. As a result, planning is much more complex than in the previous case.

Theorem 7 If we restrict $\mathbf{P}$ to be positive and deletion-free, then PLAN EXISTENCE is EXPTIME-complete.

Theorem 8 If we restrict $\mathbf{P}$ to be deletion-free, then PLAN EXISTENCE is NEXPTIME-complete.

Theorem 9 PLAN EXISTENCE is EXPSPACE-complete. It is still EXPSPACE-complete if we restrict $\mathbf{P}$ to be positive.

We show below that when we restrict preconditions of planning operators to contain at most one *atom*, then the planning problem is PSPACE-complete.

Theorem 10 If we restrict $\mathbf{P}$ to be context-free, positive, and deletion-free, then PLAN EXISTENCE is PSPACE-complete.

Theorem 11 If we restrict $\mathbf{P}$ to be deletion-free, positive, and context-free, then PLAN LENGTH is PSPACE-complete.

Theorem 12 PLAN LENGTH is NEXPTIME-complete in each of these cases:

$\mathbf{P}$ is deletion-free and positive;
$\mathbf{P}$ is deletion-free;
$\mathbf{P}$ is positive;
no restrictions on $\mathbf{P}$.

Fixed Sets of Operators

The above results are for the case in which the set of operators is part of the input. However, in many well known planning problems, the set of operators is fixed. For example, in the blocks world (see Example 1), we have only four operators: stack, unstack, pickup and putdown.

In this section we will present complexity results on planning problems in which the set of operators is fixed, and only the initial state and goal are allowed to vary. The problems we will consider will be of the form: "given the initial state S_0 and the goal G, is there a plan that achieves G?" We assume no predicate/proposition that does not appear in the operators appears in G or S_0. Since the operators can neither add nor delete atoms constructed from these predicates, this is a reasonable restriction.

Case 1: Propositional Operators

Propositional planning with a fixed set of operators is very restrictive. The number of possible plans is constant. We include the following two results just for the sake of completeness.

Theorem 13 PLAN EXISTENCE can be solved in constant time if we restrict $\mathbf{P} = (S_0, \mathcal{O}, G)$ to be propositional and $\mathcal{O}$ to be a fixed set.

Corollary 5 PLAN LENGTH can be solved in constant time if we restrict $\mathbf{P} = (S_0, \mathcal{O}, G)$ to be propositional and $\mathcal{O}$ to be a fixed set.

Case 2: Datalog Operators

When the set of operators is fixed, we can enumerate all ground instances in polynomial time, reducing the problem to propositional planning with a varying set of operators.[3] Thus, the following theorem follows from

[3]This is similar, but not identical, to the reformulation of the Blocks World given in Example 2. The reformulation in Example 2 also involved replacing the "on" and "clear" predicates by a single "off" predicate, as well as some changes to the nature of the planning operators.

Theorems 2–5 and their corollaries.

Theorem 14

1. If we restrict **P** to be fixed, deletion-free, context-free and positive, then PLAN EXISTENCE is in NLOGSPACE and PLAN LENGTH is in NP.
2. If we restrict **P** to be fixed, deletion-free, and positive, then PLAN EXISTENCE is in P and PLAN LENGTH is in NP.
3. If we restrict **P** to be fixed and deletion-free, then PLAN EXISTENCE and PLAN LENGTH are in NP.
4. If we restrict **P** to be fixed, then PLAN EXISTENCE and PLAN LENGTH are in PSPACE.

The above theorem puts a bound on how hard planning can be with a fixed set of operators. The following theorems state that we can find fixed sets of operators such that their corresponding planning problems are complete for these complexity classes.

Theorem 15 There exists a fixed positive deletion-free set of operators $\mathcal{O}$ for which PLAN LENGTH is NP-hard.

Theorem 16 There exist fixed deletion-free sets of operators $\mathcal{O}$ for which PLAN EXISTENCE and PLAN LENGTH are NP-hard.

Theorem 17 There exists a fixed set of positive operators $\mathcal{O}$ for which PLAN EXISTENCE and PLAN LENGTH are PSPACE-hard.

Conclusion

The primary aim of this paper has been to develop an exhaustive analysis of the complexity of planning with STRIPS-style planning operators (i.e., operators comprised of preconditions, add lists, and delete lists). Based on various syntactic restrictions on the planning operators, we have developed a comprehensive theory of the complexity of planning.

In order to guarantee that PLAN EXISTENCE and PLAN LENGTH are decidable, we have restricted the planning language $\mathcal{L}$ to be a datalog language. Thus, $\mathcal{L}$ has no function symbols, as is the case in STRIPS and many other planning systems. In (Erol *et al.*, 1992; Erol *et al.*, 1991), we show that if $\mathcal{L}$ is allowed to contain function symbols (and thus contain infinitely many ground terms), then PLAN EXISTENCE and PLAN LENGTH are both undecidable in general.

In summary, planning is a hard problem even under severe restrictions on the nature of planning operators. Thus, in order to construct efficient planners, it is important to find other ways to prevent the complexity from getting out of hand. For example, Yang *et al.* (1990; 1992) show how to develop efficient algorithms for merging plans to achieve multiple goals, given certain kinds of restrictions on what kinds of goal and subgoal interactions can occur.

References

Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1976. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

Bylander, Tom 1991. Complexity results for planning. In *IJCAI-91.*

Chapman, David 1987. Planning for conjunctive goals. *Artificial Intelligence* 32:333–378.

Charniak, Eugene and McDermott, Drew 1985. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, MA.

Erol, K.; Nau, D.; and Subrahmanian, V. S. 1991. Complexity, decidability and undecidability results for domain-independent planning. CS TR-2797, UMIACS TR-91-154, and SRC TR 91-96; under review.

Erol, K.; Nau, D.; and Subrahmanian, V. S. 1992. When is planning decidable? In *Proc. First Internat. Conf. AI Planning Systems.* To appear.

Garey, Michael R. and Johnson, David S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York.

Graham, R. L.; Knuth, D. E.; and Patashnik, O. 1989. *Concrete Mathematics: a Foundation for Computer Science.* Addison-Wesley.

Gupta, Naresh and Nau, Dana S. 1991. Complexity results for blocks-world planning. In *Proc. AAAI-91.* Honorable mention for the best paper award.

Gupta, N. and Nau, D. 1992. On the complexity of blocks-world planning. *Artificial Intelligence.* To appear.

McAllester, D. and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proc. AAAI-91.*

Minton, S.; Bresna, J.; and Drummond, M. 1991. Commitment strategies in planning. In *Proc. IJCAI-91.*

Nilsson, N. J. 1980. *Principles of Artificial Intelligence.* Tioga, Palo Alto.

Waldinger, R. 1990. Achieving several goals simultaneously. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning.* Morgan Kaufman. 118–139. Originally appeared in *Machine Intelligence 8*, 1977.

Warren, D. H. D. 1990. Extract from Kluzniak and Szapowicz APIC studies in data processing, no. 24, 1974. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning.* Morgan Kaufman. 140–153.

Yang, Q.; Nau, D. S.; and Hendler, J. 1990. Optimization of multiple-goal plans with limited interaction. In *Proc. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control.*

Yang, Q.; Nau, D. S.; and Hendler, J. 1992. Merging separately generated plans with restricted interactions. *Computational Intelligence.* To appear.