AAAI 1993 SPRING SYMPOSIUM ON AI AND NP-HARD PROBLEMS

A Theoretical Study of Domain-Independent Planning*

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

University of Maryland
College Park, MD 20742

Introduction

In two previous conference papers, we examined how the decidability (Erol *et al.*, 1992c) and complexity (Erol *et al.*, 1992b) of domain-independent planning with STRIPS-style planning operators depends on the nature of the operators. However, space constraints did not allow us to include much discussion of the meaning and significance of this work—and in addition, we have subsequently produced several additional results that did not appear in those papers. In this report, we briefly summarize both our previous and current results, and then concentrate on discussing their meaning, significance, and relation to others' work.

We discuss conditions under which planning is decidable and undecidable. Our results on this topic solve an open problem posed by Chapman (1987), and clear up some difficulties with his undecidability theorems.

For those cases where planning is decidable, we discuss how the time complexity varies depending on a wide variety of conditions: (1) whether or not function symbols are allowed; (2) whether or not delete lists are allowed; (3) whether or not negative preconditions are allowed; (4) whether or not the predicates are restricted to be propositional (i.e., 0-ary); and (5) whether the planning operators are given as part of the input to the planning problem, or instead are fixed in advance.

Chapman (1987) and Dean and Boddy (1988) studied planning with conditional operators, and showed that the problem of deciding whether a proposition is necessarily true after a partially ordered plan (a.k.a modal truth criterion) is NP-hard in the presence of conditional operators. The same problem can be solved in polynomial time when conditional operators are not allowed, and this led researchers to believe that planning with conditional operators is harder than planning with regular STRIPS operators. However, our new results show that, contrary to the expectations, conditional operators do not affect the complexity of plan existance, nor the complexity of plan optimality problems.

For a more extensive treatment, including all of the mathematical details, see Erol *et al.* (1992a).

*This work was supported in part by Army Research Office Grant DAAL-03-92-G-0225, and NSF Grants NSFD CDR-88003012, IRI-8907890, and IRI-9109755.

[†]Department of Computer Science.

[‡]Department of Computer Science, and Institute for Systems Research.

[§]Department of Computer Science, and Institute for Advanced Computer Studies.

Basic Definitions

If $\mathcal{L}$ is a first-order language, then a *state* is a set of ground atoms in $\mathcal{L}$. Intuitively, a state tells us which ground atoms are currently true: if a ground atom A is in state S, then A is true in state S, and if $B \notin S$, then B is false in state S. Thus, a state is simply an Herbrand interpretation (cf. Shoenfield 1967) 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.

We use STRIPS-style planning operators similar to those used by Nilsson (1980). In particular, a *planning operator* α is a 4-tuple $(\text{Name}(\alpha), \text{Pre}(\alpha), \text{Add}(\alpha), \text{Del}(\alpha))$, where

1. $\text{Name}(\alpha)$ 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. $\text{Pre}(\alpha)$ is a finite set of literals (i.e., atoms and negated atoms), called the *precondition list* of α, whose variables are all from the set $\{X_1, \ldots, X_n\}$;
3. $\text{Add}(\alpha)$ and $\text{Del}(\alpha)$ are both finite sets of atoms (possibly non-ground) whose variables are taken from the set $\{X_1, \ldots, X_n\}$. $\text{Add}(\alpha)$ is called the *add list* of α, and $\text{Del}(\alpha)$ is called the *delete list* of α.

Observe that negated atoms are allowed in the precondition list, but not in the add and delete lists.

A *planning domain* is a pair $\mathbf{P} = (S_0, \mathcal{O})$, where S_0 is a state called the *initial state*, and $\mathcal{O}$ is a finite set of planning operators. The *language* of $\mathbf{P}$ is the first-order language $\mathcal{L}$ generated by the constant, function, predicate, and variable symbols appearing in $\mathbf{P}$, along with an infinite number of additional variable symbols.

A *goal* is a conjunction of atoms which is existentially closed (i.e., the variables, if any, are existentially quantified). 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.

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 for states S and S':

$\{A\theta : A \text{ is a positive literal in } \mathrm{Pre}(\alpha)\} \subseteq S$;

$\{B\theta : \neg B \text{ is a negative literal in } \mathrm{Pre}(\alpha)\} \cap S = \emptyset$;

$S' = (S - (\mathrm{Del}(\alpha)\theta)) \cup (\mathrm{Add}(\alpha)\theta)$.

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 $\mathbf{P} = (S_0, \mathcal{O})$ is a planning domain and G is a goal. A *plan that achieves* G is a sequence $S_0, \ldots, S_n$ of states, a sequence $\alpha_1, \ldots, \alpha_n$ of planning operators, and a sequence $\theta_1, \ldots, \theta_n$ of substitutions such that

$$S_0 \stackrel{\alpha_1,\theta_1}{\Longrightarrow} S_1 \stackrel{\alpha_2,\theta_2}{\Longrightarrow} S_2 \cdots \stackrel{\alpha_n,\theta_n}{\Longrightarrow} S_n$$

and 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.

We now define two decision problems:

- PLAN EXISTENCE is the 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 problem, "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?"

Special-Case Definitions

Acyclicity and Boundedness.

A *level mapping* for a language L is a mapping $\ell : AT(L) \rightarrow \mathbf{N}$ where $AT(L)$ is the set of ground atoms in language L and $\mathbf{N}$ is the set of natural numbers. A *predicate level mapping* for L is a mapping $\sharp : Pred(L) \rightarrow \mathbf{N}$ where $Pred(L)$ is the set of predicate symbols in language L.

Suppose $\mathbf{P} = (S_0, \mathcal{O})$ is a planning domain in which no operator has negative preconditions or delete lists. $\mathbf{P}$ is said to be *atomically acyclic* iff there exists a level mapping ℓ such that for all ground instances, α, of operators in $\mathbf{P}$, it is the case that $\ell(A) > \ell(B)$ for all $A \in Add(\alpha)$ and $B \in Pre(\alpha)$. $\mathbf{P}$ is said to be *predicate acyclic* iff there exists a predicate level mapping $\sharp$ such that for all operators α in $\mathbf{P}$, it is the case that $\sharp(p) > \sharp(q)$ for all predicates p occurring in $Add(\alpha)$ and all predicates q occurring in $Pre(\alpha)$.

A planning domain $\mathbf{P} = (S_0, \mathcal{O})$ is *weakly recurrent* iff there exists a level mapping ℓ such that for every ground instance α of an operator in $\mathcal{O}$, if $A \in Add(\alpha)$ is such that there is no plan to achieve A from $\mathbf{P}$, then there is a $B_i \in Pre(\alpha)$ such that there is no plan to achieve B_i from $\mathbf{P}$ and $\ell(A) > \ell(B_i)$.

Suppose $G = (\exists)(A_1 \& \ldots \& A_n)$ is a goal. Let $Grd(G)$ denote the set of all ground instances of the quantifier-free conjunction $(A_1 \& \ldots \& A_n)$. G is *bounded* w.r.t. a level mapping ℓ iff there is an integer b such that for every ground instance $(A_1 \& \ldots \& A_n)\theta$ in $grd(G)$, it is the case that $\ell(A_i) < b$.

Conditional Planning Operators.

Several researchers (Chapman, 1987; Dean and Boddy, 1988; Peot, 1992; Pednault, 1988) have been interested in actions whose effects depend on the input situation. The following formulation of conditional planning operators is due to Dean and Boddy (1988). A *conditional operator* α is a finite set $\{t_1, t_2, \ldots, t_n\}$, where each t_i is a triple of the form $(\mathrm{Pre}_i, \mathrm{Del}_i, \mathrm{Add}_i)$. Pre_i, Del_i, and Add_i correspond to the precondition list, delete list and add list associated with the i'th triple, respectively. Hence each of these lists are sets of atoms.

Suppose α is a conditional operator, θ is a ground substitution for the variables appearing in α, S is a state, $I = \{i : S \text{ satisfies } \mathrm{Pre}_i\theta\}$, and

$$S' = (S - \bigcup_{i \in I} \mathrm{Del}_i\theta) \cup \bigcup_{i \in I} \mathrm{Add}_i\theta.$$

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

Our results are independent of whether we use conditional operators such as the ones defined above, or the ordinary STRIPS-style planning operators in the "Basic Definitions" section.

Decidability and Undecidability

Summary.

Our decidability and undecidability results are summarized in Table 1; for their details, see (Erol *et al.*, 1992a). Whether the planning operators are fixed in advance or given as part of the input, and whether or not they are conditional, does not affect these results.

If we use the conventional definitions of a first-order language and a state (i.e., the language contains only finitely many constant symbols and all states are finite), then whether or not PLAN EXISTENCE is decidable depends largely on whether or not function symbols are allowed:

1. If the language is allowed to contain function symbols (and hence infinitely many ground terms), then, in general, PLAN EXISTENCE is undecidable, regardless of whether or not the operators have delete lists, negative preconditions, or more than one precondition.
2. When certain syntactic (predicate and atomic acyclicity) and semantic properties (weak recurrence) are satisfied by planning domains (even those

Table 1: Decidability of domain-independent planning. These results are independent of whether the operators are fixed in advance, and whether conditional operators are allowed.

Allow function symbols?	Allow infinitely many constants?	Allow infinite initial states?	Allow delete lists and/or negated preconditions?	Telling if a plan exists
yes	yes/no	yes/no	yes/no/no$^{\alpha}$	semidecidable
	no	no	no$^{\beta}$	decidable
no	yes	yes	yes/no	semidecidable
		no	yes	semidecidable
			no	decidable
	no	no$^{\gamma}$	yes/no	decidable

$^{\alpha}$No operator has more than one precondition.
$^{\beta}$With acyclicity and boundedness restrictions as described in the "Special-Case Definitions" section.
$^{\gamma}$The other restrictions ensure that the initial state will always be finite.

containing function symbols) in which there are no delete lists or negative preconditions, then plan existence for bounded goals is decidable.

3. If the language does not contain function symbols (and hence has only finitely many ground terms), then PLAN EXISTENCE is decidable, regardless of whether or not the planning operators have negative preconditions, delete lists, or more than one precondition.

Discussion.

The basis for these results is that we have shown that logic programming is essentially the same as planning without delete lists. This is established by showing how to transform a planning domain without delete lists into a logic program such that for all goals G, the goal G is achievable from the planning domain iff the logical query that G represents is provable from the corresponding logic program. Furthermore, we have shown that every logic program may be transformed to an equivalent planning domain in which each operator has no negative preconditions and no delete lists. This equivalence has allowed us to transport many results from logic programming to planning, leading to a number of decidability and undecidability results.

For comparison with Chapman's (Chapman, 1987) results, Table 1 also includes decidability and undecidability results for the cases where we allow infinitely many constant symbols, infinite initial states, and conditional operators. These results relate to Chapman's work as follows:

1. We have solved an open problem stated by Chapman in (Chapman, 1987): whether or not planning is undecidable when the language contains infinitely many constants but the initial state is finite. In particular, our results show that this problem is decidable in the case where the planning operators have no negative preconditions and no delete lists. If the planning operators are allowed to have negative preconditions and/or delete lists, then the problem is undecidable.
2. Chapman's Second Undecidability Theorem states that "planning is undecidable even with a finite initial situation if the action representation is extended to represent actions whose effects are a function of their input situation" (Chapman, 1987), i.e., if the language contains function symbols and infinitely many constants. Our results show that even with a number of additional restrictions, planning is still undecidable.
3. In Chapman's Second Undecidability Theorem, the phrase "actions whose effects are a function of their input situation" has been thought by some researchers (Peot, 1992; Erol *et al.*, 1992c) to refer to conditional operators. However, this is an incorrect interpretation. First, as we mentioned above, our decidability and complexity results are unaffected by whether or not conditional operators are allowed. Second, a careful examination of Chapman's proof makes it clear that he is referring to the case where the planning operators are allowed to contain function symbols. Our results show that if function symbols are allowed, then even if there are only finitely many constant symbols, then PLAN EXISTENCE is undecidable. Thus, our results subsume the Second Undecidability Theorem.

Complexity

Summary.

Based on various syntactic criteria on what planning operators are allowed to look like, we have developed a comprehensive theory of the complexity of planning. The results are summarized in Table 2; for details see (Erol *et al.*, 1992a). When there are no function symbols and only finitely many constant symbols (so that planning is decidable), the computational complexity varies from constant time to EXPSPACE-complete, depending on the following conditions:

- whether or not we allow delete lists and/or negative preconditions,
- whether or not we restrict the predicates to be propositional (i.e., 0-ary),

Table 2: Complexity of domain-independent planning. These results are independent of whether conditional operators are allowed.

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 finitely many constant symbols)	given in the input	yes	yes/no	EXPSPACE-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no$^{\alpha}$	PSPACE-complete	PSPACE-complete
	fixed in advance	yes	yes/no	PSPACE $^{\gamma}$	PSPACE $^{\gamma}$
		no	yes	NP $^{\gamma}$	NP $^{\gamma}$
			no	P	NP $^{\gamma}$
			no$^{\alpha}$	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes/no	PSPACE-complete$^{\delta}$	PSPACE-complete
		no	yes	NP-complete$^{\delta}$	NP-complete
			no	P $^{\delta}$	NP-complete
			no$^{\alpha}$/no$^{\beta}$	NLOGSPACE-comp.	NP-complete
	fixed in advance	yes/no	yes/no	constant time	constant time

$^{\alpha}$No operator has more than one precondition.
$^{\beta}$Every operator with more than one precondition is the composition of other operators.
$^{\gamma}$With PSPACE- or NP-completeness for some sets of operators.
$^{\delta}$Results due to Bylander (1991).

- whether we fix the planning operators in advance, or give them as part of the input.

The presence or absence of conditional operators does not affect these results.

Discussion.

Examination of Table 2 reveals several interesting properties:

1. Comparing the complexity of PLAN EXISTENCE in 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 of PLAN LENGTH, finding the plan is easier than in the worst case of 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 is because what makes the problem hard is how to handle *enabling-condition interactions*. Enabling-condition interactions are discussed in more detail in (Gupta and Nau, 1992), but the basic idea is that a sequence of actions that achieves one subgoal might also achieve other subgoals or make it easier to achieve them. Although such interactions will not affect PLAN EXISTENCE, they will affect PLAN LENGTH, because they make it possible to produce a shorter plan. It is not possible to detect and reason about these interactions if we plan for the subgoals independently; instead, we have to consider all possible operator choices and orderings, making PLAN LENGTH NP-hard.

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.

Below, we summarize how and why our parameters affect the complexity of planning:

- If no restrictions are put on **P**, any operator instance might need to appear many times in the same plan, forcing us to search through all the states, which are double exponential in number. Since the size of any state is at most exponential, PLAN EXISTENCE can be solved in EXPSPACE.
- If the planning operators are restricted to have no delete lists, then any predicate instance asserted remains true throughout the plan, hence no operator instance needs to appear in the same plan twice. Since the number of operator instances is exponential, this reduces the complexity of PLAN EXISTENCE to NEXPTIME.
- If the planning operators are further restricted to have no negative preconditions, then we get the nice property that no operator clobbers another. Thus the order of the operators in the plan does not matter, and the complexity of PLAN EXISTENCE reduces to EXPTIME.
- Inspite of the restrictions above, PLAN LENGTH remains NEXPTIME. Since we try to find a plan of length at most k, which operator instances we choose, and how we order them makes a difference.
- If each planning operator is restricted to have at most one precondtion, then we can do backward search, and since each operator has at most one precondition, the number of the subgoals does not increase. Thus both PLAN EXISTENCE and PLAN LENGTH with these restrictions can be solved in PSPACE.
- The previous arguments also hold for propositional planning, with the exception of the anomaly in the unrestricted case for PLAN LENGTH, which we have discussed before. As a result of restricting predicates to be 0-ary, the number of operator instances, the size of states reduce to polynomial from exponential, hence in general, the complexity results for propositional planning are one level lower than the complexity results with datalog operators.

Related Work

Planning.

Bylander (1991; 1992) has done several studies on the complexity of propositional planning. We have stated some of his results in and Table 2. More recently, he has studied the complexity of propositional planning extended to allow a limited amount of inference in the domain theory (Bylander, 1992). His complexity results for this case range from polyomial time to PSPACE-complete.

Chapman was the first to study issues relating to the undecidability of planning; we have discussed his work in detail in the "Decidability and Undecidability section.

Backstrom and Klein (1991) found a class of planning problems called SAS-PUBS, for which planning can be done in polynomial time. Their planning formalism is somewhat different from ours: they make use of *state variables* that take values from a finite set, and consider a planning state to be an assignment of values to these state variables. Since they restrict each state variable to have a domain of exactly two values, we can consider each state variable to be a proposition; thus, in effect they are doing propositional planning. However, their operators have further restrictions: they restrict each operator to change at most one state variable, and do not allow more than one operator to change a state variable to a given value. Their restrictions are so strict that they were unable to find any domains (not even blocks world) that they could represent in their formalization. They tried to overcome this problem by weakening some of their restrictions, making the complexity of their algorithm go to exponential time—but still could not find any reasonable domain. It is not very easy to compare our results with theirs, because we use a different formalism—but we can safely state that we analyze a much broader range of problems, and we require less severe restrictions to get polynomial-time results.

Korf (1987) has pointed out that given certain assumptions, one can reduce exponentially the time required to solve a conjoined-goal planning problem, provided that the individual goals are independent. Yang, Nau, and Hendler (1992) have generalized this, showing that one can still exponentially reduce the time required for planning even if the goals are not independent, provided that only certain kinds of goal interactions are allowed. Under this same set of goal interactions, they have also developed some efficient algorithms for merging plans to achieve multiple goals (Yang *et al.*, 1990; Yang *et al.*, 1992).

Complexity results have been developed for blocks-world planning by Gupta and Nau (1991; 1992) and also by Chenoweth (1991). Gupta and Nau (1991; 1992) have shown that the complexity of blocks-world planning arises not from deleted-condition interactions as was previously thought, but instead from enabling-condition interactions. Their speculations that enabling-condition interactions are important for planning in general seem to be corroborated by some of our results, as discussed above.

Temporal Projection.

Another problem that is closely related to planning is the problem of temporal projection, or what Chapman calls the "modal truth" of an atom (Chapman, 1987). Given an atom a, an initial state S_0, and a partially ordered set of actions P, the question is whether a is necessarily/possibly true after execution of P. This question is especially important in partial-order planners such as NOAH (Sacerdoti, 1990), NONLIN (Tate, 1977), and SIPE (Wilkins, 1990). For ex-

ample, McDermott (1991) says "unfortunately, partial orders have a big problem, that there is no way of deciding what is true for sure before a step without considering all possible step sequences consistent with the current partial order," and Pednault (1988) also expresses similar sentiments.

One problem is what it means for a to be necessarily true if not all total orderings of P are executable. Chapman (Chapman, 1987) assumes that a is necessarily true after executing P only if every total ordering of P is both executable and achieves a; and in return, he comes up with a polynomial-time algorithm for determining the necessary truth of a. However, his algorithm does not work correctly for establishing the possible truth of a (Nau (1993) proves that problem is NP-hard).

Chapman also proves that with conditional planning operators, establishing the necessary truth of a is co-NP-hard; and Dean and Boddy (1988) prove a similar result with a more general notion of conditional planning operators (the same definition we gave in the "Special-Case Definitions" section).[1] Dean and Boddy (1988) also try to come up with approximate solutions for the problem. They present algorithms for computing a subset of the propositions that are necessarily true, and for computing a superset of the propositions that are possibly true. Furthermore, the complexity of these algorithms is polynomial if the number of triples for each operator is bounded with a constant. However, we do not know of any results concerning how close the approximations are.

Conclusion

Although our equivalence between planning and logic programming only holds in certain limited cases, this equivalence has allowed us to transport many decidability and undecidability results from logic programming to planning. Among other things, our results solve an open problem posed by Chapman (1987), and clear up some difficulties with his undecidability theorems. It is not a trivial task to extend this equivalence, because negation has different semantics for logic programming and planning—but it is certainly worth investigating, and we intend to do so in the future.

For those cases where planning is decidable, we have shown how the time complexity varies depending on a wide variety of conditions. Among other things, our results suggest that enabling-condition interactions (first described by Gupta and Nau (1992)) are probably just as important for planning as the better-known deleted-condition interactions. We have also been able to show that conditional operators do not affect the complexity of planning.

Most theoretical studies of planning have been confined to planning with STRIPS-style operators—but most of the practical work on AI planning systems for the last fifteen years has been based on hierarchical task-network (HTN) decomposition (Sacerdoti, 1990; Tate, 1977; Wilkins, 1990) as opposed to STRIPS systems. Thus, much of the current practice in AI planning lacks a clear theoretical basis. For our future work, we are beginning to develop a formalization of HTN planning, which we hope will enable us to correctly define, explicate, and analyze the properties of HTN planning systems (Erol *et al.*, 1993).

[1] In both cases, they state that the problem is NP-hard, but their proofs establish co-NP-hardness instead.

Acknowledgement

We appreciate the useful comments we received from Tom Bylander and Jim Hendler.

References

Backstrom, Christer and Klein, Inger 1991. Planning in polynomial time: the sas-pubs class. *Computational Intelligence* 7.

Bezem, M. zem. Characterizing termination of logic programs with level mappings. In Lusk, E. and Overbeek, R., editors zem, *Proc. 1989 North American Conf. on Logic Programming.* MIT Press. 69–80.

Blair, H.A. 1989. Canonical conservative extensions of logic program completions. In *Proceedings of the 4th IEEE Symposium on Logic Programming.*

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

Bylander, Tom 1992. Complexity results for extended planning. In *Proc. First International Conference on AI Planning Systems.*

Chapman, David 1987. Planning for conjunctive goals. *Artificial Intelligence* 32:333–379.

Chenoweth, Stephen V. 1991. On the NP-hardness of blocks world. In *AAAI-91: Proc. Ninth National Conf. Artificial Intelligence.* 623–628.

Dean, Thomas and Boddy, Mark 1988. Reasoning about partially ordered events. *Artificial Intelligence* 36:375–399.

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

Erol, K.; Nau, D.; and Subrahmanian, V. S. 1992b. On the complexity of domain-independent planning. In *Proc. AAAI-92.* 381–386.

Erol, K.; Nau, D.; and Subrahmanian, V. S. 1992c. When is planning decidable? In *Proc. First Internat. Conf. AI Planning Systems.* 222–227.

Erol, K.; Nau, D. S.; and Hendler, J. 1993. Toward a general framework for hierarchical task-network planning. In *AAAI Spring Symposium.* To appear.

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, Naresh and Nau, Dana S. 1992. On the complexity of blocks-world planning. *Artificial Intelligence* 56(2-3):223–254.

Korf, Richard 1987. Planning as search: A quantitative approach. *Artificial Intelligence* 33(1):65–88.

Lifschitz, Vladimir 1990. On the semantics of strips. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning.* Morgan Kaufman. 523–530.

Lloyd, J.W. 1987. *Foundations of Logic Programming.* Symbolic Computation. Springer-Verlag, second, extended edition edition.

McAllester, David and Rosenblitt, David 1991. Systematic nonlinear planning. In *AAAI-91.* 634–639.

McDermott, Drew 1991. Regression planning. *International Journal of Intelligent Systems* 6:357–416.

Nau, D. 1993. On the complexity of possible truth. In *AAAI Spring Symposium.* To appear.

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

Pednault, Edwin 1988. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence* 4:356–372.

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

Plaisted, David 1984. Complete problems in the first-order predicate calculus. *Journal of Computer and System Sciences* 29:8–35.

Sacerdoti, Earl D. 1990. The nonlinear nature of plans. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning.* Morgan Kaufman. 162–170. Originally appeared in *Proc. IJCAI-75,* pp. 206-214.

Sebelik, J. and Stepanek, P. 1980. Horn clause programs for recursive functions. In *Logic Programming.* Academic Press. 325–340.

Shoenfield, J. 1967. *Mathematical Logic.* Academic Press.

Stockmeyer, L. J. and Chandra, A. K. 1978. Provably difficult combinatorial games. Technical Report RC 6957, IBM T. J. Watson Research Ctr., Yorktown Heights, NY.

Tate, Austin 1977. Generating project networks. In *Proc. 5th International Joint Conf. Artificial Intelligence.*

Vardi, Moshe 1982. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing,* San Francisco, CA. 137–146.

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.

Wilkins, David E. 1990. Domain-independent planning: Representation and plan generation. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning.* Morgan Kaufman. 319–335. Originally appeared in *Artificial Intelligence* 22(3), April 1984.

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* 8(2):648–676.