An Investigation into the Computational Complexity of Planning in Nondeterministic and Information-Incomplete Domains

P@trik Haslum
pahas@ida.liu.se

January 16, 1998

Abstract

Though the planning problem, including the computational complexity of planning, has been long studied in AI, the problem of planning with incomplete information has recieved relatively little attention. Planning with incomplete information may mean any one of a number of different things; It may mean that certain facts of the initial state are not known, that plans may contain sensing operations, or that operators can have probabilistic or nondeterministic effects. The computational complexity of this problem has been even less investigated. We examine the complexity of *contingent* planning, constructing plans that will work in every forseable case. Varying restrictions on the operator and state description incompletness, ranging from totaly unconstrained to allowing only incomplete initial state and determinstic contex-independent operators, result in complexities ranging from EXPSPACE-complete to PSPACE-complete, which is also the known complexity of propositional planning with determinism and complete information.

Chapter 1

Domain-Independent Planning

The *plan synthesis problem* or *planning problem* is a formalised version of the problem *given a description of the initial state, the desired goal state (or states) and a set of operators, find a sequence of operator instances such that the state resulting from execution of the entire sequence is a goal state.* Planning is sometimes refered to as *domain independent planning*, since the domain, as described by the available operators and their effects, is a part of the problem instance. Even though the problem has been much studied since the beginnings of artificial intelligence [13, 11, 6, 27, 29, 30, 19, 26, 22, 23, 21, 20, 24, 2], the most common formalisation is still the same as was used in the STRIPS system [11]. The STRIPS formalism requires operators to have conjunctive preconditions and deterministic context-independent effects. A later development, ADL [23], extends STRIPS to incorporate context-dependent effects and preconditions including limited disjunctions. Both do however represent domains and operators in first order predicate logic. When the domain contains an infinite number of ground terms, the planning problem becomes undecidable [10]. First order predicate logic over a finite domain of ground terms is equivalent to propositional logic, but allows for more compact encodings.

Though most research into the planning problem has focused only on the case of *complete information*, i.e. when all relevant information about the initial state and the effects of executing operators is available to the planner, several approaches to dealing with more complex and realistic cases have been presented. These fall roughly in three categories; *conditional*, *contingent* and *probabilistic*[1]. In addition, some classical planners can deal with the limited

[1]There exists some confusion over names here; conditional planning is sometimes used to refer to complete-information planning with context-dependent operators, and contingent planning sometimes means planning with sensing actions. We will consistently use the termi-

form of incompleteness that we refer to as $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ below [1, 6].

The conditional planning approach assumes the plan executor to be equipped with sensors, so that all relevant information will be available when the plan is executed. Conditional planners that construct branching plans, involving choices to be made based on observations made at execution time [30, 19, 25] A contingent planner constructs instead plans for every contingency, meaning plans that achieve their goal no matter what circumstances they encounter. The two approaches may be combined by planning with sensing operations and branching plans when such operators are available, and planning for contingencies when they are not [7, 12]. Probabilistic planners require probabilities to be assigned to all possible outcomes of any uncertanity, and construct plans that maximise the probability of achieving their goals [17]. Probabilistic measures have also been integrated in conditional and contingent planning [8].

1.1 Computational Complexity of Planning

The computational complexity of the planning problem in the complete information case has been investigated rather extensively in the last decade [3, 14, 10, 9, 1].

Bylander [3] shows that plan existance for propositional deterministic context-independent planning (STRIPS) with complete information is PSPACE-complete, even under additional restrictions on operators. PSPACE-completeness extends naturally to any deterministic operator such that the state resulting from application of the operator is computable in polynomial space, which is true of for instance the context-dependent operators in ADL. Erol *et al.* [10] present a detailed analysis of decidability and complexity for planning with first order state and operator descriptions. With deterministic context-independent operators (again, STRIPS) and predicate descriptions over a finite set of ground terms, plan existance is EXPSPACE-complete, and when operators are restricted to have only positive effects, i.e. an operator can not make a fact previously known to be true false, and preconditions do not contain any negated fact, plan existance is EXPTIME-complete. When the set of ground terms is infinite, plan existence is undecidable. Bäckström and Jonsson [16] present a STRIPS-like formalism with additional structural restrictions on problem instances, in which planning can be tractable.

There appears, however, only to be one study of the complexity of planning with incomplete information; Littman [18] shows propositional probabilistic planning with context-dependent operators to be EXPTIME-complete. This contrasts interestingly with our result of Chapter 2, that planning with nondeterministic context-dependent operators is EXPSPACE-complete.

nology introduced in this section.

Operator restrictions	Initial state		
	Complete	$\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$	Arbitrary
Context-independent deterministic	PSPACE-complete[I]	PSPACE-complete[II]	PSPACE-complete[III]
Context-dependent deterministic	PSPACE-complete	PSPACE-complete[II]	?
Context-independent non-deterministic	EXPTIME-hard[IV]		
Context-dependent nondeterministic	EXPSPACE-complete		

[I] This result is due to Bylander [3].
[II] When operators are $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$, as detailed in Chapter 5
[III] When operators are interpreted in the opaque semantics.
[IV] When operators are interpreted in the transparent semantics.

Table 1.1: Cases Considered and Results Found

1.2 Terminology and Contents of this Report

In this report, we investigate the computational complexity of contingent planning in several different cases, as shown in table 1.1.

We differentiate between two kinds of uncertanity in state descriptions. By a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ *state*, we mean a state where the value of every propositional variable is either $\mathcal{T}$rue, $\mathcal{F}$alse or $\mathcal{U}$nknown, but where there are no interdependencies between the unknown propositions, i.e. every assignment of $\mathcal{T}$ and $\mathcal{F}$ to any unknown proposition is possible. An operator is $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ if it produces a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ state whenever it is executed in a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ state.

When we consider incompletely specified states, we need to reason about the result of executing an operator when the truth of its precondition is not known, and this requires defining the result of executing an operator in a state where its precondition is false. There are at least two ways to do this; executing an operator when its precondition does not hold may have no effect at all, or may result in an undefined state, and plan failure. For context-dependent operators, the first view is natural; when no branch is enabled, no branch is executed. For context-independent operators, the second view seems more appropriate, as the first actually encodes context dependency, although limited to two complementary cases. Both varieties have appeared in the planning litterature, but due to the assumption that all relevant information is or will be available when the operator is executed, the issue has not been important and thus not much discussed. However, the context-dependency of operators appears to be a crucial factor in deciding the complexity of the planning problem, which makes the question motivated.

We approach the problem by defining two different operator semantics; the *transparent*, in which an operator has no effect when executed in a state where its precondition is false, and the *opaque* in which an operator executed in a state where its precondition does not hold results in an undefined state. We will be carefull to point out in which semantic we are interpreting operators whenever it makes a difference. Finaly, we will only consider propositional planning, as we wish to show that the complexities are inherent in the problems and not merely a consequence of compact encodings.

The following basic definitions are used throughout the report.

Definition 1.1 By a *planning scenario* we mean a tuple $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$, where the *domain* $\mathcal{D}$ is a set of propositional symbols, $\mathcal{O}$ is a set of operators and I and G are the initial state and goal state descriptions, respectively. The definition of operators varies, depending on the case under consideration.

A *state description*, or *state description formula*, is a boolean combinations of literals over $\mathcal{D}$. A state is represented by the subset of $\mathcal{D}$ consisting of all propositions true in the state. A state description formula σ denotes a set of possible states, as follows.

$$mod(\sigma) \quad = \quad \{\xi \subseteq \mathcal{D} | \xi \models \sigma\}$$

For a set of states Ξ and an operator O we define the semantics of operators in terms of the *successor state set function*, $S(O, \Xi)$, denoting the set of states that may result from applying O in any state $\xi \in \Xi$. The exact definition of S varies with the restrictions placed on operators.

A sequence of operators $O_1; \ldots; O_n$ constitutes a *plan* for scenario $\mathcal{P}$ iff $S(O_n, S(O_{n-1}, \ldots S(O_1, mod(I)) \ldots)) \models G$.$\Box$

Chapter 2

The Worst Case: Nondeterminism with Context Dependency

In this chapter we examine the complexity of contingent planning with operators that can have nondeterministic as well as context-dependent effects. There may also be incompleteness in the initial state description, but this is of no consequence as a nondeterministc operator can destroy any information present in any state.

The next definition (2.1) introduces the operators, which we call *actions* (for want of a better word) and their semantics. The rest of the chapter is devoted to determining the complexity of plan existance for scenarios containing actions; theorem 2.7 shows hardness, while theorem 2.9 shows inclusion. In theorem 2.10 the results are combined to show that the problem is EXPSPACE-complete. In the process, we also show a bound on the length of plans (lemma 2.8), and we end the chapter by presenting a scenario such that the only plan for the scenario does actually meet this bound.

Definition 2.1 A *reassignment* is an expression $x{:=}v$, where $x{\in}\mathcal{D}$ and $v{\in}\{\mathcal{T},\mathcal{F}\}$. A nondeterministic and context dependent operator, called an *action*, is a boolean combination of literals over $\mathcal{D}$ and reassignments.

Given a set of possible states Ξ and an action A, the successor state function is defined inductively as

$$\begin{aligned}
S(x,\Xi) &= \{\xi\in\Xi | x{\in}\xi\} \\
S(\neg x,\Xi) &= \{\xi\in\Xi | x{\notin}\xi\} \\
S(x{:=}\mathcal{T},\Xi) &= \{\xi' | \xi\in\Xi \wedge \xi' = \xi\cup\{x\}\} \\
S(x{:=}\mathcal{F},\Xi) &= \{\xi' | \xi\in\Xi \wedge \xi' = \xi-\{x\}\}
\end{aligned}$$

$$\begin{aligned} S(\alpha \wedge \beta, \Xi) &= S(\alpha, S(\beta, \Xi)) \\ S(\alpha \vee \beta, \Xi) &= S(\alpha, \Xi) \cup S(\beta, \Xi) \end{aligned}$$

A boolean combination of only literals is called a *condition*. A boolean combination of reassignments is called an *effect*. A *conditional effect* is a formula of the form $c \rightarrow e$, where c is a condition and e a boolean combination of effects and conditional effects. For the semantics to reflect the intuitive notion of conditional effects we interpret $p \rightarrow e$ as $\neg p \vee (p \wedge e)$. An action is restricted to be a conjunction of conditional effects.$\Box$

We now turn to showing that the problem of deciding plan existance for a scenario $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$, where $\mathcal{D}$ is propositional and $\mathcal{O}$ contains actions, is EXPSPACE-hard. This will be done through a reduction from the EXPSPACE-complete problem of deciding universality for regular expressions with exponentiation, via the same problem for a new kind of nondeterministic automaton.

For definitions of regular expressions and various finite automata, see for instance [15].

Definition 2.2 A *regular expression with exponentiation* is either

(i) a regular expression r, or

(ii) r^n, where r is a regular expression with exponentiation and n is a number written in binary.

The set of strings denoted by r^n are those that consist of n consecutive substrings such that each substring is in the set denoted by r, i.e. $L(r^n) = \{w^n | w \in L(r)\}$$\Box$

Lemma 2.3 The problem of determining if a regular expression with exponentiation is *universal*, i.e. denotes all strings over its alphabet, is EXPSPACE-complete.
Proof: [15], theorems 13.14 and 13.15$\Box$

To clarify the reduction, we will first convert the regular expression into a new kind of nondeterministic finite automaton, augmented with counters of bounded capacity.

Definition 2.4 A *nondeterministic finite automaton with counters*, or NFAC, is a nondeterministic finite automaton augmented with a set of *counters* C. Each counter c is associated with a set of states $states_c$, a natural number $limit_c$, and the two states $loop_c$ and $continue_c$. Initially, all counters are set to 0. Whenever the automaton enters any state $q \in states_c$ two things happen.

(i) The counter c is incremented by one.

(*ii*) If $c < limit_c$, the automaton immediately moves to state $loop_c$, and if $c \geqslant limit_c$ the automaton instead moves to state $continue_c$. This move is not nondeterministic, it happens automatically and consumes no input.

The automaton terminates as usual when there is no more input, and accepts if there is some sequence of choices that makes it reach a final state.□

Lemma 2.5 For any regular expression r, there exists a NFA accepting the language $L(r)$. This automaton has a number of states that is linear in the number of operators in r
Proof: Follows from theorems 2.3 and 2.2 in [15].□

Lemma 2.6 For any regular expression with exponentiation r, there exists a NFAC accepting the language $L(r)$. This automaton also has a number of states that is linear in the number of operators in r.
Proof: Structural induction on r. If r is atomic it can not contain exponentiation and is therefore covered by 2.5. Assume this holds for expressions r_1 and r_2. The three cases $r = r_1 r_2$, $r = r_1 + r_2$ and $r = r_1^*$ are identical to the proof of 2.5.

If $r = r_1^n$, there exists by assumption a NFAC $M_1 =< Q, \Sigma, \delta, q_1, F, C >$ accepting $L(r_1)$. Construct $M =< Q \cup \{f_0\}, \Sigma, \delta, q_1, \{f_0\}, C \cup \{c\} >$ by setting $states_c = F$, $limit_c = n$, $loop_c = q_1$ and $continue_c = f_0$.

Any string denoted by r will consist of n copies of r_1. M starts in state q_1, and since M_1 accepts $L(r_1)$ ends in state f_1 only after having read one copy of r_1. When, and only when, M enters f_1 is c incremented, and therefore M will reach its final state f_0 only after having read n copies of r_1. Any string s accepted by M will consist of n consecutive substrings accepted by M_1 and therefore in $L(r_1)$, so $s \in L(r)$.

Since the construction adds a constant number of states for each operator in r, the total number of states is bounded by some linear function in the size of r.□

Theorem 2.7 Deciding plan existance for a scenario $\mathcal{P}$ with propositional domain and containing actions, is EXPSPACE-hard.
Proof: We will show this by a polynomial reduction from the universality problem for regular expressions with exponentiation, via the same problem for NFACs. More specifically, we will show that for any regular expression with exponentiation, r, there exists a set of actions $\mathcal{O}$ and partial state descriptions I and G, polynomially bounded in the size of r, such that there exists a plan for $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$ iff there exists some string in Σ^* not accepted by the corresponding NFAC $M =< Q, \Sigma, \delta, q_0, \{q_f\}, C >$, where Σ is assumed to include only characters actually present in r. This shows plan existance to be co-EXPSPACE-hard, which is equivalent to EXPSPACE-hard.

We model the state of M and its counters as the state of the scenario, and the transition function δ as the behavior of actions. Assume the states in Q are numbered, and let the propositions $s_0 \ldots s_{log(n)}$ denote the current state. We write $s = m$ for the formula $\bigwedge b_i$ where b_i is s_i if the ith bit of m in binary is 1 and $\neg s_i$ if the ith bit of m in binary is 0, and $s{:=}m$ for the reassignment formula $\bigwedge s_i{:=}m_i$, where m_i is $\mathcal{T}$ if the ith bit of m in binary is 1 and $\mathcal{F}$ if the ith bit of m in binary is 0.

For each counter $c \in C$ with $limit_c = n$, let the propositions $c_0 \ldots c_{log(n)}$ represent the value of c in binary. We will use the same notation as for the state propositions, and also write $c{:=}c+1$ for the conditional reassignment

$$\begin{array}{rcl} \neg c_0 & \rightarrow & c_0{:=}\mathcal{T} \wedge \\ \neg c_1 \wedge c_0 & \rightarrow & c_1{:=}\mathcal{T} \wedge c_0{:=}\mathcal{F} \wedge \\ & \ldots & \wedge \\ \neg c_{log(n)} \wedge c_{log(n)-1} \wedge \ldots \wedge c_0 & \rightarrow & c_{log(n)}{:=}\mathcal{T} \wedge c_{log(n)-1}{:=}\mathcal{F} \wedge \ldots \wedge c_0{:=}\mathcal{F} \end{array}$$

For each character $\tau \in \Sigma$, define the action

$$A_\tau \Rrightarrow \bigwedge (s = i) \rightarrow (\bigvee_{q_j \in \delta(\tau, q_i)} s{:=}j)$$

For $q_j \in state_c$, $s{:=}j$ is replaced by

$$\begin{array}{rcl} c < limit_c - 1 & \rightarrow & s{:=}loop_c \wedge c{:=}c+1 \wedge \\ c = limit_c - 1 & \rightarrow & s{:=}continue_c \wedge c{:=}c+1 \end{array}$$

The initial state I is $s = 0 \wedge (\bigwedge_{c \in C} c = 0)$ and the goal G is $\bigwedge_{q_i \in F} \neg s = i$.

If there exists a plan for $\mathcal{P}$, this plan corresponds to a sequence of characters that takes M from its initial state to halt in a non-accepting state. Since M does not accept that string it is not in $L(r)$, and r is not universal.

The initial state and goal descriptions are bounded by

$$log_2(|Q|) + \Sigma_{c \in C} log_2(limit_c) \text{ and } log_2(|Q|) \cdot |F|$$

respectively, both polynomial in the size of r. The number of actions is bounded by $|\Sigma|$, which is at most linear in the size of r, and each action will have no more than $|Q|$ branches. The effect formulas of each branch are bounded by

$$log_2(limit_{c_1}) + \ldots + log_2(limit_{c_k})$$

which is also linear in the size of r. Thus, the size of the action set, and therefore the entire planning scenario, is bounded by $c \cdot r^3$, for some constant c.□

Next, we show that propositional planning with actions is in EXPSPACE. To do this, we need the following bound on plan length.

Lemma 2.8 Let $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$ be a planning scenario, and $|\mathcal{D}| = n$. If there exists a plan for $\mathcal{P}$, there exists a plan consisting of no more than 2^{2^n} steps.
Proof: Since each state description consists of at most n propositions, there are 2^n possible states, but since the planner may have incomplete information about initial and goal states, as well as the states resulting from execution of non-deterministic actions, the state space consists, from the planners point of view, of 2^{2^n} possible *state-sets*. Any action, including non-deterministic actions and actions executed in incompletely known states, is a deterministic transition in the state-set space. If there exists a plan, it may be viewed as a path through the state-set space, and if this path contains cycles these can be removed without affecting the plan. Therefore, no more than 2^{2^n} steps are necessary.□

Theorem 2.9 Deciding plan existance for a scenario $\mathcal{P}$ with propositional domain and actions is in EXPSPACE.
Proof: By nondeterministic algorithm, as follows. Let $\mu = mod(I)$ and $i = 0$ initially.

(i) Nondeterministically select an action A and let $\mu = S(A, \mu)$. Let $i = i+1$.

(ii) If $\mu \models G$, return $\mathcal{T}$. If $i > 2^{2^n}$, return $\mathcal{F}$.

(iii) Repeat from (i).

By definition, the selected sequence of actions is a plan iff $\mu \models G$ at some stage and by lemma 2.8, if no plan has been found in 2^{2^n} steps there exists no plan at all. A single model can be represented by n propositional variables, and thus the set of models can be represented by 2^n variables. Since i can be represented by 2^n bits, the algorithm requires no more than exponential space.□

Summarising theorems 2.7 and 2.9, we have the following.

Theorem 2.10 The plan existance problem for a scenario $\mathcal{P}$ with propositional domain and actions is EXPSPACE-complete.
Proof: Theorem 2.7 shows hardness, theorem 2.9 shows inclusion.□

As mentioned, we end the chapter by presenting a scenario, or rather a class of scenarios, such that the only existing plan does actually meet the bound shown in lemma 2.8.

Let $\mathcal{D} = \{p_1, \ldots, p_n\}$. The powerset of the set of states over $\mathcal{D}$ ordered by set inclusion forms a lattice with the state set corresponding to the least informative state description formula, $\mathcal{T}$, at the top and successively more information contained in the state sets below. We construct a scenario that traverses this lattice by levels, proceeding to the next level of specification only after all state sets on the current level have been visited. Thus, the initial state description is $I = \mathcal{T}$ and the goal formula a conjunction of one literal for each proposition $p_i \in \mathcal{D}$, for instance $p_1 \wedge \ldots \wedge p_n$.

Writing the general actions for the scenario with n propositions would be too complicated; we only present them for the case of $n = 2$.

$$
\begin{array}{rcl}
A_0 & = & \mathcal{T} \rightarrow p_1{:=}\mathcal{F} \vee p_2{:=}\mathcal{F} \\
A_{(1,1)} & = & \neg p_1 \vee \neg p_2 \rightarrow p_1{:=}\mathcal{T} \vee p_2{:=}\mathcal{F} \\
A_{(1,2)} & = & p_1 \vee \neg p_2 \rightarrow p_1{:=}\mathcal{F} \vee p_2{:=}\mathcal{T} \\
A_{(1,3)} & = & \neg p_1 \vee p_2 \rightarrow p_1{:=}\mathcal{T} \vee p_2{:=}\mathcal{T} \\
A_{(1,4)} & = & p_1 \vee p_2 \rightarrow p_1{:=}\mathcal{T} \\
A_{(2,1)} & = & p_1 \rightarrow p_1{:=}\mathcal{F} \wedge (p_2{:=}\mathcal{T} \vee p_2{:=}\mathcal{F}) \\
A_{(2,2)} & = & \neg p_1 \rightarrow (p_1{:=}\mathcal{T} \vee p_1{:=}\mathcal{F}) \wedge p_2{:=}\mathcal{T} \\
A_{(2,3)} & = & p_2 \rightarrow (p_1{:=}\mathcal{T} \vee p_1{:=}\mathcal{F}) \wedge p_2{:=}\mathcal{F} \\
A_{(2,4)} & = & \neg p_2 \rightarrow (p_1{:=}\mathcal{F} \wedge p_2{:=}\mathcal{F}) \vee (p_1{:=}\mathcal{T} \wedge p_2{:=}\mathcal{T}) \\
A_{(2,5)} & = & (p_1 \wedge p_2) \vee (\neg p_1 \wedge \neg p_2) \rightarrow (p_1{:=}\mathcal{F} \wedge p_2{:=}\mathcal{T}) \vee (p_1{:=}\mathcal{T} \wedge p_2{:=}\mathcal{F}) \\
A_{(2,6)} & = & (p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2) \rightarrow p_1{:=}\mathcal{F} \wedge p_2{:=}\mathcal{F} \\
A_{(3,1)} & = & \neg p_1 \wedge \neg p_2 \rightarrow p_1{:=}\mathcal{T} \\
A_{(3,2)} & = & p_1 \wedge \neg p_2 \rightarrow p_1{:=}\mathcal{F} \wedge p_2{:=}\mathcal{T} \\
A_{(3,3)} & = & \neg p_1 \wedge p_2 \rightarrow p_1{:=}\mathcal{T}
\end{array}
$$

These actions, in the sequence listed above, constitute a plan for $\mathcal{P}$, when $G = p_1 \wedge p_2$.

Note that the number of operators in $\mathcal{P}$ is also exponential in n. Therefore, we can not realy say that the plan is of length twice exponential in the size of $\mathcal{P}$, only in the size of the domain $\mathcal{D}$. Wether a scenario of length twice exponential in the size of $\mathcal{P}$ can be constructed is an unresolved question.

Chapter 3

Intermediate Cases: Nondeterminism without Context Dependency

In this chapter we study the complexity of planning with operators that have nondeterministic, but not context-dependent, effects. As operators can have nondeterministic effects, incompleteness of the initial state is irrelevant.

However, the incompleteness of information implies that the status of the precondition of an operator may at some point in a plan be undetermined. This raises the question concerning the admissibility of operators, and results in two different semantics which we term *transparent* and *opaque*. In the transparent semantics, an operator executed in a state set where its precondition does not hold has no effect, and the successor state set is unchanged. In the opaque semantics, an operator executed in a state set where its precondition does not necessarily hold results in an undefined state, and in plan failure.

We begin by defining the operators and their semantics, and proceed to show a hardness result for the transparent case (theorem 3.4).

Definition 3.1 A *nondeterministic STRIPS operator* is a list $(P, E_1, \ldots, E_n)$. The precondition P is a boolean combination of literals and each possible effect E_i is a tuple (A_i, D_i) where A_i is the set of propositions to be added and D_i the set of propositions to be deleted.

We define two successor state functions, yielding different semantics for the nondeterministic STRIPS operator. Given a set of possible states Ξ and a nondeterministic STRIPS operator O, the *transparent* semantics of O is defined by

$$\begin{aligned} S^T(O,\Xi) &= \{\xi' | \xi \in \Xi \wedge \xi \models P_O \wedge \exists E_i \in \alpha : \xi' = (\xi - D_i) \cup A_i\} \cup \\ & \quad \{\xi \in \Xi | \xi \not\models P_O\} \end{aligned}$$

and the *opaque* semantics of O is defined by

$$S^O(O,\Xi) \quad = \quad \{\xi' | \xi \in \Xi \wedge \exists E_i{\in}\alpha : \xi' = (\xi - D_i) \cup A_i\}$$

if $\Xi{\models}P_O$ and undefined otherwise.□

Transparent operators can encode a limited form of context dependence, with only two complementary cases.

Next, we show that under the transparent semantics the plan existence problem for scenarios with nondeterministic STRIPS operators is EXPTIME-hard. This is done by reduction from the acceptance problem for alternating Turing machines [4, 5].

Definition 3.2 (Chandra and Stockmeyer [5])
An *alternating Turing machine* (ATM) is a tuple $M = (Q, \Gamma, \Sigma, \#, \delta, q_0, U)$, where Q is the set of states, Γ the tape alphabet, Σ the input alphabet ($\Sigma \subseteq \Gamma$), $\#$ the *blank symbol* ($\# \in \Gamma$, $\# \notin \Sigma$), δ the next state relation ($\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, S, R\})$, q_0 the initial state ($q_0 \in Q$) and U the set of *universal states* ($U \subseteq Q$). The states in $Q - U$ are called *existential states*.

A *configuration* of M is a tuple (q, γ, i), where $q \in Q$ is the state of the configuration, $\gamma \in \Gamma^*$ specifices the contents of the tape up to the rightmost non-blank symbol and i represents the position of the tape head. The set of all configurations is denoted C_M. A configuration $C' = (q', \gamma', i')$ is *single-step reachable* from another configuration $C = (q, \gamma, i)$, written $C{\vdash_M}C'$, iff $(q', \gamma'(i), d) \in \delta(q, \gamma(i))$ and $i' = i - 1$ if $d = L$, $i' = i + 1$ if $d = R$ and $i' = i$ if $d = S$. A configuration C is *halting* iff there are no configurations single-step reachable from C.

A trace of an ATM is a set $T \subseteq C_M{\times}\mathbf{N}$ satisfying the following two conditions.

(i) If $C = ((q, \gamma, i), k) \in T$ and q is a universal state then $(C', k - 1) \in T$ for all configurations C' such that $C{\vdash_M}C'$.

(ii) If $C = ((q, \gamma, i), k) \in T$ and q is an existential state then there exists some configuration C' such that $(C', k - 1) \in T$ and $C{\vdash_M}C'$.

M accepts $w \in \Sigma^*$ iff there exists a trace T of M such that $((q_0, w, 1), k) \in T$ for some k. By this definition, M accepts iff it halts in a universal state.□

An ATM is *linearly bounded* iff it uses no more than $|w| + 1$ space on input w. A *standard linear ATM M* is a linearly bounded ATM that satisfies the following additional constraints:

(i) M never scans tape cell $|w| + 2$.

(ii) The initial state q_0 is existential.

(iii) If $C \vdash_M C'$ then the state of C is existential iff the state of C' is universal.

Theorem 3.3 $\mathrm{DTIME}(2^n) = \{\mathrm{L}(M) \mid M$ is a standard linear ATM$\}$.
Proof: Chandra and Stockmeyer [5].□

Theorem 3.4 Deciding plan existance for propositional planning scenarios with nondeterministic STRIPS operators, interpreted in the transparent semantics, is EXPTIME-hard.
Proof: By reduction from the acceptance problem for a standard linear ATM. Let M be such a machine and w its input. The planning scenario is constructed by representing configurations as states and two-step transitions as operators, as described below.

Let s_q, for each $q \in Q$, represent the state of M so that s_q is $\mathcal{T}$ iff the state of the current configuration is q (since states are assumed to be numbered $0, \ldots, n$, s_i will be taken mean s_q for $q = q_i$. Let t^i_τ, for each $\tau \in \Gamma$ and $i = 1 \ldots |w| + 1$, represent the contents of the tape so that t^i_τ is $\mathcal{T}$ iff tape cell i contains τ. Let p_i, for $i = 1 \ldots |w| + 1$, represent the position of the tape head so that p_i is $\mathcal{T}$ iff M is scanning cell i.

In addition to this there is a proposition called *accept* which will be $\mathcal{T}$ iff M has reached a universal state from which it can not continue.

The operators will represent the two-step transitions possible from the existential states of M, with the transitions from universal states as nondeterministic effects. For each tuple $(q, \tau, q', \tau', d) \in \delta$ such that q is an existential state, and for $i = 1 \ldots |w| + 1$, define the a set of operators $MOVE_\zeta$, where ζ is any suitable index, as follows.

If $d = L$ and $i > 1$ then for each $\tau'' \in \Gamma$ define an operator with precondition

$$P = s_q \wedge p_i \wedge t^i_\tau \wedge t^{i-1}_{\tau''}$$

and one possible effect

$$E_\iota = (\{s_{q''}, t^i_{\tau'}, t^{i-1}_{\tau'''}, p_j\}, \{s_q, p_i, t^i_\tau, t^{i-1}_{\tau''}\})$$

for each tuple $(q', \tau'', q'', \tau''', d') \in \delta$, where $j = i - 2$ if $d' = L$ and $i > 2$, $j = i - 1$ if $d' = S$ and $j = i$ if $d' = R$ (in which case both reassignments of p_i are removed from E_ι as a proposition can not be assigned both $\mathcal{T}$ and $\mathcal{F}$ at the same time). If there are no tuples like this, the operator has only one effect $E = (\{accept\}, \{\})$.

If $d = R$ and $i < |w| + 1$ then for each $\tau'' \in \Gamma$ define an operator with precondition

$$P = s_q \wedge p_i \wedge t^i_\tau \wedge t^{i+1}_{\tau''}$$

and one possible effect

$$E_\iota = (\{s_{q''}, t^i_{\tau'}, t^{i+1}_{\tau'''}, p_j\}, \{s_q, p_i, t^i_\tau, t^{i+1}_{\tau''}\})$$

for each tuple $(q', \tau'', q'', \tau''', d') \in \delta$, where $j = i + 2$ if $d' = R$ and $i < |w|$, $j = i + 1$ if $d' = S$ and $j = i$ if $d' = L$ (in which case both reassignments of p_i are removed from E_ι as above). Again, if there are no tuples like this, the operator has only one effect $E = (\{accept\}, \{\})$.

If $d = S$ define an operator with precondition

$$P = s_q \wedge p_i \wedge t^i_\tau$$

and one possible effect

$$E_\iota = (\{s_{q''}, t^i_{\tau''}, p_j\}, \{s_q, p_i, t^i_\tau\})$$

for each tuple $(q', \tau', q'', \tau'', d') \in \delta$, where $j = i-1$ if $d' = L$ and $i > 1$, $j = i+1$ if $d' = R$ and $i < |w|+1$, and $j = i$ if $d' = S$ (in which case both reassignments of p_i are removed from E_ι). If there are no tuples like this, the operator has only one $E = (\{accept\}, \{\})$.

The initial state (of the planning scenario, not of M) describes the initial configuration of M with input w, represented as described above, conjoined with $\neg accept$. The goal is simply $accept$.

For each existential state q there is an operator representing a move to some state q', which is by the standardness of M universal. This operator sets $accept$ to $\mathcal{T}$ iff there are no possible moves from q' under the resulting configuration, and otherwise nondeterministically selects one of the possible moves into some state q'', which is existential. That configurations are correctly maintained should be clear from the definitions of the operators. Therefore, if there exists a plan for the scenario it corresponds to an accepting trace of M, by numbering the steps in reverse. Conversely, if there exists an accepting trace of M then construct a plan by traversing the trace in reverse order and at each move from an existential state appending to the plan the corresponding operator.

Finaly, the number of propositions is $|Q| + |\Gamma|\cdot(|w| + 1) + |w| + 2$, which is polynomial in the size of (M, w). The number of tuples in δ are at most $r = |Q|^2\cdot|\Gamma|^2\cdot 3$ so the number of operators is bounded by $|\Gamma|\cdot r\cdot(|w| + 1)$, with the number of possible effects per operator bounded by r. Thus, the size of the operator set is also polynomial in the size of (M, w) (allthough a rather large polynomial). From the above reduction and 3.3, it follows that plan existance for a scenario with propositional domain and nondeterministic STRIPS operators, interpreted in the transparent semantics, is EXPTIME-hard.□

Chapter 4

The Lower End: Determinism with Incomplete State Information

In this chapter, we examine the complexity of planning with deterministic, context-independent operators and incompleteness of information about the initial state. The plan existance problem for this class of scenarios turns out to be PSPACE-complete; the same as for the class of scenarios with propositional domain, deterministic context-dependent operators and complete initial state.

As in chapter 3, there are two possible interpretations of the results of executing an action when its preconditions are not satisfied, leading us to define two different semantics; the transparent and the opaque. The complexity analysis will concern only the later.

Definition 4.1 introduces operators and semantics, theorem 4.5 establishes inclusion and theorem 4.6 completeness. With the additional restriction that operator preconditions are on CNF, the inclusion may also be shown using regression; we do this at the end of the chapter.

Definition 4.1 A *deterministic STRIPS operator* (or just STRIPS operator), is a tuple (P, A, D), where P is a boolean combination of literals representing the operators precondition, and A and D are the sets of propositions added and deleted, respectively.

Given a set of possible states Ξ and a STRIPS operator O, the transparent and opaque semantics of O are defined by the successor state functions

$$S^T(O, \Xi) \quad = \quad \{\xi' | \xi \in \Xi \wedge \xi \models P_O \wedge \xi' = (\xi - D_O) \cup A_O\} \cup$$

$$\{\xi \in \Xi | \xi \not\models P_O\}$$

and

$$S^O(O, \Xi) \quad = \quad \{\xi' | \xi \in \Xi \wedge \xi' = (\xi - D_O) \cup A_O\}$$

if $\Xi \models P_O$ and undefined otherwise.□

As is easily seen, the deterministic STRIPS operator is a special case of the nondeterministic STRIPS operator, with only one possible effect. To show inclusion in PSPACE, we need an alternative characterisation of the state set resulting from the execution of a sequence of operators.

Definition 4.2 The *cumulative effect* of a sequence of operators, interpreted in the opaque semantics, is described by the two functions C^+ and C^-, defined as follows;

$$\begin{aligned}
C^+(\epsilon) &= \emptyset \\
C^-(\epsilon) &= \emptyset \\
C^+(O_1; \ldots; O_m; O) &= (C^+(O_1; \ldots; O_m) - D_O) \cup A_O \\
C^-(O_1; \ldots; O_m; O) &= (C^-(O_1; \ldots; O_m) \cup D_O) - A_O
\end{aligned}$$

where ϵ denotes the empty sequence.□

Lemma 4.3 Let $O_1; \ldots; O_m$ be a sequence of operators that are all admissible in their respective state sets, and I an initial state formula. Then

$$S^O(O_1; \ldots; O_m, mod(I)) \quad = \quad \bigcup_{\xi \in mod(I)} (\xi - C^-(O_1; \ldots; O_m)) \cup C^+(O_1; \ldots; O_m)$$

Proof: By induction on the length of the sequence, m. For $m = 1$,

$$\begin{aligned}
&\bigcup_{\xi \in mod(I)} (\xi - C^-(O)) \cup C^+(O) = \\
&\bigcup_{\xi \in mod(I)} (\xi - (C^-(\epsilon) \cup D_O)) - A_O)) \cup (C^+(\epsilon) - D_O) \cup A_O) = \\
&\bigcup_{\xi \in mod(I)} (\xi - ((\emptyset \cup D_O) - A_O)) \cup (\emptyset - D_O) \cup A_O) = \\
&\bigcup_{\xi \in mod(I)} (\xi - (D_O - A_O)) \cup A_O = (\text{since } D_O \cap A_O = \emptyset) \\
&\bigcup_{\xi \in mod(I)} (\xi - D_O) \cup A_O = \\
&\{\xi' | \xi \in mod(I) \wedge \xi' = (\xi - D_O) \cup A_O\} = \\
&S^O(O_1, mod(I))
\end{aligned}$$

Assuming the equality holds for $O_1;\ldots;O_m$,

$$
\begin{aligned}
&S^O(O_1;\ldots;O_m;O,mod(I)) = \\
&\{\xi|\xi' \in S^O(O_1;\ldots;O_m,mod(I)) \wedge \xi = (\xi' - D_O)\cup A_O\} = \\
&\{\xi|\xi' \in (\bigcup_{\xi''\in mod(I)} (\xi'' - C^-(O_1;\ldots;O_m))\cup C^+(O_1;\ldots;O_m)) \wedge \\
&\quad \xi = (\xi' - D_O)\cup A_O\} = \\
&\{\xi|\xi'' \in mod(I) \wedge \\
&\quad \xi' = (\xi'' - C^-(O_1;\ldots;O_m))\cup C^+(O_1;\ldots;O_m) \wedge \\
&\quad \xi = (\xi' - D_O)\cup A_O\} = \\
&\{\xi|\xi'' \in mod(I) \wedge \\
&\quad \xi = ((\xi'' - C^-(O_1;\ldots;O_m)) - D_O \cup C^+(O_1;\ldots;O_m) - D_O) \\
&\qquad \cup A_O\} = \\
&\{\xi|\xi'' \in mod(I) \wedge \\
&\quad \xi = (\xi'' - (C^-(O_1;\ldots;O_m)\cup D_O))\cup A_O \cup (C^+(O_1;\ldots;O_m) - D_O) \\
&\qquad \cup A_O\} = \\
&\{\xi|\xi'' \in mod(I) \wedge \\
&\quad \xi = (\xi'' - ((C^-(O_1;\ldots;O_m)\cup D_O)) - A_O)\cup(C^+(O_1;\ldots;O_m) - D_O) \\
&\qquad \cup A_O\} = \\
&\{\xi|\xi'' \in mod(I) \wedge \xi = (\xi'' - C^-(O_1;\ldots;O_m;O))\cup C^+(O_1;\ldots;O_m;O)\} = \\
&\bigcup_{\xi''\in mod(I)} (\xi'' - C^-(O_1;\ldots;O_m;O))\cup C^+(O_1;\ldots;O_m;O)
\end{aligned}
$$

□

Next, we show an upper bound on the length of plans.

Proposition 4.4 Let $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$ be a planning scenario, where the domain $\mathcal{D}$ is propositional and $|\mathcal{D}| = n$, $\mathcal{O}$ contains only deterministic STRIPS operators, interpretated in the opaque semantics, and I is any boolean formula over $\mathcal{D}$. If there exists a plan for $\mathcal{P}$, there exists a plan of length $\leqslant 2^{n+1}$.
Proof: For any set of states Ξ, partition the propositions in $\mathcal{D}$ in two sets K_Ξ and U_Ξ, such that K_Ξ contains exactly the propositions that are determined by Ξ. If a deterministic STRIPS operator O applied to Ξ affects a proposition p, and the result is defined, then $p \in K_{S^O(O,\Xi)}$. Since a plan that contains loops in the state space can be shortened to one that does not, any plan starting in state set Ξ, containing more than $2^{|K_\Xi|}$, and never moving any proposition from U to K, can be shortened to one containing no more than $2^{|K_\Xi|}$ steps. By repeated application of the same argument, any plan containing more than

$$\Sigma_{i=1}^{n} 2^i \quad \leqslant \quad 2^{n+1}$$

steps can be shortened to one that does not.□

Now, we can show that the plan existence problem can be solved in PSPACE.

Theorem 4.5 Let $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$ be a planning scenario, where the domain $\mathcal{D}$ is propositional and $|\mathcal{D}| = n$, $\mathcal{O}$ contains only deterministic STRIPS operators, interpretated in the opaque semantics, and I and G are boolean formulae over $\mathcal{D}$. Then plan existance for $\mathcal{P}$ is decidable in PSPACE.
Proof: By nondeterministic algorithm. We will construct the cumulative effects incrementally, so let $C^+ = C^- = \emptyset$ initially, and let $i = 0$.

(i) If, for all models ξ such that $\xi \models I$, $(\xi - C^-) \cup C^+ \models G$, return $\mathcal{T}$.

(ii) If $i > 2^{n+1}$, return $\mathcal{F}$.

(iii) Nondeterministically select an operator O such that for all models ξ such that $\xi \models I$, $(\xi - C^-) \cup C^+ \models P_O$, and let $C^- = (C^- \cup D_O) - A_O$, $C^+ = (C^+ - D_O) \cup A_O$ and $i = i + 1$.

(iv) Repeat from (i).

If the test in step (i) returns $\mathcal{T}$, the selected sequence of operators is a plan by lemma 4.3, and in the same way the test in step (iii) ensures that the selected operator is admissible. If the test in step (ii) returns $\mathcal{F}$, no plan exists by lemma 4.4. As C^+ and C^- are subsets of $\mathcal{D}$, they require at most space $2 \cdot n$. The tests in step (i) and (iii) consider only one model at a time, and therefore requires at most space n. The counter i can be represented in space $n + 1$. Thus, the algorithm is in NPSPACE, which is equal to PSPACE [28].□

Theorem 4.6 Deciding plan existance for a scenario $\mathcal{P}$ with propositional domain, deterministic STRIPS operators, with opaque semantics, and arbitrary formulae I and G, is PSPACE-complete.
Proof: Theorem 4.5 shows that plan existance for $\mathcal{P}$ is in PSPACE. Bylander [3] shows that planning with propositional domain, deterministic STRIPS operators with conjunctive preconditions and complete initial state information is PSPACE-hard. This is clearly a special case of $\mathcal{P}$, and the result follows.□

For an alternative proof of inclusion, though valid only for the restricted case of operators with preconditions on CNF, we use a goal regression function for the opaque semantics.

Definition 4.7 The expression $\alpha[p \leftarrow v]$, where α is a formula over $\mathcal{D}$, $p \in \mathcal{D}$ and $v \in \{\mathcal{T}, \mathcal{F}\}$, denotes the formula α with every occurance of p replaced by the constant v. The expression $\alpha[p_1 \leftarrow v_1, \ldots, p_n \leftarrow v_n]$ expands recursively to $(\alpha[p_1 \leftarrow v_1])[p_2 \leftarrow v_2, \ldots, p_n \leftarrow v_n]$. The expression within brackets is refered to as a *constant substitution*. For two constant substitutions σ_1, σ_2, by definition

$(\alpha[\sigma_1])[\sigma_2] = \alpha[\sigma_1, \sigma_2]$. Note that if some propositional symbol p occurs more than once in a constant substitution, all occurances after the first have no effect.

Given a STRIPS operator O and a formula φ, the *goal regression function* for the opaque semantics is defined as

$$R(O, \varphi) \quad = \quad P_O \wedge \varphi[a_i \leftarrow \mathcal{T}, d_i \leftarrow \mathcal{F}], \forall a_i{\in}A_O, d_i{\in}D_O$$

The regression function extends straightforwardly to sequences of operators through $R(O_1; \ldots; O_n, \varphi) = R(O_1, R(O_2; \ldots; O_n, \varphi))$.□

The following lemmas establish important properties of the regression function R. Lemma 4.8 shows correctness, i.e. that R agrees with the opaque operator semantics, while lemmas 4.9, 4.10 and 4.11 show that the formula returned by R is bounded in size.

Lemma 4.8 $\Xi{\models}R(O, \varphi)$ iff $S^O(O, \Xi){\models}\varphi$.
Proof: Assume $\Xi{\models}R(O, \varphi)$. Since P_O is a conjunct of $R(O, \varphi)$, $S^O(O, \Xi)$ is defined. Thus, for any $\xi \in \Xi$ there is a corresponding $\xi' = (\xi - D_O){\cup}A_O$ in $S^O(O, \Xi)$. For any propositional symbol p in φ, if $p \notin A_O \cup D_O$, then $\xi(p) \equiv \xi'(p)$. If $p \in A_O$, then p is replaced by $\mathcal{T}$ in $R(O, \varphi)$ and $\xi'(p) \equiv \mathcal{T}$. If $p \in D_O$, then p is replaced by $\mathcal{F}$ in $R(O, \varphi)$ and $\xi'(p) \equiv \mathcal{F}$. Thus $\xi(R(O, \varphi)) \equiv \xi'(\varphi)$, for all $\xi \in \Xi$. Since $S^O(O, \Xi) = \bigcup_{\xi\in\Xi}(\xi - D_O){\cup}A_O$ there can be no other models in $S^O(O, \Xi)$ and this implies $S^O(O, \Xi){\models}\varphi$.

Assuming $S^O(O, \Xi){\models}\varphi$ implies that $S^O(O, \Xi)$ must be defined, which in turn implies that $\Xi{\models}P_O$. For any $\xi' \in S^O(O, \Xi)$, there are one or more $\xi \in \Xi$ such that $\xi' = (\xi - D_O){\cup}A_O$, and by the same reasoning as above $\xi'(\varphi) \equiv \xi(R(O, \varphi))$. Since $\Xi = \bigcup_{\xi'\in S^O(O,\Xi)}\{\xi | \xi' = (\xi - D_O){\cup}A_O\}$, there can be no other models in Ξ and this implies $\Xi \models R(O, \varphi)$.□

Lemma 4.8 is easily extended to sequences of operators by induction.

Lemma 4.9 $|R(O, \varphi)| \leqslant |P_O| + |\varphi|$ and $|R(O_1; \ldots; O_n, \varphi)| \leqslant (c{\cdot}n) + |\varphi|$, where c is the size of the largest operator precondition in the sequence.
Proof: The first assertion follows by definition. From this,

$$\begin{aligned} |R(O_1; \ldots; O_n, \varphi)| &\leqslant (\Sigma_{i=1}^{n}|P_{O_i}|) + |\varphi| \\ |R(O_1; \ldots; O_n, \varphi)| &\leqslant (|P_{O_j}|{\cdot}n) + |\varphi| \end{aligned}$$

for the operator O_j with the largest precondition.□

When the number of different operators is limited by a constant and the operator preconditions are of a limited form, the size of the result of regressing a formula through a sequence can be shown to be even more tightly bounded.

Lemma 4.10 If α is a formula on CNF and σ a constant substitution, there exists a formula γ such that $\gamma \equiv \alpha \wedge \alpha[\sigma]$ and $|\gamma| \leqslant |\alpha|$, and such that γ can be effectively computed.
Proof: Let $\alpha = c_1 \wedge \ldots c_n$, where each c_i is a disjunction of literals $l_1 \vee \ldots \vee l_{m_i}$. We have

$$\alpha \wedge \alpha[\sigma] \equiv c_1 \wedge c_1[\sigma] \wedge \ldots \wedge c_n \wedge c_n[\sigma]$$

Consider any conjunct $c_i \wedge c_i[\sigma]$. If c_i contains a literal $l_j = p$ and σ contains the substitution $p \leftarrow \mathcal{T}$ or $l_j = \neg p$ and σ contains the substitution $p \leftarrow \mathcal{F}$, $c_i[\sigma] \equiv \mathcal{T}$ and $c_i \wedge c_i[\sigma] \equiv c_i$. If c_i contains no such literal, then $c_i[\sigma] \rightarrow c_i$ and thus $c_i \wedge c_i[\sigma] \equiv c_i[\sigma]$. This is because for each literal l_j in $c_i[\sigma]$, either $l_j \equiv \mathcal{F}$ under σ or l_j is unaffected by σ. Therefore, any assignment satisfying at least one literal in $c_i[\sigma]$ also satisfies the same literal in c_i.

The presence of a literal $l_j \equiv \mathcal{T}$ in $c_i[\sigma]$ can be determined in time linear in m_i. Thus, we can construct γ by selecting for each conjunct $c_i \wedge c_i[\sigma]$ the part that is equivalent to the whole, in time linear in $|\alpha|$. Since $|c_i[\sigma]| \leqslant |c_i|$, $|\gamma| \leqslant |\alpha|$ follows.□

Lemma 4.11 Let $O_1, \ldots, O_n$ be a set of distinct STRIPS operators with preconditions on CNF, and let $O'_1; \ldots; O'_m$, with $m > n$ be a sequence containing only these operators, in zero or more instances. Then there exists a formula γ such that

$$R(O'_1; \ldots; O'_m, \varphi) \equiv \gamma \text{ and } |\gamma| \leqslant (\Sigma_{i=1}^{n} |P_{O_i}|) + |\varphi|$$

and this γ can be computed in the same space.
Proof: We will construct formulae γ_i, for $i = 1 \ldots m+1$, which are conjunctions $c_0 \wedge c_1 \wedge \ldots \wedge c_n$, where each conjunct is of bounded size, such that $\gamma_{m+1} \equiv \varphi$ and $\gamma_i \equiv R(O_i'; \ldots; O_m', \varphi)$, for $i = 1 \ldots m$, and such that γ_i can be computed from γ_{i+1} within the given bounds. Intuitively, the conjuncts c_k will correspond to P_{O_k} for $k = 1 \ldots n$ and c_0 to φ.

For γ_{m+1}, let $c_0 = \varphi$ and $c_k = \mathcal{T}$ for $k = 1 \ldots n$. Obviously, $\gamma_{m+1} \equiv \varphi$. Assume $\gamma_i \equiv R(O_i'; \ldots; O_m', \varphi)$, and that the conjuncts of γ_i are such that $c_0 = \varphi[\sigma_0]$ and $c_k = P_{O_k}[\sigma_k]$ or $c_k = \mathcal{T}$ for $k = 1 \ldots n$, for some constant substitutions σ_k. Assume further that O'_{i-1} is an instance of the operator O_v. We have

$$\begin{aligned}
&\gamma_{i-1} \equiv \\
&R(O'_{i-1}; \ldots; O'_m, \varphi) \equiv \\
&R(O'_{i-1}, R(O'_i; \ldots; O'_m, \varphi)) \equiv \\
&R(O'_{i-1}, \gamma_i) \equiv \\
&P_{O_v} \wedge \gamma_i[a_l \leftarrow \mathcal{T}, d_l \leftarrow \mathcal{F}], \forall a_l \in A_{O_v}, d_l \in D_{O_v}
\end{aligned}$$

This is equivalent to replacing every substitution σ_k in γ_i with

$$\sigma'_k = \sigma_k, a_l \leftarrow \mathcal{T}, d_l \leftarrow \mathcal{F}, \forall a_l \in A_{O_v}, d_l \in D_{O_v}$$

and replacing $c_v = P_{O_v}[\sigma'_v]$ with $c_v = P_{O_v}[\sigma'_j] \wedge P_{O_v}$.

Since P_{O_v} is on CNF, we may by lemma 4.10 replace $c_v = P_{O_v}[\sigma'_j] \wedge P_{O_v}$ by either $P_{O_v}[\sigma'_j]$ or P_{O_v} in γ_{i-1} preserving the equality with $R(O'_{i-1}; \ldots; O'_m, \varphi)$.

Thus, $\gamma_1 \equiv R(O'_1; \ldots; O'_m, \varphi)$, and by lemma 4.10 $|\gamma_i| \leqslant (\Sigma_{i=1}^n |P_{O_i}|) + |\varphi|$ for $i = 1 \ldots m+1$.□

Theorem 4.12 Let $\mathcal{P} = (\mathcal{D}, \mathcal{O}, I, G)$ be a planning scenario, where the domain $\mathcal{D}$ is propositional and $|\mathcal{D}| = n$, $\mathcal{O}$ contains only deterministic STRIPS operators, interpretated in the opaque semantics and having preconditions on CNF, and I and G are boolean formulae over $\mathcal{D}$. Then plan existance for $\mathcal{P}$ is in PSPACE.

Proof: By algoritm, as follows. Let $\gamma = G$ and $i = 0$ initially.

(i) If $I \models \gamma$, return $\mathcal{T}$. If $i > 2^{n+1}$, return $\mathcal{F}$.

(ii) Nondeterministically select an operator O, and let $\gamma = R(O, \gamma)$. Let $i = i + 1$.

(iii) Repeat from (i).

By lemma 4.8, the selected sequence of operators is a plan for $\mathcal{P}$ iff $I \models \gamma$, and by lemma 4.4 if there is no plan in 2^{n+1} steps, there is exists no plan at all. By lemma 4.11 we have $|\gamma| \leqslant (\Sigma_{i=1}^n |P_{O_i}|) + |\varphi|$, i can be represented by $n+1$ bits and since propositional entailment can be decided in co-NP, the algorithm requires at most nondeterministic polynomial space. Since NPSPACE = PSPACE [28], plan existance for $\mathcal{P}$ is in PSPACE.□

Chapter 5

$\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ Planning

In this chapter, we examine the complexity of planning with $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ states and operators. The plan existence problem for this class of scenarios turns out to be PSPACE-complete; the problem, however, lies in characterising $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ operators. This problem is not yet fully solved.

Definition 5.1 Let $\mathcal{V}$ denote the set $\{\mathcal{T}, \mathcal{F}\}$ and $\mathcal{V}_+$ the set $\{\mathcal{T}, \mathcal{F}, \mathcal{U}\}$. Given a propositional domain $\mathcal{D}$, with $|\mathcal{D}| = n$, a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ *description* is a function from $\mathcal{D}$ to $\mathcal{V}_+$. A state ξ is *compatible* with a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ description τ if $\xi(p) = \tau(p)$ for all $p \in \mathcal{D}$ such that $\tau(p) \neq \mathcal{U}$. For a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ description τ, $mod(\tau) = \{\xi | \xi \text{ is compatible with } \tau\}$. Given a set of models Ξ, the *most specific* $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ *description* of Ξ, denoted by δ_Ξ is defined as

$$\begin{aligned} \delta_\Xi(p) = \quad & \mathcal{T} \quad \text{iff } \xi(p) = \mathcal{T}, \forall \xi \in \Xi \\ \delta_\Xi(p) = \quad & \mathcal{F} \quad \text{iff } \xi(p) = \mathcal{F}, \forall \xi \in \Xi \\ \delta_\Xi(p) = \quad & \mathcal{U} \quad \text{otherwise} \end{aligned}$$

A state set Ξ is a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ *state* iff $mod(\delta_\Xi) = \Xi$. An operator O is $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ iff whenever O is executed in a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ state Ξ, $S(O, \Xi)$ is also a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ state.$\Box$

Theorem 5.2 Deciding plan existance for a scenario $\mathcal{P}$, where $\mathcal{O}$ contains only $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ operators such that the computation of $S(O, \{\xi\})$ requires at most polynomial space, and I is a $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ state, is PSPACE-complete.

Proof: Planning in STRIPS with complete information is clearly a special case of $\mathcal{P}$, thus hardness follows from Bylander [3].

We show inclusion by nondeterministic algorithm. Let $\tau = \delta_I$ and $i = 0$ initially.

(i) If $mod(\tau) \models G$, return $\mathcal{T}$. If $i > 3^n$, return $\mathcal{F}$.

(ii) Nondeterministically select an operator O and let $\tau = \delta_{S(O,mod(\tau))}$. Let $i = i + 1$.

(iii) Repeat from (i).

The selected sequence of operators constitutes a plan iff the final state τ entails G, by definition. Since the initial state and all operators are $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$, $mod(\delta_{S(O,mod(\tau))}) = S(O, mod(\tau))$ throughout. Since there are only 3^n distinct states, any plan longer than 3^n steps contains loops that can be removed, and thus if no plan has been found in 3^n steps there exists no plan. The state τ as well as the counter i can be represented by $log_2(3){\cdot}n$ bits. To compute $\delta_{S(O,mod(\tau))}$ in polynomial space, we proceed as follows. We assume that models are enumerable (they are, after all, elements of $\mathcal{V}^n$) and let ξ_k denote the kth model.

For $k = 1 \ldots 2^n$ do

(i) If ξ_k is compatible with τ, let $\xi' = S(O, \{\xi_k\})$.

(ii) If ξ_k is the first compatible model found, let $\delta = \xi'$. Else for each $p \in \mathcal{D}$ such that $\delta(p) \neq \xi'(p)$ redefine $\delta(p)$ to be $\mathcal{U}$.

When the loop terminates, $\delta = \delta_{S(O,mod(\tau))}$. The computation of $S(O, \{\xi_k\})$ requires only polynomial space, by assumption, and the representations of k and $delta$ require space n and $log_2(3){\cdot}n$, respectively.

Thus, the algorithm requires only polynomial space, and since NPSPACE = PSPACE, the problem is in PSPACE.

Combining the two parts, we conclude that plan existance for $\mathcal{P}$ is PSPACE-complete.$\Box$

Chapter 6

Conclusions and Continuations

Having analysed the complexity of propositional contingent planning with varying restrictions, we can conclude that all versions of the plan existance problem that we have considered are intractable; this should come as no great surprise. There are, however, degrees even to intractability. On the lower end, propositional planning with deterministic context-independent operators and incomplete initial state, as well as $\mathcal{T}$-$\mathcal{F}$-$\mathcal{U}$ planning, was shown to be PSPACE-complete, which is no worse than the complexity of propositional planning in the complete information case. This represents a not insignificant extension of the class of planning problems known to be solvable in polynomial space.

Interesting to note is also the fact that planning with nondeterministic context-dependent operators is EXPSPACE-complete, while planning with probabilistic context-dependent operators is EXPTIME-complete [18]. This clearly shows that not only do nondeterministic descriptions contain less information than probabilistic descriptions, but so much less that it has consequences for complexity.

The central discovery, however, is the apparent importance of context dependency to the complexity of the planning problem; EXPTIME-hardness of planning with nondeterministic context-independent operators was shown only under transparent operator semantics, which allows a limited case of context dependency, and decidability in PSPACE for planning with context-independent operators and incomplete initial state was shown only under opaque semantics. The grayzone of table 1.1 includes not only the case of context-dependent operators, and the more limited transparent context-independent operators, with incomplete initial state, but also the case of opaque nondeterministic operators. From the present results, we suspect that the line between planning problems provably exponential and problems solvable in polynomial space can be drawn with the former on the high side and the later on the lower.

While the introduction of nondeterminism in operators and arbitrarily incomplete state descriptions relaxes some of the classical planning assumptions, several problems remain. In most domains, constructing a plan that covers every contingency is not a realistic approach, but neither is it to assume sensing operations will be available for any information that may be required. A combination of conditional and contingent planning is likely to be necessary in complex domains. From the complexity theoretic point of view however, contingent planning is interesting as sensing operations can be modelled as context-dependent operators changing a set of "known" propositions depending on corresponding "unknown" propositions. Branching plans can similarly be modelled as context-dependent operators.

Also from the complexity theoretic point of view, the results we have presented are interesting as they show some of the considered problems to be *provably* intractable, i.e. complete for exponential space or time. While there are many problems known to be complete for the class NP, and not so few complete for PSPACE, the library of EXPTIME-complete and EXPSPACE-complete problems is not extensive; this work makes a small contribution.

Bibliography

[1] Christer Bäckström. *Computational Complexity of Reasoning about Plans.* PhD thesis, Linköpings Universitet, 1992.

[2] Avrim L. Blum and Merrick L. Furst. Fast planning through graph analysis. *Artificial Intelligence*, 90, 1997.

[3] Tom Bylander. Complexity results for planning. In *Proceedings of the twelwth International Joint Conference on Artificial Intelligence (IJ-CAI'91)*, 1991.

[4] A.K. Chandra and L.J. Stockmeyer. Alternation. In *Proc. 17th IEEE Symposium on Foundations of Computer Science*, 1976.

[5] A.K. Chandra and L.J. Stockmeyer. Provably difficult combinatorial games. *SIAM Journal on Computation*, 8(2):151–174, 1979.

[6] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[7] Gregg Collins and Louise Pryor. Planning under uncertanity: Some key issues. In *Proceedings of the fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[8] Denise Draper, Steve Hanks, and Daniel Weld. Probabilistic planning with information gathering and contingent execution. In *Artificial Intelligence Planning Systems: Proceedings of the 2nd International Conference (AIPS'94)*, 1994.

[9] Kutluhan Erol, James Hendler, and Dana S. Nau. Complexity results for htn planning. Technical Report CS-TR-3240, Computer Science Department, University of Maryland, 1995.

[10] Kutluhan Erol, Dana S. Nau, and V.S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning: A detailed analysis. Technical Report CS-TR-2797, Computer Science Department, University of Maryland, 1991.

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

[12] Keith Golden, Oren Etzioni, and Daniel Weld. Omnipotence without omniscience: Efficient sensor management for planning. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, 1994.

[13] Cordell Green. Applications of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1969.

[14] Naresh Gupta and Dana S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence*, 56:223–254, 1992.

[15] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata theory, Languages and Computation*. Addison-Wesley, 1979.

[16] Peter Jonsson and Christer Bäckström. State-variable planning under structural restrictions: Algorithms and complexity. *Artificial Intelligence*, 1997. To appear.

[17] Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic least-commitment planning. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, 1994.

[18] Michael L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, 1997.

[19] Zohar Manna and Richard Waldinger. How to clear a block: A theory of plans. *Journal of Automated Reasoning*, 3:343–377, 1987.

[20] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the 9th National Conference on Artificial Intelligence(AAAI'91)*, 1991.

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

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

[23] Edwin P.D. Pednault. Generalizing nonlinear planning to handle complex goals and actions with context-dependent effects. In *Proceedings of the twelwth International Joint Conference on Artificial Intelligence (IJCAI'91)*, 1991.

[24] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the International Confrence on Knowledge Representation and Reasoning (KR'92)*, 1992.

[25] M. Peot and D. Smith. Conditional nonlinear planning. In *Artificial Intelligence Planning Systems: proceedings of the International Confrence (AIPS'92)*, 1992.

[26] Stanley J. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of the seventh International Joint Conference on Artificial Intelligence (IJCAI'81)*, pages 331–37, 1981.

[27] Earl D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the fourth International Joint Conference on Artificial Intelligence (IJCAI'75)*, pages 206–214. 1975.

[28] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and Systems Sciences*, 4(2):177–192, 1970.

[29] Austin Tate. Generating project networks. In *Proceedings of the fifth International Joint Conference on Artificial Intelligence (IJCAI'77)*, pages 88–93, 1977.

[30] D.H.D. Warren. Generating conditional plans and programs. In *Proceedings of the Summer Conference on AI and Simulation of Behaviour*, 1976.