Planning Graphs and Knowledge Compilation*

Héctor Geffner
ICREA & Universitat Pompeu Fabra
Paseo de Circunvalación 8
08003 Barcelona, Spain
hector.geffner@upf.edu

Abstract

One of the major advances in classical planning has been the development of Graphplan. Graphplan builds a layered structure called the planning graph, and then searches this structure backwards for a plan. Modern SAT and CSP approaches also use the planning graph but replace the regression search by a constrained-directed search. The planning graph uncovers implicit constraints in the problem that reduce the size of the search tree. Such constraints encode lower bounds on the number of time steps required for achieving the goal and account for the huge performance gap between Graphplan and its predecessors. Still, the form of local consistency underlying the construction of the planning graph is not well understood, being described by various authors as a limited but efficient form of negative binary resolution (Kautz & Selman 1999; Brafman 2001), k-consistency (Smith & Weld 1999), or 2-j consistency (Lopez & Bacchus 2003). In this paper, we aim to shed light on this issue by showing that the computation of the planning graph corresponds exactly to the iterative computation of prime implicates of size one and two over the logical encoding of the problem with the goals removed. The correspondence between planning graphs and a precise form of knowledge compilation (Selman & Kautz 1996; Cadoli & Donini 1997) provides a well-founded basis for understanding and developing extensions of the planning graph to non-Strips settings, and suggests novel and effective forms of knowledge compilation in other contexts. We explore some of these extensions in this paper and relate planning graphs with bounded variable elimination algorithms as studied by Rina Dechter and others.

Introduction

One of the major advances in classical planning has been the development of Graphplan. Given a Strips planning problem, Graphplan builds a layered structure called the planning graph, and then searches this structure backwards for a plan (Blum & Furst 1995). Modern SAT and CSP approaches also use the planning graph but replace the regression search by a constrained-directed search (Kautz & Selman 1999; Do & Kambhampati 2000; Rintanen 1998; Baioletti, Marcugini, & Milani 2000). The planning graph uncovers implicit constraints in the problem that reduce the size of the search tree. Such constraints are known to encode lower bounds on the number of time steps required for achieving the goal and account for the huge performance gap between Graphplan and its predecessors. Still, the form of local consistency underlying the construction of the planning graph is not well understood, being described by various authors as a limited but efficient form of negative binary resolution (Kautz & Selman 1999; Brafman 2001), k-consistency (Smith & Weld 1999), or 2-j consistency (Lopez & Bacchus 2003). In this paper, we aim to shed light on this issue by showing that the computation of the planning graph corresponds exactly to the iterative computation of prime implicates of size one and two over the logical encoding of the problem with the goals removed. The correspondence between planning graphs and a precise form of knowledge compilation (Selman & Kautz 1996; Cadoli & Donini 1997) provides a well-founded basis for understanding and developing extensions of the planning graph to non-Strips settings, and suggests novel and effective forms of knowledge compilation related to those studied Rina Dechter and colleagues over the last few years (Dechter 1999; Rish & Dechter 2000).

The paper is organized as follows. We start reviewing the planning graph and how it encodes lower bounds for Strips planning problems. We then analyze how lower bounds can be defined from logical encodings of planning problems, and how the two types of lower bounds are related. We do so by considering what we call the PI-k task: the computation of the sets of prime implicates of size bounded by a parameter k over the layers of a general class of stratified propositional theories. Stratified theories provide a suitable, logical abstraction of the theories found in planning. They are defined over a set of indexed boolean variables \( x_i \), for \( i = 0, \ldots, n \) and comprise clauses that involve variables \( x_i \) from the same layer or from adjacent layers only. We show that for the stratified theories that encode Strips planning problems, the planning graph computation solves the PI-k task exactly for \( k = 2 \). We conclude discussing extensions for dealing with conditional effects, an alternative scheme for deriving lower bounds from planning theories, and related work.

---

* I thank Rina Dechter, Javier Larrosa, Pedro Meseguer, and Alvaro del Val for valuable comments. Partial support from grant TIC2002-04470-C03-02, MCyT, Spain. Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.
The Planning Graph

Consider a Strips planning problem with initial situation
\[ I = \{p_0\}, \text{goal } G = \{p_n\} \text{ and operators } \alpha_i \text{ with precondition } p_i \text{ and postcondition } p_{i+1} \text{ for } i = 0, \ldots, m \text{ for } m \geq n. \]
Clearly, this problem has cost equal to \( n \); namely, \( n \) actions \( \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \) are needed in order to achieve \( G \) from \( I \). While in general determining the cost of a planning problem is intractable [Bylander 1994], there are simple and efficient procedures for obtaining lower bounds. Lower bounds are critical for optimal planning and search as they can prune large parts of the search space from consideration.

We represent Strips planning problems by tuples of the form \( P = (A, O, I, G) \), where \( A \) is the set of all possible (ground) atoms, \( I \subseteq A \) and \( G \subseteq A \) represent the initial and goal situations, and \( O \) is the set of (ground) actions \( a \), each with a precondition, add, and delete list, \( \text{pre}(a) \), \( \text{add}(a) \), and \( \text{del}(a) \). A simple procedure for computing lower bounds for a problem \( P \) can then be obtained by constructing iteratively the following sets:
\[
\begin{align*}
P_0 &= I \\
A_i &= \{a \in O \mid \text{pre}(a) \subseteq P_i\} \\
P_{i+1} &= P_i \cup \{p \in \text{add}(a) \mid a \in A_i\}
\end{align*}
\]
Namely, \( P_0 \) contains the atoms in the initial situation \( I \), \( A_i \) contains all the actions applicable in \( P_i \), and \( P_{i+1} \) contains all the atoms in \( P_i \) along with the atoms that can be added by the actions in \( A_i \). This construction can be continued until a fixed point \( P_n = P_{n+1} \) is reached.

We say then that a set of atoms \( G \) is reachable at level \( i \) if \( i \) is the index of the first layer \( P_i \) that contains \( G \). It is simple to verify that this index represents a lower bound on the number of steps needed to achieve \( G \) from \( I \). This lower bound, that we will denote as \( h_{\text{max}}(G) \), is not exact in general because it presumes that all applicable actions can be executed in parallel. Moreover, while the estimate \( h_{\text{max}}(G) \) provides an exact bound for the problem above, it is normally too weak to speed up the search for plans sufficiently.

The planning graph constructed by Graphplan is an elaboration of this idea that yields more informative (tighter) lower bounds and hence a more effective search. Graphplan computes optimal parallel plans where sets of non-interfering actions can be done in parallel. Two actions are interfering in Graphplan when one deletes a precondition or positive effect of the other. Plans in Graphplan are thus sequences of sets of actions \( A_0, A_1, \ldots, A_{n-1} \), and Graphplan minimizes the number of time steps \( n \) rather than the total number of actions. The measure \( h_{\text{max}}(G) \) above remains a lower bound on the ‘parallel’ cost \( h^*(G) \) for achieving \( G \) from \( I \), yet the lower bound \( h_g(G) \) computed by Graphplan is more informative (i.e., \( 0 \leq h_{\text{max}} \leq h_g \leq h^* \)).

The construction of the planning graph differs from the construction of the sets \( P_i \) and \( A_i \) above in two ways. First, in addition to these sets, Graphplan keeps track of pairs of propositions and actions that while ‘reachable’ individually in \( i \) steps, are not ‘jointly reachable’. These pairs are said to be mutually exclusive or mutex at level \( i \). For example, if \( p \) is not in \( P_i \), and some actions in \( A_i \) add \( p \) but all of them delete \( q \), then the pair \( p \) and \( q \) will be marked as mutex at level \( i + 1 \), meaning that while \( p \) and \( q \) may be achievable separately in \( i + 1 \) steps, the conjunction \( p \) and \( q \) may not. Graphplan uses information about mutexes in one layer to infer mutexes in the next layer, starting with the mutexes that correspond to interfering actions. Then, in order to make the propagation of mutexes more effective, atoms \( p \) in a layer \( P_i \) are no longer automatically copied into layer \( P_{i+1} \), rather ‘dummy’ actions \( \text{NO-OP}(p) \) are introduced for each atom \( p \), with precondition and effect \( p \), that interfere with all the actions the delete \( p \). The resulting sets in the planning graph are then:
\[
\begin{align*}
P_0 &= I \\
MP_0 &= \emptyset \\
A_i &= \{a \mid \text{pre}(a) \subseteq P_i \land \text{pre}(a) \not\subset MP_i\} \\
MP_i &= \{a \mid \text{add}(a) \mid a \in A_i\} \\
MP_{i+1} &= \{(p, q) \mid p, q \in P_i \land \forall a, a' \in A_i \text{ s.t. } p \in \text{add}(a) \land q \in \text{add}(a') \mid (a, a') \in M_i\}
\end{align*}
\]
where \( M_i \) stands for the sets of atom and action mutexes, \( a \land a' \) expresses that \( a \) and \( a' \) interfere, and \( Y \cap M_i \) \( Y \cap M_i \) indicates that \( Y \) contains (resp. contains no) pair in \( M_i \).

The iteration can be continued until a fixed point \( P_n = P_{n+1} \) and \( MP_n = MP_{n+1} \) is reached, yet Graphplan stops at the first layer \( m \leq n \) at which the goals are reached without a mutex \( G \subseteq P_m \) and \( G \not\subset MP_m \). This index \( m \) is a lower bound for \( h^*(G) \) which we will denote as \( h_g(G) \). Graphplan then searches the planning graph backwards for a plan, starting with the last layer, and if the search fails, a new layer is added, and the process repeats. Graphplan can actually be seen as a performing an IDA* regression search informed by the heuristic \( h_g \) implicitly encoded in the planning graph for all goals \( G' \) with estimate \( h_g(G') \) no greater than the planning horizon [Bonet & Geffner 2001].

Graphplan, introduced in [Blum & Furst 1995], was shown to be orders of magnitude faster than previous planners, a difference that can be traced to the derivation and use of an informative and admissible heuristic function encoded in the planning graph. Modern SAT and CSP approaches, make use of the planning graph but replace the regression search by a constraint-directed search [Kautz & Selman 1999; Do & Kambhampati 2000; Rintanen 1998; Baioletti, Marcugini, & Milani 2000]. While direct SAT and CSP encodings of Strips problems are possible [Kautz & Selman 1992; 1996], the planning graph encoding is known to speed up the search by uncovering implicit constraints in the problem like the bounds \( h_g \) above. The form of local consistency underlying this process, however, is not well understood. In [Haslum & Geffner 2000] a dynamic programming formulation of a parametric family of
polynomial and admissible heuristics \( h^m \), \( m = 1, 2, \ldots \) was given that generalizes Graphplan’s heuristic \( h_g \) and the \( h_{\text{max}} \) heuristic above. The heuristics \( h^m \) assume recursively that the cost of achieving a set of atoms \( s \) is given by the cost of achieving the most costly subset of size \( m \) in the set. In the parallel setting, \( h_{\text{max}} \) corresponds to the \( h^1 \) heuristic, \( h_g \) corresponds to \( h^2 \), and so on. Our aim in this paper is to understand the form of local consistency that yields these bounds.

**Deductive Lower Bounds for Planning**

The logical encoding of a Strips planning problem is a propositional theory whose models are in correspondence with the plans; namely, every model encodes a plan, and every plan is reflected in some model (Kautz & Selman 1992). For a given planning problem \( P = (A, O, I, G) \) with horizon \( n \), these encodings \( T \) feature boolean variables \( p_0, p_1, \ldots, p_n \), and \( a_0, a_1, \ldots, a_{n-1} \) for each fluent \( p \) and action \( a \) in the problem, and clauses encoding the initial situation, the action pre and postconditions, the frame axioms, and the goal. Early SAT approaches fed such encodings to a SAT solver that returned a model, and hence a plan, when the horizon was such that the problem was solvable. Modern SAT approaches do not work on such direct encodings but on encodings obtained from the planning graph. Here we consider such direct encodings but for deriving lower bounds on the optimal cost \( h^*(T) \) of achieving \( G \) from \( I \). For this, we remove from \( T \) the clauses encoding the goal \( G \), and consider the status of the formulas \( G_i \) encoding the goal condition at the different time points \( i = 0, \ldots, n \) in the resulting consistent theory. Clearly, \( i \) is a lower bound for achieving \( G \) if \( \neg G_j \) follows from \( T \) for all \( j = 0, \ldots, i - 1 \). This suggests the first class of deductive lower bounds \( h_N(G) \) defined as:

\[
h_N(G) \equiv \begin{cases} 
\min i \leq n & \text{such that } T \not\models \neg G_i \\
n + 1 & \text{if } T \models \neg G_i \text{ for all } i \in [0, n]
\end{cases}
\]  

(1)

Namely, \( h_N(G) \) encodes the first time point at which the formula \( G_i \) becomes consistent with \( T \); if no such \( i \) exists before the planning horizon \( n \), the lower bound is defined as \( n + 1 \).

The definition of the lower bound \( h_N(G) \) implies that no matter what actions are done, \( G \) cannot be made true in less than \( h_N(G) \) steps. We call \( h_N(G) \) a negative lower bound because it is based on the entailment of the negation of the goal. Later on we will discuss a type of positive lower bound defined in terms of the entailment of the goal itself but from a revised theory.

Negative lower bounds may be quite informative, and indeed for planning problems that do not involve incomplete information or non-determinism (as Strips), such bounds are exact; namely \( h_N(G) = h^*(G) \). From the results in (Bylander 1994), it follows though that the computation of these bounds is intractable. We thus focus on the definition of informative and tractable lower bounds \( h^W_N(G) \) whose general form can be captured in terms of sets \( \Gamma_i(T) \) of deductive consequences of \( T \) at times \( i = 0, \ldots, n \) as follows:

\[
h^W_N(G) \equiv \begin{cases} 
\min i \leq n & \text{such that } \Gamma_i(T) \not\models \neg G_i \\
n + 1 & \text{if } \Gamma_i(T) \models \neg G_i, \; \forall i \in [0, n]
\end{cases}
\]  

(2)

The bounds \( h^W_N \) are normally weaker than the bounds \( h_N \) i.e., \( h^W_N \leq h_N \), yet they become equal when the sets of deductive consequences \( \Gamma_i(T) \) corresponds to the sets \( PI_i(T) \) of prime implicants of \( T \) at time \( i \). The prime implicants of \( T \) are the inclusion-minimal, non-tautological clauses that follow from \( T \), and the prime implicants of \( T \) at time \( i \) are the prime implicants of \( T \) including variables with index \( i \) only. Provided with such sets of prime-implicants, it is possible to determine the bounds \( h^W_N(G) \) for arbitrary goals \( G \) efficiently; namely, \( G_i \) follows from \( PI_i(T) \) either if \( G_i \) contains complementary literals or is subsumed by a clause in \( PI_i(T) \) (Cadoli & Donini 1997; Marquis 2000). Of course, the computation of the sets of prime implicants \( PI_i(T) \) itself cannot be done efficiently, yet a suitable approximation of these sets will provide us with the tools for understanding the meaning of the planning graph from a logical point of view.

**Stratified Theories**

For the choice and computation of the sets \( \Gamma_i(T) \) for defining the lower bounds \( h^W_N \) in (2), we consider first the general logical form of planning theories. A key characteristic of them is their stratified nature as reflected in the definitions below.

**Stratified theories** are propositional theories defined over a language made up of a number of indexed variables \( x_i, i = 0, 1, \ldots, n \), where \( n \) is the given horizon. We refer to the language defined by the variables \( x_i \) for a particular index \( i \), as \( L_i \), and use the notation \( C_i, T'_i, \ldots \), to refer to clauses in \( L_i \).

A stratified theory \( T \) is made up of layers \( T_0, T_1, \ldots, T_n \) such that \( T = \cup_{i=0}^n T_i \). The first layer \( T_0 \) is given by a set of clauses \( C_0 \subseteq L_0 \), while each layer \( T_{i+1} \) for \( 0 \leq i < n \) is comprised of two sets of clauses: a set \( T_{i+1} \) of transition clauses \( C_i \vee C_{i+1} \) for \( C_i \in L_i \) and \( C_{i+1} \) in \( L_{i+1} \), and a set \( R_{i+1} \) of constraints \( C_{i+1} \subseteq L_{i+1} \). For a transition clause \( C_i \vee C_{i+1} \) we assume that neither \( C_i \) nor \( C_{i+1} \) is empty, and call \( C_i \) the head of the clause and \( C_{i+1} \) the body. For uniformity, we set \( R_0 = T_0 \).

Stratified theories \( T \) are classical propositional theories with a classical semantics and in particular \( T \) entails a formula \( A_i \in L_i \) when \( A_i \) is true in all models of \( T \). It is also useful, however, to interpret the models of \( T \) as state trajectories \( s_0, s_1, \ldots, s_n \), where \( s_i \) is a possible world (truth assignment) over the language \( L_i \). A state trajectory \( s_0, s_1, \ldots, s_n \) then satisfies \( T = \cup_{i=0}^n T_i \) when \( s_0 \) satisfies \( T_0 \) and each transition \( s_i, s_{i+1} \) satisfies \( T_{i+1}, 0 \leq i < n \); namely, \( s_{i+1} \) satisfies all constraints \( C_{i+1} \) in \( T_{i+1} \), and all heads \( C_{i+1} \) of the transition clauses \( C_i \vee C_{i+1} \) in \( T_{i+1} \) whose body \( C_i \) is not satisfied by \( s_i \). Clearly \( T \) entails \( A_i \in L_i \) when for all state trajectories \( s_0, s_1, \ldots, s_n \) satisfying \( T, s_i \) satisfies \( A_i \).
We also say that a stratified theory is **logically consistent** if there is a least one state trajectory that satisfies the theory, and is **causally consistent** if any partial state trajectory $s_0, \ldots, s_i$ satisfying the stratified theory $T' = T_0 \cup \cdots \cup T_i$ can be extended into a full state trajectory $s_0, \ldots, s_i, \ldots, s_n$ satisfying the theory $T = T_0 \cup \cdots \cup T_n$, $0 \leq i \leq n$. Clearly, if $T$ is causally consistent, then $T$ will be logically consistent if the first layer $T_0$ is consistent.

As an example, the clauses $x_0 \lor y_0, \neg x_0 \lor x_1, \neg y_0 \lor \neg x_1$ represent a logically consistent stratified theory $T = T_0 \cup T_1$ with $T_0 = \{x_0 \lor y_0\}$ and $T_1 = \{\neg x_0 \lor x_1, \neg y_0 \lor \neg x_1\}$. This theory, however, is not causally consistent as the state $s_0$ satisfying $T_0$ with $x_0$ and $y_0$ true, cannot be extended into a full trajectory satisfying $T_1$. On the other hand, the theory $T'$ that is like $T$ but includes in $T_0$ the resolvent $\neg x_0 \lor \neg y_0$ of the two clauses in $T_1$ is logically and causally consistent.

### PI-$k$ Inference

Consider now the following sets $\Gamma_i(T)$ of deductive consequences of a stratified theory $T$ over the languages $L_i$, defined iteratively for $i = 0, \ldots, n - 1$ as

$$\Gamma_0(T) \overset{\text{def}}{=} PI_0(T_0) \quad (3)$$

$$\Gamma_{i+1}(T) \overset{\text{def}}{=} PI_{i+1}(\Gamma_i(T) \cup T_{i+1}) \quad (4)$$

where $PI_{i+1}(T')$ stands for the prime implicates of $T'$ over the language $L_{i+1}$ (del Val 1999); namely $PI_{i+1}(T') = PI(T') \cap L_{i+1}$. It is possible to prove that if $T$ is causally consistent, then these sets, defined iteratively one in terms of the other, capture exactly the prime implicates of $T$ over the various sublanguages:

**Proposition 1 (Markov)** If the stratified theory $T$ is causally consistent then $\Gamma_i(T) = PI_i(T)$ for $i = 0, \ldots, n$.

The computation of these sets, however, is intractable in general. In order to obtain informative sets that can be computed in polynomial time, we focus on the iterative computation of prime implicates of bounded size. If we let $PI^k(T)$ stand for the prime implicates of $T$ of size at most $k$ for an integer $k \geq 1$, and let $PI^k(T)$ stand for the prime implicates of size no greater than $k$ over the language $L_i$, i.e., $PI^k(T) = PI(T) \cap L_i$, then a sequence of sets of clauses $\Gamma^k_i(T)$ for $i = 0, \ldots, n$ involving the iterative computation of prime implicates of bounded size can be defined as:

$$\Gamma^k_0(T) \overset{\text{def}}{=} PI^k_0(T_0) \quad (5)$$

$$\Gamma^k_{i+1}(T) \overset{\text{def}}{=} PI^k_{i+1}(\Gamma^k_i(T) \cup T_{i+1}) \quad (6)$$

We call these sets $\Gamma^k_i(T)$, the **iterated PI-$k$ sets**, and distinguish them from the non-iterated PI-$k$ sets defined simply as $PI^k_i(T)$. Iterated and non-iterated PI-$k$ sets are not equivalent in general as the ‘markovian’ property captured by Proposition 1 does not hold for a bounded $k$.

For example, for a stratified theory $T$ given by the clauses $x_0 \lor y_0, \neg x_0 \lor x_1, \neg y_0 \lor \neg x_1$, and for $k = 1$, $\Gamma^1_i(T) = \emptyset$ while $PI^1_i(T) = \left\{z_1\right\}$. On the other hand, the sets $\Gamma^2_i(T)$ and $PI^2_i(T)$ coincide for $k = 2$.

We refer to the computation of the iterated PI-$k$ sets, as the **PI-$k$ task**. This task is intractable in general yet we will determine conditions under which it is tractable, and establish a formal correspondence between the iterated PI-$k$ sets and the sets computed by Graphplan in the construction of the planning graph.

The first condition for making the computation of the $\Gamma^k_i(T)$ sets tractable has to do with the way the stratified theory is **compiled**. Basically, the compilation makes explicit certain consequences that follow in isolation from each layer $T_i$ of the theory. We will say that a stratified theory $T$ is **strongly compiled** if the resolvent of every pair of clauses in each layer $T_i$ is subsumed by a clause in $T$ or is tautological (contains a pair of complementary literals). For us, however, a weaker form of compilation suffices where this condition is imposed only on resolvents obtained over variables $x_i \in L_i$:

**Definition 1** A stratified theory $T = \cup_{i=0}^n T_i$ is compiled **iff** every clause $C$ obtained by resolving two clauses in $T_i$ upon variables $x_i \in L_i$, for $i = 0, \ldots, n$, is subsumed by a clause in $T$ or contains a pair of complementary literals.

Clearly, strongly compiled theories are compiled, but not the other way around. For example, the theory $T$ with clauses $x_0 \lor y_1$ and $\neg x_0 \lor y_1$ is compiled as both clauses belong to layer $T_1$ and they cannot be resolved upon variables in $L_1$ (i.e., variables with index equal to 1). At the same time, $T$ is not strongly compiled because the two clauses can be resolved upon the variable $x_0$, leading to the clause $y_1$ which is not subsumed. As another example, the theory $T'$ with clauses $x_0 \lor y_1$ and $y_0 \lor \neg y_1$ is not compiled as the resolvent $x_0 \lor y_0$ of the two clauses in $T'_1$ over variable $y_1 \in L_1$ is not subsumed in $T'$. Similarly for the theory $T''$ with clauses $x_0 \lor y_1$ and $y_0 \lor \neg y_1 \lor x_1$ which lacks the resolvent $x_0 \lor y_0 \lor x_1$.

The compilation of a stratified theory $T$ involves a closure operation under a restricted form of resolution where only clauses in the same layer $T_i$ are resolved, and they are resolved only upon non-body variables $x_i \in L_i$. We will later see that Strips theories are compiled in this way. More generally, the compilation of a stratified theory is intractable yet similar (intractable) compilation schemes have been successfully used in the ‘Planning as Model-Checking’ approach where planning theories are compiled into OBDDs (Cimatti et al. 2003). Indeed, prime implicates and OBDDs are two among several canonical logical forms that are appealing as they render certain kinds of boolean operations tractable (Darwiche & Marquis 2002). Some important properties of compiled theories are the following:

---

1The notion of causal consistency is closely related to a similar notion in probabilistic and logical causal networks; see (Pearl 1988; Darwiche & Pearl 1994).
Proposition 2  Let \( T = \bigcup_{i=0}^{n} T_i \) be a compiled stratified theory. Then, 1) \( T \) is logically and causally consistent if \( T \) does not contain an empty clause; 2) \( PI(R_i) \subseteq R_i \), for \( i = 0, \ldots, n \).

Recall that every layer \( T_i \) of a stratified theory \( T \), except for \( T_0 \), is given by a set \( R_i \) of primitive constraints \( C_i \in L_i \) and a set \( T_{ri} \) of transition clauses \( C_{i-1} \vee C_i \). This proposition implies that the set of constraints in a compiled theory is strongly compiled.

Compiled stratified theories yield also a decomposition property useful for computing the iterated PI-k sets. We use the notation \( L_i^k \) to denote the set of clauses in \( L_i \) with size no greater than \( k \), and assume that all the primitive constraints in \( R_i \) are in \( L_i^k \) for \( i = 0, \ldots, n \). When this is not the case, the term \( \Gamma_i^k(T) \) should be replaced by the term \( \Gamma_i^k(T) \cup R_i \), both in the following proposition and in (6). For simplicity, we will not consider that case further.

Proposition 3 (Decomposition)  For a compiled stratified theory \( T \) and any clause \( C_{i+1}^t \in L_{i+1}^k \), \( \Gamma_i^k(T), T_{ri+1} \models C_{i+1}^t \) iff \( \Gamma_i^k(T), T_{ri+1} \models C_{i+1} \) or \( R_{i+1} \models C_{i+1}^t \) Indeed, if \( C_{i+1}^t \) belongs to \( \Gamma_i^k(T) \), it must follow from \( \Gamma_i^k(T) \cup T_{ri+1} \) by resolution, and moreover, the resolution steps can be ordered so that all resolvents upon \( x_{i+1} \) variables are computed before resolvents upon \( x_i \) variables (Tison 1967; del Val 1999; Marquis 2000). Since the former resolvents can only involve clauses from \( T_{ri+1} \), due to the compilation, they are all subsumed by clauses in \( T_{ri+1} \) or \( R_i \), and hence by clauses in \( T_{ri+1} \) or \( \Gamma_i^k(T) \). Thus resolutions upon \( x_{i+1} \) variables are not needed for deriving \( C_{i+1} \), and therefore any such clause must be subsumed by a clause in \( R_{i+1} \) or must follow from resolutions upon \( x_i \) variables involving clauses from \( \Gamma_i^k(T) \) and \( T_{ri+1} \) only.

Proposition 3 suggests an approach for computing the \( \Gamma_{i+1}^k(T) \) sets. We initialize this set to be empty and then consider each of the clauses \( C_{i+1}^t \) in \( L_{i+1}^k \) in increasing order of size. If the clause is subsumed by a clause already in \( \Gamma_{i+1}^k(T) \), we skip it. Else, we test whether \( C_{i+1}^t \) follows from \( R_{i+1} \) or \( \Gamma_i^k(T) \cup T_{ri+1} \). The first part of this test is easy in a compiled theory, where it can be checked by subsumption. The second part is more subtle and is not tractable in general. We provide conditions however under which this second part can be computed in polynomial time. Then, since there is a polynomial number of clauses \( C_{i+1}^t \) in \( L_{i+1}^k \), it follows that the computation of \( \Gamma_{i+1}^k(T) \) from \( \Gamma_i^k(T) \) will run in those cases in polynomial time too.

A procedure for computing these sets based on these ideas is shown in Fig. 1. The procedure exploits also an additional property that is explained next.

The PI-k Inference Procedure

Let us refer to the transition clauses \( C_i \land C_{i+1} \) in \( T_{ri+1} \) with \( C_{i+1} \subseteq C_{i+1}^t \), and to the bodies \( C_i \) of such clauses, as the supporting clauses and supports of \( C_{i+1}^t \) respectively, and let the clause true be the single support of \( C_{i+1}^t \) when \( C_{i+1}^t \) has no supporting clauses. Then for a compiled stratified theory, the following property allows us to ‘regress’ a formula \( C_{i+1}^t \) in \( L_{i+1} \) into a formula in \( L_i \):

Proposition 4 (Regression)  Let \( C_1, \ldots, C_n \) be the supports of \( C_{i+1}^t \) in a compiled stratified theory \( T = \bigcup_{i=0}^{n} T_i \). Then \( \Gamma_i^k(T), T_{ri+1} \models C_{i+1} \iff \Gamma_i^k(T) \models \neg (C_1 \land \cdots \land C_n) \)

We will refer to the formula \( \neg (C_1 \land \cdots \land C_n) \) involving all the supports of \( C_{i+1}^t \) as the regression of \( C_{i+1}^t \), in analogy to the notion of regression used in planning, and formulate conditions under which the number of clauses encoding this formula is polynomial and the entailment of such clauses can be checked efficiently. In relation to this latter point we have that

Proposition 5  For a compiled theory \( T \), \( PI^k(\Gamma_i^k(T)) = \Gamma_i^k(T) \), and for \( k \leq 2 \), \( PI(\Gamma_i^k(T)) = \Gamma_i^k(T) \).

The first property follows from \( PI^k(PI^k(X)) = PI^k(X) \); the second from the closure of 2-CNF formulas: the resolvent of two clauses containing two literals at most, cannot contain more than 2 literals. This second property is what we need, it reduces validity checks to subsumption tests. For generalizing it to higher values of \( k \), let us say that a stratified theory \( T = \bigcup_{i=0}^{n} T_i \) is pure when each of the variables \( x_i \in L_i \), for \( 0, \ldots, n \) appears only positively or negatively in all clauses in \( T_i \) (notice that variable \( x_i \) may also occur in the body of a transition clause in \( T_{ri+1} \) yet such occurrences are not considered). Clearly if \( T \) is pure, so will be the \( \Gamma_i^k(T) \) sets, and hence

Proposition 6  For a compiled stratified theory \( T \) that is pure, \( PI(\Gamma_i^k(T)) = \Gamma_i^k(T) \).

Consider now the regression \( \neg (C_1 \land \cdots \land C_n) \) of \( C_{i+1}^t \).

The CNF representation of this formula is given by the conjunction of all the clauses \( \sim l_1 \lor \sim l_2 \lor \cdots \lor \sim l_r \) where \( U \) is a literal from \( C_i^t \). The number of such clauses is exponential in the number of disjunctive supports of \( C_{i+1}^t \); namely the number of supports \( C_i \) of \( C_{i+1}^t \) with size \( \mid C_i \mid > 1 \).

Let \( w_T(l_{i+1}) \) stand for the number of transition clauses \( C_i \land C_{i+1} \) in \( T \) with disjunctive bodies \( C_i \) such that \( l_{i+1} \in C_i \), and let \( w_T \) stand for the max such number over all literals \( l_{i+1} \) in \( L_i \) and all \( i = 0, \ldots, n - 1 \). We will call the parameter \( w_T \), the support width of \( T \) as it plays a role analogous to the width parameter in variable elimination algorithms (Dechter 1999). For Strips theories, we will show their support width to be 1.

The results above lead to the procedure shown in Fig. 1 for computing the sets \( \Gamma_i^k(T) \), \( i = 0, \ldots, n \) defined in (6) that we call the PI-k procedure. The PI-k procedure projects a set \( \Gamma_i^k \) of clauses in \( L_i^k \) into a set \( \Gamma_{i+1}^k \) of valid consequences over \( L_{i+1}^k \) given a stratified theory \( T \). For theories \( T \) with bounded support width, the procedure runs in polynomial time, and if in addition, the theory is pure and compiled, the procedure is complete; namely, given a PI-k set, it computes the next PI-k set in the sequence exactly.
Set \( \Gamma_{i+1}^k := \emptyset \) for each non-taut clause \( C_{i+1}^k \in L_{i+1}^k \). For any \( C_{i+1}^k \) subsumed by clause in \( \Gamma_{i+1}^k \), continue.

For any \( C_{i+1}^k \) subsumed by clause in \( R_{i+1} \), add \( C_{i+1}^k \) to \( \Gamma_{i+1}^k \). For each \( C_{i+1}^k \), \( j = 1, \ldots, r \) be the supports of \( C_{i+1}^k \) in \( T \). If no supports, continue.

For each \( C_{i+1}^k \), empty, add \( C_{i+1}^k \) to \( \Gamma_{i+1}^k \). If \( C_{i+1}^k \) has \( k \)-disjunctive supports, the \( k \)-disjunctive supports are the supports of the target clause \( C_{i+1}^k \). The \( k \)-disjunctive supports are needed for completeness. A sufficient condition for this is the following:

\[
\Gamma_{i+1}^k \subseteq C_{i+1}^k \land C_1 \land \ldots \land C_r.
\]

We say in that case \( \Gamma_{i+1}^k \) is subsumed by a clause of size 2 of higher, we will never have to check whether one such clause is subsumed by a unit clause. Moreover the simplification may reduce the support width of the theory as well.

The second optimization involves the elimination of redundant transition clauses. The notion of redundancy needs to be defined carefully as the compilation adds entailed clauses to \( T \) that are needed for completeness. A transition clause \( C_i \lor C_{i+1} \) is redundant in \( T_{i+1} \) given \( \Gamma_i^k \), if there is another transition clause \( C_i' \lor C_{i+1}' \) in \( T_{i+1} \) such that \( C_i' \subseteq C_{i+1}' \) and \( C_i \) follows from \( \Gamma_i^k \) and \( C_i' \). Since by construction, \( R_i \) is subsumed by \( \Gamma_i^k \), then this last condition can be safely approximated by \( C_i \) being entailed by \( R_i \) and \( C_i' \). We will see that in Strips, ‘delete clauses’ are made redundant by the primitive action mutexes and the frame clauses.

Finally, we are often interested in the consequences of \( T \) over some target language (del Val 1999). For example, in planning, we are commonly interested in goal conditions \( G \) expressed by a conjunction of atoms, and then, in order to derive (negative) lower bounds from \( T \) we care only about the negative consequences of \( T \) that may allow us to disprove the encoding of such goals. For such cases, rather working with the original, compiled theory \( T \), we may as well work with a simplified theory \( T' \), where all transition clauses \( C_i \lor C_{i+1} \) and constraints \( C_{i+1} \) with \( C_{i+1} \not\in L^- \) are removed (\( L^- \) is the set of negative clauses). Under some conditions it is possible to show that no relevant information is lost in the simplification, namely that \( \Gamma_i^k(T) \cap L^- = \Gamma_i^k(T') \). We say in that case that the restriction of \( T \) over \( L^- \) is admissible. A sufficient condition for this is the following:

**Proposition 8** The restriction \( T' \) of a compiled theory \( T \) over the target language \( L^- \) is admissible, namely \( \Gamma_i^k(T') \cap L^- = \Gamma_i^k(T') \) for \( i = 0, \ldots, n \), if the variables \( x_i \) occurring positively in \( T_i \) do not occur negatively in the body of a transition clause in \( T_{i+1} \), for \( i = 0, \ldots, n-1 \).

We will see that this condition holds naturally for planning theories, and hence, they can be simplified in this way resulting in theories that are pure.
Stratified Strips Theories

A Strips planning problem \( P = (A, O, I, G) \) with horizon \( N \), where \( A \) is the set of relevant actions, \( O \) is the set of (ground) actions (including NO-ops), and \( I \) and \( G \) stand for the initial and goal situations, can be encoded by the theory (Kautz & Selman 1996)

1. **Init:** \( p_0 \) for \( p \in I \), \( \neg q_0 \) for \( q \in A \) but not in \( I \)

2. **Actions:** For \( i = 0, 1, \ldots, N - 1 \) and \( a \in O \):
   - \( p_i \lor \neg a_i \) for each \( p \in \text{pre}(a) \)
   - \( \neg a_i \lor p_{i+1} \) for each \( p \in \text{add}(a) \)
   - \( \neg a_i \lor \neg p_{i+1} \) for each \( p \in \text{del}(a) \)

3. **Frame:** If \( a_1, a_2, \ldots, a_{n_p} \) are the actions in \( O \) that add \( p \), then for \( i = 0, 1, \ldots, N - 1 \):
   - \( a_1^i \lor a_2^i \lor \cdots \lor a_{n_p}^i \lor \neg p_{i+1} \)

4. **Mutex:** If \( a, a' \) interfere, \( \neg a_i \lor \neg a'_i \) for \( i = 0, 1, \ldots, N - 1 \)

5. **Goal:** \( p_N \) for each \( p \in G \).

In order to stratify this theory we define first the languages \( L_i \) for each layer. We do this as in Graphplan: the languages \( L_i \) for \( i = 0, 2, 4, \ldots, 2N \) represent propositional layers and are defined in terms of the atoms \( p_i \) for \( p \in A \); while the languages \( L_j \) for \( j = 1, 3, \ldots, 2N - 1 \) represent action layers and are defined in terms of the actions \( a_j \) for \( a \in O \). The resulting stratified theory \( T = \cup_{i=0}^n T_i \) for \( n = 2N \), is a simple rearrangement of the clauses and indices, with the goal clauses excluded:2

1. **Init** \( T_0 \): \( p_0 \) for \( p \in I \), and \( \neg q_0 \) for \( q \in A \) not in \( I \)

2. **Action Layers** \( T_{i+1} \): for \( i = 0, 2, \ldots, n - 2 \)
   - \( p_i \lor \neg a_{i+1} \) for each \( a \in O \) and \( p \in \text{pre}(a) \)
   - \( \neg a_{i+1} \lor \neg a'_{i+1} \) for interfering \( a, a' \) in \( O \)

3. **Propositional Layers** \( T_{i+1} \): for \( i = 1, 3, \ldots, n - 1 \)
   - \( \neg a_i \lor p_{i+1} \) for each \( a \in O \) and \( p \in \text{add}(a) \)
   - \( \neg a_i \lor \neg p_{i+1} \) for each \( a \in O \) and \( p \in \text{del}(a) \)
   - \( a_1^i \lor a_2^i \lor \cdots \lor a_{n_p}^i \lor \neg p_{i+1} \) for each \( p \in A \)

We call these theories, **stratified Strips theories**. Provided that no atom is added and deleted by the same action, it’s simple to establish the following property:

**Proposition 9** Stratified Strips theories are compiled, and are logically and causally consistent.

Notice that without the constraints \( \neg a_i \lor \neg a'_i \) for interfering actions \( a \) and \( a' \), the theory would not be causally consistent or compiled. Indeed, if \( a \) adds \( p \) and \( a' \) deletes \( p \), then the non-tautological clause \( \neg a_i \lor \neg a'_i \) follows from resolving the transition clauses \( \neg a_i \lor p_{i+1} \) and \( \neg a'_i \lor \neg p_{i+1} \) over their head variable \( p_{i+1} \). Such a clause must be subsumed by a constraint in \( R_t \) for the theory to be compiled.

It is also simple to show, using Proposition 8, that the restriction of a stratified Strips theory \( T \) over the target language \( L^- \) is admissible: the only variables \( x_i \) appearing positively in \( T_i \), for \( i = 0, \ldots, n \), are the propositional variables \( p_i \) in the positive clauses in \( T_0 \) or in Add clauses, and such variables do not appear negatively in the body of any clause. As a result, we can remove the positive clauses in \( T_0 \) and the Add clauses in each \( T_i \), for \( i = 2, 4, \ldots, n \), without losing information about the consequences over the target language \( L^- \). The simplified theory \( T^- \) remains compiled and is pure. In addition, for any literal \( l_{i+1} \), the number of supporting clauses \( C_l \lor C_{l+1} \) with disjunctive bodies \( C_l \) is at most one, this occurring only for the frame clauses and \( l_{i+1} = \neg p_{i+1} \). Thus the support width of stratified Strips theories is 1. Applying Theorem 7 we thus obtain that:

**Theorem 10** (PI-k sets for Strips) For a stratified Strips theory \( T \), the \( \pi \)-k procedure computes the sets of clauses \( \Gamma_k^T\cap L^- \), \( i = 0, \ldots, n \) in polynomial time.

We note that the sets \( \Gamma_k^T(T) \) may contain clauses that are not in \( L^- \). For example, if all the actions that add \( q \) also add \( p \), then the clause \( \neg q_i \lor p_i \) will be derivable in each propositional layer. However, since propositions \( p_i \) do not occur negatively in the body of any transition clause in \( T \), such clauses do not lead to further inferences of either positive or negative clauses.

We are finally ready to state the correspondence between iterated \( \pi \)-k sets and the sets of atoms and mutexes in the planning graph:

**Theorem 11** (Planning Graphs) Let \( P \) be a Strips planning problem, let \( T \) be the stratified theory encoding \( P \) with horizon \( N \), and let \( i = 0, \ldots, 2N \) stand for the propositional and actions levels in the planning graph. Then

1. For \( x_i \in L_i \), \( x \) is reachable at level \( i \) of the planning graph \( \text{iff} \neg x_i \notin \Gamma_i^T \)
2. For \( x_i, y_i \in L_i \), the mutex \( (x, y) \) is inferred at level \( i \) of the planning graph \( \text{iff} \neg x_i \lor \neg y_i \in \Gamma_i^T \).

This correspondence extends also to the methods used for inferring these sets of propositions, in particular after the simplifications in the \( \pi \)-k procedure discussed earlier. The restriction of the Strips theory over the language \( L^- \) allows us to remove the Add and positive clauses in \( T_0 \) from consideration. The deletion of subsumed clauses allows us to eliminate the Delete clauses as well. Indeed, the delete clauses \( \neg a_i \lor \neg p_{i+1} \) in \( T \) are made redundant by the frame clause \( a_1^i \lor a_2^i \lor \cdots \lor a_{n_p}^i \lor \neg p_{i+1} \) and the ‘mutex’ clauses \( \neg a_i \lor \neg a'_i \), as by deleting \( p \), \( a \) interferes with each of the actions \( a \) that adds \( p \). As a result, the computation of the iterated \( \pi \)-k sets over the target language \( L^- \) can indeed proceed with the precondition, frame, and mutex clauses only. Moreover, with unit simplification, the correspondence between the two inference procedures draws even closer. Due to the formulation, however, the \( \pi \)-k procedure can handle a broader range of theories, including theories not arising from planning domains. At the same time it suggests incomplete approximations, like the \( \pi \)-k procedure discussed earlier, that runs over much a broader set of theories in polynomial time.

---

2The theory with the goals is stratified but is not compiled or causally consistent.
Extensions

We consider briefly the application of the proposed framework to a planning language that includes negation and conditional effects, and an alternative compilation scheme for deriving positive rather than negative lower bounds.

Negation and Frame Axioms

Strips does not accommodate negated literals in either the preconditions or goals, and neither can the above logical encodings. For example, the encoding of a problem in which \( p \) is true initially and no action deletes \( p \), does not entail the truth of \( p \) at times \( i > 0 \); namely \( p_0 \) is entailed but not \( p_1 \) or \( p_2 \). The limitation for handling negation in Strips arises from the semantics: states are sets of atoms, and no assumption is made about the truth of atoms not in the set. The limitation of the logical encoding arises from the way it handles persistence; like Graphplan and the SAT encodings based on Graphplan, it uses NO-OP actions, which like the other actions, are assumed to be exogenous. As a result, the encodings admit ‘abnormal models’ where a fluent \( p \) true at \( i \) becomes false at \( i + 1 \) without being deleted, just because the model does not make the action NO-OP(\( p \)) true at \( i \). Such models do not hurt in the Strips setting, because of a monotonicity property: any such ‘abnormal’ model \( M \) can always be extended into a ‘normal’ model \( M' \) that encodes the same plan, where fluents persist as they should. Yet ‘abnormal’ models do hurt in other settings where the monotonicity property does not hold, as for example, those including actions with conditional effects. In this sense, actions with conditional effects require a proper handling of negation, a point already made in (Anderson, Smith, & Weld 1998). While the encodings can be fixed by forcing the truth of NO-OP actions when the corresponding fluent should persist, here we will follow an approach that also works in Strips and is commonly used: for every atom \( p \) in the problem, a new atom \( \hat{p} \) is added that stands for the negation of \( p \). To enforce this interpretation, \( \hat{p} \) is included in the initial situation \( I \) if \( p \not\in I \), in each delete list that contains \( p \) in the add list, and in each add list that contains \( p \) in the delete list. For an Strips problem, we refer to the problem that results from these modifications, the Strips problem with negation. Notice that in the logical encoding of the problem with negation it is not true that the equivalence \( \hat{p}_{i+1} \equiv \neg p_{i+1} \) holds in all models. Yet as before, for any ‘abnormal’ model where the equivalence does not hold, there is a ‘normal’ model that validates the same plan, where fluents persist as they should and the equivalence holds.

Actions with Conditional Effects

Conditional effects are important in planning for two reasons. First, from a syntactic point of view, conditional effects cannot be compiled away in Strips without causing an exponential blow up (Nebel 2000); second, from a semantic point of view, they are an essential component for planning with incomplete information (Smith & Weld 1998). Extensions of the planning graph concept for dealing with conditional effects have been considered in (Koehler et al. 1997) and (Anderson, Smith, & Weld 1998). These proposals differ in their semantics, and while valuable computationally, they lack a clear theoretical justification. Here we follow the semantics proposed by Anderson, Smith, and Weld (from here on ASW), and derive the ‘planning graphs’ from the iterated PI-k sets obtained from the corresponding stratified theory.

We represent a planning problem involving actions with conditional effects by a tuple \( P_c = (A, \Omega, I, G) \), with \( A, I, \) and \( G \) as before, and \( \Omega \) as a set of actions \( \alpha_c \), each represented by a precondition list \( prec(\alpha_c) \) and a set of conditional effects \( \{cond^d(\alpha_c), add^d(\alpha_c), del^d(\alpha_c)\} \), \( i = 1, \ldots, n_{\alpha_c} \), with \( prec(\alpha_c), cond^d(\alpha_c), add^d(\alpha_c), del^d(\alpha_c) \) all being sets of atoms (possibly empty). Actions with conditional effects are like Strips actions but with add, delete, and precondition lists that depend on the state \( s \) where the actions are applied, given by the active effects and conditions respectively. The semantics of parallel planning given by ASW follows from this correspondence; in particular, two different actions \( \alpha_c \) and \( \alpha'_c \) interfere in a state \( s \) when in the state \( s \), one action deletes a precondition or add-effect of the other. Likewise, a valid parallel plan is a sequence of sets of applicable and non-interfering actions that map the initial situation \( I \) into the goal \( G \). For simplicity, we will assume that the various conditional effects of the same action are not in conflict.

For the encoding of the problem \( P_c \), we follow the intuition of ASW and convert it into a normal Strips planning problem with negation \( P \) augmented with a number of constraints. The resulting set of actions \( O \) in \( P \) is given by the components of \( P_c \): namely, for each action \( \alpha_c \) with conditional effects \( \{cond^d(\alpha_c), add^d(\alpha_c), del^d(\alpha_c)\} \), \( i = 1, \ldots, n_{\alpha_c} \), \( O \) will contain the Strips actions \( \alpha' \) with preconditions \( prec(\alpha_c) \cup cond^d(\alpha_c) \), and add and delete lists \( add^d(\alpha_c) \) and \( del^d(\alpha_c) \). We say in that case that action \( \alpha' \) is of type \( \alpha_c \), meaning that action \( \alpha' \) in \( P \) comes from action \( \alpha_c \) in \( P_c \).

The constraints must express that actions of the same type in \( P \) are not independent; namely, that the execution of an action \( \alpha \in O \) in a state \( s \) implies the execution of all the actions \( \alpha' \in O \) of the same type whose preconditions hold in \( s \). This means that in order to encode \( P_c \) in logic, in addition to the encoding of the Strips problem \( P \) with negation, we need for every pair of actions \( \alpha \) and \( \alpha' \) of the same type in \( P \), and all \( i = 0, \ldots, n - 1 \), the clauses

\[
\bar{p}_i^1 \lor \ldots \lor \bar{p}_i^r \lor \neg a_i \lor a'_i
\]

where \( p_1^1, \ldots, p_r^r \) are the atoms in \( prec(\alpha') - prec(\alpha) \), and \( \bar{p}_1^1, \ldots, \bar{p}_r^r \) are the atoms encoding their negation.\(^3\)

\(^3\)Using the literals \( \neg p \) instead of the atoms \( \bar{p} \) would yield an incorrect encoding for the reasons discussed above; namely, the monotonicity property would not hold, and hence, the theory may have ‘abnormal’ models that cannot be extended into ‘normal’ models validating the same plans.
resulting encoding and consider its compilation. The stratification works out as for normal Strips theories, with the addition of the clauses (7) indexed as transition clauses in the action layers. With the addition of these new clauses, however, the resulting stratified theory is no longer compiled. If as before, we restrict our attention to the negative clauses, the compilation yields the following conditional mutexes:

\[ p^1_i \lor \ldots \lor p^r_i \lor q^1_i \lor \ldots \lor q^r_i \lor -a_{i+1}^1 \lor -a_{i+1}^2 \]  

(8)

for actions \( a^1 \) of the same type as \( a \), and \( a^2 \) of the same type as \( a' \), such that \( a \) and \( a' \) interfere, and \( p^1, \ldots, p^r = \text{pre}(a) - \text{pre}(a^1) \) and \( q^1, \ldots, q^r = \text{pre}(a') - \text{pre}(a^2) \). That is, if \( a \) and \( a' \) interfere, and \( a^1 \) and \( a^2 \) are of the same type as \( a \) and \( a' \) respectively, then \( a^1 \lor a^2 \) is a mutex unless the conditions that make \( a^1 \lor a^2 \) trigger \( a \) and \( a' \) respectively do not hold.

The compiled theory \( T \) can be fed to the PI-k procedure for computing the iterated PI-k sets \( \Gamma^k_i(T) \). The complexity of this computation, as before, is given by the support width of the theory; namely the max number of transition clauses \( C_i \lor C_{i+1} \) with disjunctive bodies \( C_i \) with a common literal \( l_{i+1} \) in their heads. While for Strips, however, this parameter is equal to 1, for these theories, this parameter is not bounded and hence the procedure may run in exponential time.

For example, consider two actions \( a_0 \) and \( b_0 \) with 'conflicting' conditional effects; e.g., for \( i = 1, \ldots, r \), if \( c_i \) is true, then \( a_0 \) adds \( c_i \), and if \( d_i \) is true, \( b_0 \) deletes \( c_i \). The above encoding yields a number of conditional mutexes of the form \( c^2_i \lor d^2_i \lor -a_{i+1}^1 \lor -a_{i+1}^2 \) which is quadratic in \( r \) and hence the support width of the theory becomes quadratic in \( r \) as well. The extension of the planning graph in (Anderson, Smith, & Weld 1998) is polynomial but incomplete. The polynomial procedure PI-k-m discussed above that considers at most \( m \) disjunctive supports at a time yields a stronger approximation for the same \( k = 2 \) and \( m = 2 \). In any case, stratified theories, the notion of compilation, and the various projection procedures seem to provide a convenient framework for studying and evaluating such tradeoffs.

**Positive Deductive Lower Bounds**

We have considered the definition and computation of lower bounds for planning theories \( T \) considering the first time point \( i \) at which the encoding \( G_i \) of a goal \( G \) is consistent with the theory, or equivalently, the first time point \( i \) at which the negation of the goal \( \overline{G} \) is not entailed by \( T \). We called these lower bounds negative in Section 3 for this reason. We want to consider now positive lower bounds defined in terms of the first time point \( i \) at which the goal \( G_i \) itself is entailed, rather than its negation, from an slightly different and stronger theory \( T' \). We will consider one such definition for the Strips setting.

For a given Strips planning problem \( P \), we consider the stratified encoding \( T' \) of the delete-relaxation of \( P \) where all delete effects are assumed empty. This is a common and useful relaxation in planning that has given rise to a number of informative heuristics. The stratified theory \( T' \) is thus given by the following clauses:

1. **Init** \( T_0 \): \( p_0 \) for \( p \in I \), and \( \neg q_0 \) for \( q \in A \) not in \( I \)
2. **Action Layers** \( T_{i+1} \): for \( i = 0, 2, \ldots, n - 2 \\
   \bullet p_i \lor \neg a_{i+1} \) for each \( a \in O \) and \( p \in \text{Prec}(a) \)
3. **Propositional Layers** \( T_{i+1} \): for \( i = 1, 3, \ldots, n - 1 \\
   \bullet \neg a_i \lor p_{i+1} \) for each \( a \in O \) and \( p \in \text{Add}(a) \), and
   \bullet \neg a^1_i \lor a^2_i \lor \ldots \lor a^m_i \lor \neg p_{i+1} \) for each \( a \in A \) and the actions \( a^j \in O \) that add \( p \)

Since delete clauses and mutex constraints are no longer part of the theory, the theory \( T' \) is weaker than the theory \( T \) encoding the original problem \( P \), and thus we cannot expect it to yield positive lower bounds. We thus build our target theory \( T'' \) from \( T' \) by forcing the equivalences \( \text{pre}(a) \equiv a_{i+1} \), where \( \text{pre}(a) \) is the formula encoding the preconditions of \( a \) at level \( i \).

4. **Action Closure**: \( \neg p_i \lor a_{i+1}, i = 0, 2, \ldots, n - 2 \)

The resulting stratified theory \( T'' \) is compiled as the resolutions upon \( x_{i+1} \) variables in each layer \( T_{i+1} \) all yield clauses that include complementary literals. As a result, the theory is causally consistent, and since \( T_0 \) is consistent, the theory is consistent as well. Moreover, the theory \( T'' \) is complete in the sense that it has a single state trajectory \( s_0, s_1, \ldots, s_n \) satisfying it. Thus, for any variable \( x_i \), either \( x_i \in P I_i(T') \) or \( \neg x_i \in P I_i(T') \). This also implies that these sets are equivalent to the iterated PI-k sets \( \Gamma^k_i(T') \) for any \( k \geq 1 \) which are also complete. Hence for any goal \( G_i \in L_i \), \( \Gamma^k_i(T') \) either entails \( G_i \) or \( \neg G_i \). Moreover, it can also be shown that the sets \( \Gamma^k_i(T') \cap L^- \) are equivalent to the sets \( \Gamma^k_i(T) \cap L^- \) obtained from the theory \( T \) encoding the original problem \( P \) with \( k = 1 \). It follows then that \( \Gamma^1_i(T) \neq \Gamma^k_i(T') \) iff \( \Gamma^k_i(T) \neq \Gamma^k_i(T') \) iff \( \Gamma^k_i(T') \neq \Gamma^k_i(T) \). Thus, the positive lower bounds obtained from \( T'' \) are equivalent to the negative lower bounds obtained from the original theory \( T \) for \( k = 1 \), and hence that the former bounds are weaker than the latter for values of \( k \) greater than 1.

Positive lower bounds, however, may be useful in non-Strips settings. Recently, for example, an heuristic estimator for conformant planning has been proposed in (Brafman & Hoffmann 2004) which seems to be based on a closed, relaxed theory like \( T'' \) that allows for disjunctive information in the initial situation \( I \). However, rather than computing a representation of the sets \( \Gamma^k_i(T') \) using a procedure like \( \text{PI-k} \) of Brafman and Hoffmann rely on an approximation based on a compilation into 2-CNF; a closed and tractable fragment of CNF including the clauses with two literals at most. Indeed, any theory \( T'' \) stronger than \( T'' \) would yield positive lower bounds, and in particular, if \( T'' \) is in 2-CNF, the computation of these lower bounds is tractable. In this sense, the notion of lower bounds used in knowledge compilation, that refers to stronger theories (Selman & Kautz 1996), may turn out to be useful for
computing lower bounds in planning. The ideal compila-
tion target for a theory like $T^+$ in situations where the
domain features incompleteness or non-determinism, is
the weakest theory $T^+_{\forall}$ over the target tractable
language (whether 2-CNF, Horn, etc) that is as strong as
$T^+$; the so-called greatest lower bound of $T^+$. However, since such a
compilation is intractable in general, one must settle for a
good lower bound theory rather than for the best one.
The scheme developed by Brafman and Hoffmann follows this
approach building the corresponding 2-CNF theory incre-
mentally. The reasons they don’t get a lower bound is that,
as in FF, they do not use the index of the layer where the
goals become derivable but the number of actions involved
in the derivation; an heuristic that appears more informa-
tive but is not admissible.

Related Work
The ideas in this work are closely related to the idea of
variable or bucket elimination as formulated in (Dechter
1999). In a CSP context, variable elimination algorithms
process the problem variables in a fixed order $x_1, \ldots, x_m$
eliminating the variables one by one. The elimination of a
variable $x_i$ in a problem $P_i$ induces a constraint $\Gamma_{i+1}$ over
the remaining variables $x_{i+1}, \ldots, x_m$, which is added to the
existing constraints to yield a subproblem $P_{i+1}$. The
nature of this subproblem is such that its solutions can all
be extended to solutions of the problem $P_i$ involving vari-
able $x_i$ as well. Thus after eliminating all variables, all
solutions to the original problem $P = P_0$ can be obtained
backtrack free in reverse order starting with the subprob-
lem $P_n$ involving only the variable $x_n$. The computation of
the constraints $\Gamma_{i+1}$ induced by the elimination of a
variable, however, may be exponential in time and space.
A tractable alternative is a bounded form of variable elim-
ination in which the elimination of a variable $x_i$ induces
constraints of size at most $k$ among the remaining vari-
ables $x_{i+1}, \ldots, x_n$. Such weak form of elimination runs
in polynomial time but the backward search for solutions is
no longer backtrack free: the constraints that are inferred
and posted as variables are eliminated some but not
necessarily all the backtracks. This is actually what
Graphplan does: a forward weak variable elimination pass
that induces constraints of size 1 and 2, followed by a
backtracking search over the resulting graph starting from
the last layer. The inference scheme based on the compu-
tation of the iterated $\Gamma^k(T)$ sets by means of the PI-k pro-
cedure provides a generalized logical account of the com-
putations performed by Graphplan in the weak elimination
pass. We have shown the conditions under which the PI-
k procedure runs in polynomial time and computes these
sets exactly.

Variable elimination algorithms for SAT take the form
of Directed Resolution (DR) where variables $x_i$ are elim-
inated by computing the resolvents of clauses $C_i$ and
$C_j'$ involving the literal $x_i$ and the literal $\neg x_i$ respec-
tively, inducing new clauses over the remaining variables
$x_{i+1}, \ldots, x_n$ (Rish & Dechter 2000). Bounded Directed
Resolution, is the bounded version of DR which runs
in polynomial time and only posts resolvents of size no
greater than $k$. For example, Bounded DR-2, generates
resolvents of size 1 and 2 only. In this sense, BDR-2 is
related to Graphplan, and more generally, BDR-k is rel-
lated to the inference scheme based on the compu-
tation of the iterated PI-k sets that also derives consequences of
size $k$ or less. The two inference methods, however, are
not equivalent; the latter method is stronger, and is poly-
nomial only under certain conditions (such as those ex-
pressed in Theorem 7). Indeed, while the computation of
the PI-k sets results in clauses of size bounded by $k$, it may
involve intermediate resolution steps producing clauses of
arbitrary size. For example, consider two frame axioms
$a_i^1 \lor \cdots \lor a_i^k \land \neg p_{i+1}$ and $b_i^1 \lor \cdots \lor b_i^k \land \neg q_{i+1}$ along
with the ‘mutex’ clauses $\land a_i^j \lor \neg b_i^j$ for each $a^j$ and
$b^j$. In such a situation, the PI-k procedure for any $k \geq 2$ in-
er the ‘mutex’ $\neg p_{i+1} \lor q_{i+1}$, yet the first level resolvents of
these clauses have all size $r$ which may be much higher
than $k$. Such inferences are common in planning and are
critical in the construction of the planning graph.

The computation of the sets $\Gamma^k(T)$ stands thus halfway
between full and bounded forms of variable elimination:
it involves full resolution over the $x_i$ variables but results
in clauses of size bounded by $k$ over the $x_{i+1}$ variables.
Without the restriction of the $k$ parameter, these sets be-
come equal to the sets of prime implicates of $T$ over the
languages $L_i$, i.e., $\Gamma_i(T) = PI(T)$, an intractable, com-
piled representation that corresponds to the one obtained
by the clause-based tree-clustering algorithm in (Rish &
Dechter 2000).

Other works on propositional inference methods that
exploit the structure of the underlying theories, are (Dar-
wiche 1996), (del Val 1999), (Kohlas, Haenni, & Moral
1999), and (McIlraith & Amir 2001).

Discussion
We have formulated an inference scheme over stratified
propositional theories, and have shown the conditions un-
der which this inference scheme is tractable and com-
putes exactly the iterated sets of prime implicates of size
bounded by $k$. We have also established a corresponden-
t between this computation and the construction of the
planning graph, and more generally, the derivation of lower
bounds in planning. The scheme is also related to bounded
variable elimination methods considered in constraint sat-
sification and optimization.

We have focused only in ‘classical’ planning theories
where the initial state is known and actions, with or with-
out conditional effects, are deterministic. The account,
however, may provide a basis for defining and deriving
cost-effective lower bounds in non-classical planning set-
tings such as those involving incomplete information and
non-deterministic actions. The use of Graphplan as a front
end of current SAT and CSP planners indicates that in the
classical setting, the proposed PI-k inference methods pro-
vide indeed a cost-effective preprocessing filter. In more
complex planning tasks that cannot be reduced to SAT, we expect the leverage gained by the use of these inference methods to be even greater.

As a concrete example of the challenges involved in the definition and derivation of lower bounds in problems that involve incomplete information in the initial state, consider a robot that can move deterministically, one step at a time in the four directions, in a square grid of side $n$. Moves that would take the robot out of the grid leave the robot in the same location. The robot does not know its initial location and the goal is to get to the center of the square with certainty. Any optimal plan for the problem involves a self-localization stage where the robot takes $n$ steps in one direction, and $n$ steps in an orthogonal direction. The robot will be then at a particular corner of the grid and will know its location, and from there, it can reach the goal in $n$ steps more. The problem has thus cost equal to $3n$. The challenge is to have a tractable and general inference scheme capable of deriving such bounds.

The ideas we have laid out above are not sufficient for handling these situations, yet some of them may turn out to be relevant. A possible way for approaching the derivation of lower bounds in such settings may involve the computation of positive lower bounds over encodings in a propositional modal logic that distinguishes atoms being true from atoms being known. This will be an interesting possibility, as if true, it would lead naturally to further connections between propositional and problem solving methods.

References

---

4 This problem is from (Cimatti & Roveri 2000), which in turn is a variation of a problem from (Parr & Russell 1995).