Planning Graphs and Propositional Clause-Learning

Jussi Rintanen
NICTA and the Australian National University
Canberra, Australia

Abstract

The planning graph of Blum and Furst is one of the frequently used tools in planning. It is a data structure which can be visualized as a bipartite graph with state variables and actions as nodes and which approximates (upper bound) the set of reachable states with a given number of sets of simultaneous actions.

We show that the contents of planning graphs follow from two more general notions: extended clause learning restricted to 2-literal clauses and the representation of parallel plans consisting of STRIPS actions in the classical propositional logic. This is the first time planning graphs have been given an explanation in terms of the inference methods used in SAT solvers. The work helps in bridging the gap between specialized algorithms devised for planning and general-purpose algorithms for automated reasoning.

Introduction

Much of the development of efficient general-purpose reasoning algorithms for problems such as SAT is driven by restricted but efficient inference methods. Examples of such methods are conflict-directed clause learning (Bayardo, Jr. and Schrag 1997; Marques-Silva and Sakallah 1996) and unit propagation look-ahead (Li and Anbulagan 1997), both of which are based on unit resolution.

One early important application of SAT to "real-world" problems was planning as satisfiability (Kautz and Selman 1996). Most of the work on planning in the SAT context has concentrated on finding more efficient ways of expressing the planning problems as a propositional formula. Much less research has focused on looking at the SAT algorithms directly and tried to see why SAT algorithms are successful in solving planning problems and how the SAT algorithms could become still better.

Many ad hoc inferences performed by planners as a preprocessing step, such as testing whether all preconditions of a STRIPS action could be made true, can be derived from the logic representation of the planning problem with repeated application of the unit resolution rule: if l is initially false and none of the actions making l true is applicable in the initial state, the unit resolution rule allows inferring that l will be false in the successor state as well, and so on. Clearly, for any planner that uses a SAT solver to do search such specialized preprocessors are superfluous.

However, there are some specialized inference methods for planning that clearly fall in the domain of logical inference, but that are still implemented by non-logical ad hoc means. An interesting research problem is to understand why this is the case and how general-purpose reasoning methods or their implementations can be modified to cover the specialized methods.

In this work we address one class of specialized inference methods which has been vital for the efficiency of a number of approaches to classical planning such as the GraphPlan algorithm (Blum and Furst 1997) and Planning as Satisfiability (Kautz and Selman 1992; 1996) and its adaptations to other formalisms than SAT.

Planning graphs (Blum and Furst 1997) represent an (over) approximation of the reachable states of a planning problem. They are closely related to the notion of *invariants* which are facts that are guaranteed to hold in all reachable states of a planning problem (Gerevini and Schubert 1998). Geffner (2004) has shown how the contents of the planning graph logically follow from the representation of the planning problem in the propositional logic and can be inferred by logical inference methods such as binary resolution in polynomial time.

Our work takes a different look at planning graphs, motivated by the current SAT solving technology. In particular, we are interested in understanding why it is necessary to use planning graphs or related reachability information when using a SAT solver for finding plans, why current SAT solvers are not able to make the corresponding inferences automatically, and how the inference methods of SAT solvers could be extended to make constructions such as planning graphs obsolete. Our results carry over to invariants (Gerevini and Schubert 1998; Rintanen 1998) which contain the same kind of reachability information as planning graphs, but restricted to its time-independent part that holds for all reachable states. For the purposes of efficient planning, invariants are sufficient and have been very successfully used in place of planning graphs for notions of parallel plans to which planning graphs do not apply (Rintanen, Heljanko, and Niemelä 2006).

In this work, we show that the planning graph construction is covered by a form of *clause learning*. Clause learning is a general purpose inference method (Marques-Silva and Sakallah 1996) that is used by many of the most efficient SAT solvers, often as their sole inference method.

Since planning graphs contain only literals and binary mutexes, it is sufficient to use a restricted form of clause learning for 2-literal clauses for constructing them. Unlike the standard form of clause learning (Marques-Silva and Sakallah 1996) which involves assigning values to variables and then performing unit resolution, it turns out that, for our purposes in this paper, it is necessary to replace unit resolution by a stronger inference method called unit propagation look-ahead (Li and Anbulagan 1997). While this inference method is rarely used in practice, this discovery, the discrepancy between the standard clause learning procedure and the requirement posed by planning graphs, both explains why planning graphs and similar reachability information such as invariants (Gerevini and Schubert 1998; Rintanen 1998) have been useful for SAT/CSP-based planning and suggests stronger inference methods SAT/CSP-based planning would benefit from. Since the inferences performed during the construction of the planning graph and other planning-specific methods such as invariant algorithms often seem a practical prerequisite for efficient planning, it would seem worthwhile to investigate more efficient tractable inference methods in the context of planning in more depth.

The structure of the paper is as follows. The first two sections explain the prerequisite concepts of planning graphs and encodings of the classical planning problem in the propositional logic. Then we outline the polynomial-time inference algorithms that are the basis of some of the most efficient SAT solvers. The following section explains how the construction of planning graphs can be understood in terms of restricted propositional inference. The last three sections relate the results to earlier work, including long-distance mutex constraints, and conclude the paper.

Preliminaries: Planning and Planning Graphs

Planning has been traditionally formalized in terms of an initial state I, a finite set A of actions (c, a, d) where c, d and a are sets of state variables, and a goal. The initial state I is represented as the set of state variables that are initially true. V is the finite set of all state variables in the problem. In an action (c, a, d) the *precondition* c requires that the state variables c are true for the action to be possible, the *add list* a tells which state variables become *true*, and the *delete list* d tells which state variables become false when the action is taken. For an action $o = (c, a, d)$ we define $\mathrm{prec}(o) = c$, $\mathrm{add}(o) = a$ and $\mathrm{del}(o) = d$.

The GraphPlan algorithm and some other approaches to planning use the notion of *parallel plans* in which more than one action can be taken simultaneously. The parallelism in this context does not mean temporal parallelism because the actions have to satisfy an independence condition that makes it possible to take the parallel actions independently of each other in any order. Two actions may be in parallel if they do not interfere. This notion of parallelism is useful because it avoids imposing a total ordering on actions which don't have to be ordered.

Definition 1 (Interference) *Actions* $o = (c, a, d)$ *and* $o' = (c', a', d')$ interfere *if* $o \neq o'$ *and* $d \cap (c' \cup a') \neq \emptyset$ *or* $d' \cap (c \cup a) \neq \emptyset$.

If the actions interfere, then taking one action disables the other action or different states are reached depending on which action is taken first.

Blum and Furst's planning graph construction (Blum and Furst 1997) uses special NOOP actions for expressing that state variables do not change. For a state variable v we have $\mathrm{NOOP}(v) = (\{v\}, \{v\}, \emptyset)$. NOOPs somewhat simplify the way the graph is defined but otherwise does not have an important role. Let $\mathrm{NOOP} = \{\mathrm{NOOP}(v) | v \in V\}$.

Mutexes $(v, v') \in M_i^v$ on state variables indicate that v and v' cannot both be true at time i. Mutexes $(o, o') \in M_i^a$ on actions indicate that not both actions o and o' can be taken at time i (because they interfere or not both preconditions can be true.)

Definition 2 *A* planning graph *consists of the sets* A_i *and* V_i *and the mutual exclusion (mutex) relations* M_i^a *and* M_i^v *defined as follows for all* $i \geq 0$.

$$\begin{aligned}
V_0 &= I \\
M_0^v &= \emptyset \\
A_0 &= \{(c, a, d) \in A \cup \mathit{NOOP} | c \subseteq V_0, (v, v') \notin M_0^v \\
&\quad \textit{for all } \{v, v'\} \subseteq c\} \\
M_0^a &= \{(o, o') \in A_0 \times A_0 | o \textit{ and } o' \textit{ interfere}\} \\
V_{i+1} &= \{v \in V | o \in A_i, v \in \mathit{add}(o)\} \\
M_{i+1}^v &= \{(v, v') \in V_{i+1} \times V_{i+1} | \\
&\quad (o, o') \in M_i^a \textit{ for all } \{o, o'\} \subseteq A_i \\
&\quad \textit{such that } v \in \mathit{add}(o) \textit{ and } v' \in \mathit{add}(o')\} \\
A_{i+1} &= \{(c, a, d) \in A \cup \mathit{NOOP} | c \subseteq V_{i+1}, (v, v') \notin M_{i+1}^v \\
&\quad \textit{for all } \{v, v'\} \subseteq c\} \\
M_{i+1}^a &= \{(o, o') \in A_{i+1} \times A_{i+1} | o \textit{ and } o' \textit{ interfere or} \\
&\quad (v, v') \in M_i^v \textit{ for some } v \in \mathit{prec}(o), v' \in \mathit{prec}(o')\}
\end{aligned}$$

Planning graphs have many interesting properties. We state only one which will be used in later proofs.

Lemma 3 *If for* $i \geq 1$ *we have* $(v, v') \in M_i^v \backslash M_{i-1}^v$, *then either* $v \notin V_{i-1}$ *or* $v' \notin V_{i-1}$.

Preliminaries: Planning as Satisfiability

Before the work by Kautz and Selman (1992; 1996) logical formalizations of planning were always based on deduction. A plan would correspond to a proof that a sequence of actions reaches the goals. Kautz and Selman had the idea of representing the planning problem as a satisfiability problem instead. Each plan corresponds to an assignment that satisfies a propositional formula.

Let the set of state variables be V and the set of actions be A. We consider a finite horizon length $T \geq 0$. The atomic propositions are v^t for $v \in V$ and $t \in \{0, \ldots, T\}$ and $N(o)^t$ for $o \in A$ and $t \in \{0, \ldots, T-1\}$ where $N(o)$ is a name for action o. Intuitively, these atomic propositions express the values of the state variables $v \in V$ at different time points and tell whether a given action $a \in A$ is taken at a given time point.

Atomic propositions are formulas, and if ϕ and ϕ' are formulas, then so are $\neg\phi$, $\phi \vee \phi'$ and $\phi \wedge \phi'$. The formula $\phi \rightarrow \phi'$ is a shorthand for $\neg\phi \vee \phi'$.

As is well known, every formula can be transformed into an equivalent formula $c_1 \wedge c_2 \wedge \cdots \wedge c_k$ where each of the conjuncts c_i is a *clause*, that is, a disjunction $l_1 \vee l_2 \vee \cdots \vee l_j$ of literals. The constant $\perp$ (*false*) can be viewed as *the empty disjunction* because $\phi \vee \perp$ is equivalent to ϕ (we will use this in defining the unit resolution rule). Literals l are atomic propositions p or negated atomic propositions $\neg p$. For a literal l we define its *complement* $\overline{l}$ by $\overline{p} = \neg p$ and $\overline{\neg p} = p$.

Parallel Encoding

Let I and A be the initial state and the actions, respectively.

The clause set $\mathcal{T}(i, i+1)$ which describes the possible changes of state variable values between times i and $i+1$ consists of the following clauses.

1. $N(o)^i \rightarrow v^i$ for all $o \in A$ and $v \in \text{prec}(o)$
2. $N(o)^i \rightarrow v^{i+1}$ for all $o \in A$ and $v \in \text{add}(o)$
3. $N(o)^i \rightarrow \neg v^{i+1}$ for all $o \in A$ and $v \in \text{del}(o)$
4. $(v^i \wedge \neg v^{i+1}) \rightarrow (N(o_1)^i \vee \cdots \vee N(o_n)^i)$ for all $v \in V$ where $o_1, \ldots, o_n$ are the actions such that $v \in \text{del}(o_j)$ (remember that the empty disjunction for the case $n = 0$ is defined as $\perp$)
5. $(\neg v^i \wedge v^{i+1}) \rightarrow (N(o_1)^i \vee \cdots \vee N(o_n)^i)$ for all $v \in V$ where $o_1, \ldots, o_n$ are the actions such that $v \in \text{add}(o_j)$
6. $\neg(N(o)^i \wedge N(o')^i)$ for all $\{o, o'\} \subseteq A$ such that o and o' interfere

The initial state is described by the clause set I_C consisting of the following unit clauses.

1. v^0 for all $v \in I$
2. $\neg v^0$ for all $v \in V \backslash I$

Note that we don't include information about the goal in the encoding. This is because the goal is not used when constructing the planning graph.

The above formulae are clauses modulo elimination of implications by $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$ and pushing negations in front of atoms by the De Morgan rule $\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi)$.

Now the planning problem with horizon length $T \geq 0$ can be encoded as $I_C \cup \bigcup_{i=0}^{T-1} \mathcal{T}(i, i+1)$.

This encoding has been used in some of the early works (Ernst, Millstein, and Weld 1997) and has sometimes been called a *state-based encoding* (Kautz and Selman 1996).

Serial Encoding

For plans that allow only one action in parallel we can use the most traditional form of frame axioms that spell out explicitly for every action which state variables change and which do not. This is the encoding first used for planning as satisfiability (Kautz and Selman 1992).

The formula for the possible transitions between two consecutive time points is denoted by $\mathcal{T}^s(i, i+1)$ and consists of the following clauses.

```
1: procedure UP($C$)
2: for each unit clause $l \in C$ and clause $\overline{l} \vee \phi \in C$ do
3:     $C := C \cup \{\phi\}$;
4: end do
5: return $C$;
```

Figure 1: An algorithm for Unit Propagation

1. $N(o)^i \rightarrow v^i$ for all $o \in A$ and $v \in \text{prec}(o)$.
2. $N(o)^i \rightarrow v^{i+1}$ for all $o \in A$ and $v \in \text{add}(o)$,
3. $N(o)^i \rightarrow \neg v^{i+1}$ for all $o \in A$ and $v \in \text{del}(o)$,
4. $(N(o)^i \wedge \neg v^i) \rightarrow \neg v^{i+1}$ for all $o \in A$ and $v \in V \backslash \text{add}(o)$,
5. $(N(o)^i \wedge v^i) \rightarrow v^{i+1}$ for all $o \in A$ and $v \in V \backslash \text{del}(o)$,
6. $\bigvee_{o \in A} N(o)^i$

Polynomial-Time Inference Algorithms

We present the tractable inference algorithms used in many SAT solvers that implement the DPLL procedure or some of its variants.

Unit Propagation

Unit propagation (see Figure 1) is the application of the unit resolution rule which infers from a unit clause l and a clause $\overline{l} \vee \phi$ a new clause ϕ (where ϕ may be $\perp$, a unit clause, or a non-unit clause), until no more new clauses can be obtained. It is an important component of SAT solvers that are based on the Davis-Putnam-Logemann-Loveland procedure (Davis, Logemann, and Loveland 1962), including recent efficient implementations such as zChaff and MiniSat.

Clauses inferred by unit propagation are all logical consequences of the clause set. If the empty clause $\perp$ is obtained (which is inferring $\perp$ from l and $\overline{l}$) then the original clause set was unsatisfiable.

Unit Propagation Look-Ahead

Unit propagation can be used as a component of many powerful and efficient methods for SAT solving, one of which is known as Unit Propagation Look-Ahead. Look-Ahead can be used as an inference method and as an informed heuristic estimator (Li and Anbulagan 1997). In this work we use it for inference only. Figure 2 contains the basic procedure, which involves adding a unit clause (a literal) l to a clause set and running the unit propagation algorithm UP($C \cup \{l\}$). If the empty clause $\perp$ is obtained, then $\overline{l}$ is a logical consequence of the clause set, and can be added to it.

Some of the basic properties of UPLA(C) we will later use are stated in the following lemma.

Lemma 4 *1. If $C \subseteq C'$ then UPLA(C) $\subseteq$ UPLA(C').*

2. UP(C) $\subseteq$ UPLA(C)

3. If $\{l_1, \ldots, l_n\} \subseteq$ UPLA(C) and $\overline{l_1} \vee \cdots \vee \overline{l_n} \vee l \in C$ then $l \in$ UPLA(C).

```
 1:  procedure UPLA($C$)
 2:  $C$ := UP($C$);
 3:  repeat
 4:    change := false;
 5:    for each literal $l$ do
 6:      if $\bot \in$ UP($C \cup \{l\}$) then
 7:        begin
 8:          $C$ := UP($C \cup \{\bar{l}\}$);
 9:          change := true;
10:        end
11:    end do
12:  until $\bot \in C$ or change = false;
13:  return $C$;
```

Figure 2: An algorithm for Unit Propagation Look-Ahead

```
 1:  procedure learn2l($C$)
 2:  $C$ := UPLA($C$);
 3:  repeat
 4:    change := false;
 5:    for each pair $l, l'$ of literals such that $l \neq \overline{l'}$ do
 6:      if $\{\bar{l}, \overline{l'}\} \cap C = \emptyset$
 7:        and $\bar{l} \vee \overline{l'} \notin C$
 8:        and $\bot \in$ UPLA($C \cup \{l, l'\}$)
 9:      then
10:        begin
11:          change := true;
12:          $C$ := UPLA($C \cup \{\bar{l} \vee \overline{l'}\}$);
13:        end
14:    end do
15:  until change = false;
16:  return $C$;
```

Figure 3: An algorithm for learning 2-literal clauses

Extended Clause Learning for 2-Literal Clauses

Clause learning in algorithms for testing the satisfiability of propositional formulae proceeds by setting the values of literals and performing unit resolution until a contradiction is obtained. If $l_1, \ldots, l_n$ are the literals that were set true, then $\overline{l_1} \vee \cdots \vee \overline{l_n}$ is a conflict clause. Similarly to the unit clauses found with UPLA (which can be thought of as a simple special case of clause learning), the conflict clause is a logical consequence of the original clause set. Efficient implementations of clause learning attempt to find a small subset of $l_1, \ldots, l_n$ that leads to a contradiction to obtain short conflict clauses.

We define an *extended* clause learning procedure restricted to 2-literal clauses in Figure 3. This procedure differs from the clause learning procedures employed by most SAT solvers in that it uses the UPLA procedure for deriving conflicts, instead of the UP procedure, and that it restricts to 2-literal clauses and goes systematically through all of them.

Algorithms for learning clauses of arbitrary lengths include a (relatively complex) relevance test for the literals to avoid learning clauses that are subsumed by a stronger/shorter clause that is logically entailed by C. In the 2-literal case this reduces to the test on line 6.

Lemma 5 *Let C be a set of clauses and $C' =$ learn2l(C). Then $C \models C'$.*

Proof: Sketch: If UPLA produces the empty clause $\bot$ from $C \cup \{l, l'\}$, then $\bar{l} \vee \overline{l'}$ is a logical consequence of the set. All clauses $\bar{l} \vee \overline{l'}$ added to the clause set have this property. $\square$

Clause learning restricted to 2-literal clauses is not capable of inferring all 2-literal clauses that are logical consequences. Often more complex inferences are needed.

The procedure in Figure 3 runs in polynomial time: there is a quadratic number of pairs of literals to consider, the number of iterations is proportional to the number of pairs of literals, and at each iteration one call to unit resolution is made which can be implemented to run in linear time (Dowling and Gallier 1984).

Resolution Proofs of Planning Graphs

In this section we give some intuitions about the resolution proofs involved in computing the contents of planning graphs by the tractable inference algorithms and the parallel encoding of planning. The algorithms are based on the unit resolution rule, the unit propagation look-ahead algorithm, and the clause learning algorithm.

The resolution proofs we outline here, or similar ones, can be obtained from the computations of those inference algorithms.

Reasoning by unit propagation directly yields conventional resolution proofs. Resolution proofs in look-ahead and in clause-learning are less explicit in the computation but can be extracted in a systematic way in polynomial time in the size of the clause set and the size of the derivation of the conflict-clause.

The resolution proof for the mutex $\neg v^i \vee \neg v'^i$ uses the unit clauses v^i and v'^i which are first resolved with the frame axioms for the two state variables,

$$v^{i-1} \vee \neg v^i \vee N(o_1)^{i-1} \vee \cdots \vee N(o_m)^{i-1}$$
$$v'^{i-1} \vee \neg v'^i \vee N(o'_1)^{i-1} \vee \cdots \vee N(o'_{m'})^{i-1}$$

as well as binary clauses that indicate that literals in $\{v^{i-1}, N(o_1)^{i-1}, \ldots, N(o_m)^{i-1}\}$ pairwise contradict literals in $\{v'^{i-1}, N(o'_1)^{i-1}, \ldots, N(o'_{m'})^{i-1}\}$, and hence the two frame axioms cannot be true simultaneously with v^i and v'^i. The binary clauses express the direct and indirect mutual exclusion relations between the actions in the two sets and between v'^{i-1} and the actions that make v^i true (symmetrically for v^{i-1} and v'^i.)

The proofs can be constructed in different ways but it is probably easiest to start from one of the frame axioms and resolve each of the literals with binary clauses until only the complement of one of the literals in the second frame axiom is left. This is done for each of the literals in the second frame axiom. Finally, the resulting unit clauses are resolved with the second frame axiom to obtain the empty clause.

Assume there are n actions that make v true and n' actions that make v' true. Then the number of resolution steps is

$$2 + (n+1)(n'+1) + (n'+1)$$

$\neg s^2 \vee \neg t^2$		
$\neg m^2_{s,t} \vee s^2$	$\neg m^2_{s,t} \vee t^3$	$\neg m^2_{s,t} \vee \neg s^3$
$\neg m^2_{s,u} \vee s^2$	$\neg m^2_{s,u} \vee u^3$	$\neg m^2_{s,u} \vee \neg s^3$
$\neg m^2_{t,s} \vee t^2$	$\neg m^2_{t,s} \vee s^3$	$\neg m^2_{t,s} \vee \neg t^3$
$\neg m^2_{t,u} \vee t^2$	$\neg m^2_{t,u} \vee u^3$	$\neg m^2_{t,u} \vee \neg t^3$
$\neg m^2_{u,s} \vee u^2$	$\neg m^2_{u,s} \vee s^3$	$\neg m^2_{u,s} \vee \neg u^3$
$\neg m^2_{u,t} \vee u^2$	$\neg m^2_{u,t} \vee t^3$	$\neg m^2_{u,t} \vee \neg u^3$
$\neg m^2_{s,t} \vee \neg m^2_{s,u}$	$\neg m^2_{t,s} \vee \neg m^2_{t,u}$	$\neg m^2_{u,s} \vee \neg m^2_{u,t}$
$\neg m^2_{s,t} \vee \neg m^2_{u,s}$	$\neg m^2_{s,u} \vee \neg m^2_{t,s}$	$\neg m^2_{t,s} \vee \neg m^2_{u,t}$
$\neg m^2_{t,u} \vee \neg m^2_{s,t}$	$\neg m^2_{u,s} \vee \neg m^2_{t,u}$	$\neg m^2_{u,t} \vee \neg m^2_{s,u}$
$s^2 \vee \neg s^3 \vee m^2_{t,s} \vee m^2_{u,s}$	$\neg s^2 \vee s^3 \vee m^2_{s,t} \vee m^2_{s,u}$	
$t^2 \vee \neg t^3 \vee m^2_{s,t} \vee m^2_{u,t}$	$\neg t^2 \vee t^3 \vee m^2_{t,s} \vee m^2_{t,u}$	
$u^2 \vee \neg u^3 \vee m^2_{s,u} \vee m^2_{t,u}$	$\neg u^2 \vee u^3 \vee m^2_{u,s} \vee m^2_{u,t}$	

Table 1: Clauses Formalizing the Example Problem

if only one mutex (binary clause) is needed to show that each pair of literals in the two frame axioms is incompatible. This is the best case which occurs when all action pairs' mutual exclusivity is explicit in one binary clause. In the worst case one may need three binary clauses per action pair (precondition axioms for both plus one fact mutex).

We illustrate the resolution proofs with a simple planning problem. The basic step in the computation of planning graphs (and invariants) is to establish that a mutex (v, v') that holds at a given time continues to hold at the next.

Example 6 Consider the planning problem in which one can move between locations s, t and u. This is formalized with actions

$$\begin{aligned} m_{s,t} &= (\{s\},\{t\},\{s\}), & m_{s,u} &= (\{s\},\{u\},\{s\}),\\ m_{t,s} &= (\{t\},\{s\},\{t\}), & m_{t,u} &= (\{t\},\{u\},\{t\}),\\ m_{u,t} &= (\{u\},\{t\},\{u\}), & m_{u,s} &= (\{u\},\{s\},\{u\}). \end{aligned}$$

The clauses formalizing these actions for time point 2 are listed in Table 1.[1] We construct a refutation proof for $\neg s^3 \vee \neg t^3$. Hence we can use the negation $s^3 \wedge t^3$ of this formula in the resolution proof. We assume that we have already derived the corresponding clause $\neg s^2 \vee \neg t^2$ for the previous layer of the planning graph.

We can derive $\neg m^2_{u,t}$ from the frame axiom for s as shown in Figure 4. The derivations of $\neg m^2_{s,t}$ and $\neg t^2$ are similar.

After deriving the literals $\neg m^2_{s,t}$, $\neg m^2_{u,t}$ and $\neg t^2$ the empty clause is obtained from the frame axiom for t by the obvious consecutive resolution steps (see Figure 5.) ■

Planning Graphs Reduced to Clause Learning

We will show that the computation of the mutexes in the planning graph is subsumed by the computation of 2-literal clauses by the extended clause learning algorithm for the parallel encoding of planning.

[1]This could have been any other time point just as well.

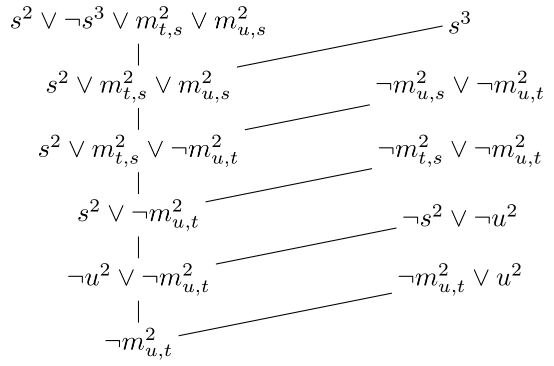
$s^2 \vee \neg s^3 \vee m^2_{t,s} \vee m^2_{u,s}$
s^3
$s^2 \vee m^2_{t,s} \vee m^2_{u,s}$
$\neg m^2_{u,s} \vee \neg m^2_{u,t}$
$s^2 \vee m^2_{t,s} \vee \neg m^2_{u,t}$
$\neg m^2_{t,s} \vee \neg m^2_{u,t}$
$s^2 \vee \neg m^2_{u,t}$
$\neg s^2 \vee \neg u^2$
$\neg u^2 \vee \neg m^2_{u,t}$
$\neg m^2_{u,t} \vee u^2$
$\neg m^2_{u,t}$

Figure 4: Derivation of $\neg m^2_{u,t}$

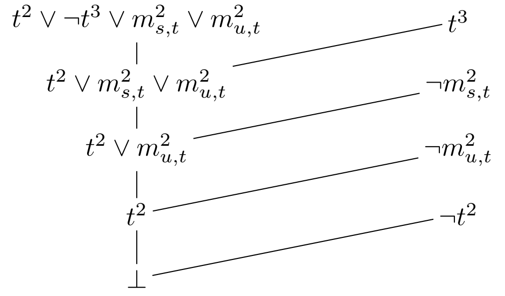
$t^2 \vee \neg t^3 \vee m^2_{s,t} \vee m^2_{u,t}$
t^3
$t^2 \vee m^2_{s,t} \vee m^2_{u,t}$
$\neg m^2_{s,t}$
$t^2 \vee m^2_{u,t}$
$\neg m^2_{u,t}$
t^2
$\neg t^2$
$\perp$

Figure 5: Refutation Proof for $\neg s^3 \vee \neg t^3$

Definition 7 *For a planning problem with a given horizon length $T \geq 0$ and actions A define the clause set $C = \{v^0 | v \in I\} \cup \{\neg v^0 | v \in V \backslash I\} \cup \bigcup_{i=0}^{T-1} \mathcal{T}(i, i+1)$ for expressing reachability in T steps.*

Theorem 8 *Let V_i and M^v_i for all $i \in \{0, \ldots, T\}$ and A_i and M^a_i for all $i \in \{0, \ldots, T-1\}$ form the planning graph as in Definition 2. Let $C_0 = C$ be the clause set from Definition 7. Let $C_i, i \geq 1$ be the sets obtained from C_0 inside learn2l(C_0) after trying out all pairs of literals v^i and v'^i on line 5 (following the construction of the planning graph level by level, with $C_{i-1} \subseteq C_i$.)*

Then for all $i \geq 0$ and $\{v, v'\} \subseteq V$ and $o \in A$,

1. *if $v \notin V_i$ then $\neg v^i \in$ UPLA(C_i),*
2. *if $(v, v') \in M^v_i$ then $\neg v^i \vee \neg v'^i \in C_i$, and*
3. *if $o \notin A_i$ then $\neg N(o)^i \in$ UPLA(C_i),*

Proof: We prove the claim by induction on i.

Base case $i = 0$:

1. For any $v \in V$, $v \notin V_0$ implies $v \notin I$ implies $\neg v^0 \in C_0 \subseteq$ UPLA(C_0).
2. There are no fact mutexes at the 0th level of the planning graph and the claim therefore trivially holds.
3. If $o \notin A_0$ then prec$(o) \not\subseteq V_0$, that is, there is $v \in$ prec(o) such that $v \notin I$. Hence $\neg N(o)^0 \vee v^0 \in C_0$ and $\neg v^0 \in C_0$. By unit resolution we have $\neg N(o)^0 \in$ UP$(C_0) \subseteq$ UPLA(C_0).

Inductive case $i \geq 1$:

1. Assume $v \notin V_i$.

   Hence NOOP$(v) \notin A_{i-1}$ and $v \notin V_{i-1}$ and by the induction hypothesis $\neg v^{i-1} \in$ UPLA$(C_{i-1}) \subseteq$ UPLA(C_i).

   This also implies that $o \notin A_{i-1}$ for any other action with $v \in$ add(o). Hence by the induction hypothesis we have $\neg N(o)^{i-1} \in$ UPLA$(C_{i-1}) \subseteq$ UPLA(C_i).

   Hence complements of all literals in the frame axiom $v^{i-1} \vee \neg v^i \vee N(o_1)^{i-1} \vee \cdots \vee N(o_n)^{i-1}$ except $\neg v^i$ are in UPLA(C_i), and therefore $\neg v^i \in$ UPLA(C_i).

2. Assume $(v, v') \in M_i^v$. We will show that $\perp \in$ UPLA$(C_{i-1} \cup \{v^i, v'^i\})$ from which the claim follows by the definition of *learn2l*. The proof is symmetric with respect to v and v', so we show one case only.

   We will be looking at the frame axiom

   $$(\neg v'^{i-1} \wedge v'^i) \rightarrow (N(o_1)^{i-1} \vee \cdots \vee N(o_n)^{i-1})$$

   where $o_1, \ldots, o_n$ are all the actions that make v' true. Expressed as a clause it is

   $$v'^{i-1} \vee \neg v'^i \vee N(o_1)^{i-1} \vee \cdots \vee N(o_n)^{i-1}.$$

   Let o be any action that makes v true. At some point the UPLA algorithm tries out $N(o)^{i-1}$, that is, sets it true.

   By definition of M_i^v, $(o, o') \in M_{i-1}^a$ for any action $o' \in A_{i-1} \backslash$NOOP such that $v' \in$ add(o'). Take any such o'. Hence either o and o' interfere and $\neg N(o)^{i-1} \vee \neg N(o)'^{i-1} \in C_{i-1}$, or for some $(v_p, v_p') \in M_{i-1}^v$ we have $v_p \in$ prec(o) and $v_p' \in$ prec(o') and hence $\{N(o)^{i-1} \rightarrow v_p^i, N(o')^{i-1} \rightarrow v_p'^i, \neg v_p^{i-1} \vee \neg v_p'^{i-1}\} \subseteq C_{i-1}$. Hence $\neg N(o')^{i-1} \in$ UPLA(C_i) for all $o' \in A_{i-1}$ such that $v' \in$ add(o'), and the complements of all the action literals in the frame axiom for v' are obtained by unit resolution from $N(o)^{i-1}$. Hence v'^{i-1} is the only unassigned literal in the frame axiom.

   By definition of M_i^v also $(o, \text{NOOP}(v')) \in M_{i-1}^a$, that is, the two actions interfere or have mutually exclusive preconditions, which means that either

   (a) $v' \in$ del(o) or

   (b) $v'' \in$ prec(o) and $(v'', v') \in M_{i-1}^v$ for some v''.

   In the first case we have $N(o)^{i-1} \rightarrow \neg v'^i \in C_{i-1}$ which directly leads to contradiction because we already had set both $N(o)^{i-1}$ and v'^i true. In the second case we have $\{N(o)^{i-1} \rightarrow v''^{i-1}, \neg v''^{i-1} \vee \neg v'^{i-1}\} \subseteq C_{i-1}$, from which we obtain $\neg v'^{i-1}$ by unit propagation, falsifying the remaining literal v'^{i-1} in the frame axiom and leading to contradiction.

   Because every action o that makes v true leads to a contradiction, the UPLA algorithm, for given $\{v^i, v'^i\}$, will add every such literal $\neg N(o)^{i-1}$ to the clause set. Hence the frame axioms for v^i and v'^i yield v^{i-1} and v'^{i-1} by unit resolution.

   It remains to be shown that $\perp \in$ UPLA$(C_{i-1} \cup \{v^i, v'^i\})$. We consider two cases.

   a) $(v, v') \in M_{i-1}^v$: By the induction hypothesis C_{i-1} contains $\neg v^{i-1} \vee \neg v'^{i-1}$. By unit resolution a contradiction is derived with the v^{i-1} and v'^{i-1} obtained from the frame axioms.

   b) $(v, v') \notin M_{i-1}^v$: By Lemma 3 this means that either $v \notin V_{i-1}$ or $v' \notin V_{i-1}$. By the induction hypothesis either $\neg v^{i-1} \in$ UPLA(C_{i-1}) or $\neg v'^{i-1} \in$ UPLA(C_{i-1}). In both cases a contradiction follows with unit resolution.

3. Assume $o \notin A_i$.

   Hence either prec$(o) \not\subseteq V_{i-1}$ or $(v, v') \in M_{i-1}^a$ for some $\{v, v'\} \in$ prec(o).

   Hence either $\neg v^{i-1} \in$ UPLA(C_{i-1}) for some $v \in$ prec(o) (by the induction hypothesis) or $\neg v^{i-1} \vee \neg v'^{i-1} \in C_{i-1}$ for some $\{v, v'\} \in$ prec(o) (by the previous case 2.) In both cases it follows that $\neg N(o)^i \in$ UPLA(C_i).

□

Why does the standard clause learning procedure not infer the mutexes?[2] To derive the mutex (v, v') at time $i+1$ as a refutation proof, we can use the frame axiom

$$v^i \vee \neg v^{i+1} \vee N(o_1)^i \vee \cdots \vee N(o_n)^i,$$

the frame axiom for v', and the clauses $\neg v^i \vee \neg v'^i$, v^{i+1} and v'^{i+1}. The only literal in the frame axiom that is available at a unit clause is v^{i+1}, and hence no unit resolution steps are possible.[3] Therefore no contradiction can be derived and the mutex cannot be inferred. To derive a contradiction it has to be shown that all possible ways of making v true are mutually exclusive of all possible ways of making v' true. The case analysis over different ways of making v true is performed by UPLA but not by UP.

Similar arguments about the weakness of the inference methods carry over to other known encodings of the classical planning problem in the classical propositional logic.

Encodings with a notion of plans differing from the parallel plans in the first encoding we gave are not, of course, directly related to planning graphs. The sequential encoding of planning, which allows at most one action per time point, can be shown to sanction similar inferences but an exact match with planning graphs does not exist.

Example 9 Consider actions $o_a = (\emptyset, \{a\}, \emptyset)$ and $o_b = (\emptyset, \{b\}, \emptyset)$ and an initial state in which both a and b are false. The planning graph contains both a and b on the first level but these facts are not mutex. But $\neg a^1 \vee \neg b^1$ is inferred by learn2l with the sequential encoding $\mathcal{T}^s(0,1)$: from unit clauses a^1 and b^1 and the initial state literals $\neg a^0$ and $\neg b^0$ one obtains by unit resolution respectively $\neg o_a^0$ and $\neg o_b^0$ which contradicts $o_a^0 \vee o_b^0$. ■

We do not show in this work but believe that the sequential encoding $\mathcal{T}^s(i, i+1)$ allows to infer the same invariants as some of the iterative invariant algorithms (Rintanen 1998).

[2]Some mutexes can be obtained by learning other (possibly non-binary) auxiliary clauses first as an intermediate step. We don't consider this possibility in this work.

[3]Assuming that there is at least one action that can make v true.

Discussion

So planning graphs are a specialized technique for inferring a class of 2-literal clauses. These clauses are very useful for speeding up plan search by satisfiability algorithms (Kautz and Selman 1996) but they are not systematically inferred by SAT solvers which is the reason why specialized algorithms have emerged.

General clause learning algorithms can infer some 2-literal clauses $l \vee l'$ in which l and l' say something about different time points. Consider a planning problem in which the only action making a true also makes b true and in which no action makes b false. Now $\neg a^t \vee b^{t+i}$ follows for any $t \geq 0$ and $i \geq 0$. Planning graphs don't say anything about such clauses but SAT solvers can learn such clauses nevertheless.

Planning graphs are not naturally definable for more general planning languages than STRIPS, and, the well-known tractable inference methods for the propositional logic do not infer the desired mutexes for such languages. In fact, inferring mutexes for languages that sanction arbitrary propositional formulae as preconditions is NP-hard because testing the possibility of simultaneous actions involves a satisfiability test. To circumvent this problem, polynomial-time approximations may be used.

Unit resolution, unit propagation look-ahead, and conflict-directed clause learning are the main inference methods used in the best systematic SAT solvers. There are other powerful polynomial-time inference methods that have been used as preprocessors but rarely as part of a SAT solver, including binary hyper-resolution (Bacchus and Winter 2004), simplification methods based on the implication graphs of binary clauses (Brafman 2001), and restricted forms of resolution (Subbarayan and Pradhan 2005). None of these preprocessors infer the contents of the planning graph. The binary hyper-resolution rule (Bacchus and Winter 2004) performs inferences that are close to those that are required (and, interestingly, can be implemented in terms of unit-resolution look-ahead which also turned out to be necessary to capture planning graphs with clause learning) but falls slightly short.

Long Distance Mutexes

Chen et al. (2007) propose a form of constraints they call *long distance mutual exclusion* or londex. Similarly to the mutexes in the planning graph, the londex constraints express that the truth of two state variables is mutually exclusive but possibly at different time points.

In this section we show that long-distance mutexes follow from the basic formalization of planning as satisfiability by unit resolution. This means that the efficiency gains Chen et al. obtain from long-distance mutexes are not because of some additional inferences long-distance mutexes would sanction (as those inferences are performed by any standard systematic SAT solver already) but because of some secondary effect of the additional constraints on the functioning of SAT solvers.

Chen et al. derive long-distance mutual exclusion constraints from graphs (N, E) that represent, for sets N of Boolean state variables of which exactly one is true at a time, how the values of the variables can change. These graphs are sometimes called Domain Transition Graphs (DTG). There is an edge $(v, v') \in E$ iff there is an action o that can be taken only if v is true and that makes v' true and v false, and variables in N are made true by actions of this form only. In typical planning benchmarks such sets express the difference possible locations of an object.

There is a long-distance mutex (v, d, v') if the shortest path in the graph from v to v' has length d. This means that one needs at least d actions to make v' true when starting from a state in which v is true, and hence $\neg v'$ must hold in all states that are reachable by less than d actions.

Identification DTGs (N, E) is typically based on detecting that $\neg v \vee \neg v'$ is an invariant for every $\{v, v'\} \subseteq N$ (Chen, Xing, and Zhang 2007), so we assume that these invariants are included in the encoding of the planning problem, as the planning graph or otherwise.

Our result about long-distance mutual exclusion constraints shows that the unreachability of $v \in N$ in a DTG can be inferred by unit resolution. For a variable $v \in N$, from v^t we can derive $\neg v'^t$ for all other variables $v' \in N$ by unit resolution from the invariants $\neg v^t \vee \neg v'^t$. In the inductive step we can use the frame axiom for v' to show that $\neg v'^{t+i}$ must hold for all $i < n$ when the distance of v' from v is n: we already have $\neg v'^{t+i-1}$, we show that none of the actions making v' true at $t + i$ can be taken, and finally infer $\neg v'^{t+i}$ from the frame axiom by unit resolution.

Theorem 10 *Let C be clauses encoding the planning problem and including invariants $\neg v_1 \vee \neg v_2$ for all DTGs (N, E) and $\{v_1, v_2\} \subseteq N$ such that $v_1 \neq v_2$. Let $v \in N$ be a variable in a DTG (N, E). Let $t \geq 0$ be an integer. Then for all variables $v' \in N$, if the shortest path from v to v' in the DTG has length $\geq n$, then $\neg v'^{t+i}$ can be derived by unit resolution from $C \cup \{v^t\}$ for all $i \in \{0, \ldots, n-1\}$.*

Proof: By induction on n. We show for variables further and further away from v that their negations are derivable for time points $< t + n$. The inductive case is proved by a nested induction proof.

Base case $n = 0$: The claim is trivially true, because $i \in \{0, \ldots, 0-1\}$ does not exist.

Inductive case $n \geq 1$: The proof is by induction on i.

Base case $i = 0$: By assumption $\neg v^t \vee \neg v'^t \in C$ and the claim follows immediately.

Inductive case $i \geq 1$ and $i < n$:

Let $v' \in N$ be a variable with a distance $\geq n$ from v in (N, E). By the induction hypothesis we can derive $\neg v'^{t+j}$ by unit resolution from $C \cup \{v^t\}$ for all $j \in \{0, \ldots, i-1\}$.

We show that $\neg v'^{t+i}$ is derivable by unit resolution from $C \cup \{v^t\}$. Every action o that makes v' true has a precondition v'' with distance $\geq n - 1$ from v. Hence by the outer induction hypothesis $\neg v''^{t+i-1}$ is derivable from $C \cup \{v^t\}$ by unit resolution and hence by the clause $N(o)^{t+i-1} \rightarrow v''^{t+i-1}$ we can further derive $\neg N(o)^{t+i-1}$.

Consider the frame axiom $v'^{t+i-1} \vee \neg v'^{t+i} \vee N(o_1)^{t+i-1} \vee \cdots \vee N(o_k)^{t+i-1}$ where $o_1, \ldots, o_k$ are all the actions that make v' true. We have derived all other literals in this clause by unit resolution from $C \cup \{v^t\}$ except $\neg v'^{t+i}$, and hence we can derive $\neg v'^{t+i}$ by unit resolution. $\Box$

Long-distance mutual exclusion constraints for actions follow from the long-distance mutual exclusion constraints for facts by unit resolution. This is a direct consequence of the way the long-distance mutexes for actions are defined: two actions have distance $\geq d$ if they have preconditions with distance d or they have effects with distance d. Let two actions o and o' respectively have the preconditions a and b. If the first action is taken at time t, then we can infer a^t from o^t and $o^t \rightarrow a^t$ by unit resolution, and as we have shown above, $\neg b^{t+i}$ follows by unit resolution from a^t for all $i \in \{0, \ldots, d-1\}$, and consequently we get o'^{t+i} from $o'^{t+i} \rightarrow b^{t+i}$ by a further unit resolution step. Distances between a precondition and an effect or vice versa similarly induce distance lower bounds for two actions.

Chen et al. (2007) show a speed-up in the running times of a SAT solver when the long-distance mutex constraints are included in the problem encoding. They claim that the constraints "effectively prune the search space". As we have shown above, the search space is not affected by the addition of these constraints. The remaining possible explanations for the speed-up include faster detection of contradictions by unit propagations through "short cuts" provided by the long-distance mutex constraints, more useful learned conflict-clauses, or changes in the variable selection heuristic due to the additional constraints. However, none of these explanations is directly related to the main idea of the long-distance mutexes, which strongly suggests that the same or better efficiency gains can be obtained by planning-specific modifications to the SAT solver.

Conclusions

We have shown that the *planning graph construction* of Blum and Furst can be viewed as an instance of an extended form of the clause learning inference method. An earlier work by Geffner (2004) gives a logical explanation of planning graphs, but does not use those tractable inference methods that are used by SAT solvers.

We believe that the approach adopted in this paper, reducing apparently planning specific algorithms and constructions to well-known general-purpose algorithms and inference methods, helps putting important planning techniques in the proper perspective and recognizing further avenues to powerful general-purpose planning techniques.

The work raises important questions about SAT algorithms and encodings of planning and other problems. Is it possible to bridge the gap between the strengths of our extended clause learning algorithm (which uses UPLA) and the standard clause learning algorithms (which use UP) by devising more powerful encodings of planning which sanction the inferences which currently require UPLA? Can current SAT solvers be efficiently strengthened to capture the kind of inferences for which researchers have felt compelled to devise specialized inference algorithms?

Acknowledgements

The research was funded by Australian Government's Department of Broadband, Communications and the Digital Economy and the Australian Research Council through NICTA and the DPOL project.

References

Bacchus, F., and Winter, J. 2004. Effective preprocessing with hyper-resolution and equality reduction. In Hoos, H. H., and Mitchell, D. G., eds., *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*, 341–355. Springer-Verlag.

Bayardo, Jr., R. J., and Schrag, R. C. 1997. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, 203–208.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):281–300.

Brafman, R. I. 2001. A simplifier for propositional formulas with many binary clauses. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 515–522. Morgan Kaufmann Publishers.

Chen, Y.; Xing, Z.; and Zhang, W. 2007. Long-distance mutual exclusion for propositional planning. In Veloso, M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1840–1845. AAAI Press / International Joint Conference on Artificial Intelligence.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem proving. *Communications of the ACM* 5:394–397.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* 1(3):267–284.

Ernst, M.; Millstein, T.; and Weld, D. S. 1997. Automatic SAT-compilation of planning problems. In Pollack, M., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1169–1176. Morgan Kaufmann Publishers.

Geffner, H. 2004. Planning graphs and knowledge compilation. In Dubois, D.; Welty, C. A.; and Williams, M.-A., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004)*, 662–672. AAAI Press.

Gerevini, A., and Schubert, L. 1998. Inferring state constraints for domain-independent planning. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 905–912. AAAI Press.

Kautz, H., and Selman, B. 1992. Planning as satisfiability. In Neumann, B., ed., *Proceedings of the 10th European Conference on Artificial Intelligence*, 359–363. John Wiley & Sons.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, 1194–1201. AAAI Press.

Li, C. M., and Anbulagan. 1997. Heuristics based on unit propagation for satisfiability problems. In Pollack, M., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 366–371. Morgan Kaufmann Publishers.

Marques-Silva, J. P., and Sakallah, K. A. 1996. GRASP: A new search algorithm for satisfiability. In *Proceedings of International Conference on Computer-Aided Design*, 220–227.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence* 170(12-13):1031–1080.

Rintanen, J. 1998. A planning algorithm not based on directional search. In Cohn, A. G.; Schubert, L. K.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, 617–624. Morgan Kaufmann Publishers.

Subbarayan, S., and Pradhan, D. K. 2005. NiVER: Non increasing variable elimination resolution for preprocessing SAT instances. In Hoos, H. H., and Mitchell, D. G., eds., *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT-2004. Vancouver, BC, Canada, May 10-13, 2004. Revised selected papers*, number 3542 in Lecture Notes in Computer Science, 276–291. Springer-Verlag.