This paper appears in the *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96), Boston, MA, 1996.*

Encoding Plans in Propositional Logic

Henry Kautz, David McAllester, and Bart Selman
AT&T Labs
600 Mountain Avenue
Murray Hill, NJ 07974
{kautz, dmac, selman}@research.att.com

Abstract

In recent work we showed that planning problems can be efficiently solved by general propositional satisfiability algorithms (Kautz and Selman 1996). A key issue in this approach is the development of practical reductions of planning to SAT. We introduce a series of different SAT encodings for STRIPS-style planning, which are sound and complete representations of the original STRIPS specification, and relate our encodings to the Graphplan system of Blum and Furst (1995). We analyze the size complexity of the various encodings, both in terms of number of variables and total length of the resulting formulas. This paper complements the empirical evaluation of several of the encodings reported in Kautz and Selman (1996). We also introduce a novel encoding based on the theory of *causal* planning, that exploits the notion of "lifting" from the theorem-proving community. This new encoding strictly dominates the others in terms of asymptotic complexity. Finally, we consider further reductions in the number of variables used by our encodings, by compiling away either state-variables or action-variables.

1 INTRODUCTION

In recent work we have shown that planning problems can be efficiently solved by general propositional satisfiability algorithms (Kautz and Selman 1996). On instances drawn from logistics and blocks-world planning problems, both systematic and local-search SAT algorithms often dramatically outperform earlier domain-independent planning systems. A key issue in this approach is the development of practical reductions of planning to SAT. Because both SAT and bounded-length STRIPS-style planning are NP-complete, it is evident that polynomial reductions between the problems exist. However, there are both relative and absolute practical limits on the size complexity of an acceptable reduction. For example, an $O(n^3)$ increase in the size of problem may be acceptable, while a reduction with an $O(n^4)$ may yield SAT instances that are too large to be solved by current algorithms. Of course, not all SAT problems of a given size are equally difficult, and the known results that quantify the hardness of randomly-generated SAT problems based on its size and clause-ratio (Mitchell, Selman, and Levesque 1992) cannot be directly applied to structured problems. However, experiments reported in Kautz and Selman (1996) using the SATPLAN system on problems derived from planning provide some rough guidelines. Formulas containing around 2,000 variables could be solved by both systematic and stochastic search in a few seconds. The limits of the systematic algorithm ("tableau", Crawford and Auton (1993)) were reached at 2,800 variables and 6 hours of running time. For the stochastic algorithm ("Walksat", Selman *et al.* (1994; 1996)), problems containing 6,700 variables were solved in 15 minutes; informal experiments show that the stochastic methods are currently feasible up to around 10,000 or so variables.

Kautz and Selman (1992) described one possible reduction of blocks-world planning to SAT, and some associated encoding techniques for reducing the number of variables in the SAT encoding. Kautz and Selman (1996) also showed that the "planning graphs" used by Blum and Furst's Graphplan system could be interpreted as propositional CNF formulas, and informally described an alternative reduction of planning to SAT called "state-based encodings". Both of these papers, however, concentrated on experiments in actually solving the resulting SAT formulas. In this paper, we focus on the reductions themselves. We present a analysis of the size of the encoded problems, noting the degree to which various properties of the original STRIPS problem (*e.g.*, the number of operators, the size of the domain, the length of the minimal solution, *etc.*) affect the size of the encoding under different reductions. This kind of analysis and comparison allows the gross statistical properties of the statement of a given planning problem to be used to select an efficient encoding.

The first reduction, "linear encodings", is most reminiscent of classical situation calculus representations of planning. Second, we show how the use of (the SAT equivalent of) "explanatory" frame axioms (Haas 1987; Schubert 1989) can both reduce the problem size and allow parallel actions to be efficiently encoded, in a manner very similar to the planning graphs of Blum and Furst. Two related encoding techniques, "operator splitting" and "lifting", are described, that reduce the number of variables required in the SAT reduction. Third, we present a new encoding of planning into SAT based on the lifted version of "causal plans" introduced by McAllester and Rosenblitt (1991). Lifted casual plan encodes have the best size complexity in the limit (although possibly with a larger constant factor).

Finally, we consider ways to reduce the size of an encoded problem by "compiling away" certain classes of variables. Because resolution can be used to eliminate any set of variables from a SAT problem, our encodings can be further transformed so that either they contain no variables referring to actions, or no variables referring to states. The first transformation gives an automated procedure for creating the "state-based encodings" that were created manually for the experiments reported in Kautz and Selman (1996). For the Graphplan system, we show that the worst-case size of the transformed formula is strictly better when state-variables are eliminated, rather than action-variables. However, these worst-case results should be tempered by the fact that we have, by hand, created state-based encodings of *particular* domains (*e.g.* logistics planning) that are much more compact than those given by general reductions. The results in this paper are dealt with in more detail in Kautz *et al.* (1996).

While this paper concentrates on reductions of planning to Boolean satisfiability, it is important to note that there is related work that views planning as general constraint satisfaction (*e.g.,* Joslin and Pollack 1995). We believe that the various techniques introduced in our reductions, such as operator splitting, lifting, and compilation, will also prove to be applicable in the more general CSP setting. The operation of our SATPLAN system can also be viewed as a form of refinement planning, as shown by Kambhampati and Yang (1996).

2 DEFINITIONS AND GENERAL FRAMEWORK

Planning problems are specified using standard STRIPS notation (Fikes and Nilsson 1971). A planning problem Π is a tuple $\langle \text{Ops}, \text{Dom}, S_0, S_1 \rangle$, where Ops is a set of operator definitions, Dom is a domain of individuals, and S_0 and S_1 are the initial and goal states, respectively. Operators are defined by schemas using precondition, add, and delete lists; for example:

```
MOVE(x, y, z)
  PRE: CLEAR(x), ON(x, y), CLEAR(z)
  ADD: CLEAR(y), ON(x, z)
  DEL: CLEAR(z), ON(x, y)
```

All of the variables that appear in the definition of an operator must appear in its head. (In particular, note that no negated atoms appear anywhere in this formalism.) The instantiation of an operator over the domain is called an *action* (*e.g.*, MOVE(A,B,C)), and the instantiation of a predicate is called a *fluent* (*e.g.*, ON(A,B)).[1] Given an operator definition, it is straightforward to define functions Pre(), Add(), and Del() that map an operator or an action to the corresponding lists of predicates or fluents; *e.g.*, Add(MOVE(A,B,C)) $=$ $\{$ON(A,C), CLEAR(B)$\}$. States are sets of fluents.

An action applied to a state adds or deletes the specified fluents from the state. A sequence of actions is a *solution* to a planning problem if it transforms the initial state into a superset of the goal state (the goal need only be partially specified), the precondition of each action appears in the state to which it applies, and no action both adds and deletes the same fluent. In a *bounded* planning problem, the solution must be length $\leq n$ for some fixed n. Given this notion of a planning problem, we can then define what it means to reduce a planning problem to SAT.

Definition: A function $\mathcal{T}$ which takes a planning problem Π and a length bound n and returns a SAT problem is said to *reduce bounded planning to SAT* provided that $\mathcal{T}(\Pi,\ n)$ is solvable by a plan with of at most n operations if and only if $\mathcal{T}(\Pi,\ n)$ is satisfiable. A SAT embedding is called constructive if a solution to Π can be efficiently extracted from any solution to $\mathcal{T}(\Pi,\ n)$ — more specifically if there exists a polynomial time operation $\mathcal{E}$ such that for any Π, n, and truth assignment σ satisfying $\mathcal{T}(\Pi,\ n)$, we have that $\mathcal{E}(\Pi,\ n,\ \sigma)$ is a solution to Π.

Each of the reductions described below is constructive. All the reductions are to CNF, which is the form used by most satisfiability-testing programs. (For clarity in the exposition we sometimes write down formulas which are not strictly CNF, but which can obviously put into CNF form with only a linear increase in size.) Whenever we talk of the "size" of a formula, we mean the total number of literals it contains. When we are considering clauses that are of fixed length, we can also talk about size in terms of the number of clauses without confusion.

For each reduction we also analyze the size of the CNF SAT encoding, as a function of various features of the planning problem instance. These features include the number of operators $|\text{Ops}|$, predicates $|\text{Pred}|$, domain elements $|\text{Dom}|$, and length bound n. The linear and Graphplan based reductions are also critically dependent on maximum number of arguments (*i.e.*, arity) of the operators and predicates, noted as $\mathcal{A}_{\text{Ops}}$ and $\mathcal{A}_{\text{Pred}}$ respectively. (The size of the lifted causal encoding is independent of the arity of the operators and predicates.) We abbreviate the expression for the number of actions, namely $|\text{Ops}||\text{Dom}|^{\mathcal{A}_{\text{Ops}}}$, as $|\mathcal{A}|$. Similarly, we abbreviate the number of fluents, $|\text{Pred}||\text{Dom}|^{\mathcal{A}_{\text{Pred}}}$, as

[1] For simplicity we assume a single domain over which all variables range, but it is easy to extend the definitions to allow typed domains and variables.

$|\mathcal{F}|$. Finally, we use m to denote the maximum combined length of the precondition, add, and delete lists for any operator.

3 LINEAR ENCODINGS

Linear encodings, introduced by Kautz and Selman (1992), are quite similar to a propositionalized version of the situation calculus (McCarthy and Hayes 1969), but with additional axioms included to rule out certain "unintended models" that are harmless for deductive formalizations, but problematic for the model-finding (SAT) approach. The intuition behind the encoding is that an additional time-index parameter is added to each action or fluent, to indicate the state at which the action begins or the fluent holds; for a problem bounded by n, the actions are indexed by 0 through $n-1$, and the fluents by 0 through n. Variables corresponding to dummy "null" action are also included: these handle the case where the solution to the planning problem is actually shorter than n. In brief, the reduction yields the following kinds of clauses:

1. The initial state is completely specified: if fluent $f \notin S_0$, then $\neg f_0$.[2] The goal state may be either fully or partially specified.
2. If an action holds at time i, its preconditions hold at i, its added fluents hold at $i+1$, and the negation of each of its deleted fluents holds at $i+1$.
3. Classical frame conditions hold for all actions (*i.e.*, if an action does not add or delete a fluent, then the fluent remains true or remains false when the action occurs).
4. Exactly one action occurs at each time instant (exclusiveness).

The correctness of the embedding can be shown by induction on n: propositions indexed by 0 exactly correspond to the given initial state, and those indexed by $i+1$ describe a legal state that can be reached from one described by propositions indexed by i. The function $\mathcal{E}$ simply extracts the sequence of action instances that are true in a satisfying truth assignment.

The number of variables used is $O(n|\mathcal{A}| + n|\mathcal{F}|)$. The size of the CNF formula (number of literal occurrences) is dominated by the clauses corresponding to the exclusiveness and frame axioms, and is thus $O(n|\mathcal{A}|^2 + n|\mathcal{A}||\mathcal{F}|)$. The most critical factor in determining whether this reduction is practical is clearly the arity of the operators and predicates. For many planning domains the predicate arity is 2 (*e.g.*, `ON`, `IN`, and `NEXT-TO`). Operators that take 3 or more arguments (*e.g.* `move`) generally make this reduction infeasible. We therefore consider two modifications to this reduction that shrink the number of variables and clauses.

[2]Note that this encoding makes an explicit closed-world assumption about the initial state. The STRIPS formalism itself does not necessarily require this assumption: STRIPS can be consistently interpreted so that the absence of a fluent from a state means that it is unknown, rather than false. However, this subtle difference does not actually change the set of solutions to a given problem, due to the restricted form of the operator definitions.

3.1 Operator Splitting

The first modification, operator splitting, is based on the observation that since only a single action occurs at a given time step, an m-place action can be represented as the conjunction of m 1-place actions. For example, instead of encoding "the action `MOVE(A,B,C)` occurs at time 3" as a single proposition "`MOVE(A,B,C,3)`", one could use an expression like "`SOURCE(A,3)` $\wedge$ `OBJECT(B,3)` $\wedge$ `DESTINATION(C,3)`". This technique can also be viewed as a special case of a "lifted" representation, that is, the use of propositions to represent bindings of the arguments of an operator or predicate. Lifting is used more extensively in the causal encodings described below. Operator splitting reduces the number of variables to $O(n\mathcal{A}_{\text{Ops}}|\text{Ops}||\text{Dom}| + n|\mathcal{F}|)$. The exclusiveness axioms become small, requiring only $O(n{\mathcal{A}_{\text{Ops}}}^2|\text{Ops}|^2|\text{Dom}|^2)$ binary clauses, because one can separately assert that each of the 1-place actions has a unique argument. Furthermore, in many cases operator splitting reduces the number of clauses corresponding to frame axioms, because not all arguments to an action may need appear in a frame axiom. Using this modification to the basic linear encodings, Kautz and Selman (1996) were able to solve blocks-world problems requiring 20 blocks and 20 moves, without using any kind of domain-specific search control knowledge.

3.2 Explanatory Frame Axioms

Haas (1987) and Schubert (1989) proposed an alternative to McCarthy and Hayes' style frame axioms, in the context of first-order deductive formalizations of planning. An "explanatory" frame axiom says that if the truth value of a fluent changes, then one of the actions that adds or deletes it must have occured. If none of those actions occurs, then by *modus tolens* the truth value of the fluent must persist through whatever other action does occur. These kind of frame axioms can be incorporated directly into the linear encoding framework. The basic schemas (which can be expanded into a set of clauses) are:

$$f_i \wedge \neg f_{i+1} \supset \quad \bigvee \{a_i | f \in \text{Del}(a_i)\}$$

$$\neg f_i \wedge f_{i+1} \supset \quad \bigvee \{a_i | f \in \text{Add}(a_i)\}$$

While classical frame axioms require $O(n|\mathcal{A}||\mathcal{F}|)$ clauses, this modification uses only $O(n|\mathcal{F}|)$ clauses. Unfortunately, each clause may be longer, and in the worst case, the total size of the formula is the same. However, in practice it appears that this formulation leads to smaller encodings, because the number of actions that could explain a given change is usually small, and so the clauses are of moderate length. Thus, if at most k actions could account for a change in the truth of a fluent, then the total size of the frame axioms is $O(nk|\mathcal{F}|)$.

When both operator splitting and explanatory frame axioms are employed, the size of the entire reduction is then dominated by the size of the frame axioms. We will discuss this issue in the expanded version of this paper (Kautz *et al.* 1996).

When explanatory frame axioms are used, one may optionally weaken the exclusiveness axioms, so that they assert that at *most* one action occurs at each time instance. Furthermore, it is no longer necessary to introduce "null" actions to account for solutions that are shorter than n. This is because the explanatory axioms imply that an action occurs whenever the (encodings of) two adjacent states differ. Thus, if two adjacent states are identical, then no action occurs at that instant.

4 PARALLELIZED ENCODINGS

So far, we have considered encodings in which only one action can occur at each time step. The number of time steps n occurs as a factor in both the number of variables and the total length of the encodings. An obvious way therefore to reduce the size of our encodings is by allowing several actions to occur at each time step, *i.e.*, parallel actions. The encodings below are such that if several actions occur at the same time, it means that they can be serialized in any order. Therefore the solution extraction function $\mathcal{E}$ for these encodings would identify the partially-ordered set of actions in a satisfying model and then arrange the actions in an arbitrary total order.

4.1 Graphplan-based Encodings

The Graphplan system of Blum and Furst (1995) works by converting a STRIPS-style specification into a planning graph. This is an ordered graph, where alternating layers of nodes correspond to grounds facts (indexed by the time step for that layer) and fully-instantiated operators (again indexed by the time step). Arcs lead from each fact to the operators that contain it as a precondition in the next layer, and similarly from each operator to its effects in the next layer. For every operator layer and every fact there is also a no-op "maintain" operator that simply has that fact as both a precondition and "add" effect.

A solution is a subgraph of the planning graph that contains all the facts in the initial and goal layers, and contains no two operators in the same layer that conflict (*i.e.*, one operator deletes a precondition or an effect of the other). Thus, a solution corresponds to a partially-ordered plan, which may contain several operators occuring at the same time step, with the semantics that those operators may occur in any order (or even in parallel). For planning problems that can take advantage of this kind of parallelism, the planning graph can have many fewer layers than the number of steps in a linear solution — and therefore be much smaller.

A planning graph is quite similar to a propositional formula, and in fact, we can automatically convert planning graphs into CNF notation. The translation begins at goal-layer of the graph, and works backward. Using the "rocket" problem in Blum and Furst (1995, Fig. 2) as an example (where "`LOAD(A,R,L,i)`" means "load `A` into `R` at location `L` at time `i`", and "`MOVE(R,L,P,i)`" means "move `R` from `L` to `P` at time `i`"), the translation is:

1. The initial state holds at layer 1 (fully specified), and the goals hold at the highest layer.
2. Operators imply their preconditions; *e.g.,*
$$\mathtt{LOAD(A,R,L,2)} \supset (\mathtt{AT(A,L,1)} \wedge \mathtt{AT(R,L,1)}$$
3. Each fact at level i implies the disjunction of all the operators at level $i-1$ that have it as an add-effect (backward-chaining); *e.g.,*
$$\mathtt{IN(A,R,3)} \supset (\mathtt{LOAD(A,R,L,2)} \vee \mathtt{LOAD(A,R,P,2)} \vee \mathtt{MAINTAIN(IN(A,R),2)})$$
4. Conflicting actions are mutually exclusive; *e.g.,*
$$\neg\mathtt{LOAD(A,R,L,2)} \vee \neg\mathtt{MOVE(R,L,P,2)}$$

The axioms of type (2) above assert that actions imply their preconditions, but not that actions imply their effects. This can admit solutions that contain spurious actions. The extraction function $\mathcal{E}$ can simply delete actions from the solution whose effects do not hold. The axioms for conflicting actions (4) prevent the solution from actually *depending* upon the co-occurance of actions whose effects would conflict.

The resulting CNF formula has $O(n|\mathcal{A}| + n|\mathcal{F}|)$ variables. The size of the formula is given by $O(nm|\mathcal{A}| + n|\mathcal{A}|^2 + nk|\mathcal{F}|)$. Note that is the same as the expression for the size of our linear encoding with explanatory frame axioms, but *without* operator splitting. This is consistent with the empirical results reported in Kautz and Selman (1996), where we observed that on the larger benchmark instances, the Graphplan-based encodings became too large for our satisfiability procedures. It should also be noted that these are worst-case bounds. In the worst-case, all operators are mutually exclusive, and therefore, we do not see an advantage in allowing parallel actions. In many practical domains, however, parallel actions lead to fewer "exclusiveness" axioms, fewer time steps (smaller n), and thus shorter encodings.

A very similar encoding for plans with parallelized actions can be generated by using the axioms for linear encodings with explanatory frame axioms (Section 3.2), but where the exclusiveness axioms only assert that conflicting actions are mutually exclusive, as in the Graphplan-based encoding. In fact, explanatory frame axioms can be generating by resolving together the backward-chaining axioms (3) above with the action/precondition axioms (2) for the "maintain" actions. For example, resolving the backward-chaining axiom:
$$\mathtt{IN(A,R,3)} \supset (\mathtt{LOAD(A,R,L,2)} \vee \mathtt{LOAD(A,R,P,2)} \vee \mathtt{MAINTAIN(IN(A,R),2)})$$

with the action/precondition axiom:

$$\texttt{MAINTAIN(IN(A,R),2)} \supset \texttt{IN(A,R,1)}$$

yields the the frame axiom:

$$(\neg\texttt{IN(A,R,1)} \wedge \texttt{IN(A,R,3)}) \supset (\texttt{LOAD(A,R,L,2)} \vee \texttt{LOAD(A,R,P,2)})$$

4.2 Compiling Away Actions or Fluents

Our linear and Graphplan encoding have a special structure in terms of the dependencies among variables. More specifically, we have alternating layers of variables representing operators and variables representing states (fluents). Since we are interested in reducing the number of variables as much as possible, we now consider whether we can "compile away" the action layers or the state layers. Given two consecutive states, it's generally straightforward to determine what action(s) led to the changes between the states. The action variables can thus be viewed as logically dependent variables. Similarly, the state variables can be taken as the dependent ones, since a set of actions applied at a certain state directly defines the changes in that state. So, we can compile away the dependent variables: either the state variables or the action variables.

First, note that for any propositional theory, *any* variable can be eliminated by performing all possible resolutions on that variable, and then removing all clauses containing the variable. Thus, any subset of propositions, whether they represent actions or fluents, can in principle be eliminated. The key question is what happens to the total number of clauses and the size of the clauses in the encodings. In general the resolution procedure gives an exponential blowup in size. For Graphplan, however, we know the following:

> **Observation:** For Graphplan-based encodings, compiling away actions can lead to exponential blowup in the size of the encoding, but compiling away states gives only a polynomial (in $|\mathrm{Dom}|$) increase in size.

Consider the result of resolving clauses between action- and state- layers. When resolving away action variables, we find that a given fluent, say, $\mathtt{s}_2$, can have been added by a number of different actions, *e.g.*, $\mathtt{a}_1^1$, $\mathtt{a}_1^2$, or $\mathtt{a}_1^3$. Each of these actions may have a different pair of pre-conditions, *e.g.*, $\mathtt{a}_1^1$ may require $\mathtt{s}_1^1$ and $\mathtt{s}_1^2$, while $\mathtt{a}_1^2$ requires $\mathtt{s}_1^3$ and $\mathtt{s}_1^4$, etc. We get $\mathtt{s}_2 \supset (\ \mathtt{a}_1^1 \vee \mathtt{a}_1^2 \vee \mathtt{a}_1^3\ )$. This gives us $\mathtt{s}_2 \supset ((\ \mathtt{s}_1^1 \vee \mathtt{s}_1^2\ ) \vee (\ \mathtt{s}_1^3 \vee \mathtt{s}_1^4\ ) \vee (\ \mathtt{s}_1^5 \vee \mathtt{s}_1^6\ ))$. Going back to CNF, we get clauses of the form $\mathtt{s}_2 \supset (\ \mathtt{s}_1^1 \vee \mathtt{s}_1^3 \vee \mathtt{s}_1^5\ )$, $\mathtt{s}_2 \supset (\ \mathtt{s}_1^2 \vee \mathtt{s}_1^3 \vee \mathtt{s}_1^5\ )$, etc. In general, we get exponentially many, *i.e.*, a worst-case blowup of $O(|\mathcal{F}|^{|\mathcal{A}|})$. More precisely, the blowup is $O(|\mathcal{F}|^{k})$, where k is maxium number of actions that could account for a change in the truth of a fluent.

However, to compile away states, *e.g.*, $\mathtt{s}_2^1$, $\mathtt{s}_2^2$, you simply go from $\mathtt{a}_2^1 \supset \mathtt{s}_2^1$, $\mathtt{a}_2^1 \supset \mathtt{s}_2^2$, $\mathtt{a}_2^2 \supset \mathtt{s}_2^1$, $\mathtt{a}_2^2 \supset \mathtt{s}_2^2$, and $\mathtt{s}_2^1 \supset (\ \mathtt{a}_1^1 \vee \mathtt{a}_1^2\ )$ to $\mathtt{a}_2^1 \supset (\ \mathtt{a}_1^1 \vee \mathtt{a}_1^2\ )$, $\mathtt{a}_2^2 \supset (\ \mathtt{a}_1^1 \vee \mathtt{a}_1^2\ )$, which gives a total worst case increase of $O(|\mathcal{A}|)$.

An intuitive explanation for the fact that Graphplan-based representations do not blow-up when explicit variables are eliminated is that the dummy "maintain" actions implicitly encode state information.

Encodings with the actions compiled away were called *state-based* encodings in Kautz and Selman (1996). For a further discussion of state-based encodings see Bendrax-Weiss *et al.* (1996). Encodings resulting from compiling away states, leaving "action-based " encodings, are similar to the causal encodings considered in McAllester and Rosenblitt (1991), Penberthy and Weld (1992), and Barrett and Weld (1994). Our analysis shows that the worst-case size of the action-based encoding is strictly better than that of the the state-based encodings. It should be noted however that in Kautz and Selman (1996), we still obtained good experimental results with state-based encodings. This is because in the domains considered, the number of actions that could account for a change in the truth of a fluent was small. It therefore appears that in this case our worst-case results may be overly pessimistic for many domains.

One cannot show in general that state-based encodings are constructive, because it may be difficult to reconstruct the set of actions that admit the sequence of states represented by models of the axioms. The general problem of finding unordered plans of length 1 is NP-complete. However, in domains we have examined so far, including the logistics and blocks world domains, there is a linear-time algorithm for finding such plans – again, because there are few ways of changing the truth-value of a single fluent.

5 LIFTED CAUSAL ENCODINGS

We now give a reduction inspired by the lifted version of the SNLP causal link planner of McAllester and Rosenblitt (1991). This encoding gives our best asymptotic results. Unlike our other reductions, which are exponential in $\mathcal{A}_{\mathrm{Ops}}$ and $\mathcal{A}_{\mathrm{Pred}}$, this reduction is polynomial in all parameters. The analysis can be simplified by considering a fixed but arbitrary set of operators. Furthermore, we consider the case where $|S_0| = |S_G| = |\mathrm{Dom}| = n$. This is reasonable in the blocks world where the size of the initial and final state, the number of blocks, and the plan length are all roughly linearly related. Under these conditions (with an arbitrary operator set) this encoding gives a SAT formula with $O(n^2)$ Boolean variables and $O(n^3)$ literal occurrences.

We will describe a lifted causal solution to the Sussman anamoly in three steps. First we review ground nonlinear causal planning and discuss a reduction to SAT based on the ground case. We then introduce the notion of a lifted SAT problem, an NP-complete problem somewhat "more general" than SAT. Every SAT problem is a lifted SAT problem but not vice-versa. We then give a reduction from planning to lifted SAT. Finally we give a general reduction from lifted SAT to SAT.

We will use the Sussman anomaly as an explanatory example. Suppose that we have one operator definition for the

MOVE operation as follows.

```
MOVE(x, y, z)
  PRE: CLEAR(x), ON(x, y), CLEAR(z)
  ADD: CLEAR(y), ON(x, z)
  DEL: CLEAR(z), ON(x, y)
```

Suppose that we have block C on block A, block A on PLACE1; block B on PLACE2, with C, B, and PLACE3 clear. Suppose we want A on B on C. The shortest plan is to move C from A to PLACE3; move B from PLACE2 to C; and finally move A from PLACE1 to B.

In this formulation of the Sussman anomaly we have six objects (three blocks and three places). This yields 6^3, or 216 possible actions of the form $\texttt{MOVE}(x,\ y,\ z)$. In this formulation of the blocks world we have $|\mathrm{Dom}|^3$ possible move actions. Each move action has three prerequisites, two deletions and two additions. A ground planner works with the individial ground actions and ignores the general operator definition.

We can think of the initial state as an operation which adds the initial assertions and the final state as an operation which requires, as prerequisites, the final assertions. The basic principle of causal link planning is that every prerequisite, including the assertions in the final state, must have a causal source, which may be the initial state. In the Sussman anomaly there are two assertions required by the final state and three required by each of the three actions. So we have 11 total prerequisites. A ground causal link is an assertion of the form $o_i \xrightarrow{\Phi} o_j$ where o_i and o_j are plan steps (possibly the initial or final step), and Φ is a ground fluent that is a prerequisite of o_j. The causal link assertion is true if o_i adds Φ, o_j needs Φ as a prerequisite and no step between o_i and o_j either adds or deletes Φ.

A nonlinear ground causal planner works with a set of step names where two different step names might be assigned the same action (some plans require the same action to be performed twice at different times). We let $o_1, \ldots, o_n$ be the step names. There is no *apriori* temporal order on the step names, and assertions of the form $o_i < o_j$ are used to state that step o_i occurs before step o_j.

A complete causal plan is an assignment of ground actions to step names, a set of casual links, and a set of step ordering assertions satisfying the following conditions.

1. Every prerequisite has a cause. If Φ is a prerequisite of the action assigned to step o_i then the plan includes a causal link of the form $o_j \xrightarrow{\Phi} o_i$.

2. Every causal link is true. If the plan contains $o_j \xrightarrow{\Phi} o_i$ then the action assigned to o_j adds Φ, the plan contains the ordering assertion $o_j < o_i$, and for every step name o_k other than o_i and o_j such that the action assigned to o_k deletes Φ, the plan contains either $o_k < o_j$ or $o_i < o_k$.

3. The ordering constraints are consistent. If the ordering constraints contain $o_i < o_k$ and $o_k < o_j$ then they contain $o_i < o_j$ and no step precedes itself, *i.e.*, the ordering constraints do not contain $o_i < o_i$ for any step o_i.

Theorem: In any topological sort of a complete nonlinear clausal plan, *i.e.*, in any total order consistent with the ordering constraints, all prerequisites are true when executed.

Recall that the initial and final states can be modeled as initial and final steps in the plan. For the Sussman anomaly the three most significant causal links are the following:

$$\texttt{MOVE(B,PLACE2,C)} \xrightarrow{\texttt{ON(B, C)}} \texttt{FINAL}$$
$$\texttt{MOVE(A,PLACE1,B)} \xrightarrow{\texttt{ON(A, B)}} \texttt{FINAL}$$
$$\texttt{MOVE(C,A,PLACE3)} \xrightarrow{\texttt{CLEAR(A)}} \texttt{MOVE(A,PLACE1,B)}$$

Since there are a total of 11 prerequisites, there are 8 other casual links with the initial state as the source. Also note that moving B onto C deletes CLEAR(C), a prerequisite of MOVE(C,A,PLACE3). The definition of a complete plan forces the step moving B to C to occur after the step moving C to A. Hence, once the actions and causal links are selected in the Sussman anomaly, the ordering of the steps is forced.

Condition 2 above does not ensure systematicity, *i.e.*, that every solution corresponds to a unique complete causal plan. To ensure systematicity we must require that actions that *add* the fluent in a causal link are also ordered to occur outside the link. We assume that systematicity is not important in local search.

We can reduce ground causal nonlinear planning directly to SAT, although this turns out not to be nearly as concise as the lifted encoding. We let I and F be initial and final actions. Let A be the set of actions *not* including actions representing the final or initial state. We let O be the set of step names not including initial and final steps. We let $\mathcal{F}$ be the set of all ground fluents. The ground clauses consist of the clauses in the conjunctive normal form of the Boolean formulas in Table 1.

We give a complexity analysis based on the assumption that the ground planning problem is derived from a fixed set of lifted operator definitions. This implies that each action has a bounded number of preconditions, additions, and deletions. For a fixed set of operation definitions we have that $|A|$ is $O(|\mathrm{Dom}|^{\mathcal{A}_{\mathrm{Ops}}})$ and $|\mathcal{F}|$ is $O(|\mathrm{Dom}|^{\mathcal{A}_{\mathrm{Pred}}} + |S_0| + |S_G|)$. The number of Boolean variables is dominated by the causal links, which is $O(n^2|\mathcal{F}|)$, and the number of variables of the form $\texttt{ACTION}(o,\ a)$, which is $O(n|A|)$. The number of literal occurances is dominated by schema 10 which involves $O(n^3|\mathcal{F}|)$ clauses (and literal occurances). For the blocks world with the three place move operation, and with $|\mathrm{Dom}| = |S_0| = |S_G| = n$ (where n is the number

1.	$\texttt{ACTION}(o,\ a_1) \vee \ldots \vee \texttt{ACTION}(o,\ a_m)$	$o \in O,\ a_i \in A$
2.	$\neg(\texttt{ACTION}(o,\ a) \wedge \texttt{ACTION}(o,\ b))$	$o \in O,\quad a, b \in A,\ a \neq b$
3.	$\neg\texttt{ADDS}(I,\ \Phi)$	$\Phi \in \mathcal{F} - S_o$
4.	$\texttt{NEEDS}(F,\ \Phi)$	$\Phi \in S_G$
5.	$\texttt{ADDS}(o,\ \Phi) \equiv (\texttt{ACTION}(o,\ a_1) \vee \ldots \vee \texttt{ACTION}(o,\ a_k))$	$o \in O,\ \Phi \in \mathrm{Add}(a_i)$
6.	$\texttt{DELS}(o,\ \Phi) \equiv (\texttt{ACTION}(o,\ a_1) \vee \ldots \vee \texttt{ACTION}(o,\ a_k))$	$o \in O,\ \Phi \in \mathrm{Del}(a_i)$
7.	$\texttt{NEEDS}(o,\ \Phi) \equiv (\texttt{ACTION}(o,\ a_1) \vee \ldots \vee \texttt{ACTION}(o,\ a_k))$	$o \in O,\ \Phi \in \mathrm{Pre}(a_i)$
8.	$\texttt{NEEDS}(p,\ \Phi) \Rightarrow (o_1 \xrightarrow{\Phi} p \vee \ldots \vee o_n \xrightarrow{\Phi} p)$	$o_i \in O \cup \{I\},\ p \in O \cup \{F\},\ \Phi \in \mathcal{F}$
9.	$o \xrightarrow{\Phi} p \Rightarrow \texttt{ADDS}(o,\ \Phi)$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ \Phi \in \mathcal{F}$
10.	$o \xrightarrow{\Phi} p \wedge \texttt{DELS}(r,\ \Phi) \Rightarrow (r < o \vee p < r)$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ r \in O - \{o,\ p\},\ \Phi \in \mathcal{F}$
11.	$I < o$	$o \in O$
12.	$o < F$	$o \in O$
13.	$\neg(o < o)$	$o \in O \cup \{I,\ F\}$
14.	$o < p \wedge p < r \Rightarrow o < r$	$p, q, r \in O \cup \{I,\ F\}$

Table 1: Reduction of ground causal nonlinear planning (non-lifted).

of plan steps) we get $O(n^4)$ Boolean variables and $O(n^5)$ literal occurrences.

The causal encoding eliminates the need for frame axioms. The frame axioms are implicit in schema 10. The lifted version of schema 10 involves $O(n^3)$ clauses instead of $O(n^5)$.

Now we define the notion of a lifted SAT problem. A lifted clause is just a first order clause — a disjunction of first order literals possibly containing free variables. A lifted SAT problem is just a set of lifted clauses. A lifted SAT problem is *satisfiable* if there exists a ground substitution (mapping from variables to ground terms) such that the resulting ground SAT problem is satisfiable, subject to the constraint that a ground atomic equality of the form $t = w$ is true if and only if t and w are the same term. As an example, consider the following simple clause set.

$$P(x) \Rightarrow Q(x)$$

$$Q(x) \Rightarrow W(x)$$

$$P(y)$$

$$x = a \vee x = b$$

$$P(a),\quad \neg W(a),\quad P(b),\quad \neg W(b)$$

This lifted SAT problem is not satisfiable. x must be either a or b, and in either case one of the first two clauses must be violated. However, if we remove the second clause which intuitively defines the range of x, then the clause set becomes satisfiable because we can now interpret x as some new constant, say c, and set $P(c)$, $Q(c)$, and $W(c)$ all to true. Note that the clause $P(y)$, which intuitively constrains the value of y, has no effect on the satisfiability of the problem. This existential interpretation of the variables in the clauses is very different from the universal interpretation assigned to clauses in resolution theorem proving.

Theorem: A lifted SAT problem is satisfiable if and only if there exists an equivalence relation on the atomic formulas and a truth assignment to the atomic formulas respecting that relation (equivalent literals are assigned the same truth value), such that the truth assignment satisfies all clauses and the unification problem defined by the equivalence relation is solvable.

Proof: If there exists such a truth assignment and equivalence relation then the most general unifier of the equivalences in the equivalence relation yields a substitution satisfying the problem. On the other hand, any substitution and truth assignment satisfying the problem defines an equivalence relation and truth assignment satisfying the required conditions. ■

Clearly lifted SAT includes SAT as a special case, and hence is NP-hard. The above theorem shows that the problem is in NP because we can nondeterministically guess the equivalence relation and truth assighment.

We will now reduce planning to lifted SAT, and then reduce lifted SAT to SAT. The composition of these two reductions will yield the desired concise reduction from planning to SAT. In the reduction from planning to lifted SAT we create a "fresh copy" of each operator definition at each step. For each step name o we let A_o consist of a fresh copy of each operator definition. Fresh copies are built by renaming the formal parameters. The variables appearing in A_o are disjoint from the variables appearing in A_p for distinct o and p. In the blocks world we have that A_o contains only a single element of the form $\texttt{MOVE}(x',\ y',\ z')$ where x', y' and z' are fresh variables for the step o. For any fixed operator set we have that $|A_o|$ is $O(1)$. We let A be the union of all A_o. Note that $|A|$ is $O(n)$. This should be contrasted to the ground case where $|A|$ is $O(|\mathrm{Dom}|^{\mathcal{A}_{\mathrm{Ops}}})$. This reduction in $|A|$ is the main benefit of lifting. For any set of actions S we let $\mathrm{Pre}(S)$ be the set of all prerequisites of operations in S and similarly for Add(S) and Del(S). Note that $|\text{Pre}(A_o)|$ is $O(1)$, as are $|\text{Add}(A_o)|$ and $|\text{Del}(A_o)|$. Also note that Pre(A), Add(A), and Del(A) all have size $O(n)$. For convenience we also define A_I and A_F to be singleton sets of operators where A_I adds the initial assertions and A_F requires the final assertions. We will also use Vars(A) to denote the set of all variables appearing in A. Note that $|\text{Vars}(A)|$ is $O(n)$. The reduction from planning to lifted SAT can now be defined as in Table 2.

Selecting a way of satisfying schemas 1 and 2 selects the set of actions without making any commitment to the order of those actions. In our fomulation of the blocks world we have that A_o is a singleton set containing an action of the form $\texttt{MOVE}(x,\ y,\ z)$. Assuming that we are looking for a three step plan we have that O contains three step names and the first schema generates the following unit clauses.

$$\texttt{ACTION}(o_1,\ \texttt{MOVE}(x_1,\ y_1,\ z_1))$$

$$\texttt{ACTION}(o_2,\ \texttt{MOVE}(x_2,\ y_2,\ z_2))$$

$$\texttt{ACTION}(o_3,\ \texttt{MOVE}(x_3,\ y_3,\ z_3))$$

We now have nine variables. For an n step blocks world plan we will have $3n$ variables (independent of domain size). For any fixed set of operator definitions the total size of the instances of schemas 1 and 2 is $O(n)$.

Selecting a way of satisfying schema 3 selects a value for each variable. For each of the nine variables in the Sussman anomaly we have an instance of this schema. For example we have the following.

$$x_1 = \texttt{A} \vee x_1 = \texttt{B} \vee x_1 = \texttt{C} \vee$$
$$x_1 = \texttt{PLACE1} \vee x_1 = \texttt{PLACE2} \vee x_1 = \texttt{PLACE3}$$

For any fixed set of operator definitions the total size of instances of schema 3 is $O(n|\text{Dom}|)$.

We can think of the schemas as "running" nondeterministically from the top to the bottom. Schemas 1 and 2 select the actions and schema 3 selects the argument values. Schemas 4 and 5 assert the preconditions of each action. This is straightforward and the total size of the instances of these schemas is $O(n + |S_G|)$. Schema 6 selects a causal source for every precondition, i.e., it selects the set of causal links in the plan. Every causal link atomic formula of the form $o \xrightarrow{\Phi} p$ has the property that $\Phi \in \text{Pre}(A_p)$. Since $|\text{Pre}(A_p)|$ is $O(1)$ for $p \neq F$ and $O(|S_G|)$ for $p = F$, we have that the total number of causal link formulas is $O(n^2 + n|S_G|)$. The total size of the instances of schema 6 is $O(n^2 + n|S_G|)$.

Schema 7 states that the source of each link must precede its destination. The total size of this schema has the same order as schema 6. Schema 8 places a "nonlocal addition demand" on the source of each causal link. Since there is one such demand for each causal link, the total size of this schema is again $O(n^2 + n|S_G|)$. Note that the number "nonlocal" formulas of the form $\texttt{ADDS}(o,\ \Phi)$ is quadratic ($O(n^2 + n|S_G|)$). This should be contrasted with the linear number of "local" atomic formulas of the form $\texttt{NEEDS}(o,\ \Phi)$. In the Sussman anomaly we get formulas such as $\texttt{ADDS}(o_1,\ \texttt{CLEAR}(x_3))$.

After schema 8 forces certain addition formulas to be true, schemas 9, 10, and 11 select equations between the things to be added and the approriate elements of the add lists. The truth of these equations is "checked" by the semantics of lifted SAT. For example, in the Sussman anomaly we have (essentially) the following.

$$\texttt{ADDS}(o_1,\ \texttt{CLEAR}(x_3)) \Rightarrow \texttt{CLEAR}(x_3) = \texttt{CLEAR}(y_1)$$

The equation $\texttt{CLEAR}(x_3) = \texttt{CLEAR}(y_1)$ is equivalent to $x_3 = y_1$ and this equivalence is handled by the semantics of lifted SAT. The total size of the instances of schemas 9, 10, and 11 is $O(n^2 + n|S_0|)$.

Schema 12 forces nonlocal delete statements to be true. In the Sussman anomaly we have, essentially, the following.

$$\texttt{CLEAR}(z_3) = \texttt{CLEAR}(x_2) \Rightarrow \texttt{DELS}(o_3,\ \texttt{CLEAR}(x_2))$$

Note that there is a quadratic number of "nonlocal" deletetion atomic formulas of the form $\texttt{DELS}(o,\ \Phi)$. Again this should contrasted with the linear number of "local" prerequisite assertions of the form $\texttt{NEEDS}(o,\ \Phi)$. The total size of the instances of schema 12 is $O(n^2 + n|S_G|)$.

Schema 13 handles the frame axioms by ensuring that deletions do not interfere with causal links. Schema 13 in the lifted encoding is analogous to schema 10 in the ground encoding. Schema 10 in the ground encoding generates $\Omega(n^3|\text{Dom}|^{\mathcal{A}_{\text{Pred}}})$ clauses. However, schema 13 in the lifted version generates only $O(n^3 + n^2|S_G|)$ clauses. The key observation is that in the lifted encoding there are only $O(n^2 + n|S_G|)$ causal link formulas.

Schemas 14 and 15 check that the order conditions selected in schema 13 are consistent, i.e., that I comes first, F comes last, and that the transitive closure does not contain loops. The instances of these schemas involve $O(n^2)$ atomic formulas and order $O(n^3)$ clauses.

Overall we have a quadratic number of atomic formulas and a cubic number of literal occurrences. The number of atomic formulas is dominated by formulas of the form $x = c$, $\Phi = \Psi$, and causal links of the form $o \xrightarrow{\Phi} p$. All schemas have linear or quadratic size except for schemas 13 and 14 which are cubic.

To complete the reduction to SAT we give a reduction from general lifted SAT to SAT. Let $\mathcal{L}$ be a lifted SAT problem. In our reduction we simply add clauses to $\mathcal{L}$. The additional clauses ensure that equations occuring in $\mathcal{L}$ have the proper truth value and that two atomic formulas which are equal have the same truth value. We let $T(\mathcal{L})$ be the set of terms in $\mathcal{L}$ and we let $F(\mathcal{L})$ be the set of all atomic formulas in $\mathcal{L}$ other than equations. If $\mathcal{L}$ is the lifted SAT encoding of a planning problem then (for a fixed set of operator definitions) $|T(\mathcal{L})|$ is $O(|\text{Dom}| + n + |S_0| + |S_G|)$. In other words,

1.	$\texttt{ACTION}(o,\ a_1) \vee \ldots \vee \texttt{ACTION}(o,\ a_m)$	$o \in O,\ a_i \in A_o$
2.	$\neg(\texttt{ACTION}(o,\ a) \wedge \texttt{ACTION}(o,\ b))$	$o \in O,\ \ a, b \in A_o,\ \ a \neq b$
3.	$x = c_1 \vee \ldots \vee x = c_n$	$x \in \mathrm{Vars}(A),\ \ c_i \in \mathrm{Dom}$
4.	$\texttt{ACTION}(o,\ a) \Rightarrow \texttt{NEEDS}(o,\ \Phi)$	$o \in O,\ a \in A_o,\ \ \Phi \in \mathrm{Pre}(a_i)$
5.	$\texttt{NEEDS}(F,\ \Phi)$	$\Phi \in \mathrm{Pre}(F)$
6.	$\texttt{NEEDS}(p,\ \Phi) \Rightarrow (o_1 \xrightarrow{\Phi} p \vee \ldots \vee o_n \xrightarrow{\Phi} p)$	$p \in O \cup \{F\},\ \ \Phi \in \mathrm{Pre}(A_p),\ \ o_i \in O \cup \{I\}$
7.	$o \xrightarrow{\Psi} p \Rightarrow o < p$	$o \in O \cup \{I\},\ \ p \in O \cup \{F\},\ \ \Psi \in \mathrm{Pre}(A_p)$
8.	$o \xrightarrow{\Phi} p \Rightarrow \texttt{ADDS}(o,\ \Phi)$	$o \in O \cup \{I\},\ \ p \in O \cup \{F\},\ \ \Phi \in \mathrm{Pre}(p)$
9.	$\texttt{ACTION}(o,\ a) \wedge \texttt{ADDS}(o,\ \Psi)$ $\Rightarrow (\Psi = \Phi_1 \vee ... \vee \Psi = \Phi_n))$	$o \in O,\ a \in A_o,\ \ \Psi \in \mathrm{Pre}(A \cup \{F\}),\ \ \Phi_j \in \mathrm{Add}(a)$
10.	$\texttt{ADDS}(I,\ \Psi) \Rightarrow (\Psi = \Phi_1 \vee \ldots \vee \Psi = \Phi_n)$	$\Psi \in \mathrm{Pre}(A),\ \ \Phi_i \in \mathrm{Add}(I)$
11.	$\neg\texttt{ADDS}(I,\ \Psi)$	$\Psi \in \mathrm{Pre}(F) - \mathrm{Add}(I)$
12.	$\texttt{ACTION}(o,\ a) \wedge \Psi = \Phi \Rightarrow \texttt{DELS}(o,\ \Psi)$	$o \in O,\ a \in A_o,\ \ \Psi \in \mathrm{Pre}(A \cup \{F\}),\ \ \Phi \in \mathrm{Del}(a)$
13.	$o \xrightarrow{\Psi} p \wedge \texttt{DELS}(r,\ \Psi) \Rightarrow (r < o \vee p < r)$	$o \in O \cup \{I\},\ \ p \in O \cup \{F\},\ \ r \in O - \{o,\ p\}, \Psi \in \mathrm{Pre}(A_p)$
14.	$o < r \wedge r < p \Rightarrow o < p$	$o, r, p \in O \cup \{I,\ F\}$
15.	$\neg(p < p)$	$p \in O \cup \{I,\ F\}$

Table 2: Lifted reduction of ground causal nonlinear planning.

we have only a linear number of terms. Furthermore, most atomic formulas in the lifted SAT encoding of a planning problem involve step name constants. This observation can be used to show that for planning problems there are only a quadratic number of unifiable pairs of formulas in $F(\mathcal{L})$. For any lifted SAT problem $\mathcal{L}$ we define the augmented clause set $\mathcal{L}'$ to be $\mathcal{L}$ plus the clauses in Table 3.

Schemas 1 through 5 ensure that the true equations define an equivalence relation on terms consistent with the interpretation of each function as a Herbrand constructor. Two terms "clash" if they are either distinct constants, one is a constant and one is a function application, or they are applications of different function symbols. Schema 5 states that clashing terms must not be equal.

Schemas 1 through 5 introduce $|T(\mathcal{L})|^2$ new atomic formulas — the equations of the form $t = u$. The number of instances of schema 4 can be no larger than $|T(\mathcal{L})|^2$. If we bound the arity (number of arguments) of functions then the total size of literal occurances in instances of schema 4 is $O(|T(\mathcal{L})|^2)$. For bounded arity, the the number of literal occurances in instances of schemas 1 through 5 is dominated by the transitivity schema, schema 3. The number of instances of schema 3 is $|T(\mathcal{L})|^3$.

Schemas 6, 7, and 8 handle the occurs-check condition. Without these schemas $x = f(y)$ and $y = f(x)$ appear consistent. But these equations (interpreted over the Herbrand universe of first order terms) imply that x is a subterm of itself. Since the Herbrand universe does not include infinite terms, this is impossible. These equations are inconsistent with schemas 6,7, and 8 because we can now infer $\texttt{OCCURS-IN}(x,\ x)$. Assuming a bounded arity for function symbols we have that schemas 6, 7, and 8 involve $O(|T(\mathcal{L})|^2)$ atomic formulas and $O(|T(\mathcal{L})|^3)$ clauses (and literal occurances).

Schema 7 enforces the condition that equivalent atomic literals have the same truth value. The number of instances of schema 7 equals the number of unifiable pairs of atomic formulas. We now have the following theorem.

Theorem: If $\mathcal{L}$ is a lifted SAT problem then $\mathcal{L}'$ as defined by the above augmentation is a satisfiable as a SAT problem if and only if $\mathcal{L}$ is satisfiable as a lifted problem. Furthermore, the number of additional atomic formulas in $\mathcal{L}'$ is $O(|T(\mathcal{L})|^2 + |U(\mathcal{L})|)$ where $U(\mathcal{L})$ is the number of unifiable pairs of atomic formulas. For bounded arity functions and predicates the number of additional literal occurances is $O(|T(\mathcal{L})|^3 + |U(\mathcal{L})|)$.

It is interesting to note that the above schemas can be converted to Horn clauses in linear time. When combined with a linear time algorithm for Boolean Horn clauses, the schemas define a cubic time unification algorithm. The classical Robinson unification algorithm is exponential time but a linear time algorithm is known (Patterson78).

It is also worth noting that in the lifted SAT problems that result from encodings of planning problems we need only schemas 1 through 5. In these lifted problems variables can only be bound to constants so no occurs-check reasoning is needed. Furthermore, all atomic formulas other than equations are forced to have appropriate truth values even in the absense of schema 7.

The total reduction from planning to SAT yields a quadratic number of atomic formulas and a cubic number of literal occurances.

6 CONCLUSIONS

Kautz and Selman (1996) challenged the widespread belief in the AI community that planning is not amenable to gen-

1.	$t = t$	$t \in T(\mathcal{L})$
2.	$t = u \Rightarrow u = t$	$t, u \in T(\mathcal{L})$
3.	$t = u \wedge u = w \Rightarrow t = w$	$t, u, w \in T(\mathcal{L})$
4.	$(f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n))$ $\equiv (t_1 = u_1 \wedge \ldots \wedge t_n = u_n)$	$f(t_1, \ldots, t_n) \in T(\mathcal{L})$ $f(u_1, \ldots, u_n) \in T(\mathcal{L})$
5.	$\neg(t = u)$	$t, u \in T(\mathcal{L})$, t, u clash
6.	$\texttt{OCCURS-IN}(t_i,\ f(t_1, \ldots, t_n))$	$f(t_1, \ldots, t_n) \in T(\mathcal{L})$
7.	$\texttt{OCCURS-IN}(u,\ w) \wedge \texttt{OCCURS-IN}(w,\ t)$ $\Rightarrow \texttt{OCCURS-IN}(u,\ t)$	$u, w, t \in T(\mathcal{L})$
8.	$\neg\texttt{OCCURS-IN}(t,\ t)$	$t \in T(\mathcal{L})$
9.	$P(t_1, \ldots, t_n) \wedge t_1 = u_1 \wedge \ldots \wedge t_n = u_n$ $\Rightarrow P(u_1, \ldots, u_n)$	$P(t_1, \ldots, t_n) \in F(\mathcal{L})$ $P(u_1, \ldots, u_n) \in F(\mathcal{L})$ $\forall i\ \ t_i, u_i$ unifiable

Table 3: Additional clauses for reduction from lifted SAT to ordinary SAT.

eral theorem-proving techniques, by showing that general propositional satisfiability algorithms can outperform specialized planning systems on a range of benchmark problems. Critical to the success of this approach is the use of concise SAT encodings of the planning problems. This paper described general, polynomial-time reductions from STRIPS-style planning to CNF formulas. We compared the various encodings, both in terms of the number of variables used and the total size of the formulas. This kind of analysis will allow one to use the gross statistical properties of a given planning problem to select a practical encoding.

This paper also introduced lifted causal encodings, a new kind of reduction. We showed that this encoding strictly dominates others in asymptotic terms. In future work, we will empirically evaluate this new class of encodings, both to determine whether the constant factors in the size of the encoding are reasonable for practical problems, and to determine whether our satisfiability procedures also perform well on this class of formulas.

References

Bacchus, F. and Kabanza, F. (1995). Using temporal logic to control search in a forward chaining planner. *Proc. EWSP-95*, 157–169.

Backstrom, C. (1992). *Computational complexity of reasoning about plans*, Ph.D. thesis, Linkoping University, Linkoping, Sweden.

Barrett, A. and Weld, D. (1994). Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67:71-112, 1994.

Bedrax-Weiss, T., Jonsson, A.K., and Ginsberg, M.L. (1996). Partial-order planning: evaluating possible efficiency gains. Unpublished manuscript.

Blum, A. and Furst, M.L. (1995). Fast planning through planning graph analysis. *Proc. IJCAI-95*, Montreal, Canada.

Bylander, T. (1991). Complexity results for planning. *Proc. IJCAI-91*, Sidney, Australia, 274-279.

Crawford, J.M. and Auton, L.D. (1993) Experimental Results on the Cross-Over Point in Satisfiability Problems. *Proc. AAAI-93*, Washington, DC, 21–27.

Davis, M., Logemann, G., and Loveland, D. (1962). A machine program for theorem proving. *Comm. ACM*, 5, 1962, 394–397.

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

Fikes, R.E. and Nilsson, N.J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4), 189–208.

Gupta and Nau (1992). On the complexity of blocks-world planning. *Artificial Intelligence*, **56**, 139–403.

Haas, A. (1987). The case for domain-specific frame axioms. *The Frame Problem in Artificial Intelligence, Proceedings of the 1987 Workshop*, F.M. Brown, ed., Lawrence, KS, 1987. Morgan Kaufmann Publishers, Los Altos, CA.

Joslin, D. and Pollack, M. (1995). Passive and Active Decision Postponement in Plan Generation. *European Workshop on Planning (EWSP)*, Assisi, Italy, Sept. 1995.

Kambhampati, S. and Yang, X. (1996). On the role of Disjunctive Representations and Constraint Propagation in Refinement Planning. *Proc. KR-96.*

Kautz, H. , McAllester, D., and Selman, B. (1996). Planning in Propositional Logic. In preparation.

Kautz, H. and Selman, B. (1992) Planning as Satisfiability. *Proc. ECAI-92*, Vienna, Austria, 1992, 359–363.

Kautz, H., and Selman, B. (1996). Pushing the envelope: planning, propositional logic, and stochastic search. *Proc. AAAI-96*, Portland, OR, 1996.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence 4*, D. Michie, ed., Ellis Horwood, Chichester, England, 1969, page 463ff.

McAllester, D. and Rosenblitt, D. (1991). Systematic nonlinear planning. *Proc. AAAI-91*, Anaheim, CA.

Mitchell, D., Selman, B., and Levesque, H.J. (1992). Hard and Easy Distributions of SAT Problems. *Proc. AAAI-*

92, San Jose, CA, 459–465.

Patterson, M.S. and Wegman, M.N. (1978). Linear unification. *JCSS*, 16 1978, 158–167.

Penberthy, J. and Weld, D. (1992). UCPOP: A sound, complete, partial order planner for ADL. In the *Proc. KR-92*, Boston, MA, 103–114.

Schubert, L. (1989). Monotonic Solution of the Frame Problem in the Situation Calculus: an Efficient Method for Worlds with Fully Specified Actions. *Knowledge Representation and Defeasible Reasoning*, H. Kyburg, R. Loui, and G. Carlson, eds.

Selman, B. (1994). Near-Optimal Plans, Tractability, and Reactivity. *Proc. KR-94*, Bonn, Germany, 1994, 521–529.

Selman, B., Kautz, H., and Cohen, B. (1996) Local Search Strategies for Satisfiability Testing. *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*. (to appear)

Selman, B., Levesque, H., and Mitchell, D. (1992). A New Method For Solving Hard Satisfiability Problems. *Proc. AAAI-92*, San Jose, CA, 1992, 440-446.

Slaney, J. and Thiebaux, S. (1996). Linear Time Near-Optimal Planning in the Blocks World. *Proc. AAAI-96.*

Stone, P., Veloso, V., and Blythe, J. (1994). The need for different domain-independent heuristics. In *AIPS94*, pages 164-169, Chicago, 1994.

Veloso, M. (1992). Learning by analogical reasoning in general problem solving. Ph.D. Thesis, CMU, CS Techn. Report CMU-CS-92-174.