Planning as Satisfiability

Henry Kautz and Bart Selman
AI Principles Research Department
AT&T Bell Laboratories
600 Mountain Avenue, Room 2C-407
Murray Hill, NJ 07974
{kautz, selman}@research.att.com

Abstract

We develop a formal model of planning based on satisfiability rather than deduction. The satisfiability approach not only provides a more flexible framework for stating different kinds of constraints on plans, but also more accurately reflects the theory behind modern constraint-based planning systems. Finally, we consider the computational characteristics of the resulting formulas, by solving them with two very different satisfiability testing procedures.

1 Introduction

Planning has traditionally been formalized as deduction [Green, 1969; McCarthy and Hayes, 1986; Rosenschein, 1981; Pednault, 1988; Allen, 1991]. Although the details of the different formalisms vary, all use axioms which state that the effects of an action are implied by the occurrence of the action when its preconditions hold. Planning is then formalized as the process of finding a deductive proof of a statement that asserts that the initial conditions together with a sequence of actions imply the goal conditions.

The complementary problem to deducibility is satisfiability — that is, finding a model of a set of axioms. While deduction is a very hard problem, it has long been supposed in AI that satisfiability is even harder. In the first-order case, theoremhood is at least semi-decidable, but there can be no complete procedure for finding models of arbitrary formulas. In the propositional case, however, there

This paper appears in the *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 92)*, Vienna, Austria, August 1992.

appears to be no theoretical basis for supposing that the one problem is harder than the other. incomparable. Nevertheless, it appears that no one has tried to explicitly formulate planning as propositional satisfiability. It may have been that the task of searching the exponentially large space of truth assignments for a model seemed far more daunting than searching the space of resolution proofs (although the latter could be equally large, since blocks world planning is NP-hard [Gupta and Nau, 1991] regardless of the formalization).

Recently, we have been developing algorithms that solve extremely large and difficult satisfiability problems expressed as sets of propositional clauses. As reported in [Selman *et al.*, 1992], we have used a randomized greedy algorithm GSAT to solve empirically hard random formulas as well as encodings of hard graph-coloring problems. This impressive performance led us to try to formulate other problems in AI as propositional satisfiability, both in order to test our algorithms on different kinds of formulas, and to potentially uncover a new practical approach to solving those problems. Although this was our primary motivation for investigating planning as satisfiability, we believe that the resulting framework is actually a more flexible and accurate way of formalizing planning than the standard deductive approach.

The first part of this paper deals with the formalism. We will demonstrate that the axiomatizations of actions used for deductive planning are not adequate for the satisfiability approach, because they admit many unintended models. We then present an alternative approach for axiomatizing actions using the blocks world as an example, and show how a technique for replacing predicates that take three or more arguments by predicates that take no more than two arguments can dramatically cut the size of the corresponding instantiated propositional theory.

The formalization of planning as satisfiability turns out to have a number of attractive properties: It is easy to state arbitrary facts about any state of the world, not just the initial and goal states. It is likewise easy to state arbitrary constraints on the plan — for example, that it contain a specified action performed at a specified time. Finally, the approach provides a more accurate formal model of modern constraint-based planners [Stefik, 1981; Chapman, 1987].

The second part of the paper reports on our preliminary experiments in solving the Boolean satisfiability problems generated by specific blocks world planning problems. We tried two algorithms: the GSAT program mentioned above, and DP, an implementation of the standard Davis-Putnam backtracking algorithm [Davis and Putnam, 1960]. The backtracking algorithm did surprisingly well on most of the planning problems, despite that fact that it failed to solve similarly-sized coloring and random problems. The results with GSAT were the reverse — the planning formulas appeared harder than the coloring and hard random problems.

A significant finding of the experimental work is that the performance of GSAT is significantly improved by the addition of axioms which are not logically necessary, but which explicitly rule out the existence of certain "impossible" states (such as block being on itself). This is an interesting case where a larger problem can be much easier to solve than a smaller one.

2 Planning as Deduction

The best-known logical formalization of planning is the situation calculus [McCarthy and Hayes, 1986]. In this system, the execution of an action is explicitly represented by the application of a function to a term representing the state in which the action is performed. For example, to express the fact that block A is on B after performing $move(A, B)$ in state S_0, one might write

$$on(A, B, result(move(A, B), S_0)))$$

A disadvantage of the situation calculus for our purposes is that it is inherently first-order: an infinite number of state terms can be constructed by repeated application of the $result$ function. On the other hand, we are interested only in finite plans containing no more than some given number of actions. The following approach is equivalent to a finite propositional system.

The basic language is function and quantifier-free typed predicate logic with equality. Each of the finite set of types contains a finite set of individuals, named by *unique* constant terms. In this paper we will only use two types, BLOCK and TIME. We will use A, B, C, $\cdots$ for constants of type BLOCK. The constants of type TIME are always a finite range of integers.

A finite set (or conjunction) of formulas may be abbreviated by a *schema*. A schema looks just like a formula, but may also contain typed variables bound to the quantifiers $\exists$ or $\forall$. A schema stands for the set of all of the formulas that can be generated by iterating the quantifiers over the constants of the appropriate types. We will use the letters i and j for variables of type TIME, and other letters for variables for type BLOCK. Arithmetic expressions like "$i + 1$" are interpreted at instantiation time — the basic language does not actually contain any function symbols such as "+". Finally, we make the convention that we simply disregard any instantiation of a schema which would contain a time greater than N, the largest constant of type TIME.

In the blocks world we will use the predicates $on(x, y, i)$ to mean that x is on y at time i; $clear(x, i)$ to mean there is room to move something on top of x at time i; and $move(x, y, z, i)$ to mean that x is moved from y to z between times i and $i+1$. Note that actions are represented by propositions, not terms or functions. It is straightforward to write down the usual kinds of axioms which state that if the preconditions of an action hold then the action achieves its effects. For example:

$$\begin{aligned}\forall x, y, z, i.\ & on(x, y, i) \wedge clear(x, i) \wedge clear(z, i) \wedge \\ & move(x, y, z, i) \supset on(x, z, i+1) \wedge clear(y, i+1)\end{aligned}$$

We also include *frame axioms* to describe the propositions an action does *not* affect. A total of 11 such schemas can capture a simple blocks world. We also introduce a special block named $Table$ which is never moved but which can support any number of blocks (i.e., it is always "clear").

In the traditional deductive approach, a planning problem is formalized as a theorem which states that the initial conditions together with a sequence of actions implies the goal conditions. For example, suppose in the initial condition A is on B, and we wish to find a two step plan such that B will be on A. The problem is represented by the existentially-quantified schema:

$$\begin{aligned}&\exists x_1, y_1, z_1, x_2, y_2, z_2. \\ &\quad on(A, B, 1) \wedge on(B, Table, 1) \wedge clear(A, 1) \wedge \\ &\quad move(x_1, y_1, z_1, 1) \wedge move(x_2, y_2, z_2, 2) \supset \\ &\qquad on(B, A, 3)\end{aligned}$$

A proof of such a schema is taken to be a proof of a particular instantiation of the schema; the plan then corresponds to the instantiation of the two instances of the *move* predicate.

3 Anomalous Models

The axioms sketched above (or the equivalent in the situation calculus) are all true in the (idealized) blocks world. But it is not the case that all worlds described by these axioms are anything like the blocks world! Or, in other words, it is safe to use them deductively: whatever follows from them is true. But it is not correct to say that any plan which *satisfies*, or is *consistent* with, these axioms is reasonable.

A *model* is a truth-assignment to the atomic propositions of the language, and can be identified with the set of propositions it assigns "true". Consider the two step planning problem described above. The theorem is true in all models of the axioms. But there are many models of the axioms which satisfy the theorem but do not correspond to valid plans. For example, the axioms allow the model

$$\left\{\begin{array}{l} on(A, B, 1), on(B, Table, 1), clear(A, 1), \\ on(B, A, 2), on(B, A, 3), clear(Table, 1), \\ clear(Table, 2), clear(Table, 3) \end{array}\right\}$$

where the world changes, yet no known action occurs. (For brevity, we henceforth omit the propositions $clear(Table, i)$ from the description of models.)

Furthermore, the axioms only state what happens when an action is performed when its preconditions are satisfied. Thus

$$\left\{ \begin{array}{l} on(A, B, 1), on(B, Table, 1), clear(A, 1), \\ move(B, Table, A, 1), on(B, A, 2), on(B, A, 3) \end{array} \right\}$$

is also a model: because the preconditions of $move(B, Table, A, 1)$ are false, it is *consistent* that the effect $on(B, A, 2)$ still occurs!

These kind of anomalous models are not unique to the particular language we choose. They occur in the axioms developed for the situation calculus, or any other system designed for deductive planning.

4 Planning as Satisfiability

In the planning as satisfiability approach, a planning problem is not a theorem to be proved; rather, it is simply a set of axioms with the property that *any* model of the axioms corresponds to a valid plan. Some of these axioms describe the initial and goal states. For the simple example discussed above, this is

$$on(A, B, 1) \wedge on(B, Table, 1) \wedge clear(A, 1) \wedge on(B, A, 3)$$

The other axioms describe the actions in general. These include the standard effect and frame axioms described above, plus others that rule out the anomalous models.

First, we rule out the possibility that an action executes despite the fact that its preconditions are false. This can be done by asserting that an action implies its preconditions as well as its effects; e.g.,

$$\begin{array}{c} \forall x, y, z, i.\ move(x, y, z, i) \supset (clear(x, i) \wedge \\ clear(z, i) \wedge on(x, y, i)) \end{array}$$

It is interesting to note that in this formulation preconditions and effects are treated *symmetrically.*

Next, we state that only one action occurs at a time.

$$\begin{array}{c} \forall x, x', y, y', z, z', i.\ (x \neq x' \vee y \neq y' \vee z \neq z') \supset \\ \neg move(x, y, z, i) \vee \neg move(x', y', z', i) \end{array}$$

Finally, we assert that some action occurs at every time. This is not a significant restriction, since we can always introduce an explicit "do nothing" action if desired. In the simple blocks world the axiom schema is

$$\forall i < N.\ \exists x, y, z.\ move(x, y, z, i)$$

(An existentially-quantified formula expands to the disjunction of its instantiations.) If a planning problem is specified by asserting a *complete* initial state then these axioms guarantee that all models correspond to valid plans. This is so because every model contains a sequence of actions whose preconditions are satisfied, and the execution of an action in a state completely determines the truth-values of all propositions in the next state.

The *only* model of the simple two step planning problem is the intended model containing $move(A, B, Table, 1)$ and $move(B, Table, A, 2)$. A simple planning system can be constructed by linking a routine that instantiates such a given set of axiom schemas and initial and goal state specifications to a Boolean satisfiability algorithm.

5 Advantages of the Framework

The formulation of planning as satisfiability turns out to have a number of advantages over the purely deductive approach. First, it is easy to specify conditions in *any* intermediate state of the world, not just the initial and goal states. For example, if you want to insure that something is on either block C or D at time 5, you simply add the assertion to the problem specification $\neg clear(C, 5) \vee \neg clear(D, 5)$. Such conditions can involve arbitrary quantifiers and disjunction, as can the statements in the goal description. The conditions can also include events in a changing world that are beyond the control of the agent.

It is difficult to assert such conditions in the deductive approach. In the situation calculus it appears necessary to resort to syntactic, non-logical constraints on the form of the compound term which names the goal state. Even so, there remain conditions that cannot be guaranteed using only the deductive planning axioms. For example, suppose you want the plan to *not* contain the action $move(A, B, C, 3)$. It is not correct to conjoin the formula $\neg move(A, B, C, 3)$ to the antecedent of the theorem, because then a proof of the theorem becomes trivial: instantiate the plan so it *does* contain that action, and then "false" implies anything. On the other hand, it may be impossible to prove the theorem if $\neg move(A, B, C, 3)$ is added to the consequence, because the basic axioms cannot in general be used to prove that an action does *not* occur.

In the satisfiability framework the plan requirements are all *constraints* on the models. This view turns out to tie in with the notion of *planning with constraints* [Stefik, 1981; Chapman, 1987], widely used in modern planning systems. Chapman describes the functioning of his planning system TWEAK by using a formula in temporal logic called the "modal truth criteria", which roughly can be interpreted as saying that a proposition p holds in a state if and only if it is asserted (added by an action) in that or an earlier state, and not falsified by an

problem	using *move*			using *object*, *source*, *dest*		
	# props	# clauses	size	# props	# clauses	size
anomaly	127	2,364	5,529	94	375	933
reversal	429	22,418	51,753	215	993	2,533
medium	641	68,533	155,729	244	1,185	3,025
hanoi	1,005	63,049	137,106	288	1,554	3,798
huge	>7,000	>3,500,000	> 8,000,000	996	5,945	15,521

Table 1: Comparison of size of propositional theories using one ternary predicate versus three binary predicates.

intermediate action. He goes on to say that the "truth criteria can usefully be thought of as a completeness/soundness theorem for a version of the situation calculus," but does not specify the exact relationship. The logical status of the modal truth criteria is made clear by the planning as satisfiability approach: it is one way of expressing the axioms that must be added to the situation calculus so that all models of the axioms correspond to valid plans. The criteria rules out anomalous models in which propositions become true but are not added by an action whose preconditions are satisfied.

6 Experimental Results

In the rest of this paper we shall talk about issues that arise when solving planning problems using *general* satisfiability procedures. In all the work described henceforth, instantiated planning problems were represented as sets (that is, conjunctions) of propositional clauses (disjunctions of literals).

One of the first issues that arises in solving the Boolean satisfiability problems generated by planning is their sheer size. Taking c to be the number of elements (constants) in the largest type, d to be the maximum depth of quantifier nesting in any schema, and k to be the number of literals in the longest schema, the total length of the instantiated theory is bounded by $O(kc^d)$. It is clear that the greatest reduction in size can be had by reducing the quantifier depth. A simple scheme for so doing also greatly reduces the number of propositions.

The basic idea is to replace predicates that take three or more arguments by several predicates that take no more than two arguments. The $move(x, y, z, i)$ predicate takes four arguments, so we will replace it with three new predicates: $object(x, i)$, $source(y, i)$, and $dest(z, i)$.

Consider the axiom in section 4 that states that only a single action occurs at a time. It has seven universally quantified variables, but can be replaced by the following three schemas, each with only three quantifiers:

$$\forall i, x_1, x_2 . x_1 \neq x_2 \supset \neg object(x_1, i) \vee \neg object(x_2, i)$$
$$\forall i, y_1, y_2 . y_1 \neq y_2 \supset \neg source(y_1, i) \vee \neg source(y_2, i)$$
$$\forall i, z_1, z_2 . z_1 \neq z_2 \supset \neg dest(z_1, i) \vee \neg dest(z_2, i)$$

Table 1 shows the dramatic reduction in size made possible by the shift to 2-place predicates.

The most widely-used algorithm for solving satisfiability problems is the Davis-Putnam procedure [Davis and Putnam, 1960], which is in essence a resolution method [Vellino, 1989]. This algorithm incrementally builds up a truth assignment and backtracks when it determines the assignment does not satisfy the formula. It also simplifies the clauses as it goes along by deleting literals which are false in the current partial assignment, and when a clause containing a single literal is created, immediately assigns that literal true (or backtracks if necessary).

Recently we developed a new approach to solving large satisfiability problems called GSAT, a randomized local search procedure. The algorithm works by guessing a complete random truth-assignment. It then repeatedly changes the value ("flips") assigned to the proposition that results in the largest number of clauses being satisfied. If several propositions are equally good, it picks one at random. It continues to flip until either a satisfying model is found or a predetermined number of flips are performed. If a satisfying model is not found, GSAT repeats the procedure, starting with a different random assignment.

In [Selman *et al.*, 1992], we report on the details of GSAT, and show that it can solve non-trivial satisfiability problems that are an order of magnitude larger than can be solved by DP. These problems were generated from graph-coloring problems that were known to be hard [Johnson *et al.*, 1991], and random problems from a "hard" random distribution [Mitchell *et al.*, 1992]. A few of the results comparing the speed of the two algorithms appear in Table 2, along with some of the tests we ran on the planning formulas. A dash — in the table indicates that the algorithm failed to find a solution after running overnight.

DP performed well on moderately large planning formulas: compare its time of 1.2 seconds on the "medium" planning problem, versus its time on 4.7 *hours* on the much *smaller* "random B" problem. On the largest planning formulas DP failed as expected. GSAT did not fair as well, but its performance can be improved, as we shall see.

One of the ways in which a model can fail to satisfy all the axioms of a planning problem is if it depends on an "impossible" state of the world — for example, one in which a block is *on* itself. It is not *logically necessary* to rule out such states, since the axioms for actions are such that they can never lead from a legal state

problem	vars	size	GSAT	DP
random A	100	1,290	6 sec	2.8 min
random B	140	1,806	14 sec	4.7 hours
random C	500	6,450	1.6 hours	—
coloring A	2,125	168,419	8 hours	—
coloring B	2,250	180,576	5 hours	—
anomaly	94	933	26 sec	0.1 sec
reversal	215	2,533	—	4 sec
medium	244	3,025	—	1.2 sec
hanoi	288	3,798	—	13 hours
huge	996	15,521	—	—

Table 2: Comparison of speed of GSAT versus DP on solving sample coloring, random, and planning satisfiability problems.

problem	original		expanded	
	size	time	size	time
anomaly	933	26 sec	1,325	1.9 sec
reversal	2,533	—	3,889	1.2 min
medium	3,025	—	5,235	1.2 min

Table 3: Improvement in performance of GSAT by adding additional axioms to rule out impossible states.

to an impossible state. None the less, we decided to try adding axioms which *explicitly* rule out various impossible conditions.

Expanding the problems in this way significantly improved the performance of GSAT, as shown in Table 3. **(Note: see also the paper "Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems", by Bart Selman and Henry Kautz, for more recent extensions to GSAT that greatly improve its performance on these formulas.)** GSAT only requires a small fraction of the time it previously required to solve the Sussman anomaly, and can now solve the next two bigger problems. Interestingly, the performance of DP was *not* improved by the additional constraints.

It appears that the larger number of constraints in the expanded problem helped guide GSAT toward the global solution. It is important to note, however, that the additional constraints do not give any *new* information to the problem solver; they can all be derived from the original set of constraints. This suggests an interesting line of future research: given a initial set of logical constraints, determine how to derive additional constraints that help guide a greedy local search type problem solver.

The relatively good performance of DP on moderately-sized planning problems can be at least partly explained by the fact that the planning formulas consist of 99% Horn clauses. (A Horn clause contains at most one positive literal.) DP performs *unit resolution*, which propagates very efficiently — in linear time, in the best implementation — through Horn clauses. Our current version of GSAT has no special mechanism for handling Horn clauses efficiently.

Minton et al. [Minton *et al.*, 1990] claim very good results for using a greedy local search method for a large scheduling problem involving the Hubble Space Telescope. This raises the possibility that the computational nature of real-world scheduling tasks is fundamentally different from the kind of blocks world "puzzles" traditionally studied in AI (see also [Agre and Horswill, 1992] for a discussion of this claim). This issue is the focus of our current research.

7 Conclusions

We have developed a formal model of planning based on satisfiability rather than deduction. We showed how deductive planning axioms must be strengthened in order to rule out anomalous models. We then went on to argue that the satisfiability approach not only provides a more flexible framework for stating different kinds of constraints on plans than does the deductive approach, but also more accurately reflects the theory behind constraint-based planning systems.

We showed that the kinds of satisfiability problems that come from planning have different computational characteristics than the random formulas and coloring problems typically used to test satisfiability algorithms. Finally, we saw that enlarging the problems by adding more axioms can dramatically improve the performance of satisfiability algorithms based on greedy local search.

References

[Agre and Horswill, 1992] Philip E. Agre and Ian Horswill. Cultural support for improivisation. In *Proceedings of AAAI-92*, Menlo Park, CA, 1992. AAAI Press/The MIT Press.

[Allen, 1991] James Allen. Planning as temporal reasoning. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, Cambridge, MA, 1991.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, 1960.

[Green, 1969] C. Green. Application of theorem proving to problem solving. In *Proceedings of IJCAI-69*, pages 219–239, Washington, D.C., 1969.

[Gupta and Nau, 1991] Naresh Gupta and Dana S. Nau. Complexity results for blocks-world planning. In *Proceedings of AAAI-91*, page 629, Menlo Park, CA, 1991. AAAI Press/The MIT Press.

[Johnson *et al.*, 1991] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partioning. *Operations Research*, 39(3):378–406, 1991.

[McCarthy and Hayes, 1986] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. MICHIE, editor, *Machine Intelligence 4*, page 463ff. Ellis Horwood, Chichester, England, 1986.

[Minton *et al.*, 1990] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of AAAI-90*, pages 17–24, Cambridge, MA, 1990. The MIT Press.

[Mitchell *et al.*, 1992] D. Mitchell, B. Selman, and H.J. Levesque. Hard and easy distribution of sat problems. In *Proceedings of AAAI-92*, San Jose, CA, 1992.

[Pednault, 1988] E.P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, November 1988. special issue on planning.

[Rosenschein, 1981] Stanley J. Rosenschein. Plan synthesis: a logical perspective. In *Proceedings of IJCAI-81*, 1981.

[Selman *et al.*, 1992] B. Selman, Levesque H.J., and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, San Jose, CA, 1992.

[Stefik, 1981] Mark Stefik. Planning with constraints (molgen: Part 1 and 2). *Artificial Intelligence*, 16:111–170, 1981.

[Vellino, 1989] A. Vellino. The complexity of automated reasoning. Ph.D. thesis, Department of Philosophy, University of Toronto, Toronto, Canada, 1989.