Planning as QBF

Michael Cashmore and **Maria Fox**
University of Strathclyde

Abstract

Planning can be solved by general purpose satisfiability algorithms using an efficient encoding. Introduced here is a translation from STRIPS-style planning problems to an encoding as a Quantified Boolean Formula, which is sound and complete. It exploits the structure of the QBF problem to place an bound on the plan length that is exponential in the size of the formula, an improvement over the linear bounds of similar SAT encodings.

1. Introduction and Definitions

A STRIPS-style planning problem is described as a set F of propositional facts, a set of actions A, and an initial and goal state $I, G \subseteq F$. An action (a) contains three sets of states from F; adds [$adds(a)$], deletes [$dels(a)$] and preconditions [$precs(a)$]. An action can be applied to a state $S \subseteq F$, in which all of its preconditions are present ($precs(a) \subseteq S$). Applying an action a in state S_i produces a new state $S_{i+1} \subseteq F$ where $S_{i+1} := (S_i - dels(a)) \cup adds(a)$.
A planning problem has a solution if there is a sequence of actions that can be applied to each state, beginning with the initial state, ending in a final state S_g such that $G \subseteq S_g$. In a parallel plan a set of actions can be applied to a single state, as long as the resultant state is identical whatever order those actions are applied.
There has been success in solving bounded planning problems by first encoding them as SAT instances, as both problems are NP-complete. However there are some practical limits on the size complexity of the resulting transformation (Kautz, McAllester, and Selman 1996).
Quantified Boolean Formula (QBF), for which SAT is a subproblem, is PSPACE complete (Cadoli, Giovanardi, and Schaerf 1998). It is possible to encode a planning problem instance as a QBF and achieve a formula size that is exponentially smaller than that of an equivalent SAT translation. As there are QBF solvers which solve the formula without an exponential blowup in memory (Giunchiglia, Narizzano, and Tacchella 2001) this prospect seems attractive. There are already encodings into QBF for conformant planning, (c.f. (Littman 2003) and (Rintanen 2007)). These encodings make use of the QBF's structure to handle uncertainty, rather than simply reduce the size of the encoding. This paper will describe a basic encoding, with reference to an example, in section 2. Section 3. displays results on running an initial implementation of these encodings on a selection of STRIPS PDDL instances. Section 4. lists some improvements that can be made to the basic encoding, without going into detail.
An example of a simple planning problem will be used in order to demonstrate the basic ideas of the translation. The problem describes a truck moving between locations. The problem is described by:

$$Facts : (at\,A),\ (at\,B)\ and\ (at\,C)$$

$$Actions : (drive\,A\,B)\ and\ (drive\,B\,C)$$

With the obvious adds, deletes and preconditions, eg:

$$adds(drive\,A\,B) :\ (at\,B)$$

$$dels(drive\,A\,B) :\ (at\,A)$$

$$precs(drive\,A\,B) :\ (at\,A)$$

There are also three actions for doing nothing $(noop\,A)$, $(noop\,B)$ and $(noop\,C)$. These actions contain the associated fact as an add effect and precondition, with no deletes. The initial state is $(at\,A)$, and the goal is $(at\,C)$.

Quantified Boolean Formula

The QBF problem extends SAT by allowing the quantification of variables, either existentially ($\exists$) or universally ($\forall$). A QBF has the form

$$Q_1x_1, Q_2x_2, ..., Q_nx_n\phi$$

where ϕ is the propositional formula containing exactly the variables x_i, $(i = 1..n)$. Since $\exists x \exists y \phi = \exists y \exists x \phi$ and $\forall x \forall y \phi = \forall y \forall x \phi$ the formula can be written as

$$Q_1X_1, ..., Q_nX_n\phi$$

where X_i's are mutually disjoint sets of variables, and the quantifiers alternate between $\forall$ and $\exists$.
A set of variables is said to be quantified *before* or *above* another set that follows it in the quantification layer, or is *inside* or *below* a set which precedes it. These terms are used interchangeably.
A QBF is said to be *satisfied* by the assignment of truth or

falsity to its existentially quantified variables. A *truth assignment* or *assignment* is used to refer to a complete or partial satisfying certificate for the QBF instance, or proposed partial certificate during the search process. The context will make obvious which usage is intended.

The encoding uses the idea that a QBF can be represented as a tree structure, (c.f. proof of QSAT's PSpace-completeness in (Papadimitriou 1993)). However it uses a novel scheme of using existentially quantified descriptions of the state, with universal variables limited to branching the structure in order to allow far more propagation.

A QBF

$$\forall x_1, Q_2 x_2, ..., Q_n x_n \phi(x_1, ..., x_n)$$

can be written as

$$A \& B$$

$$A = Q_2 x_2, ..., Q_n x_n \phi(0, x_2, ..., x_n)$$

$$B = Q_2 x_2, ..., Q_n x_n \phi(1, x_2, ..., x_n)$$

with A and B as separate QBF instances. A satisfying assignment can be represented as a binary tree in which universally quantified variables mark branches, with the outermost universal variable forming the root. Existentially quantified variables are given truth assignments directly above the branches which immediately follow them in the quantification layer. This representation will be used to describe the translation in section 2. with the leaves of the tree referring to the innermost existentially quantified set of variables.

2. QBF Encoding

The approach described here uses the branching structure of the QBF to reuse a single set of clauses that describe a single state in the plan. The two assignments inside each universal variable represent the first and second half of the plan split around that branch. The assignments to each existential set represent action choices within a single state. The bound on the length of the plan can be increased exponentially with a linear increase in variables and clauses. This places an upper bound on the plan length of $2^{n+1} - 1$ parallel actions layers, where n is the number of universally quantified variables.

In the truck example a QBF can be formulated with a single branch node and two existential sets. These sets contain variables associated with the actions described in section 1. and written as $(action)_1$ or $(action)_2$ depending on whether it is quantified in the first or second existential set. The quantification layer will be:

$$\exists (drive\, A\, B)_1, (drive\, B\, C)_1, (noop\, A)_1,$$

$$(noop\, B)_1, (noop\, C)_1$$

$$\forall b$$

$$\exists (drive\, A\, B)_2, (drive\, B\, C)_2, (noop\, A)_2,$$

$$(noop\, B)_2, (noop\, C)_2$$

Figure 1 shows how this quantification line corresponds to the branching structure described in section 1. The clauses in the formula are drawn from a Plan Graph, which is generated until level off, as described in (Blum and Furst 1997) with improvements as described in (Fox and Long 1999). These clauses ensure that:

- An action chosen at step$_i$ implies a disjunction of achievers for each of its preconditions in step$_{i-1}$.
- Two mutex actions cannot be chosen in the same step.
- Only actions applicable in the initial state can be chosen in step$_0$.
- The effects of the actions chosen in the final step of the plan must satisfy the goal.

A clause may include a set of universal variables in order to become satisfied in every step except for those in which the implications of the clause are required - in which case one of the remaining existential literals must satisfy the clause.

Negative preconditions in the original planning problem would cause the encoding to become unsound. Additional clauses (a small number, dwarfed by the mutex clause sets) could be added to further constrain the problem - forcing a *noop* action to be chosen for any present facts that are not deleted. This would allow for negative preconditions, but since they can be easily avoided in writing the domain, these clauses have not yet been included in the encoding.

Action Preconditions and Mutual exclusions

Encoding a single step is performed in a similar way to SAT based encodings ((Kautz, Selman, and Hoffmann 2006) and (Kautz, McAllester, and Selman 1996)) except that variables are not duplicated for each step in the plan. Instead each existential set of variables has a set of clauses which are reused at each node of the tree of the appropriate depth. These clauses assert:

- If two actions are mutex then the variables corresponding to those action choices cannot both be selected.
- If an action variable is true then for each of its preconditions a disjunction of action variables, which achieve that precondition, is implied in the previous node.

The schema for mutual exclusions is:

$$(\neg a_1 \vee \neg a_2),\ for\, all\, (a_1\, mutexwith\, a_2)$$

for each existential set. For example $(drive\, A\, B)$ is mutually exclusive with $(noop\, A)$, so two clauses are asserted:

$$(\neg(drive\, A\, B)_1 \vee \neg(noop\, A)_1)$$

$$(\neg(drive\, A\, B)_2 \vee \neg(noop\, A)_2)$$

The plan under this encoding is read with a $false \rightarrow true$ traversal of the tree, hence the steps in the plan alternate between leaf and non-leaf nodes. The schema for the precondition clauses follows, with a_1, A_i referring to action variables in the leaf nodes and a_2, A_j referring to variables quantified immediately above branch node b:

$$(\neg a_1 \vee A_1 \vee B_1 \vee \neg b)$$

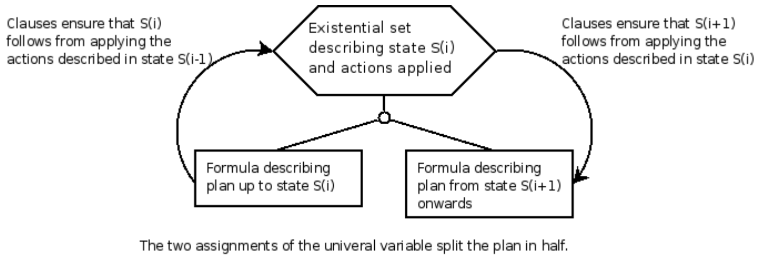
$$A_1 : \{a_j : a_j \in A_j\, st.\, f \in adds(a_j)\}$$

$$for\, all\, f \in precs(a_1),\ and\, a_1 \in A_i$$

and,

$$(\neg a_2 \vee A_2 \vee B_2 \vee b)$$

$$A_2 : \{a_i : a_i \in A_i\, st.\, f \in adds(a_i)\}$$


Clauses ensure that S(i) follows from applying the actions described in state S(i-1)
Existential set describing state S(i) and actions applied
Clauses ensure that S(i+1) follows from applying the actions described in state S(i)
Formula describing plan up to state S(i)
Formula describing plan from state S(i+1) onwards
The two assignments of the univeral variable split the plan in half.

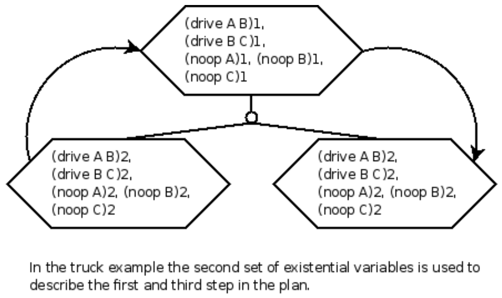

(drive A B)1, (drive B C)1, (noop A)1, (noop B)1, (noop C)1
(drive A B)2, (drive B C)2, (noop A)2, (noop B)2, (noop C)2
(drive A B)2, (drive B C)2, (noop A)2, (noop B)2, (noop C)2
In the truck example the second set of existential variables is used to describe the first and third step in the plan.

Figure 1: The universal variable is used to create a branch, and does not represent an element of the planning problem.

$$for\,all\, f \in precs(a_2)\, and\, a_2 \in A_j$$

where:

$$B_1 : \{b_i | b \prec b_i\}$$
$$B_2 : \{\neg b_i | b \prec b_i\}$$

Using the example, in order for the action $(drive\, B\, C)_1$ to be used in the second step of the plan (the root node) the following clause is involved:

$$(\neg(drive\, B\, C)_1 \vee (drive\, A\, B)_2 \vee (noop\, B)_2 \vee b)$$

This states that either the action is not performed, or one of the two actions which achieve its precondition (at B) are performed in the innermost existential set, or the branch is true - indicating that the innermost existential set represents the state after this. Figure 2 shows the variables involved. They are constrained only when the branch is assigned $false$. The other branch is formed in a similar way, with the roles of the two existential sets reversed. Since the formula must be true *for any* assignment to b then both transitions between the outer and inner sets are used to constrain the innermost existential variables, albeit in different branches of the tree - however, every precondition clause includes at least one variable from the outer existential set as it must must achieve the preconditions of the last step, as well as having its own preconditions met. Because of this overlap between clauses, information can propagate throughout the tree.

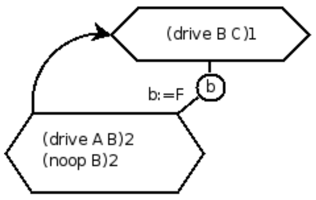


Figure 2: The variables affected by the example clause.

Initial state and goal state

It has been shown that a stage, or multiple stages, of a plan can be specified by a clause that is satisfied by any other assignment of branch variables, also that the assignment on the $false$ side of a universal branch is the preceding part of the plan. The initial state is described by the assignments in the first leaf, and therefore must hold when all universal branch variables are assigned false. To ensure the initial state holds the following clauses are included:

$$(\neg a \vee B),\ for\,all\, a\, st.\, precs(a) \not\subseteq I$$
$$B : \{b | b \in Universal\, vars\}$$

The goal state is similarly described:

$$(A \vee \neg B),\ for\,all\, f \in G, where$$
$$A : \{a | f \in adds(a)\}$$

So, in our truck example the initial state would include three clauses, which state that either it is not the initial state (b is true), or the actions whose preconditions are not met by the initial state must be false.

$$(\neg(drive\, B\, C)_2 \vee b),\ (\neg(noop\, B)_2 \vee b)$$
$$(\neg(noop\, C)_2 \vee b),$$

The goal state would be represented by a single clause, as only one fact is present in the goal state. This clause states that either it is not the goal state (b is false), or one of the achievers of this goal fact (at C) must be applied in this state.

$$((drive\, B\, C)_2 \vee (noop\, C)_2 \vee \neg b)$$

Increasing the bound

The bound on the plan length can be doubled by adding another universally quantified variables and existentially quantified variable set to the outside of the quantification layer:

$$\exists (drive\, A\, B)_3, (drive\, B\, C)_3, (noop\, A)_3,$$
$$(noop\, B)_3, (noop\, C)_3$$
$$\forall b_1\ \exists (drive\, A\, B)_1, (drive\, B\, C)_1, (noop\, A)_1,$$
$$(noop\, B)_1, (noop\, C)_1$$
$$\forall b_2\ \exists (drive\, A\, B)_2, (drive\, B\, C)_2, (noop\, A)_2,$$
$$(noop\, B)_2, (noop\, C)_2$$

The number of new clauses generated is linear in the size of the domain, as the only clauses that must be introduced are mutex clauses in the new existential set and precondition clauses for the two transitions to and from the leaf nodes, as noted previously. The new sets are added to the outside of the quantification layer, so the previous clauses do not require modification, the structure being recursive.

	Clauses						Variables					
	QBF			SAT-BASED			QBF			SAT-BASED		
prob.	15	31	63	15	31	63	15	31	63	15	31	63
dl01	4214	5352	6480	24052	74068	174100	483	604	725	1881	4761	10521
dl02	5063	6434	7787	41792	107296	238304	591	739	887	2619	6171	13275
dl03	5625	7141	8645	41369	106873	237881	659	824	989	2559	6111	13215
dl07	14956	18926	22874	149239	367063	802711	1311	1639	1967	5128	12136	26152
dl08	15991	20238	24459	162793	401929	880201	1407	1759	2111	5402	12890	27866
dl09	25438	32127	38792	292521	847449	1957305	1915	2394	2873	6480	16912	37776

Table 2: Showing the growth of SAT and QBF encodings as the plan bound is increased.

3. Results

The translation process was implemented and run on the *driverlog* and *zeno* domains from IPC-3 and the *pipesnotankage* domain from IPC-4. The QBF solvers Qube, Skizzo and Quantor were applied to the instances generated, and Ozziks was used to extract a resulting plan from Skizzo's satisfying certificate. Descriptions of the solvers can be found in (E. Giunchiglia and Tacchella 2004), (Benedetti 2005) and (Biere 2005).

The results from Quantor are shown in table 1, displaying the time taken to solve the QBF instances of the *driverlog* and *pipesnotankage* problems, with a plan bound of 15. The table also shows results from the SAT based encoding used in Blackbox42b, with the same bound on plan length. The SAT cnf encoding was solved using picosat, the same SAT solver called by Quantor. Table 2 compares the growth in the number of variables and clauses in the *driverlog* problems with increasing plan bound. Blackbox42b is used as a comparison as it uses the same actions-only encoding scheme as the QBF encoding described, without quantification. Any advanced SAT encoding could be represented in a tree structure - the same is true of many Markov Chain problems.

	QBF Encoding			SAT based encoding		
prob.	vars	clauses	time (s)	vars	clauses	time (s)
dl01	483	4214	0.14	1881	24052	0.16
dl02	591	5063	0.21	2619	41792	0.25
dl03	659	5625	0.2	2559	41369	0.24
dl07	1311	14956	0.68	5128	149239	0.83
dl08	1407	15991	0.81	5402	162793	0.91
dl09	1915	25438	7.53	6480	292521	1.63
pt01	691	12958	0.53	2791	68019	0.39
pt02	691	12960	4.86	2861	70174	0.43
pt03	1131	28966	2.09	3490	130119	0.72
pt04	1131	28968	5.08	3516	131324	0.71
pt05	1763	60852	5.04	5527	363474	2.08

Table 1: Times to solve the SAT and QBF encodings.

4. Improvements and Future Work

Further improvements to this encoding have been implemented and others considered, viable in both QBF and SAT representations, including ideas related to:

- Using landmarks, (c.f. (Porteous, Sebastia, and Hoffmann 2001) and (Richter and Westphal 2008)), to further constrain the problem, and introduce new information to encourage propagation between the initial and final states. Table 3 shows preliminary promising results on rovers problems with a subset of these improvements.
- A SAS+ based or Lifted representation, (c.f. (Kautz, McAllester, and Selman 1996)) reducing the size of the problem, break symmetry, and integrate with landmark strategies.
- Using clauses to add lower and upper bound constraints to actions based on the levels at which they appear in the Plan Graph in order to aid propagation.

prob.	extra vars	extra clauses	t_1	t_2
rov01	410	84	21.86	18.31
rov02	446	92	120.94	82.63
rov03	212	48	117.7	83.25

Table 3: Times with and without landmark clauses.

5. Conclusion

Solving this proposed encoding with the referenced QBF algorithms has shown that this approach is feasible.

It is hoped that applications such as this will encourage further improvements to QBF solving algorithms, and this approach will become comparable to other good general and specialised systematic search engines, and be applied to other PSPACE problems.

References

Benedetti, M. 2005. Evaluating qbfs via symbolic skolemization. In *Proceedings of 11th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR04)*. n. 3452 in LNCS, Springer.

Biere, A. 2005. Resolve and expand. In *Proceedings of 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'04)*. Lecture Notes in Computer Science (LNCS), vol. 3542.

Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artifcial Intelligence* 90:281–300.

Cadoli, M.; Giovanardi, A.; and Schaerf, M. 1998. An algorithm to evaluate quantied boolean formulae. In *Proceedings of 15th National Conference on Artificial Intelligence (AAAI 98)*.

E. Giunchiglia, M. N., and Tacchella, A. 2004. Qube++: an efficient qbf solver. In *Proceedings of 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD'04)*.

Fox, M., and Long, D. 1999. Efficient implementation of the plan graph in stan. *Journal of Artificial Intelligence Research* 10:87–115.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2001. Qube: a system for deciding quantified boolean formulas satisfiability. In *Proceedings of International Joint Conf. on Automated Reasoning (IJCAR'01)*.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*.

Kautz, H.; Selman, B.; and Hoffmann, J. 2006. Satplan: Planning as satisfiability.

Littman, S. M. . M. 2003. Contingent planning under uncertainty via stochastic satisfiability. *AIJ* 147.

Papadimitriou, C. H. 1993. *Computational Complexity*. Addison Wesley.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Pre-proceedings of the Sixth European Conference on Planning (ECP01)*.

Richter, S., and Westphal, M. 2008. The lama planner: Using landmark counting in heuristic search.

Rintanen, J. 2007. Asymptotically optimal encodings of conformant planning in qbf. *AAAI*.