ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-217

Planning as Quantified Boolean Formula

Michael Cashmore [1] and **Maria Fox** [2] and **Enrico Giunchiglia** [3]

Abstract. This paper introduces two techniques for translating bounded propositional reachability problems into Quantified Boolean Formulae (QBF). Both exploit the binary-tree structure of the QBF problem to produce encodings logarithmic in the size of the instance and thus exponentially smaller than the corresponding SAT encoding with the same bound. The first encoding is based on the iterative squaring formulation of Rintanen. The second encoding is a compact tree encoding that is more efficient than the first one, requiring fewer alternations of quantifiers and fewer variables. We present experimental results showing that the approach is feasible, although not yet competitive with current state of the art SAT-based solvers.

1 Introduction

Planning as Satisfiability is one of the most well-known and effective techniques for classical planning: SATPLAN [8] was an award-winning system in the deterministic track for optimal planners in the first International Planning Competition (IPC) in 1998, the 4th IPC in 2004, and the 5th IPC in 2006. The basic idea is to encode the existence of a plan with $n + 1$ (or fewer) steps as a propositional (SAT) formula obtained by unfolding, n times, the symbolic transition relation of the automaton described by the planning problem. In recent work [14], the basic SAT approach has been improved by equipping the solver with planning-specific variable and value-ordering heuristics that are similar to the helpful actions filter of FF [6]. This is very effective for solving planning problems in the SAT framework. In general, SAT-based planning, though quite successful, suffers from the problem that it is easy to come up with problems in which the number of steps required is large, making it impossible to even encode the original problem as a propositional formula. The same problem arises in bounded model checking [1]. The use of compact encoding as Quantified Boolean Formulae (QBFs) combined with the use of QBF solvers has been proposed [7, 11, 4] as a way to overcome this problem[4]. In particular, [13, 7] present an encoding that is logarithmic in n, resembling the proof of the PSPACE-hardness of solving QBFs [15, 16].

Here we introduce two techniques for translating bounded propositional reachability problems into Quantified Boolean Formulae (QBF). Both of the translations that we present exploit the PSPACE complexity of the QBF problem to produce encodings that are logarithmic in n, and thus exponentially smaller than the corresponding SAT encodings with the same bound. By encodings logarithmic in n we mean that when encoding the existence of a plan with $n + 1$ steps we require only $O(log_n)$ variables, rather than $O(n)$ as would be required in SAT. The existence of these encodings forms the first step to finding compact QBF encodings of planning problems that are more efficient than SAT encodings. The first encoding we present is a recursive formulation similar to that presented in [13]. We make no novelty claims for the first encoding - we are using it simply as an example of the standard recursive formulation which we contrast with our second encoding. The second encoding that we present reduces redundancy in the first formula and is *tree structured*. It is more efficient than the first one in that:

1. the QBF that encodes the existence of a plan of length n has one fewer universal quantifier. This means that an encoding with the same number of quantifier alternations will captures twice the makespan, and
2. even when the two formulations have the same number of universal quantifiers, it necessitates far fewer variables.

We refer to the first encoding as the *Flat Encoding* and the second as the *Compact Tree Encoding* throughout this paper. In order to determine the effectiveness of the two encodings we run a preliminary experimental analysis showing that the Compact Tree Encoding tends to perform better on hard problems, solving (or proving unsolvable) many instances that cannot be solved (or proved unsolvable) by the Flat Encoding within the 30-minute time bound that we allowed. Furthermore, our results show that the Compact Tree Encoding is much faster (often by at least one or two orders of magnitude) than the Flat Encoding on a large number of instances from classical planning benchmark domains. We show through these encodings that the QBF approach to classical planning in general is feasible. To the best of our knowledge, this is the first attempt in the literature to demonstrate this. We also stress that although we include a direct comparison with state-of-the-art planning as satisfiability techniques, we do not claim that the approaches discussed here are competitive. Indeed, such a comparison is not fair, given the maturity of the research in SAT and in planning as SAT, compared to the maturity of the much younger field of research in QBFs (and of course of classical planning as QBF, starting with this paper). Our goal is to demonstrate that QBF-based encodings are feasible for planning and to encourage further research into making them competitive. Consequently we focus on comparing the new Compact Tree Encoding with the Flat Encoding throughout this paper, comparing with SAT at the end of Section 4.

After some preliminaries in Section 2 the two encodings will be introduced in Section 3. Section 4 will detail the experiments run on these encodings and discuss the results, beginning with time based comparisons and moving onto memory consumption. Finally we conclude in Section 5.

[1] University of Strathclyde, Glasgow, G1 1XH, UK, email: michael.cashmore@strath.ac.uk
[2] King's College London, London, WC2R 2LS, UK, email: maria.fox@kcl.ac.uk
[3] Universit di Genova, 16145 Genova (GE), Italy, email: giunchiglia@unige.it

[4] It should be noted that the QBF encodings of symbolic reachability problems presented in [4, 11] have size polynomial in the makespan of the problem, and thus possibly exponential in the number of variables.

2 Preliminaries

Quantified Boolean Formula

Formally, the language of QBFs extends propositional logic by allowing for universal ($\forall$) and existential ($\exists$) quantification over variables (in our case, fluents and actions). Semantically, $\forall x.\varphi$ (resp. $\exists x.\varphi$) can be interpreted as $(\varphi_x \wedge \varphi_{\neg x})$ (resp. $(\varphi_x \vee \varphi_{\neg x})$), where φ_x (resp. $\varphi_{\neg x}$) is the formula obtained from φ by replacing x with $\top$ (resp. $\bot$).[5] The process of substituting $\forall x.\varphi$ (resp. $\exists x.\varphi$) with $(\varphi_x \wedge \varphi_{\neg x})$ (resp. $(\varphi_x \vee \varphi_{\neg x})$) is called *expansion*. By expanding all quantifiers, each QBF can be reduced to (possibly an exponentially larger) propositional formula. When every variable is quantified, such an expansion reduces to a Boolean combination of $\top$ and $\bot$ and is thus equivalent to either $\top$ or $\bot$. The QBF formula expands recursively into a tree-structured representation where all of the propositional variables are at the leaves. Expansion therefore produces an and-or tree that accords to the semantics of the QBF.

The Planning Problem

Let $\mathcal{F}$ and $\mathcal{A}$ be two finite sets representing the sets of *fluents* and *actions* respectively. A *state* (resp. *(complex) action*) is an interpretation of the set of fluents (resp. actions). In the following, we use X to denote the whole set of variables, i.e., the set of fluents unioned with the set of actions.

A *planning problem* is a triple $\langle I, \tau, G\rangle$ where

- I is a Boolean formula over $\mathcal{F}$ and represents the set of *initial states*;
- τ is a Boolean formula over $X \cup X'$ where $X' = \{x' : x \in X\}$ is a copy of the set of variables and represents the *transition relation* of the automaton describing how (complex) actions affect states (we assume $X \cap X' = \emptyset$);
- G is a Boolean formula over $\mathcal{F}$ and represents the set of *goal states*.

The above definition of the planning problem differs from the traditional ones in which the description of the effects of actions on a state is described in an high-level action language such as STRIPS. We prefer this formulation because the techniques we are going to describe are independent of the action language used. The only assumption that we make is that the description is deterministic: there is only one state satisfying I and the execution of a (complex) action α in a state s can lead to at most one state s'. More formally, for each state s and complex action α, any two interpretations extending $s \cup \alpha$ and satisfying τ interpret in the same way the variables in $\mathcal{F}' = \{f' : f \in \mathcal{F}\}$.

3 Encoding planning problems as QBFs

Consider a planning problem $\Pi = \langle I, \tau, G\rangle$. As standard in planning as satisfiability, the existence of a parallel plan with makespan n is proved by building a propositional formula with n copies of the set of variables. In the following,

- by X_α we denote one such copy of the set of variables;
- by $I(X_\alpha)$ (resp. $G(X_\alpha)$) we denote the formula obtained from I (resp. G) by substituting each $x \in X$ with the corresponding variable $x_\alpha \in X_\alpha$;
- by $\tau(X_\alpha, X_\beta)$ we denote the formula obtained from τ by substituting each variable $x \in X$ with the corresponding variable $x_\alpha \in X_\alpha$ and similarly each $x' \in X'$ with the corresponding $x_\beta \in X_\beta$.

For $n \geq 1$, the *planning problem* Π *with makespan* n is the Boolean formula Π_n defined as

$$I(X_1) \wedge \bigwedge_{i=1}^{n} \tau(X_i, X_{i+1}) \wedge G(X_{n+1}) \qquad (n \geq 0) \qquad (1)$$

and a *plan for* Π_n is an interpretation satisfying (1).

However, since the plan existence problem is a PSPACE-complete problem [3] the size of (1) can be exponential in the number of fluents - making it impossible to even build (1) in practice. QBFs are a promising alternative representation language given that:

1. there exists an encoding of the planning problem with makespan n as QBFs which are polynomial in the number of fluents (assuming, as we do, a deterministic transition relation), and
2. there is a growing interest in developing efficient solvers for QBFs; see, for example, the report from the last QBF competition [12].

Flat Encoding

The *flat encoding* of a planning problem $\langle I, \tau, G\rangle$ with makespan $n = 2^k$ is the formula:

$$I(X_I) \wedge E_k(X_I, X_G) \wedge G(X_G) \qquad (2)$$

where $k \geq 0$ is the *folding parameter* and $E_k(X_I, X_G)$ is defined as follows (in the following, given two finite sets X_α and X_β of variables, $\exists X_\alpha X_\beta$ denotes the result of existentially quantifying each variable in $X_\alpha \cup X_\beta$, and $(X_\alpha \leftrightarrow X_\beta)$ stands for $(\wedge_{x\in X} x_\alpha \leftrightarrow x_\beta)$):

$$\begin{aligned} E_k(X_I, X_G) := {} & \exists X_{st} \forall y \exists X_s X_t (\\ & (\neg y \Rightarrow ((X_{st} \leftrightarrow X_t) \wedge (X_I \leftrightarrow X_s))) \wedge \\ & (y \Rightarrow ((X_{st} \leftrightarrow X_s) \wedge (X_G \leftrightarrow X_t))) \wedge \\ & E_{k-1}(X_s, X_t) \qquad) \end{aligned}$$

if $k > 0$, and

$$\begin{aligned} E_0(X_I, X_G) := {} & \exists X_1 \exists X_2 \\ & ((X_I \leftrightarrow X_1) \wedge \tau(X_1, X_2) \wedge (X_2 \leftrightarrow X_G)) \end{aligned}$$

when $k = 0$. The correspondence between (1) and the Flat Encoding is clear when $k = 0$. When $k > 0$, by expanding the universal quantifiers we find the same correspondence. Intuitively, the branch formed by expanding y divides the formula into $E_{k(\neg y)}$ and $E_{k(y)}$, each representing one half of the total timesteps. As X_{st} remains above this branch it is used to link these two halves together with equality constraints.

The above formulation involves $(3k+2)|X|$ existential variables and k universal variables. Further, the Flat Encoding can be converted into prenex conjunctive normal form (corresponding to the QDIMACS format used by most QBF solvers) with $4(2k+1)|X| + |\tau|$ clauses, where $|\tau|$ is the number of clauses in the transition relation of the original planning problem.

Proposition 1 *For each $k \geq 0$, if $n = 2^k$ the existential closures of (1) and the Flat Encoding are equivalent.*

The proof can be found in Rintanen [13], in which the Flat encoding is first introduced.

[5] $\top$ and $\bot$ are the logical symbols we use for truth and falsity.

Compact Tree Encoding

We now describe a new encoding which removes redundancy and requires considerably fewer variables than the Flat Encoding.

Intuitively, the Flat Encoding describes a one-to-one correspondence between the states traversed and the *leaves* of the expansion corresponding to the QBF. The leaves of the expansion are composed of the values of the innermost existentially quantified variables. By contrast, in our second encoding, the Compact Tree Encoding, there is a one-to-one correspondence between the states traversed and the *nodes* of the tree corresponding to the QBF. Every existentially quantified layer in the QBF is used to represent at least one distinct state.

Given a QBF formula with k universal quantifiers, the new encoding is a tree of depth k which removes all redundancy from the formula, by only specifying equivalent states once. The key novelty of this encoding is in the traversal of its tree structure, in which edges are encoded from each leaf node to one of the nodes in each of the preceding layers of the tree. This leads to a formula that encodes 2^{k+1} transitions in a tree with k layers. The formula is quadratic in k because every edge to and from level i requires $k - i - 1$ terms to express the context which is the assignment to the variables in the intervening layers. While the Flat Encoding does not require these contexts and therefore is linear in k, twice as many variables are required to enforce the equivalence of sets of existentially quantified variables on different branches as are required in the Compact Tree Encoding to describe the traversal of the tree.

The *Compact Tree Encoding* of a planning problem $\langle I, \tau, G\rangle$ with makespan $n = 2^{k+1}$ is the formula:

$$I(X_I) \land Q_k(X_I, X_G) \land G(X_G) \tag{3}$$

where

$$\begin{array}{l} Q_k(X_I, X_G) := \exists X_k \forall y_k \ldots \exists X_1 \forall y_1 \exists X(\\ \quad ((\bigwedge_{i=1}^{k} \neg y_i) \Rightarrow \tau(X_I, X)) \\ \quad \land((\bigwedge_{i=1}^{k} y_i) \Rightarrow \tau(X, X_G)) \\ \quad \bigwedge_{i=1}^{k}(\ ((\neg y_i \bigwedge_{j=1}^{i-1} y_j) \Rightarrow \tau(X, X_i)) \\ \qquad\qquad \land((y_i \bigwedge_{j=1}^{i-1} \neg y_j) \Rightarrow \tau(X_i, X))) \quad) \end{array}$$

This formula states that the goal state G_G is reachable in 2^{k+1} applications of the transition relation. Only $k + 1$ states are quantified (X_k to X_1 and X).

The Compact Tree Encoding (3) has k universal variables and $(k + 1)|X|$ existential variables. Further, with (3) we check the existence of plans having makespan equal to 2^{k+1}, i.e., twice the makespan allowed by the Flat Encoding for the corresponding number of quantifier alternations. The conversion of (3) to prenex conjunctive normal form has $2(k + 1)|\tau|$ clauses. Although this has more clauses than the Flat Encoding and in some cases requires more memory it can be seen in Section 4 that in practice it does not make the problem more difficult to solve.

The correspondence between the Compact Tree Encoding and (1) is again found when expanding the universally quantified variables. Unlike the Flat Encoding however, when expanding y_k, X_k does not link the two halves of the plan using equality constraints, but instead with two transition relations - X_k representing a distinct state in the solution to the planning problem.

Proposition 2 *For each $k \geq 0$, if $n = 2^{k+1}$ the existential closures of (1) and the Compact Tree Encoding (3) are equivalent.*

Proof: We prove, by induction on k, that $Q_k(X_I, X_G)$ is equivalent to (1) with $n = 2^{k+1}$

$\underline{k = 0}$**:** In this case, $Q_k(X_I, X_G)$ becomes

$$\exists X(\tau(X_I, X) \land \tau(X, X_G))$$

and the thesis trivially holds.

$\underline{k = p + 1}$**:** By expanding the outermost universal quantifier, y_{p+1}, $(\bigwedge_{i=1}^{k} \neg y_i) \Rightarrow \tau(X_I, X)$ becomes false in one branch of the expansion, $(Q_{k,y_{p+1}})$, and $(\bigwedge_{i=1}^{k} y_i) \Rightarrow \tau(X, X_G)$ in the other, $(Q_{k,\neg y_{p+1}})$.
In addition

$$\bigwedge_{i=1}^{p+1}\left(\begin{array}{l} (\neg y_i \bigwedge_{j=1}^{i-1} y_j) \Rightarrow \tau(X, X_i)) \\ \land((y_i \bigwedge_{j=1}^{i-1} \neg y_j) \Rightarrow \tau(X_i, X)) \end{array} \right)$$

is reduced to

$$\bigwedge_{i=1}^{p}\left(\begin{array}{l} (\neg y_i \bigwedge_{j=1}^{i-1} y_j) \Rightarrow \tau(X, X_i)) \\ \land((y_i \bigwedge_{j=1}^{i-1} \neg y_j) \Rightarrow \tau(X_i, X)) \end{array} \right)$$

with two clauses falling out when $i = p + 1$: $(\bigwedge_{i=1}^{p} y_i) \Rightarrow \tau(X, X_{p+1})$ in $(Q_{k,y_{p+1}})$ and $(\bigwedge_{i=1}^{p} \neg y_i) \Rightarrow \tau(X_{p+1}, X)$ in $(Q_{k,\neg y_{p+1}})$.
After expansion $Q_k(X_I, X_G)$ becomes

$$\begin{array}{l} \exists X_{p+1}(\\ \quad \exists X_p \forall y_p \ldots \exists X_1 \forall y_1 \exists X(\\ \quad (\bigwedge_{i=1}^{p} \neg y_i \Rightarrow \tau(X_I, X)) \land \\ \quad (\bigwedge_{i=1}^{p} y_i \Rightarrow \tau(X, X_{p+1})) \land \\ \quad (\bigwedge_{i=1}^{p} \ ((\neg y_i \land \bigwedge_{j=1}^{i-1} y_j) \Rightarrow \tau(X, X_i)) \\ \qquad\qquad \land((y_i \land \bigwedge_{j=1}^{i-1} \neg y_j) \Rightarrow \tau(X_i, X))) \quad) \land \\ \quad \exists X_p \forall y_p \ldots \exists X_1 \forall y_1 \exists X(\\ \quad (\bigwedge_{i=1}^{p} \neg y_i \Rightarrow \tau(X_{p+1}, X)) \land \\ \quad (\bigwedge_{i=1}^{p} y_i \Rightarrow \tau(X, X_G)) \land \\ \quad (\bigwedge_{i=1}^{p} \ ((\neg y_i \land \bigwedge_{j=1}^{i-1} y_j) \Rightarrow \tau(X, X_i)) \land \\ \qquad\qquad ((y_i \land \bigwedge_{j=1}^{i-1} \neg y_j) \Rightarrow \tau(X_i, X))) \quad)) \end{array}$$

By induction hypothesis,

$$\begin{array}{l} \exists X_p \forall y_p \ldots \exists X_1 \forall y_1 \exists X(\\ (\bigwedge_{i=1}^{p} \neg y_i \Rightarrow \tau(X_I, X)) \land \\ (\bigwedge_{i=1}^{p} y_i \Rightarrow \tau(X, X_{p+1})) \land \\ (\bigwedge_{i=1}^{p} \ ((\neg y_i \land \bigwedge_{j=1}^{i-1} y_j) \Rightarrow \tau(X, X_i)) \land \\ \qquad ((y_i \land \bigwedge_{j=1}^{i-1} \neg y_j) \Rightarrow \tau(X_i, X))) \quad) \end{array}$$

is equivalent to

$$\begin{array}{l} \exists X'_1 \ldots X'_{2^{p+1}-1}(\tau(X_I, X'_1) \\ \land \bigwedge_{i=1}^{2^{p+1}-2} \tau(X'_i, X'_{i+1}) \land \tau(X'_{2^{p+1}-1}, X_{p+1})) \end{array}$$

The second half of the expansion can be equated to a similar expression in an analogous fashion - with the exception that the chain of transitions are between X_{p+1} and X_G, as they represent the second half of the plan. Thus, by combining the conjunctions we get

$$\begin{array}{l} \exists X_1 \ldots X_{2^{p+2}-1}(\tau(X_I, X_1) \\ \land \bigwedge_{i=1}^{2^{p+2}-2} \tau(X_i, X_{i+1}) \land \tau(X_{2^{p+2}-1}, X_G)) \end{array}$$

which is the thesis. □

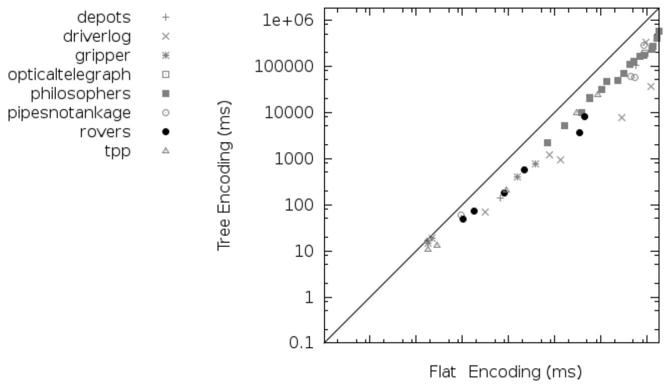

Figure 1. Times on problems solved with DepQBF using the Compact Tree Encoding and Flat Encoding, times in ms.

	Flat Encoding		Compact Tree Encoding	
solver	#solved	unique	#solved	unique
QuBE7.0	39	0	61	22
DepQBF	58	0	68	10

Table 1. Number of solved problems using the Compact Tree Encoding and Flat Encoding.

Abstract Example

The following simple abstract example emphasises the difference between the Compact Tree Encoding and the Flat Encoding.

If we build an expression containing two universal quantifiers using the Compact Tree Encoding, we express the existence of a plan of makespan 8. Below is a simple example,

$$\begin{aligned}&\exists X_2 \forall y_2 \exists X_1 \forall y_1 \exists X \cdot (\\ &\qquad (\neg y_2 \wedge \neg y_1 \Rightarrow \tau(I,X)) \quad \wedge\\ &\qquad (y_2 \wedge y_1 \Rightarrow \tau(X,G)) \quad \wedge\\ &\qquad (\neg y_2 \wedge y_1 \Rightarrow \tau(X,X_2)) \quad \wedge\\ &\qquad (y_2 \wedge \neg y_1 \Rightarrow \tau(X_2,X)) \quad \wedge\\ &\qquad (\neg y_1 \Rightarrow \tau(X,X_1)) \quad \wedge\\ &\qquad (y_1 \Rightarrow \tau(X_1,X)))\end{aligned}$$

It should be noted that the last two transitions are both invoked twice: once in the context where y_2 is true and once in the context where it is false. The other transitions are all invoked once, accounting for all 8 transitions.

By contrast, two universally quantified variables using the Flat Encoding produces:

$$\begin{aligned}&\exists X_1 \forall y_1 \exists X_2 \exists X_3 \cdot (\\ &(y_1 \Rightarrow I \leftrightarrow X_2 \wedge X_1 \leftrightarrow X_3) \wedge\\ &(\neg y_1 \Rightarrow X_1 \leftrightarrow X_2 \wedge X_3 \leftrightarrow G) \wedge\\ &\quad \exists X_4 \forall y_2 \exists X_5 \exists X_6 \cdot\\ &\quad ((y_2 \Rightarrow X_2 \leftrightarrow X_5 \wedge X_4 \leftrightarrow X_6) \wedge\\ &\quad (\neg y_2 \Rightarrow X_5 \leftrightarrow X_5 \wedge X_3 \leftrightarrow X_6) \wedge\\ &\qquad \exists X_7 \exists X_8 \cdot\\ &(X_5 \leftrightarrow X_7 \wedge X_8 \leftrightarrow X_6 \wedge \tau(X_7,X_8))))\end{aligned}$$

The single transition is invoked in all of the contexts generated by assignments to y_1 and y_2. This is only 4 contexts, so the encoding expresses the existence of a plan of 4 transitions.

It can be seen that by expanding the universal quantifiers in both Flat and Compact Tree Encoding, we get propositional formulae in which the Compact Tree Encoding has twice as many transition relations. The Flat Encoding uses twice as many variables as the Compact Tree Encoding for the same number of universal quantifiers, and encodes only half as many transitions.

4 Results

We ran some experiments to form a number of comparisons, hoping to show that:

- the Compact Tree Encoding uses less memory, and less time than the Flat Encoding,
- the QBF approach uses less memory than SAT,
- there exists a problem that can be represented using QBF, but cannot be built in practice with SAT, due to memory limitations.

In general, we do not expect that the QBF approach is competitive with SAT in terms of time, but include a comparison to illustrate this gap in performance.

For these we used encodings of STRIPS-style planning problems from the IPC benchmarks; *depots, driverlog, freecell, gripper, opticaltelegraph, philosophers, pipesnotankage, rovers, tpp* and *zeno*. These were solved with a variety of solvers in a number of experiments.

The problems were solved using Quantified Boolean Formula as follows:

A plan graph was created until level-off and then encoded as a QBF, this was passed to the choice of solver. If the QBF proved to be unsatisfiable a larger encoding (larger makespan) was created and the process repeated. Once a QBF was found to be satisfiable, or the time limit (30 minutes per encoding) was reached, the next problem

	DepQBF				QuBE7					
domain	CTE		Flat		CTE		Flat		SATPLAN'06	
	average	smoothed average	average	smoothed average	average	smoothed average	average	smoothed average	average	smoothed average
depots	**281**	292	283	**201**	1994	1211	563	329	1944	1427
driverlog	**217**	**186**	539	293	772	527	432	245	2090	2193
gripper	**68**	**89**	126	135	145	182	384	306	437	496
freecell	495	501	**143**	**175**	765	802	365	279	2087	2613
philosophers	105	70	669	370	**54**	**51**	344	240	902	763
pipesnotankage	401	313	**377**	**220**	1910	871	965	468	2095	2194
rovers	**524**	**314**	573	392	597	422	835	558	2061	1577

Table 2. Memory used solving problems using SATPLAN'06, the Compact Tree Encoding (CTE) and Flat Encoding, solving QBFs with DepQBF and QuBE7; sizes to the nearest MB.

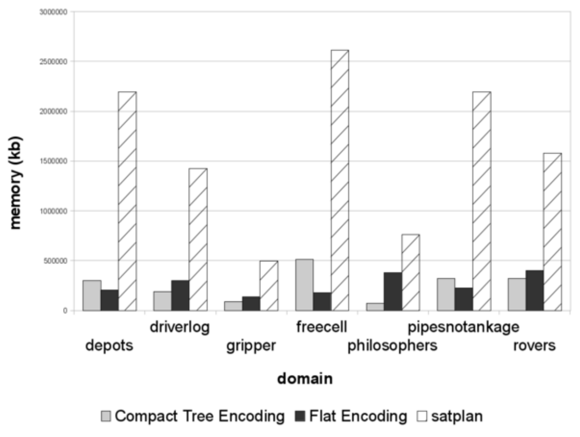

Figure 2. Average memory usage over time per domain (smoothed average), SAT and QBF encodings, solving QBFs with DepQBF; memory in (kb)

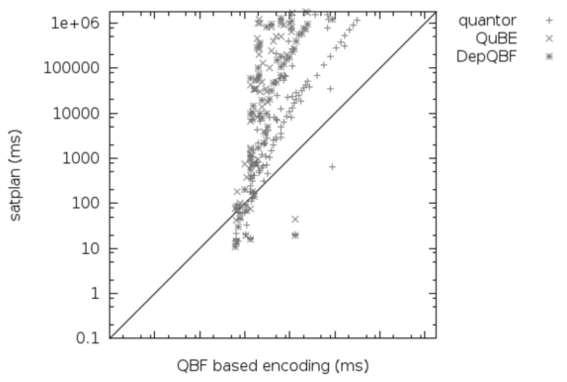

Figure 3. Times on problems solved using the Compact Tree Encoding with various solvers against SAT, times in ms.

was considered. This was repeated for each pairing of formula and solver.

Comparing the Flat with the Compact Tree Encoding

The encodings were solved using DepQBF [10] and QuBE7 [5]. QuBE7 was used as, although it is not the latest version, we have a modified version that allows for easier plan extraction. We do not present a resolution based solver in these results, such as Quantor-3.0 [2] because it resolves the formula to SAT and hence it performs similarly to our SAT based encoding. We found that DepQBF used far less memory in solving these encodings, and that, although the results are mixed, the smallest memory footprint is found when using the Compact Tree Encoding and DepQBF. As a result DepQBF will be used primarily in the discussion and figures. Figure 1 compares times (in ms) for solving the Compact Tree Encoding and Flat Encoding with DepQBF.

All problems from the benchmarks listed above were tested, with only problems solved by both techniques included in the figure 1. This comparison shows that the Compact Tree Encoding dominates the Flat Encoding in time. Table 1 reinforces this, showing the number of problems solved overall using each encoding. The column *unique* in Table 1 displays the number of problems solved by an encoding that were not solved by the alternate encoding.

Memory Comparison

The memory used for each approach was recorded. For each domain every problem was attempted, under the same time limits and on the same machines as previous experiments. Problems with greater than 5120 grounded fluent and action variables were not attempted. The memory use is displayed in Table 2 and Figure 2.

In Table 2 the *average* column denotes the mean of the maximum and minimum memory footprint of the solver. The memory used by the solver was recorded at small time intervals throughout solving, the *smoothed average* column is the mean value over time. The *average* will be sensitive to spikes in memory usage, while *smoothed average* presents a clearer view of the memory used over the course of the solving process. *smoothed average* is close to the average in the QBF-based approaches, in most cases it is smaller: the amount of memory used is low for the majority of the execution time. The opposite is true for the SAT-based approach, in which the amount of memory used increases quickly from the minimum and levels out.

As can be seen from the tables, all QBF-based approaches use far less memory than SAT.

The minimum amount of memory used is ideally the amount of space taken once the problem is grounded and translated into a boolean formula. As should be expected this is much smaller in the QBF form. The maximum amount of memory used behaves erratically in the results for QUBE7 (when compared to the SAT results). This is due to the way in which QUBE7 approaches the problem: the size of learned cubes can grow very large if the initial guesses to the solution are far from correct and a lot of backtracking is involved. This is often the case in the *depots* domain, in which QUBE7 performed very poorly.

Comparison with SAT

The experiment carried out to compare the Compact Tree Encoding with the Flat Encoding was repeated using a SAT based planner. SATPLAN'06 was used for this as it uses the same actions-only translation of the plan graph and states to boolean formula as the encodings described here [9]. Alternative representations of state, improvements in Planning level constraints and modifications to the solver have all been shown in improve the effectiveness of SAT based planners [9, 17, 14]. We do not compare against more sophisticated SAT based planners as we are concerned only with the gap in performance between QBF and basic, unenhanced SAT approaches, and so choose a SAT planner with the same state and transition representation as our QBF implementation. It should be noted however that these improvements are also applicable to QBF-based approaches with little or no modification.

The results are shown in Figure 3.

Although on some smaller problems the QBF approach is faster, overall it scales much more poorly than the SAT-based planner. However, this is only the first step of future work in QBF representation and solving for planning - the Compact Tree Encoding provides a foundation upon which more competitive, and more efficient QBF planners can be built.

5 Conclusions

In this paper we have presented methods for encoding reachability problems (specifically, propositional planning problems), as QBFs. We described two alternative encodings. The first encoding, the Flat Encoding, is an equivalence-based encoding based on [13]. The second encoding, the Compact Tree Encoding, uses fewer universal and existential quantified variables than the first when encoding problems with a given makespan. In fact, for plans encoded with the same depth of universal quantification, the Compact Tree Encoding describes a plan of double the makespan of that described by the Flat Encoding. We have experimented with sets o f problem instances taken from the IPC benchmarks, and we have shown that the second encoding leads to faster solution (and proof of unsolvability) of the harder problems in these sets.

SAT-based planners work on encodings in which every variable represents some aspect of the Planning problem. However, in the Flat Encoding the majority of the variables are redundant - essentially serving only as machinery to allow the problem to be represented in QBF. In the Compact Tree Encoding we include only variables that represent distinct aspects of the underlying Planning problem, much like SAT. We consider this to be the more natural translation of Planning to QBF, as well as the more elegant. We find this exciting because, as our results show, it holds the potential for marked improvement in the performance of planners based on QBF-solving.

REFERENCES

[1] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, 'Symbolic model checking without BDDs', in *Proceedings of the Fifth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '99)*, pp. 193–207, (1999).

[2] Armin Biere, 'Resolve and expand', in *Proc. SAT*, pp. 59–70, (2004).

[3] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *Artif. Intell.*, **69**(1-2), 165–204, (1994).

[4] Nachum Dershowitz, Ziyad Hanna, and Jacob Katz, 'Bounded model checking with QBF', in *SAT*, pp. 408–414, (2005).

[5] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella, 'QUBE: A system for deciding quantified Boolean formulas satisfiability', in *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'2001), LNAI 2083*, pp. 364–369, (2001). QUBE is available at `www.mrg.dist.unige.it/star/qube`.

[6] Jörg Hoffmann and Bernhard Nebel, 'The FF planning system: fast plan generation through heuristic search', *J. Artif. Int. Res.*, **14**, 253–302, (May 2001).

[7] Toni Jussila and Armin Biere, 'Compressing BMC encodings with QBF', *Electr. Notes Theor. Comput. Sci.*, **174**(3), 45–56, (2007).

[8] Henry Kautz and Bart Selman, 'Planning as Satisfiability', in *Proc. ECAI*, pp. 359–363, (1992).

[9] Henry Kautz and Bart Selman, 'Pushing the envelope: planning, propositional logic and stochastic search', in *Proc. AAAI-96*, pp. 1194–1201, (1996).

[10] Florian Lonsing and Armin Biere, 'Depqbf: A dependency-aware qbf solver', *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, **7**, 71–76, (2010).

[11] Hratch Mangassarian, Andreas G. Veneris, and Marco Benedetti, 'Robust QBF encodings for sequential circuits with applications to verification, debug, and test', *IEEE Trans. Computers*, **59**(7), 981–994, (2010).

[12] Claudia Peschiera, Luca Pulina, Armando Tacchella, Uwe Bubeck, Oliver Kullmann, and Ines Lynce, 'The seventh QBF solvers evaluation (QBFEVAL'10)', in *Proc. SAT*, (2010).

[13] Jussi Rintanen, 'Partial implicit unfolding in the Davis-Putnam procedure for Quantified Boolean Formulae', in *Proc. LPAR*, volume 2250 of *LNCS*, pp. 362–376, (2001).

[14] Jussi Rintanen, 'Heuristics for planning with SAT', in *Proceedings of the 16th international conference on Principles and practice of constraint programming*, CP'10, pp. 414–428, Berlin, Heidelberg, (2010). Springer-Verlag.

[15] Walter J. Savitch, 'Relationships between nondeterministic and deterministic tape complexities', *J. Comput. Syst. Sci.*, **4**(2), 177–192, (1970).

[16] Larry J. Stockmeyer and Albert R. Meyer, 'Word problems requiring exponential time: Preliminary report', in *STOC*, pp. 1–9, (1973).

[17] Zhao Xing Yixin Chen and Weixiong Zhang, 'Long-distance mutual exclusion for propositional planning', *Proc. International Joint Conference on Artificial Intelligence (IJCAI-07)*.