Asymptotically Optimal Encodings of Conformant Planning in QBF

Jussi Rintanen
NICTA Ltd & the Australian National University
Canberra, Australia

Abstract

The world is unpredictable, and acting intelligently requires anticipating possible consequences of actions that are taken. Assuming that the actions and the world are deterministic, planning can be represented in the classical propositional logic. Introducing nondeterminism (but not probabilities) or several initial states increases the complexity of the planning problem and requires the use of quantified Boolean formulae (QBF).

The currently leading logic-based approaches to conditional planning use explicitly or implicitly a QBF with the prefix $\exists\forall\exists$. We present formalizations of the planning problem as QBF which have an asymptotically optimal linear size and the optimal number of quantifier alternations in the prefix: $\exists\forall$ and $\forall\exists$. This is in accordance with the fact that the planning problem (under the restriction to polynomial size plans) is on the second level of the polynomial hierarchy, not on the third.

Introduction

Conditional planning is the problem of constructing a plan that reaches the goals despite the nondeterminism in the actions or in the world. There are different flavors of conditional planning, with different definitions of plans, observability assumptions and the form of uncertainty.

Conditional planning without probabilities can be motivated on two grounds. First, the probabilities of the different nondeterministic events may be unknown and the objective is to guarantee that the goals are reached. This is a difference to Markov decision processes. Second, nondeterminism can be understood as an intelligent opponent whose actions cannot be a priori associated probabilities with, as in game playing (Russell & Wolfe 2005).

Planning research targets problems with very big state spaces. Unlike extensive games in game theory, the state spaces of planning problems are typically expressed in terms of valuations of state variables. This makes it possible to express and solve big problem instances without explicitly enumerating all the states.

Some of the main approaches to solving non-probabilistic conditional planning problems with observability restrictions (partial observability) are search in the belief space (Bonet & Geffner 2000) and logic-based approaches more or less directly based on quantified Boolean formulae (QBF) (Rintanen 1999). The latter generalize SAT-based classical planning (Kautz & Selman 1996).

In this paper we investigate solutions methods for nondeterministic planning without observability in the QBF framework. This planning problem is also known as *conformant planning*. An efficient solution method for QBF-based planning, analogously to SAT-based planning, consists of the following three largely orthogonal components.

1. Efficient QBF encodings $\Phi(t)$ of the planning problem for given plan lengths t.

   QBF encodings have been presented by Rintanen (1999). QBFs with prefix $\exists\forall$ are implicit in (Giunchiglia 2000): search for plans (corresponding to the $\exists$ quantifier) is performed explicitly, and validity/correctness of candidate plans (corresponding to the $\forall$ quantifier) is reduced to a satisfiability test. See the discussion of related work.

2. Efficient algorithms for evaluating $\Phi(t)$.

   QBF can be evaluated by algorithms that generalize the Davis-Putnam procedure to AND-OR search (Giunchiglia, Narizzano, & Tacchella 2002) and algorithms based on eliminating quantifiers in the QBF (Biere 2005) or in its representation in some normal form like DNNF (Palacios & Geffner 2005).

3. Efficient algorithms for finding values of t such that $\Phi(t)$ corresponds to a plan.

   The simplest algorithm sequentially tries out values $0, 1, 2, 3, \ldots$ of t. More sophisticated algorithms that allow trading optimal plan length to improved runtimes are the same as for SAT planning (Rintanen, Heljanko, & Niemelä 2006).

In this work we address problem 1 by devising a new QBF encoding for nondeterministic planning without observability (conformant planning). The encoding can be directly generalized to more general conditional planning problems with partial observability, and it can be applied in connection with all the main approaches to evaluate QBF. Encodings that have been presented earlier (Rintanen 1999) are not optimal with respect to two problem parameters: asymptotic size and the number of quantifier alternations. We rectify the problem with the latter parameter in this paper. Optimality with respect to the size which is related to the interference

constraints has already been presented (Rintanen, Heljanko, & Niemelä 2006).

The existing encodings of conditional planning as a QBF use the prefix $\exists P \forall C \exists E \Phi$ saying that *there is* a plan (represented by variables P) so that *for all* contingencies (represented by variables C) *there is* an execution (represented by variables E) so that the goals are reached (Rintanen 1999).

The evaluation problem of $\exists\forall\exists$ QBF is a Σ_3^p-complete problem. A simple complexity analysis shows that conditional planning is simpler, located on the second level of the polynomial hierarchy, not on the third (Rintanen 1999). A Σ_2^p Turing machine is a nondeterministic machine with an NP oracle. The oracle Turing machine solves the planning problem by guessing a polynomial-size plan and then testing with the oracle, which runs in nondeterministic polynomial time, that there is no bad execution of the plan. This means that it is possible to formalize the planning problem as a QBF with the prefix $\exists\forall$. There is a generic way of turning the Turing machines in $\text{NP}^{\text{NP}} = \Sigma_2^p$ membership proofs into QBF, but this is not a practically useful or interesting way of obtaining 2-quantifier QBF for conformant planning.

It is presumed that QBF with a smaller number of prefix alternations are easier to evaluate (subject to complexity-theoretic assumptions such as P≠NP.) Hence an efficient QBF encoding of planning with an optimal prefix $\exists\forall$ or $\forall\exists$ should lead to more efficient planning than encodings with prefix $\exists\forall\exists$. This is the starting point of this paper.

Preliminaries

We define in detail how (partially-ordered/parallel) plans with nondeterministic operators are represented in the propositional logic. The encoding of nondeterminism is important for understanding how the QBF in the later sections work.

Definition 1. *Let A be a set of state variables. An* operator *is a pair $\langle c, e\rangle$ where c is a propositional formula over A (the* precondition*), and e is an* effect *over A. Effects over A are recursively defined as follows.*

1. *a and $\neg a$ for state variables $a \in A$ are effects over A.*
2. *$e_1 \wedge \cdots \wedge e_n$ is an effect over A if $e_1, \ldots, e_n$ are effects over A (the special case with $n = 0$ is the empty effect $\top$).*
3. *$c \rhd e$ is an effect over A if c is a formula over A and e is an effect over A.*
4. *$e_1 | \cdots | e_n$ is an effect over A if $e_1, \ldots, e_n$ for $n \geq 2$ are effects over A.*

The compound effects $e_1 \wedge \cdots \wedge e_n$ denote executing all the effects $e_1, \ldots, e_n$ simultaneously. In conditional effects $c \rhd e$ the effect e is executed if c is true in the current state. The effects $e_1 | \cdots | e_n$ denote nondeterministic choice between the effects $e_1, \ldots, e_n$. Exactly one of them is chosen randomly.

Definition 2 (Operator execution)**.** *Let $\langle c, e\rangle$ be an operator over A. Let s be a state (a valuation of A). The operator is* executable in *s if $s \models c$ and every set $E \in [e]_s$ is consistent. The set $[e]_s$ is recursively defined as follows.*

1. $[a]_s = \{\{a\}\}$ *and* $[\neg a]_s = \{\{\neg a\}\}$ *for* $a \in A$.
2. $[e_1 \wedge \cdots \wedge e_n]_s = \{\bigcup_{i=1}^n E_i | E_1 \in [e_1]_s, \ldots, E_n \in [e_n]_s\}$.
3. $[c' \rhd e]_s = [e]_s$ *if* $s \models c'$ *and* $[c' \rhd e]_s = \{\emptyset\}$ *otherwise.*
4. $[e_1 | \cdots | e_n]_s = [e_1]_s \cup \cdots \cup [e_n]_s$.

The alternative successor states of a state s when operator $\langle c, e\rangle$ is executed are those that are obtained from s by making the literals in some $E \in [e]_s$ true and retaining the values of state variables not occurring in E.

A problem instance is $\langle A, I, O, G\rangle$ where A is a set of state variables, I and G are formulae over A (respectively representing the sets of initial and goal states), and O is a set of operators over A. There is a solution plan for this problem instance if there is a sequence $o_1, \ldots, o_n$ of operators such that $\{o_1, \ldots, o_n\} \subseteq O$ and for all states s such that $s \models I$ the operator sequence is executable starting from s and every execution terminates in a state s' such that $s' \models G$.

In the following we use a notion of plans that allows the execution of one or more operators simultaneously as long as they do not *interfere*, which means that executing them in both orders $o_1; o_2$ and $o_2; o_1$ is possible and has the same effect in both cases. To obtain a sequential plan simultaneous operators may be ordered arbitrarily.

To simplify the presentation we adopt the following stricter assumption for executability than the one in Definition 2.

Assumption 1. *Every operator is executable if and only if its precondition is true.*

For example, the operator $\langle a, ((c \rhd b)|d) \wedge ((c \rhd \neg b)|d)\rangle$ is not allowed because its effects are not well-defined if c is true. The same operator with precondition $a \wedge \neg c$ is allowed.

The operators are assumed to be in a normal form in which effects e in conditional effects $c \rhd e$ do not contain nondeterminism $|$ (Rintanen 2003). For simplicity of presentation we further transform nondeterministic choices $e_1 | \cdots | e_n$ so that only binary choices exist. For example $a|b|c|d$ is replaced by $(a|b)|(c|d)$. Each binary choice can be encoded in terms of one auxiliary variable.

The condition for the atomic effect l to be executed when e is executed is $EPC_l^{nd}(e, \sigma)$. The sequence σ of integers is used for deriving unique names for auxiliary variables x_σ which control nondeterminism. The sequences correspond to paths in the tree formed by nested nondeterministic choices and conjunctions.

$$
\begin{array}{ll}
EPC_l(\top) & = \bot \\
EPC_l(l) & = \top \\
EPC_l(l') & = \bot \text{ when } l \neq l' \text{ (for literals } l') \\
EPC_l(e_1 \wedge \cdots \wedge e_n) & = EPC_l(e_1) \vee \cdots \vee EPC_l(e_n) \\
EPC_l(c \rhd e) & = c \wedge EPC_l(e) \\
EPC_l^{nd}(e, \sigma) & = EPC_l(e) \text{ if } e \text{ is deterministic} \\
EPC_l^{nd}(e_1|e_2, \sigma) & = (x_\sigma \wedge EPC_l^{nd}(e_1, \sigma 1)) \\
 & \quad \vee (\neg x_\sigma \wedge EPC_l^{nd}(e_2, \sigma 1)) \\
EPC_l^{nd}(e_1 \wedge \cdots \wedge e_n, \sigma) & = \bigwedge_{i=1}^n EPC_l^{nd}(e_i, \sigma i)
\end{array}
$$

Nondeterminism is encoded by making the effects conditional on the values of the auxiliary variables x_σ. Different valuations of these auxiliary variables correspond to different nondeterministic effects.

The following frame axioms express the conditions under which state variables $a \in A$ may change from true to false and from false to true. Let $e_1, \ldots, e_n$ be the effects of $o_1, \ldots, o_n$ respectively. Each operator $o \in O$ has a unique integer index $\Omega(o)$. For a state variable a the propositional variable a denotes the value of a in the current state and a' denotes its value in the successor state.

$$(a \wedge \neg a') \rightarrow \bigvee_{i=1}^{n}((o_i \wedge \mathit{EPC}_{\neg a}^{nd}(e_i, \Omega(o_i)))$$
$$(\neg a \wedge a') \rightarrow \bigvee_{i=1}^{n}((o_i \wedge \mathit{EPC}_{a}^{nd}(e_i, \Omega(o_i)))$$

For $o = \langle c, e\rangle \in O$ there is a formula for describing values of state variables in the predecessor and successor states when the operator is executed.

$$\begin{array}{l}(o \rightarrow c) \wedge \\ \bigwedge_{a \in A}(o \wedge \mathit{EPC}_{a}^{nd}(e, \Omega(o)) \rightarrow a') \wedge \\ \bigwedge_{a \in A}(o \wedge \mathit{EPC}_{\neg a}^{nd}(e, \Omega(o)) \rightarrow \neg a')\end{array}$$

Example 1. *Consider* $o_1 = \langle \neg a, (b|(c \rhd d)) \wedge (a|c)\rangle$ *and* $o_2 = \langle \neg b, (((d \rhd b)|c)|a)\rangle$. *The execution of these operators is described by the following formulae.*

$$\begin{array}{ll}\neg(a \wedge \neg a') & (\neg a \wedge a') \rightarrow ((o_1 \wedge x_{12}) \vee (o_2 \wedge \neg x_2)) \\ \neg(b \wedge \neg b') & (\neg b \wedge b') \rightarrow ((o_1 \wedge x_{11}) \vee (o_2 \wedge x_2 \wedge x_{21} \wedge d)) \\ \neg(c \wedge \neg c') & (\neg c \wedge c') \rightarrow ((o_1 \wedge \neg x_{12}) \vee (o_2 \wedge x_2 \wedge \neg x_{21})) \\ \neg(d \wedge \neg d') & (\neg d \wedge d') \rightarrow (o_1 \wedge \neg x_{11} \wedge c)\end{array}$$

$$\begin{array}{ll}o_1 \rightarrow \neg a & \\ (o_1 \wedge x_{12}) \rightarrow a' & (o_1 \wedge x_{11}) \rightarrow b' \\ (o_1 \wedge \neg x_{12}) \rightarrow c' & (o_1 \wedge \neg x_{11} \wedge c) \rightarrow d' \\ o_2 \rightarrow \neg b & \\ (o_2 \wedge \neg x_2) \rightarrow a' & (o_2 \wedge x_2 \wedge x_{21} \wedge d) \rightarrow b' \\ (o_2 \wedge x_2 \wedge \neg x_{21}) \rightarrow c' & \end{array}$$

■

Two operators o_1 and o_2 may be executed in parallel only if they do not interfere. Hence we use formulae $\neg(o_1 \wedge o_2)$ for all operators o_1 and o_2 that interfere.

Let X be the set of auxiliary variables x_σ in all of the above formulae. The conjunction of all the above formulae is denoted by $\mathcal{R}(A, A', O, X)$.

We need several copies of the formula $\mathcal{R}(A, A', O, X)$ to represent sequences of operators. We use propositional variables a^i to represent the values of state variables $a \in A$ in different time points i. The set of propositional variables for the values of state variables in A at time point i is A^i. Now we can for example represent (parallel) operator sequences of length two between time points 0 and 2 in terms of the formula

$$\mathcal{R}(A^0, A^1, O^0, X^0) \wedge \mathcal{R}(A^1, A^2, O^1, X^1).$$

We abbreviate $\mathcal{R}(A^i, A^{i+1}, O^i, X^i)$ by $\mathcal{T}(i, i+1)$. Given a formula ϕ over some set of variables A, we write ϕ^t for the formula over A^t that is obtained from ϕ by replacing every $a \in A$ by a^t.

Encoding with Prefix $\exists\forall\exists$

The standard encoding of conditional planning (without observability) as a quantified Boolean formula has the prefix $\exists\forall\exists$ (Rintanen 1999).

$$\exists P \forall C \exists E \left(I^0 \rightarrow \left(\bigwedge_{i=0}^{t-1} \mathcal{T}(i, i+1) \wedge G^t \right) \right) \qquad (1)$$

Here $P = \bigcup_{i=0}^{t-1} O^i$, and $C = A^0 \cup \bigcup_{i=0}^{t-1} X^i$ consists of the propositional variables that express different contingencies (initial states and nondeterminism), and $E = \bigcup_{i=1}^{t} A^i$ is the set of all remaining propositional variables.

The QBF says that *there is* a plan such that *for all* possible contingencies *there is* an execution that leads to a goal state. Since there is no observability, the plan is a sequence of operators that cannot depend on the initial state or any events that happen during plan execution.

That three quantifiers are used is slightly surprising since the conditional planning problem with the restriction to polynomial size plans is on the second level of the polynomial hierarchy, not on the third (Rintanen 1999). Given a plan (a valuation of P), one of the initial states and the nondeterministic choices (a valuation of $P \cup C$), the execution (a valuation of E) can be computed in linear time by unit resolution.

Since reasoning with QBF with a shorter prefix is in general more efficient, there is an obvious need to develop QBF encodings of conditional planning with the prefix $\exists\forall$ or $\forall\exists$.

Encoding with Prefix $\exists\forall$

Considering that the planning problem is in Σ_2^p, there is a QBF with prefix $\exists\forall$ for testing the existence of a plan. At the technical level, given the $\exists\forall\exists$ QBF of the previous section, we either have to eliminate all the innermost existential variables, or make them universal. We follow the latter idea because the obvious implementations of the former idea increase the size of the QBF exponentially in the worst case. For our QBF $\exists P \forall E \Phi$, given a valuation of P that represents a plan, all of the valuations of the variables E should satisfy Φ. These valuations represent executions of the plan (which should reach the goals) as well as state sequences that do not correspond to an execution and which therefore do not have to reach the goals. A QBF that formalizes this is

$$\exists P \forall E \left(\iota \wedge \left[\left(I^0 \wedge \bigwedge_{i=0}^{t-1} \mathcal{T}(i, i+1) \right) \rightarrow G^t \right] \wedge \nu \right) \qquad (2)$$

where

$$P = \bigcup_{i=0}^{t-1} O^i$$
$$E = \bigcup_{i=0}^{t} A^i \cup \bigcup_{i=0}^{t-1} X_i$$
$$\iota = \bigwedge_{i=0}^{t-1} \bigwedge_{o_1, o_2 \in O \text{interfere}} \neg(o_1^i \wedge o_2^i)$$

$$\nu = I^0 \rightarrow \bigwedge_{i=0}^{t-1} \left[\left(\bigwedge_{j=0}^{i-1} \mathcal{T}(j, j+1) \right) \rightarrow \bigwedge_{o \in O} (o^i \rightarrow \mathrm{prec}(o)^i) \right].$$

The precondition of an operator o is denoted by $\text{prec}(o)$.

The formula ν states that the plan is executable: the preconditions of the operators in the plan must be true after executing the preceding steps of the plan. The correctness of this formula depends on Assumption 1.

The formula ι states that the plan does not execute two interfering operators simultaneously.

Unlike in the $\exists\forall\exists$ encoding in the previous section, the formulae $\mathcal{T}(i, i+1)$ do not have to include the interference and precondition constraints because the reachability tests for states is made assuming that the plan is valid. Validity is tested separately by formulae ι and ν.

Formula 2 has $\mathcal{O}(n^2)$ size because of a quadratic number of occurrences of $\mathcal{T}(i, i+1)$ in ν and the quadratic size of ι. First we reduce the number of occurrences of $\mathcal{T}(i, i+1)$ to t. We recursively define a sequence of formulae denoted by meta-variables $\phi_i, i \in \{0, \ldots, t\}$.

$$\phi_i = \begin{cases} G^t \text{ if } i = t \\ \bigwedge_{o \in O}(o^i \rightarrow \text{prec}(o)^i) \wedge (\mathcal{T}(i, i+1) \rightarrow \phi_{i+1}), i < t \end{cases}$$

Now Formula 2 can be logically equivalently phrased as

$$\exists P \forall E(\iota \wedge (I^0 \rightarrow \phi_0)). \tag{3}$$

The subformula ι has a quadratic size. There is an asymptotically optimal linear size encoding of the interference constraints (Rintanen, Heljanko, & Niemelä 2006). Let ι' be this encoding and let X be the new auxiliary variables occurring in this formula. Our QBF encoding of the planning problem having linear size is now the following.

$$\exists P \exists X \forall E(\iota' \wedge (I^0 \rightarrow \phi_0)) \tag{4}$$

Most of the implemented QBF solvers expect the QBF to be in CNF, and this now poses a problem. A translation into CNF in general may increase the formula size exponentially. There is an inexpensive CNF transformation that uses auxiliary variables. However, this transformation cannot be applied because the auxiliary variables must be existential, not universal, and they cannot occur before E in the prefix, only after E, which would take us back to a $\exists\forall\exists$ QBF which we originally wanted to avoid. In the next section we construct a QBF that avoids this problem.

Encoding with Prefix $\forall\exists$

An encoding with prefix $\forall\exists$ is obtained from the $\exists\forall$ encoding simply by negating the QBF and pushing the negation through the quantifiers, yielding the QBF

$$\forall P \forall X \exists E(\neg\iota' \vee (I^0 \wedge \neg\phi_0)). \tag{5}$$

This QBF says that for all candidate plans, either the definition of plans is violated (interference, non-executability) or there is an execution that does not lead to a goal state.

Since the inner quantifier is $\exists$, there is no problem applying a linear CNF transformation. The main practical difference to the $\exists\forall$ QBF is that the existence of plans corresponds to the $\forall\exists$ QBF being *false*, not *true*. Hence a plan is obtained as a valuation of P that *falsifies* the remaining $\exists$ QBF.

Some experiments showed that it is better to have less variables quantified by the outermost quantifiers. We tested also a different encoding ι_2 of $\neg\iota'$ in which the associated auxiliary variables X' are existentially quantified. The formula ι_2 uses similar implication chains as ι': if an operator that makes a *true* is not executed, then a later such operator (in an arbitrarily chosen ordering) is executed, and for one executed operator that makes a *true* there is a later executed operator in the ordering that either makes a *false* or requires a not to be made *true* because it occurs in the precondition or in the antecedent of a conditional effect. The formula ι_2 is slightly more complicated than $\neg\iota'$ but both have a linear size and require a linear number of new auxiliary variables. An alternative encoding of the planning problem that uses the formula ι_2 is hence

$$\forall P \exists X' \exists E(\iota_2 \vee (I^0 \wedge \neg\phi_0)). \tag{6}$$

Experiments

We test the encodings by using some benchmark problems presented in the literature. Bonet and Geffner (2000) proposed one of the most interesting planning benchmarks that have no observability, sorting networks (Knuth 1998). The other benchmarks we consider – the BTC, blocks world (BW), ring, and square problems – have a very simple structure but pose problems for the current generation of planning algorithms. For a description see (Bonet & Geffner 2000).

The QBF that represent planning problems can be evaluated by Davis-Putnam style QBF algorithms or by other types of algorithms. In this section we consider two general-purpose QBF solvers that can be readily downloaded from the Internet, QuBE-Rel by Giunchiglia et al. (2002) and yQuaffle by Yu and Zhang&Malik (2002). Another approach to QBF evaluation was used by Palacios and Geffner (2005) in connection with the standard $\exists\forall\exists$ encoding, and it would be interesting to test the impact of the simpler $\exists\forall$ prefix on the planning runtimes with their DNNF-based compiler and a SAT solver. There is a close connection of this approach to the algorithm by Biere (2005).

All the experiments were run under Linux in a Dell Latitude D610 laptop with a 2.0 GHz Intel Pentium M 760 processor, 2MB cache and 1GB memory. Generation of all the 1300 QBF used in the experiments took 2 minutes of CPU time, with a maximum of about two seconds per QBF. We used the encoding 5 for all but the sorting network problems for which the encoding 6 turned out to be better.

The runtimes of the QBF solvers QuBE and yQuaffle for the $\forall\exists$ and $\exists\forall\exists$ encodings of the benchmarks are given in Table 1. No clear picture emerges from the runtimes. The QuBE runtimes with the $\forall\exists$ encoding are always worse than with the $\exists\forall\exists$ encoding. The yQuaffle runtimes with $\forall\exists$ are often better but in many cases worse. The strongest solver/encoding combinations on these problems are yQuaffle with $\forall\exists$ and QuBE with $\exists\forall\exists$. The strength of $\exists\forall\exists$/QuBE is in the BTC, BW and ring problems, and the strength of $\forall\exists$/yQuaffle in the sorting network problems. For the square problems the results are mixed: $\forall\exists$/yQuaffle cannot prove tight plan length lower bounds in a reasonable time but finds suboptimal solutions very quickly. However, with parallel plan search algorithms which are not guaranteed to find optimal plans (Rintanen, Heljanko, & Niemelä

instance	len	plan	prefix ∀∃		prefix ∃∀∃	
			QuBE	yQuaffle	QuBE	yQuaffle
btc04-1	6	F	3.32	0.70	0.00	0.01
btc04-1	7	T	3.84	1.92	0.00	0.01
btc05-1	7	F	88.10	13.32	0.01	0.03
btc05-1	8	F	-	158.56	0.02	0.04
btc05-1	9	T	-	-	0.01	0.05
BW2	2	F	0.05	0.02	0.00	0.01
BW2	3	T	0.11	0.05	0.01	0.02
BW3	5	F	-	25.15	2.34	200.33
BW3	6	F	-	-	25.16	-
BW3	7	T	-	-	8.39	-
BW4	4	F	-	25.46	234.76	-
ring02	4	F	7.33	0.15	0.01	0.00
ring02	5	T	98.88	0.09	0.01	0.00
ring03	5	F	257.78	1.52	0.03	0.02
ring03	6	F	-	17.53	0.11	0.08
ring03	7	F	-	-	0.48	0.39
ring03	8	T	-	136.93	1.29	1.83
ring04	4	F	14.93	0.29	0.01	0.03
ring04	5	F	363.01	2.01	0.03	0.05
ring04	6	F	-	22.51	0.14	0.16
ring04	10	F	-	-	39.85	178.37
ring04	11	T	-	-	83.46	279.18
sort5	2	F	38.66	0.02	0.01	0.01
sort5	3	F	-	0.07	0.15	0.18
sort5	4	F	-	0.37	11.17	14.61
sort5	5	T	-	0.25	2.34	16.38
sort6	4	F	-	1.30	115.91	-
sort6	5	T	-	0.66	209.78	-
sort7	3	F	-	0.34	22.54	125.21
sort7	4	F	-	3.08	-	-
sort7	5,6,7	?	-	-	-	-
sort7	8	T	-	6.45	-	-
sort7	9	T	-	7.24	-	-
sort8	3	F	-	0.83	107.82	-
sort8	4	F	-	7.70	-	-
sort8	5,6	?	-	-	-	-
sort8	7	T	-	247.66	-	-
sq3	5	F	4.26	8.90	0.02	0.01
sq3	6	F	66.16	-	0.06	0.05
sq3	7	F	-	-	0.36	0.19
sq3	8	F	-	-	4.06	1.13
sq3	9	F	-	-	37.17	7.01
sq3	10	T	-	-	63.84	166.93
sq3	11	T	-	-	22.88	184.34
sq3	12	T	-	1.28	59.41	115.16
sq3	13	T	-	1.31	87.09	102.17

Table 1: Runtimes with ∀∃ and ∃∀∃ encodings

2006) ∀∃/yQuaffle is for these problems more efficient than ∃∀∃/QuBE.

In summary, it is not clear whether the ∀∃ encoding is an improvement over the ∃∀∃ encoding, but on some problems (sorting networks) the ∀∃ leads to by far the best runtimes. The ∃∀∃ encoding dominates on the other problems which have a far simpler structure than the sorting network problems. The most difficult soluble QBF for most benchmarks series had sizes in the range between 10 KB and 30 KB.

The sizes of both the ∃∀∃ and the ∀∃ QBF grow linearly, but the ∀∃ QBF are about twice as big because of the more complicated structure with deep nesting of disjunctions at the top level. Also the number of auxiliary variables needed for CNF transformation is higher. It would be interesting to see how non-CNF QBF solvers which take arbitrary formulae or circuits as input would fare.

Related Work

Rintanen (1999) generalized the planning as satisfiability approach to nondeterministic planning problems and QBF, and is the basis of this work. A follow-up work by Giunchiglia (2000) does not represent the QBF explicitly, and splits planning to search for candidate plans and verifying their correctness. The latter is reduced to a satisfiability test. This approach can be directly compared to the ∀∃ and ∃∀ QBF presented by us but what is roughly the outermost ∃ quantifier in our ∃∀ QBF is handled by an ad hoc search algorithm and there is no exact match to the QBF. Brafman and Hoffmann (2004) present a variant of Giunchiglia's approach where the search algorithm uses a heuristic.

Palacios and Geffner (2005) do not make the QBF explicit but their implicit QBF are like the ∃∀∃ of Rintanen (1999). Their planning method is actually a general-purpose algorithm for evaluating QBF by reduction to satisfiability testing through an intermediate compilation to DNNF. The reduction proceeds by eliminating the two inner quantifiers ∀∃ by universal and existential quantification, and a plan is found by finding a satisfying assignment to the resulting formula. Palacios and Geffner demonstrate that for some problems their approach has a clear advantage over an earlier DNNF-based approach by Palacios et al. (2005).

The DNNF/SAT-based approach to QBF evaluation seems more efficient for some of the planning problems than Davis-Putnam style AND-OR search. Palacios and Geffner (2005) report finding depth 6 sorting networks for 7 and 8 inputs respectively in 46 and 4256 seconds and being able to prove the optimality for 7 inputs in 355 second and for 8 inputs not in less than 2 hours. The lower and upper bounds we could show with the ∀∃ QBF for nets with 7 and 8 inputs within a time limit of 400 seconds are looser. However, with the ∃∀∃ encoding it would only be possible to show very loose lower bounds for these nets and no upper bounds at all. It would be interesting to test whether similar dramatic improvement could be obtained with the DNNF approach by using ∃∀ QBF instead of ∃∀∃ QBF.

Planning by heuristic search in the space of belief states is very efficient for many types of problems, for example for finding sorting networks that are relatively close to the optimal (Rintanen 2004). However, for finding networks with

optimal size or depth is not feasible further than up to 6 input gates, which is also about where the QBF approaches scale to: Palacios and Geffner prove the optimality of a 16 gate network for the 7 input problem in one hour.

Conclusions

We have presented an encoding of conformant planning as QBF with the prefixes $\exists\forall$ and $\forall\exists$, improving earlier encodings which have the prefix $\exists\forall\exists$. Evaluation of these QBF with AND-OR tree search QBF algorithms gives mixed results: in some cases the new encoding is more efficient, sometimes less. However, for one very challenging problem, sorting networks, the new encoding improves the efficiency to a level which had earlier only achieved with a specialized DNNF based approach.

The work underlines the importance of developing more efficient and robust algorithms for evaluating QBF. The QBF solvers used in the experiments are similar but exhibit wildly differing behaviors. We believe that the future successes of QBF in many applications is strongly dependent on the development of better algorithms for evaluating QBF.

Acknowledgements

This research was supported by NICTA Ltd in the framework of the SuperCom and DPOLP projects. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

References

Biere, A. 2005. Resolve and expand. In *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*, number 3542 in Lecture Notes in Computer Science, 59–70. Springer-Verlag.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, 52–61. AAAI Press.

Brafman, R. I., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, 355–364. AAAI Press.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2002. Learning for quantified Boolean logic satisfiability. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-2002) and the 14th Conference on Innovative Applications of Artificial Intelligence (IAAI-2002)*, 649–654.

Giunchiglia, E. 2000. Planning as satisfiability with expressive action languages: concurrency, constraints and nondeterminism. In Cohn, A. G.; Giunchiglia, F.; and Selman, B., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR 2000)*, 657–666. Morgan Kaufmann Publishers.

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

Knuth, D. E. 1998. *Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Publishing Company.

Palacios, H., and Geffner, H. 2005. Reducción de la planificación conformante a SAT mediante compilación a d-DNNF. In *XI Conferencia de la Asociación Española para la Inteligencia Artificial*. Also presented in an ICAPS'05 workshop as "Mapping conformant planning into SAT through compilation and projection".

Palacios, H.; Bonet, B.; Darwiche, A.; and Geffner, H. 2005. Pruning conformant plans by counting models on compiled d-DNNF representations. In Biundo, S.; Myers, K.; and Rajan, K., eds., *ICAPS 2005. Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling*, 141–150. AAAI Press.

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

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research* 10:323–352.

Rintanen, J. 2003. Expressive equivalence of formalisms for planning with sensing. In Giunchiglia, E.; Muscettola, N.; and Nau, D., eds., *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling*, 185–184. AAAI Press.

Rintanen, J. 2004. Distance estimates for planning in the discrete belief space. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-2004) and the 16th Conference on Innovative Applications of Artificial Intelligence (IAAI-2004)*, 525–530. AAAI Press.

Russell, S., and Wolfe, J. 2005. Efficient belief-state AND-OR search, with application to Kriegspiel. In Kaelbling, L. P., ed., *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 278–285. Morgan Kaufmann Publishers.

Zhang, L., and Malik, S. 2002. Conflict driven learning in a quantified Boolean satisfiability solver. In *Proceedings of the 2002 IEEE/ACM International Conference on Computer Aided Design (ICCAD2002)*, 442–448. ACM Press.