The Achilles' Heel of QBF*

Carlos Ansotegui, Carla P. Gomes, and Bart Selman
Dept. Computer Science
Cornell University
Ithaca, NY 14853

Abstract

In recent years we have seen significant progress in the area of Boolean satisfiability (SAT) solving and its applications. As a new challenge, the community is now moving to investigate whether similar advances can be made in the use of Quantified Boolean Formulas (QBF). QBF provides a natural framework for capturing problem solving and planning in multi-agent settings. However, contrarily to single-agent planning, which can be effectively formulated as SAT, we show that a QBF approach to planning in a multi-agent setting leads to significant unexpected computational difficulties. We identify as a key difficulty of the QBF approach the fact that QBF solvers often end up exploring a much larger search space than the natural search space of the original problem. This is in contrast to the experience with SAT approaches. We also show how one can alleviate these problems by introducing two special QBF formulations and a new QBF solution strategy. We present experiments that show the effectiveness of our approach in terms of a significant improvement in performance compared to earlier work in this area. Our work also provides a general methodology for formulating adversarial scenarios in QBF.

Introduction

There has been tremendous progress in our ability to solve large Boolean satisfiability (SAT) problems. State-of-the-art SAT solvers can solve instances with hundreds of thousands of variables and over one million clauses (Berre & Simon 2004). These solvers are now used in a range of applications, such as hardware and software verification and AI planning. In recent years, a new frontier in automated reasoning has emerged. This new challenge is focused on Quantified Boolean Formulas (QBF), their use in encodings and the development of QBF solvers. The potential of QBF is quite significant. For example, in such encodings one can capture multi-agent planning, general (unbounded) model-checking problems for verification, and planning with no apriori restriction on plan length. The price one pays for this higher level of expressiveness is that solving QBF is a PSPACE-complete problem — a problem believed to be considerably harder than NP-complete problems. Recent SAT competition results indeed show that building practical QBF solvers is more challenging than one might expect given the tremendous strides made in the design of SAT solvers. In particular, relatively small QBF problems are often beyond the reach of QBF solvers (Berre *et al.* 2004).

The goal of the work in this paper is to obtain better insights into what exactly causes many QBF formulas to be surprisingly difficult to solve. The central issue we identify is that *QBF solvers are often forced to explore much larger combinatorial spaces than the natural search space of the original problem*. We will discuss the source of this phenomenon in detail below, but at a high-level the issue can be explained as follows. In QBF, we generally have an alternation of existentially and universally quantified Boolean variables followed by a CNF expression (called the "matrix" of the formula). A formula is satisfiable if and only if there is a series of assignments to the existentially quantified variables such that for all possible assignments of the universally quantified variables the matrix is satisfied (each clause has at least one satisfied literal). QBF is closely connected to adversarial scenarios where player A tries to set the existential variables so as to satisfy the matrix and the other player B (corresponding to the universal variables) tries to invalidate the matrix. A QBF is satisfiable if player A "wins" — i.e., finds a strategy for setting the existentially quantified variables such that no matter what setting player B chooses the matrix can be satisfied. If no such strategy exists for A, B wins and the QBF is unsatisfiable.

This interpretation of QBF suggests a natural correspondence between QBFs and adversarial planning and game playing (Papadimitriou 1995). However, this correspondence hides an important difficulty: When encoding an actual adversarial planning problem or game, one has to add constraints (clauses) to the matrix that capture the legal actions for each player. More precisely, a violated clause will represent a violation of one or more of the basic rules of the game. The existential player, who is trying to satisfy the matrix, will "work hard" to satisfy those constraints. As a consequence, a QBF solver will pursue legal actions for the existential player A. (More technically, as soon as A makes an illegal move, there will be a violated clause in the matrix and the solver backtracks.) Unfortunately, the situation for the player B, the universal player, is quite different. This player is trying to violate clauses, so as to "break" (partial) assignments that may satisfy the constraints. Player B (via

*Support from AFOSR, Darpa, and NSF is acknowledged.

		Madhusudan, Nam, & Alor 2003			Model A		Model B	
N	steps	QuBEJ	Semprop	Quaffle	Best QBF solver	CondQuaffle	Best QBF solver	CondQuaffle
4	**7**	**2030**	$>$**2030**	$>$ **2030**	**7497**	**3**	**0.03**	**0.03**
4	9	–	–	–	–	28	0.06	**0.04**
8	**5**	**32429**	$>$ **32429**	$>$ **32429**	–	**1**	**0.37**	**0.37**
8	7	–	–	–	–	800	5	**5**
8	13	–	–	–	–	–	–	**2838**

Table 1: Results for Evader/Pursuer on an N×N board for Madhusudan *et al.* (2003); and our models A and B with the best performing QBF solver (non cond.) and our new conditional solver (CondQuaffle). Time in seconds. "–" for $>$ 10 hours.

the QBF solver) is therefore almost naturally drawn to make moves that violate the rules of the game or environment (so-called "illegal" actions or moves). In order to prevent B from doing so, one therefore has to formulate the matrix in such a way that all clauses become satisfied as soon as B attempts an illegal action. We will discuss how this can be done in a QBF encoding. However, unfortunately, current state-of-the-art solvers have often great difficulty recognizing the fact that all clauses in the matrix become satisfied as soon as B makes an illegal move. In particular, "top-down" search QBF solvers, instantiating the quantified variables from left to right, are often forced to explore many more existential and universal variable instantiations before they recognize that the matrix is actually "automatically" satisfied because of an illegal action by B. Most current QBF solvers perform a top-down search. Other QBF solution strategies appear to encounter related problems; we are exploring this further.

In order to alleviate these problems, we introduce two special QBF formulation schemes for encoding adversarial scenarios. We evaluate these encodings in detailed experiments using eight state-of-the-art QBF solvers: QMRES, Quantor, Skizzo, Semprop, QuBE (two versions), and Quaffle (two versions). We also introduce a new QBF solution strategy, called a conditional QBF solver. We implement this strategy by extending Quaffle. Our experiments show a significant improvement over previous QBF approaches to adversarial planning in our benchmark domain.

Our work was inspired by the original call by Toby Walsh to push research on QBF solvers by experimenting with QBF encodings for actual games (Walsh 2003). We were also inspired by work on this challenge by Ian Gent and colleagues (Gent & Rowley 2003). Finally, as a starting point and a baseline to benchmark our results, we consider the work by (Madhusudan *et al.* 2003), who studied these issues to evaluate the potential for QBF solvers for model checking, used in hardware and software verification. They introduce a basic two-player benchmark, the Evader/Pursuer game, as a starting point. Given the apparent "simplicity" of this setting, the reader may wonder what relevance this work may have for QBF solving for general model checking and multi-agent adversarial planning: The key issue, as noted in (Madhusudan *et al.* 2003), is that one will first have to overcome the difficulties of QBF solvers on such basic scenarios, before we can expect real progress in richer settings. Moreover, given the generality of the cause underlying the difficulty for QBF solvers we identify, it seems likely that these insights will also point the way to tackle QBF solving

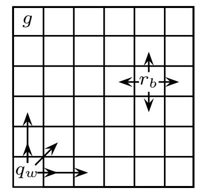

Figure 1: Evader/Pursuer scenario. q_w has to reach g without being captured by r_b.

in larger scale applications.

Preview of Results As the reader will see below, the detailed steps that lead to a full QBF encoding of adversarial games are quite involved. We therefore first briefly preview our final experimental results. We consider the so-called Evader/Pursuer game. Figure 1 shows the basic setting. Two players make alternating moves on a N×N board. The goal is for the starting player q_w to reach the goal square g without being captured by the other player r_b. The horizontal and vertical moves of q_w are restricted to one or two squares at each turn. The diagonal moves are restricted to a single square. r_b can move one square either horizontally or vertically at a time. Note that the evader cannot capture the pursuer. Variations of this game have been used in many different settings to study the complexity of basic multi-agent interactions. Mahusudan et al. (2003) were the first to consider its QBF formulation with several QBF solvers. Their results provide a baseline for our work.

Table 1 gives a summary of our results. The table also contains the results of Madhusudan *et al.* (2003) for several QBF solvers on their problem encodings. (Note that for Semprop and Quaffle they only report that these solvers perform worse than QuBEJ.) So, for example, on a 4×4 board of the Evader/Pursuer game allowing 7 steps total (4 moves by q_w and 3 moves by r_b), the table shows that QuBEJ on the Madhusudan et al. encodings takes 2,030 seconds. When going to 9 steps, the instances cannot be solved in under 10 hours. On an 8×8 board, Madhusudan *et al.* can solve only up to 5 step games.

Table 1 shows how our approach alleviates much of these difficulties, thereby significantly improving the reach of the QBF approach. We consider two encoding strategies, Model A and Model B. Each are designed to avoid much of the exploration of "illegal" moves. We see that especially our conditional QBF solver ("CondQuaffle") performs quite well.

More specifically, for our best encodings (Model B), the 4×4 board with 7 step problem is quite easy indeed, and is solved in only 0.03 seconds. Moreover, even the 8×8 case with 13 steps becomes solvable.

QBF for Adversarial Games

The task of designing a QBF formulation for adversarial settings can be surprisingly complex and prone to mistakes. In order to manage the complexity of our encodings, we proceed in three phases. First, we provide an encoding of the game that ignores the adversarial aspect. In effect, we view the game as a *planning problem*, where both players cooperate. The encoding encapsulates all constraints of the game, in terms of what the legal moves are. In a second phase, we take the plan formulation and construct a general Quantified Boolean Formula (QBF) that captures the adversarial form of the game up to k steps, for a predefined value of k. In the third and final phase of our design, we convert the general QBF formula into QBF in conjunctive normal form (CNF) to obtain the standard form readable by most QBF solvers. As we will see, depending on how we implement the third and last step we obtain encodings with dramatically different computational properties.

Phase I: Non-adversarial plan formulation

Let us denote the players by $P = \{q_w, r_b\}$, let $p \in P$. (By analogy to chess, which is the larger game setting we consider later, we denote q_w as the white queen and r_b as the black rook.) Let $C = \{c_1, \ldots, c_{N\times N}\}$ stand for the cells of the board, let $k \in \mathbb{N}$ be the bound on time steps, let N be the order of the board, and $0 \leq s, s' \leq k$ and $0 \leq i, j \leq n-1$.

Variables We introduce variables representing locations and moves at each time step: (1) Location variables: $l(p, c_i, s)$, p is located at c_i at time step s. (2) Move variables: $m(p, c_i, c_j, s)$, p moves from c_i to c_j at time step s ($c_i \neq c_j$). If s is an even time step (White step) then $p = q_w$, otherwise $p = r_b$. Note we precompute all the possible moves.

Let L^s and M^s be the set of location and move variables at time step s, respectively. We have k sets of type M^s, while we have $k+1$ sets of type L^s (the $(k+1)^{th}$ represents the locations after the last move).

Axioms
We illustrate our axiomatization by giving examples of each axiom group. For the full axiomatization, we refer to the full version of this paper.

Initial Conditions Axioms The encoding below is demonstrated for the Black player, the other case being similar mutatis mutandis. Let r_b be at c_i at time step 0. We encode the initial location for the Black player, denoted by I_b, as follows:

$$I_b \equiv l(r_b, c_i, 0)) \ \wedge \ (\bigwedge_{j\neq i} \neg l(r_b, c_j, 0)) \qquad (1)$$

Action Axioms Informally, the player can take two main actions "to move" or "to wait." For each move, $m(p, c_i, c_j, s)$, we generate a set of axioms encoding preconditions and move effects:

Action Axioms I (Preconditions)
We have different preconditions depending on the type of move. For example, the preconditions for horizontal and vertical moves of two squares (only applicable to the White queen) are:

$$m(q_w, c_i, c_j, s) \rightarrow (l(q_w, c_i, s) \wedge \neg l(r_b, c', s)) \qquad (2)$$

Where $c' \in C$ is the cell between c_i and c_j. At time step s, if q_w moves from c_i to c_j, then q_w must be at c_i and r_b cannot be on its way.

Action Axioms II (Move effects)

$$m(p, c_i, c_j, s) \rightarrow (l(p, c_j, s+1) \wedge \neg l(p, c_i, s+1)) \qquad (3)$$

If p moves from c_i to c_j at time step s, then p is located at c_j and is not located at c_i at time step $s+1$.

Frame Axioms For each location variable, $l(p, c_i, s)$, we generate a set of standard frame axioms. For example, for each player, we will have axioms that state that if a player is at a certain location at time t and does not move it remains at that location at time $t+1$; also, if a player is not at a certain location at time t and doesn't move to that location at time $t+1$, the player will not be at that location at time $t+1$.

Goal Axioms The goal for White, denoted by G^s_w, is to place the white queen at the goal position, while the goal for Black is to prevent this either by capturing the queen or by blocking its path to the goal position. We denote the goal of capturing the white queen as $G^{s'}_b$. So, we have

$$G^s_w \equiv l(q_w, c_g, s), \ \ G^{s'}_b \equiv \bigvee_{i\neq g} (l(q_w, c_i, s') \wedge l(r_b, c_i, s')),$$

where s is an odd time step and s' is an even time step.

We can now state the overall goal, G, to be achieved by the White queen as:

$$G \equiv \bigvee_{s=0}^{k} G^s_w \wedge \bigwedge_{s'<s} \neg G^{s'}_b \qquad (4)$$

I.e., White wins the game if it reaches the goal square and is not captured by Black on the way. (Note that this also covers the case where White is blocked from the goal by Black.)

Mutual Exclusion We need to express that a player cannot take more than one action at each time step. Let $m^s_1, \ldots, m^s_{|M^s|}$ be the move variables encoding potential moves at time step s (M^s are the move variables at time s). To ensure mutual exclusion at time step s is to add $\binom{|M^s|}{2}$ clauses, known as *at-most-one* (AMO) clauses:

$$\bigwedge_{(i,j), i\neq j} (\neg m^s_i \vee \neg m^s_j) \qquad (5)$$

The action "wait" is applied by setting to false every move variable at time step s.

Transitions Let Me^s, Pr^s, Mf^s and Fr^s be the clauses encoding mutual exclusion, preconditions, move effects and frame axioms, respectively, at time step s. We define the concept of transition at time step s, denoted by Tr^s, as the union of the previous sets:

$$\mathrm{Tr}^s \equiv \mathrm{Pr}^s \wedge \mathrm{Me}^s \wedge \mathrm{Mf}^s \wedge \mathrm{Fr}^s \tag{6}$$

We also define the concepts of White's and Black's transitions, denoted by Tr_w and Tr_b, as the set of clauses encoding the White's and Black's transitions, respectively.

$$\begin{aligned} \mathrm{Tr}_w &\equiv \mathrm{Tr}^0 \wedge \mathrm{Tr}^2 \wedge \ldots \wedge \mathrm{Tr}^{k-1} \\ \mathrm{Tr}_b &\equiv \mathrm{Tr}^1 \wedge \mathrm{Tr}^3 \wedge \ldots \wedge \mathrm{Tr}^{k-2} \end{aligned} \tag{7}$$

Phase II: The game as a QBF

Quantified Boolean Logic (QBL) extends Boolean logic by allowing quantification over Boolean variables. If ϕ is a propositional formula (the "matrix") over a set of Boolean variables B and σ is a sequence of $\exists b$ and $\forall b$, one for every $b \in B$, then $\sigma\phi$ is a Quantified Boolean Formula (QBF). In our case, we need to produce a QBF such that for an odd number of time steps (k) and a set of initial conditions (I_w and I_b), there exists a series of White's actions (corresponding to White's transitions (Tr_w)) such that for all legal counter moves by Black (corresponding to Black's transitions (Tr_b)), the goal state G is satisfied.

Formula (8) is a QBF describing our game.

$$\exists L^0 M^0 L^1 \forall M^1 \exists L^2 M^2 L^3 \ldots \forall M^{k-2} \exists L^{k-1} M^{k-1} L^k \\ I_w \wedge I_b \wedge \mathrm{Tr}_w \wedge (\mathrm{Tr}_b \rightarrow G) \tag{8}$$

This formula is constructed such that it is satisfiable *if and only if* there is a series of winning sequences of moves for White, no matter what counter moves Black makes (more details below). A QBF is satisfiable if there exists a series of assignments to the existential variables such that for all possible assignments of the universal variables the matrix part of the QBF is satisfied. The setting of the existential variables can depend on the instantiation of the universal variables that precede it. In a sense, we're playing a "game" on the matrix part of the formula, where one player tries to set the existential variables so as to satisfy the matrix, and its opponent instantiates the universal variables, searching for ways to "unsatisfy" the matrix. The order of the quantifiers is clearly important.

To understand (8), first consider the quantifiers. The moves for Black are universally quantified.[1] The location variables (describing the state of the board) and the variables modeling White's moves are quantified existentially. Now, in order for the matrix to be satisfied, the initial conditions and the White's transitions ($I_w \wedge I_b \wedge \mathrm{Tr}_w$) have to be satisfied. On the other hand, $(\mathrm{Tr}_b \rightarrow G)$ says that we only need to guarantee that the goal (G) has to be satisfied as long as the Black player plays according to the rules of the game, i.e., to satisfy Tr_b. Informally, a player can perform an *illegal* action (from the perspective of the game), by breaking the mutual exclusion, i.e., by trying to perform more than one action at the same time step (see 5), or by breaking the precondition axioms, for example, by moving a piece from a location c_i without being on c_i. Contradictions should only arise due to *legal* actions that do not satisfy the goal or due to *illegal* actions of the White player: On the one hand, if the White player performs an *illegal* action, Tr_w becomes unsatisfiable and therefore the matrix is unsatisfiable ("if White cheats Black wins"). On the other hand, if the Black player performs an *illegal* action, Tr_b becomes unsatisfiable, then $(\mathrm{Tr}_b \rightarrow G)$ is satisfiable and we know that $I_w \wedge I_b \wedge \mathrm{Tr}_w$ is also satisfiable and therefore the matrix is satisfiable ("if Black cheats White wins").

[1] Note that we have a slight abuse of notation: $\forall M^1$ is a short for $\forall m_1^1 \forall m_2^1 ... \forall m_{|M^1|}^1$, where $M^1 = \{m_1^1, m_2^1, ..., m_{|M^1|}^1\}$ is the set of all potential moves for Black at time step 1; $\exists L^s M^s L^{s+1}$ should be read as $\exists L^s \exists M^s \exists L^{s+1}$; and each existential quantifier is again really a sequence of quantifiers, one for each element in the sets L^s, M^s, and L^{s+1}.

Phase III: The game as a QBF in CNF form

Generally, QBF solvers require the matrix of the QBF to be in CNF form. The most straightforward way to translate the matrix in QBF (8) into CNF form is by applying the implication and distributivity rules. However, the resulting conjunctive normal form can be exponentially larger compared to the size of the original matrix, mostly due to the translation of the term $(\mathrm{Tr}_b \rightarrow G)$. The standard approach to avoiding such a blow up is to introduce new variables.

For example, let's consider a simplified version of $(\mathrm{Tr}_b \rightarrow G)$, where G is logically equivalent to the Boolean variable g, the number of time steps is five ($k = 5$) and the Black transitions (Tr_b) only include the $h_1 = \binom{|M^1|}{2}$ and $h_3 = \binom{|M^3|}{2}$ mutual exclusion clauses at time step 1 and 3, respectively (see (5)):

$$\begin{aligned} &((\neg m_1^1 \vee \neg m_2^1) \wedge \ldots \wedge (\neg m_{|M^1|-1}^1 \vee \neg m_{|M^1|}^1) \wedge \\ &(\neg m_1^3 \vee \neg m_2^3) \wedge \ldots \wedge (\neg m_{|M^3|-1}^3 \vee \neg m_{|M^3|}^3)) \rightarrow g \end{aligned} \tag{9}$$

If we apply the implication and distributivity rules to (9) we obtain $2^{h_1+h_3}$ clauses. The idea is to map the mutual exclusion clauses to a set of auxiliary Boolean variables $me_1^1, \ldots, me_h^1, me_1^3, \ldots, me_h^3$ and add the mappings as equivalences (Tseitin 1967):

$$\begin{aligned} &(me_1^1 \wedge \ldots \wedge me_{h_1}^1 \wedge me_1^3 \wedge \ldots \wedge me_{h_3}^3) \rightarrow g \\ &(me_1^1 \leftrightarrow (\neg m_1^1 \vee \neg m_2^1)) \wedge \ldots \wedge (me_{h_1}^1 \leftrightarrow (\neg m_{|M^1|-1}^1 \vee \neg m_{|M^1|}^1)) \\ &(me_1^3 \leftrightarrow (\neg m_1^3 \vee \neg m_2^3)) \wedge \ldots \wedge (me_{h_2}^3 \leftrightarrow (\neg m_{|M^3|-1}^3 \vee \neg m_{|M^3|}^3)) \end{aligned} \tag{10}$$

When we translate (10) into CNF form we obtain $3(h_1 + h_3) + 1$ clauses. We refer to the new introduced variables in (10) as *indicator* variables, in the sense that they "indicate" the validity of the logic expression they represent. So, me_1^1 "watches" the exclusion between m_1^1 and m_2^1, *i.e.*,, it is is True iff m_1^1 and m_2^1 are *not* simultaneously True.

In contrast to SAT where adding new variables does not necessarily increase the potential search space, since Boolean propagation takes care of the dependencies (Thiffault *et al.* 2004), the new variables lead to a significant performance penalty for QBF solvers. Consider a partial interpretation that covers the universal variables representing the moves at time step 1 and makes m_1^1 and m_2^1 evaluate to true, i.e., there has been at least one illegal action at time step

1. Then, in formula (10), top-down search QBF solvers, like Quaffle and QuBE, are only able to satisfy the clauses in the first and second line, and are forced to continue the search until they assign all universal variables at the third level. This leads the solvers to explore a large search space containing many "illegal" moves. Similar unnecessary search occurs at every level of universal quantifiers. In a more detailed analysis, provided in the full version of this paper, we show that *all* state-of-the-art solvers encounter similar difficulties when this particular structure is gradually extended to obtain the complete formula encoding the game.

Below, we present a formulation for which the top-down search QBF solver can avoid much of the unnecessary search. The idea is to introduce "grouped" indicator variables at each time step that flag whether an illegal move has occurred. Consider formula (10). We introduce a new indicator variable me^1 logically equivalent to the conjunction of all me_i^1 variables. This new variable watches for any possible exclusion violation at step 1. (True iff no such violation at step 1.) We can then add the negation of me_1 to all clauses at later time steps. When an exclusion violation occurs at step 1, the clauses at later levels are immediately satisfied. (Below we use tr^s variables, which watch for all possible constraint violations.) For a similar approach in a static CSP domain, see (Gent *et al.* 2004). We extend this idea to dynamic scenarios in which it is critical to factor in the dependence of constraint violations as a function of earlier moves. To achieve this we introduce a hierarchy of such grouped indicator variables.

Model A: Grouped Indicator Variables

Formula (11) is a QBF in CNF form describing our game.

$$\begin{array}{l}
\exists L^0 M^0 L^1 \forall M^1 \exists I_d^1 L^2 M^2 L^3 \ldots \forall M^{k-2} \exists I_d^{k-2} L^{k-1} M^{k-1} L^k \\
^{[a]}\ I_w \wedge I_b \\
\ldots \\
\left\lfloor \begin{array}{l}
^{[b]}\ \neg G_b^{s-1} \wedge \mathrm{Me}^{s-1} \wedge \mathrm{Pr}^{s-1} \wedge \mathrm{Mf}^{s-1} \wedge \mathrm{Fr}^{s-1} \\
^{[c]}\ G_w^s \wedge \mathrm{Me}^s \wedge \mathrm{Pr}^s \\
^{[d]}\ \neg \mathrm{ia}^s \leftrightarrow (\neg g_w^s \wedge \mathrm{me}^s \wedge \mathrm{pr}^s)
\end{array} \right\rfloor \vee \neg \mathrm{tr}^{s-2} \\
^{[e]}\ \mathrm{tr}^s \leftrightarrow (\neg \mathrm{ia}^s \wedge \mathrm{tr}^{s-2}) \\
\left\lceil \begin{array}{l}
^{[f]}\ \mathrm{Mf}^s \wedge \mathrm{Fr}^s \\
^{[g]}\ \neg G_b^{s+1} \wedge \mathrm{Me}^{s+1} \wedge \mathrm{Pr}^{s+1} \wedge \mathrm{Mf}^{s+1} \wedge \mathrm{Fr}^{s+1}
\end{array} \right\rceil \vee \neg \mathrm{tr}^s \\
\ldots \\
^{[h]}\ G_w^k \vee \neg \mathrm{tr}^{k-2} \\
\qquad I_d^s \equiv me_1^s, \ldots, me_h^s, \mathrm{me}^s, pr_1^s, \ldots, pr_t^s, \mathrm{pr}^s, \mathrm{ia}^s, g_w^s, \mathrm{tr}^s \\
\qquad h = \binom{|M^s|}{2},\ t = |M^s| \text{ , s is an odd time step}
\end{array} \tag{11}$$

We include this QBF for completeness. However, because of limited space, the description below is rather compact. See the full version of the paper, for a detailed description.

QBF (11) extends the prefix in QBF (8) by adding after each block of universal quantifiers, an existential quantification on the indicator variables, denoted by I_d^s, that indicate if there has been an action at the universal time step s. Particularly, $me_1^s, \ldots, me_h^s$ and $pr_1^s, \ldots, pr_t^s$ indicate if there is an *illegal* action at the level of the mutual exclusion and preconditions, respectively. We have: (1) $[a], [b], [f]$ and $[g]$ include the initial conditions, preconditions, move effects, frame and goal axioms and mutual exclusion, as described above, translated into CNF form. (2) $[c]$ includes the goal for the White player, the mutual exclusion and the preconditions for the Black player, translated into CNF form. Due to the translation, we obtain the indicator variables I_d^s. We consider the satisfaction of White's subgoal, g_w^s evaluates to true, as a special case of an illegal action of the Black player, i.e., if the White player has already won, any future action of the Black player should be considered illegal. (3) At equivalences in $[d]$ and $[e]$, ia^s indicates if there has been an illegal action exactly at time step s, and tr^s indicates if the transitions up to time step s do not involve an illegal action. (4) Finally, $[h]$ states that if there has not been any illegal transition, the win of White depends only on its last action at time step $(k-1)$, that has to allow White to be at the goal position at time step k.

Model B: Full Assignment to Universal Variables

Our model A formulation in QBF (11) guarantees that when there is an interpretation for the universal variables at a certain time step and contains one or more illegal actions, the QBF solvers implementing a top-down search can easily derive the empty formula (satisfied matrix) and backtrack immediately. The number of universal variables at a given time step s is $|M^s|$ and the number of possible interpretations is $2^{|M^s|}$. Ideally, QBF solvers should be able to derive the empty formula *as soon as a partial interpretation* to the universal variables already contains an illegal action (instead of only after there is a full interpretation). However, due to a similar phenomena as discussed earlier, current QBF solvers are unable to prune effectively and are forced to explore potentially all the $2^{|M^s|}$ interpretations. We can avoid this problem by encoding the mutual exclusion, over the move variables at time step s and the action of waiting, using a logarithmic mapping that employs $\lceil \log_2(|M^s|+1) \rceil$ auxiliary Boolean variables. In effect, these variables provide a much more compact encoding of the possible moves at level s.

Conditional QBF Solver and non-CNF QBF

Both models A and B are equipped with indicator variables to flag the occurrence of illegal actions. Because of the specific structure of our formula, a top-down QBF solver[2] can use these variables to determine satisfiability of the matrix as soon as one of the indicator variables is set to False as we discussed earlier. In practice, clause learning and other QBF techniques may make this detection less efficient. Moreover, in model A we still have some unnecessary branching as discussed in the section on Model B. We therefore introduce a so-called conditional top-down QBF solver. Assuming that no local inconsistency is detected, the solver will backtrack immediately when an indicator variable is set to False, signalling that the matrix is satisfiable. We have extended Quaffle to implement this strategy. We call our solver CondQuaffle. CondQuaffle takes as input a QBF instance and a list of the indicator variables, and it prevents the underlying QBF

[2] A top-down QBF solver instantiates the quantified variables from left to right. Almost all current QBF solvers are top-down (Berre *et al.* 2004).

N	mod.	k (steps)	QMRES	Quantor	Skizzo	Semprop	QuBEJ	QuBER	Quaffle(cs)	Quaffle(c)	CondQuaffle(c)
4	**A**	**7**	**–**	**–**	**7467**	**–**	**–**	**–**	**–**	**–**	**3**
4	**A**	9	–	–	–	–	–	–	–	–	28
4	**A**	15	–	–	–	–	–	–	–	–	24713
4	**B**	**7**	**–**	**–**	**408**	**26**	**2**	**368**	**0.14**	**0.03**	**0.03**
4	**B**	9	–	–	219	207	7	18134	0.18	0.06	0.04
4	**B**	13	–	–	–	11191	288	–	0.23	0.08	0.07
4	**B**	15	–	–	2865	–	2135	–	0.24	0.08	0.07
8	**B**	5	**–**	–	**114**	**1.52**	**0.12**	**583**	**0.69**	**0.37**	**0.37**
8	**B**	7	–	–	410	79	69	27604	139	5	5
8	**B**	9	–	–	1894	1251	1278	–	922	33	32
8	**B**	11	–	–	5240	15708	16824	–	21564	–	337
8	**B**	13	–	–	–	–	–	–	–	–	2838
8	**B**	15	–	–	–	–	–	–	–	–	33369

Table 2: Models A and B, 4x4 and 8x8 boards. CondQuaffle(c) is our new QBF strategy applied to Quaffle(c). Time in seconds. "–" for $>$ 10 hrs.

inst.	m	k	QMRES	Quantor	Skizzo	Semprop	QuBEJ	QuBER	Quaffle(cs)	Quaffle(c)	CondQuaffle(c)
1	**B**	5	**–**	**–**	–	3817	1324	–	449	90	91
2	B	7	–	–	–	–	2722	–	4590	–	535
3	B	5	–	–	–	–	–	–	–	–	1705
4	B	7	–	–	–	–	–	–	–	–	–

Table 3: Model B, 8x8 boards. Chess endgame instances. CondQuaffle(c) is our new QBF strategy applied to Quaffle(c). Time in seconds. "–" for $>$ 10 hrs.

solver (Quaffle) from continuing search if an indicator variable is set to False.

Our conditional solver approach can also be viewed as effectively implementing a non-CNF QBF solution strategy. In particular, a promising general strategy for designing a non-CNF QBF solver would be to read in as input a formula of, e.g., form (8); translate this into CNF, while introducing the appropriate indicator variables; and solve the resulting CNF formula using an extended CNF QBF solver that incorporates the special semantics of the indicator variables as described above for CondQuaffle. Such a non-CNF solver could avoid the illegal search space but but can still take advantage of the many special techniques developed for CNF style QBF solvers (which in turn use techniques from CNF SAT solvers).

Experimental Results

To evaluate our encodings, we consider a series of instances of the Evader/Pursuer game using the same parameter settings as Madhusudan *et al.* (2003). We considered both Model A and Model B encodings and our new QBF strategy, the conditional solver.[3] The QBF solvers we use for the experimental investigation are: QMRES (Pan & Vardi 2004), Quantor (Biere 2004), sKizzo (Benedetti 2004), Semprop (Lets 2002), two variants of QuBE (Giunchiglia *et al.* 2001): QuBEJ (with backjumping) and QuBER (QuBEJ + learning), and three variants of Quaffle (Zhang & Malik 2002): Quaffle(c) (with conflict analysis) Quaffle(cs) (with sat analysis) and CondQuaffle (Quaffle(c) + indicator-pruning, our new solver). Our experiments ran on a 0.5GHz Pentium III with 0.5 GB memory; Madhusudan *et al.* (2003), see Table 1, ran on a 1GHz Pentium III with 1.5 GB memory. Whenever a solver crashed or gave a wrong answer we reported as a result the timeout of the corresponding experiment.

Table 1 provides a summary of our results. See the "Preview of Results" (in Introduction) for a discussion of the table. We significantly outperform the results in (Madhusudan *et al.* 2003). Model B is the best performing approach, in which almost all unnecessary search in the space of illegal moves has been successfully eliminated. This results in a much better scaling to larger boards and more time steps. Note that even our Model A, using our conditional QBF solver (CondQuaffle), solves all but one of 8×8 instances in Table 1.[4]

Table 2 gives more detailed performance results on our set of instances for a total of eight state-of-the-art QBF solvers and our conditional solver. We see how Model B allows us to solve a series of non-trivial 8×8 Evader/Pursuer instances with up to 15 moves total. Our CondQuaffle solver is the most effective. But even Quaffle itself and several of the other QBF solvers also perform quite well on Model B. This again suggests that we have succeeded in eliminating much of the illegal move search space. On the largest instances,

[3] Code and data available from the authors.

[4] Our Model A formulation without the conditional solver does not perform competitively. Note that model A is much larger than model B. Furthermore, the exploration of illegal moves within a single time level, as discussed in the description of Model B, is still significant. The improvements obtained with CondQuaffle on Model A confirm this.

it appears that clause learning and other mechanisms may actually hamper the best QBF solvers by "obscuring" the role of indicator variables. Resolving this issue is a clear challenge for future solver development. Note that our conditional solver is not hampered by this phenomenon because the indicators are used directly for backtracking.

Our ultimate challenge is to use a QBF approach to solving hard endgame problems for Chess and other forms of adversarial planning. Table 3 provides preliminary results for our Model B encodings for non-trivial chess endgame instances with five pieces on a 8×8 board. Again, our conditional solver performs best. These instances were completely out of reach for our earlier encodings (model A and earlier variants). More importantly, the results show that our QBF approach is quite promising, given that these instances are much more complex than the Evader/Pursuer problems.

Conclusions

We have considered QBF encodings of adversarial scenarios. Standard QBF formulations lead to significant computational inefficiencies. We identified an important source of these difficulties in that QBF solvers tend to explore search spaces much larger than the natural search space of the original problem. This is quite unlike the experience with SAT. We introduced two new formulations (Models A and B) to alleviate much of the unnecessary search. Our encoding strategy is based on a principled three phase methodology for capturing adversarial scenarios as QBF. We also introduced a conditional QBF solution strategy which can be easily integrated with existing solvers and directly boosts their performance, by avoiding the illegal search space. We discussed how such a conditional solver can be viewed as essentially implementing a non-CNF QBF solution strategy. Finally, we presented detailed experimental results on the Evader/Pursuer game showing a significant performance improvement over earlier work, thereby increasing the reach of the QBF approach. We also presented promising results of our approach on a richer domain, Chess endgame instances. We believe our findings concerning the unnecessary exploration of the illegal search space and our conditional solution strategy will provide a framework for further improvements in QBF approaches.

References

Benedetti, M. 2004. Evaluating QBFs via symbolic Skolemization. In *LPAR04*.

Berre, D. L., and Simon, L. 2004. Fifty-five solvers in Vancouver: The SAT-04 competition. In *Proc. SAT'04*, LNCS.

Berre, D. L.; Narizzano, M.; Simon, L.; and Tacchella, A. 2004. The 2nd QBF solvers evaluation. *Proc. SAT'04*, LNCS.

Biere, A. 2004. Resolve and expand. *Proc. SAT'04*.

Buning, K.; Karpinski, M.; and Flogel, A. 1995. Resolution for QBF. *Inf. Comput.* 117(1):12–18.

Cadoli, M.; Schaerf, M.; Giovanardi, A.; and Giovanardi, M. 2002. An algorithm to evaluate QBF and its experimental evaluation. *J. of Automated Reasoning* 28(2):101–142.

Gent, I., and Rowley, A. 2003. Encoding connect-4 using QBF. In *Modelling and Reformulating CSP*, 78–93.

Gent, I.; Giunchiglia, E.; Narizzano, M.; Rowley, A.; and Tacchella, A. 2003. Watched data structures for QBF. *SAT-03*.

Gent, I. P.; Nightingale, P.; and Rowley, A. 2004. Encoding quantified CSPs as QBF. In *ECAI'04*, 176–180.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2001. QuBE: A system for deciding QBF. In *Proc. of IJCAR'01)*.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2004. QBF reasoning on real-world instances. In *SAT'04,* LNCS.

Kautz, H. A., and Selman, B. 1992. Planning as satisfiability. In *ECAI'92*, 359–363.

Lets, R. 2002. Lemma and model caching in decision procedures for QBF. In *Proc. TABLEAUX'02*, 2002.

Madhusudan, P.; Nam, W.; and Alur, R. 2003. Symb. computational techniques for solving games. *Elec. Notes Theor. Comp. Sci.* 89(4).

Otwell, C.; Remshagen, A.; and Truemper, K. 2004. An effective QBF solver for planning problems. In *MSV/AMCS*, 311–316.

Pan, G., and Vardi, M. Y. 2004. Symbolic decision procedures for QBF. In *SAT04*.

Papadimitriou, C. 1995. *Computational Complexity*. Addison.

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *JAIR'99* 10:323–352.

Thiffault, C.; Bacchus, F.; and Walsh, T. 2004. Solving non-clausal formulas with DPLL search. In *CP04*.

Tseitin, G. 1967-1970. On the complexity proofs in propositional logics. In *Autom. of Reas.: Classical Papers in Comp. Logic*.

Walsh, T. 2003. Challenges for SAT and QBF. Keynote SAT-03.

Zhang, L., and Malik, S. 2002. Conflict driven learning in a QBF. In *Proc. of ICCAD'02*.

Zhang, L., and Malik, S. 2002. Towards a symmetric treatment of satisfaction and conflicts in QBF. In *CP'02*, 200–215.

Zhao, and Buning, K. 2004. Equivalence models for QBF. In *SAT'04*, LNCS.