The QBF Cover encoding for Harary's Tic-Tac-Toe

Steve Boucher[†], Roger Villemaire[‡,*]
[†] Dept. of Computer Science, UQAM, Canada, steve.boucher@live.ca
[‡] Dept. of Computer Science, UQAM, Canada, villemaire.roger@uqam.ca

Abstract

Games present sophisticated challenges and require intricate reasoning. Quantified Boolean Formulas (QBF) enjoy a natural game-theoretic semantics and are therefore a promising representation for games. However, solving games with current QBF solvers remains challenging due to their significant runtime. In this paper, we use QBF solvers' reasoning capabilities to search for a specific kind of winning strategy, restricting the allowed moves. While this could, in principle, prevent the discovery of a winning strategy, when there are only more complex ones, we experimentally show that our method does indeed find a winning strategy when one is found by the current best QBF encoding. Furthermore, we also show that this reduction in available moves allows to reduce solving time across QBF solvers and preprocessors.

Keywords: Quantified Boolean Formulas, QBF, Game, Combinatorial Games, Harary's Tic-Tac-Toe, Winning Strategy

1. Introduction

Combinatorial games are zero-sum complete information games where players, usually two, compete, with the first player to attain the intended objective winning the game. Many board games such as Chess, Checker, Go, Tic-Tac-Toe, Hex and Gomoku all fall within this category. With deceivingly simple rules, these games raise intricate challenges that captivate players. Game playing therefore necessitates complex reasoning capabilities and it hence comes as no surprise that games are of central concern in AI. Moreover, the mathematical study of combinatorial games is a highly active field [1–4] with many striking results, open questions, and research challenges.

Harary Tic-Tac-Toe (HTTT) [5] is a combinatorial game, where two players, Black and White, alternatively claim cells on a square board, with the first player to form a specified target shape winning the game. HTTT has been very influential with many results, generalizations, and complex combinatorial problems [6–13], some of which have been open for many years. For instance, the winning status of the *Snaky* polyomino is still unknown, even on the original square boards.

Quantified Boolean Formulas (QBFs), where a sequence of existential and universal quantifiers on Boolean variables, the *prenex*, is followed by a purely propositional formula, the *matrix*, allow for a natural game semantics. Following the quantifiers' order, an existential player chooses values for existentially quantified variables while a universal player does the same for universally quantified variables. A QBF is then satisfied exactly when the existential player has a winning strategy ensuring that the matrix is satisfied. It therefore comes as no surprise that combinatorial games have attracted the attention of the QBF community [14–18]. Determining Snaky's status on 9×9 boards has, for instance, been set as a challenge for the QBF-community [16, 17]. Game encodings furthermore allow to evaluate QBF-solvers' performance with a large number of quantifier alternations, in stark contrast to the small number of alternations found in many QBF applications [19].

Solving a game consists in showing whether or not there is a winning strategy for some player. For HTTT, the usual approach to QBF game solving is to encode, as a QBF instance,

[*] villemaire.roger@uqam.ca

the existence of a winning strategy for the first player (Black) [14, 16, 17, 20], since it is known that there is no winning strategy for the second player (White).

In this paper, we leverage the fact that some moves are more sensible than others. We therefore introduce the COVER QBF encoding, which encodes the existence of a specific kind of strategy for the first player that restricts possible moves.

We experimentally show that this restriction on the strategy is not detrimental since our COVER encoding actually allows to find a winning strategy as often as the current best HTTT QBF encoding COR+. Furthermore, we also show that this reduction in the number of available moves has a beneficial effect on the QBF solving time of our COVER encoding with respect to that of COR+.

This paper is structured as follows. Section 2 introduces QBF formulas, Harary's Tic-Tac-Toe, and presents related work. Section 3 defines our COVER encoding, which is evaluated in Section 4. Section 5 finally concludes the paper.

2. Background Knowledge

2.1. QBF Formula

A *Quantified Boolean Formula (in prenex CNF form)* (QBF) is formed of a finite sequence of universal ($\forall$) and existential ($\exists$) quantifiers on Boolean ($0/1$) variables followed by a *matrix*. For its part, the matrix is a formula in *Conjunctive Normal Form* (CNF), i.e., a conjunction ("AND") of disjunctions ("OR") of *literals*, which are variables or negation of variables. We consider only *closed* QBF where every variable is quantified (existentially or universally).

QBF semantics can be defined by a game, as mentioned in the introduction, or alternatively by rewriting $\exists x\varphi(x)$ into $\varphi(0) \vee \varphi(1)$ and $\forall x\varphi(x)$ into $\varphi(0) \wedge \varphi(1)$ to remove every quantifier and then simply evaluating the obtained expression.

Determining the truth value of a QBF is PSPACE-complete [21] and is widely considered to be of a higher time complexity than CNF solving (SAT). There is therefore quite some interest in extending the striking advances in SAT solving toward QBF and accordingly many QBF solvers have been developed over the years [22–25].

2.2. Harary's Tic-Tac-Toe

Harary's Tic-Tac-Toe [5] is a game played by two players, Black and White, on a board divided into square cells called *positions*. Players alternate, with Black playing first. At each time-step the current player claims an empty position that then acquires her color. The objective of the game is to form the *target shape* in a single color. The first player that succeeds wins the game. Target shapes are edge-connected shapes, called *polyominos*. For instance, Figure 1 depicts the Snaky polyomino.

Figure 1. The Snaky polyomino

Originally, the game was played on a regular (square) board but nowadays torus boards that wrap around at horizontal and vertical edges are also considered.

By a classic strategy stealing argument one observes that a winning strategy allowing White to complete the target could as well be used by Black (pretending an arbitrary pre-game move). There is therefore no such winning strategy for White. The game outcome is hence that either Black has a winning strategy, in which case the target shape is said to be a *winner*, or White has a *blocking* strategy preventing Black's win, in which case the target shape is said to be a *loser*.

On regular boards the asymptotic (i.e., when the board is big enough) winner/loser status of all polyominos was already known to Harary [5], except for Snaky (Figure 1) whose exact status has drawn considerable attention but is still an open question. However, a polyomino that is a winner on large enough boards could be a loser on smaller boards. Also, very little has been published on the winner/loser status on torus boards, which is a distinct question.

2.3. **Related Work**

Surprisingly, encoding a combinatorial game in QBF is not as straightforward as one could expect. For instance, in order to correctly represent the game's rules, [14] introduces so-called *indicator-variables* ensuring that when a player breaks a game rule the matrix indeed reaches the correct Boolean value, i.e., False (0) if the first player breaks a rule, and True (1) if it is the second player. This QBF-encoding is then applied to the Connect-4 game, showing a rapidly increasing run time that unfortunately does not allow to solve the standard Connect-4 game on a 7×6 board within 10 minutes. A similar encoding is also developed for HTTT [16], solving all 84 instances on a 3×3 board with 10s timeout and some of the 96 instances on a 4×4 board with a 1000s timeout. Game solving is hence challenging for QBF solving technologies.

While indicator-variables have been thought to be crucial to QBF-encoding of games [15], [17] distinguish between move choices variables, existentially quantified for the first player and universally quantified for the second, and variables determining the occupied positions on the board that are always existentially quantified. Occupied board positions are then changed only when the players' moves do not break any game rules. In the context of *positional games*, such as HTTT, where players lay but do not move stones on a board, [17] introduces the very compact COR QBF encoding. This allowed to solve all HTTT instances on a 4×4 board with 1000s timeout. It is also noted in [17] that searching for a winning strategy by encoding in QBF a game of maximal length, for instance, 25 for a 5x5 board, is intractable and it is much faster to use *iterative deepening* and progressively solve for games of increasing lengths $k = 1, 2, \ldots$. Indeed, as soon as a winning strategy for Black is found, the shape is a winner.

A QBF-encoding for the PAIRING *blocking strategy* for White is introduced in [18]. This approach is then shown to be more than two orders of magnitude faster than [17] on 4×4 boards. Moreover, in a further twist to iterative deepening and making good use of the Black winning strategy/White blocking strategy duality, [18] extends this method by alternating the search for a winning strategy for Black with COR, for k odd, with that of a blocking strategy for White with PAIRING, for k even. As soon as a satisfiable instance is encountered, the shape is winning if k is odd or losing if k is even, completely solving the game. This approach allows [18] to solve 72 of 110 instances on 5×5 boards (1000s timeout) compared to only 7 for [17].

3. **The COVER Encoding**

In this section, we introduce our COVER encoding. We first present the underlying principles, then the variables, quantifiers, and clauses, and finally we justify the correctness of this QBF encoding.

3.1. **Overview and Rationale**

The COVER encoding follows the general approach of QBF game encodings [14, 16] where at each time-step $t = 0, \ldots, t_{end}$, the game configuration, i.e., the positions (x, y) occupied by Black and White, are encoded by the Boolean variables $black_{t,x,y}$, and $white_{t,x,y}$.

Furthermore, as for the COR/COR+ encodings [17, 20], the COVER encoding takes advantage of the positional nature of the game in which an occupied position remains occupied up to game's end. Therefore, as the game progresses, occupied positions remain so. Furthermore, at each time-step, it is sufficient to simply record which new position is claimed using variables $moveB_{t,j}$ for t odd (Black's moves) and $moveW_{t,j}$ for t even (White's moves) as detailed in the next subsection. Finally, Black (the existential player) can stop the game at any time point by setting the $time_t$ variable to false, at which point the game configuration is no longer allowed to change. This hence allows to check the winning condition simply at t_{end}.

QBF game encodings therefore present a long quantifier prefix, with a number of alternations proportional to the game length (t_{end}). Furthermore, each quantifier block expresses a player's move with a length proportional to the number of possible moves. The COVER encoding follows [20] and uses a logarithmic (binary) encoding for the moves. Moreover, the central objective of the COVER encoding is to reduce the number of possible choices that a player can make. This is indeed a natural objective since as the play goes on, some choices are obviously more sensible than others.

To this end, the COVER encoding restricts the players' moves. Black's very first move is simply restricted by symmetry breaking, as in the COR/COR+ encodings. Furthermore, the COVER encoding restricts further moves to the positions of a specific set, called the *cover*. This set contains the positions of all shapes that contains Black's first move. The objective is therefore to concentrate and focus the players on positions that offer the possibility of completing an already partially occupied shape.

Restricting Black's moves to the cover is sound in the sense that if a winning strategy restricting Black move's is found, this is indeed a winning strategy for Black. In the converse direction, this is, however, not complete, since there could be a winning strategy for Black, without any that restricts Black to the cover. Nevertheless, we will show in Section 4 that in our experiments this does not occur and that a winning strategy for Black is found with COVER every time that such a strategy is found with COR+.

As to restricting White's moves to the cover, this is clearly unsound. Indeed, one must ascertain that Black has a strategy that is winning and this irrespective of White's behavior. To restore soundness, our COVER encoding restricts White's moves to the cover but also allows a single additional out-of-cover (ooc) (White) move. COVER also keeps track of the number of times White plays out of cover (ooc) in order to check the winning condition, as we will now see.

In COR/COR+ the winning condition is simply that at t_{end} Black has formed the target shape and White not. As Black can halt the game at any time, preventing further positions from being claimed, this rightly ensures that Black has a winning strategy that complete the shape before White does. COVER's winning condition still checks that Black has completed the target shape, but for White COVER rather considers, for all target forms F the parts F_c lying in the cover and F_o outside of the cover, in the following way. For every target shape F, such that the size of F_o does not exceed the number of White's out-of-cover moves, COVER checks that White did not complete F_c. This ensures that White could not have completed F, regardless of where its out-of-cover moves could have been. This is therefore sound.

In summary, with a cover of size n, Black is restricted to n possible moves, while White is restricted to exactly $n+1$ moves including its out-of-cover move. COVER therefore bounds the number of moves both for Black and White.

3.2. **Variables and Quantifiers**

We now define the variables representing the game and their intended meaning. The range of parameters x, y, e, t below is as follows, unless otherwise stated: $1 \leq x \leq W, 1 \leq y \leq H$, where W is the width and H the height of the board, $e \in E$ for E the set of target shapes on the board, and $t \in T = \{0, \ldots, t_{end}\}$ the set of game turns.

$$time_t : \text{the game is running at time } t \tag{3.1}$$
$$moveB_{t,j} : j\text{-th digit of the binary encoding of Black's move at time } t \tag{3.2}$$
$$moveW_{t,j} : j\text{-th digit of the binary encoding of White's move at time } t \tag{3.3}$$
$$incoverS_e : \text{the shape } e \text{ is contained in the cover} \tag{3.4}$$
$$incoverP_{x,y} : \text{the position at } x, y \text{ is in the cover} \tag{3.5}$$
$$out\_of\_cover\_inc_{t,i} : \text{White has already played outside the cover a number } i \text{ of times at turn } t \tag{3.6}$$
$$black_{t,x,y} : \text{there is a black stone at } x, y \text{ at time } t \tag{3.7}$$
$$white_{t,x,y} : \text{there is a white stone at } x, y \text{ at time } t \tag{3.8}$$
$$win_e : \text{all cells in shape } e \text{ are black at time } t_{end} \tag{3.9}$$

In the previous variable list, j ranges over the number of bits necessary to encode the moves and i from 0 to the size (number of cells) of the target shape.

Note that, we will define the cover in two steps. First, with the $incoverS_e$ variables we will determine the **shapes** that are in the cover, which are those that contain Black's first move. Secondly, we will determine with the $incoverP_{x,y}$ variables the **positions** within the cover, which are those within some shape contained in the cover.

We now specify the quantifier blocks. They appear in turn order beginning at turn $t = 0$ where the board is initialized, as we will see in the next subsection.

$$\exists time_0 \tag{3.10}$$
$$\exists black_{0,x,y};\ \text{for all } x, y \tag{3.11}$$
$$\exists white_{0,x,y};\ \text{for all } x, y \tag{3.12}$$

At $t = 1$, Black plays and the cover is defined.

$$\exists time_1 \tag{3.13}$$
$$\exists moveB_{1,j}; \text{for all } j \tag{3.14}$$
$$\exists incoverS_e; \text{for all } e \tag{3.15}$$
$$\exists incoverP_{x,y};;\ \text{for all } x, y \tag{3.16}$$
$$\exists black_{1,x,y};\ \text{for all } x, y \tag{3.17}$$
$$\exists white_{1,x,y};\ \text{for all } x, y \tag{3.18}$$

At $t = 2, ..., t_{end}$ White plays for t even and Black for t odd. Furthermore, at White's turns the count of ooc moves are recorded with the $out\_of\_cover\_inc_{t,i}$ variables.

$$\exists time_t \tag{3.19}$$

$$\forall moveW_{t,j};\ \text{for } t \text{ even and all } j \tag{3.20}$$

$$\exists moveB_{t,j};\ \text{for } t \text{ odd and all } j \tag{3.21}$$

$$\exists out\_of\_cover\_inc_{t,i};\ \text{for } t \text{ even and all } i \tag{3.22}$$

$$\exists black_{t,x,y};\ \text{for all } x, y \tag{3.23}$$

$$\exists white_{t,x,y};\ \text{for all } x, y \tag{3.24}$$

Finally, at the last turn of the game t_{end}, we have:

$$\exists win_e; \text{for all } e \tag{3.25}$$

3.3. **Clauses**

We now give the clauses, either directly as disjunctions of literals or as conjunctions of clauses or double-implications that can readily be transformed into clauses in the usual way. Below, t is any timepoint greater of equal to 1 since we will compare with $t-1$. As before x, y is any board position.

Time Handling. If the game is still running at time t, it was running at time $t-1$.

$$\neg time_t \vee time_{t-1} \tag{3.26}$$

Structure of the board. Clause (3.27) encodes that there is no stone on the board at time $t = 0$. Furthermore, both players cannot claim the same cell as expressed by (3.28). Finally, (3.29) indicates that once a cell is claimed, it stays the same color until the end of the game.

$$\neg black_{0,x,y} \wedge \neg white_{0,x,y} \tag{3.27}$$

$$\neg black_{t,x,y} \vee \neg white_{t,x,y} \tag{3.28}$$

$$(\neg black_{t-1,x,y} \vee black_{t,x,y}) \wedge \neg(white_{t-1,x,y} \vee white_{t,x,y}) \tag{3.29}$$

Frame axioms. The following clauses (3.30), (3.31), (3.32), (3.33) specify that when the game is over, no new stone are permitted on the board and no new black, or white stones can appear on the board if it is not Black's or White's turn.

$$time_t \vee black_{t-1,x,y} \vee \neg black_{t,x,y} \tag{3.30}$$

$$time_t \vee white_{t-1,x,y} \vee \neg white_{t,x,y} \tag{3.31}$$

$$black_{t-1,x,y} \vee \neg black_{t,x,y}; \text{for } t \text{ even} \tag{3.32}$$

$$white_{t-1,x,y} \vee \neg white_{t,x,y}; \text{for } t \text{ odd} \tag{3.33}$$

Cover setup. By definition, the cover contains the positions of all shapes that contain Black's first move. Accordingly, (3.34) states that $incoversS_e$ holds exactly when one of e's cells has been played on Black's first move. Furthermore, (3.35) expresses the fact that a position x, y is in the cover exactly when it is the position of a cell that is in a shape e of the cover. Finally, (3.36) expresses that Black cannot play outside the cover, as intended.

$$incoverS_e \iff \bigvee_{(x,y)\in e} black_{1,x,y} \tag{3.34}$$

$$incoverP_{x,y} \iff \bigvee_{\{e\in E;(x,y)\in e\}} incoverS_e \tag{3.35}$$

$$incoverP_{x,y} \vee \neg black_{t_{end},x,y} \tag{3.36}$$

Black's moves. The next clauses (3.37), (3.38) represent Black actions. Here a move that sets a stone on the cell at x, y is encoded by the binary string $[x, y]$ with j-th bit $[x, y](j)$. These clauses express the fact that the cell at x, y becomes black at turn t if it was not played before and the j-th digit of the binary encoding representing Black's choice is $moveB_{t,j}$. This is the At-Most-One constraint of [20] preventing Black from playing multiple times in the same turn.

$$black_{t-1,x,y} \vee \neg black_{t,x,y} \vee moveB_{t,j}; [x, y](j) = 1 \tag{3.37}$$

$$black_{t-1,x,y} \vee \neg black_{t,x,y} \vee \neg moveB_{t,j}; [x, y](j) = 0 \tag{3.38}$$

Symmetry breaking. Symmetry breaking uses the fact that on a regular board Black's first move is irrelevant, up to reflection and rotation of the board. On a torus board the situation is even simpler since all cells are equivalent and Black can be constrained to play on a specific position without restricting generality. Therefore, for torus boards, we force Black to play at the center (3.39) whereas on a normal board, we force Black to play in the top left side triangle of the board (3.40).

$$black_{1,\lceil \frac{W}{2} \rceil,\lceil \frac{H}{2} \rceil} \tag{3.39}$$

$$\bigvee_{x=1}^{\lceil \frac{W}{2} \rceil} \bigvee_{x=1}^{\lceil \frac{H}{2} \rceil} black_{1,x,y} \tag{3.40}$$

White's moves. For White's moves we distinguish two cases. First, if White plays within the cover, clause (3.41) expresses that if the game is still on, the cell at x, y is not black, x, y is in the cover, and White's move is to the cell at x, y, then the cell at x, y is white.

$$\neg time_t \vee black_{t-1,x,y} \vee \neg incoverP_{x,y} \vee \bigvee_{j;[x,y](j)=1} \neg moveW_{t,j} \vee \bigvee_{j;[x,y](j)=0} moveW_{t,j} \vee white_{t,x,y} \tag{3.41}$$

Furthermore, if White rather chooses the out-of-cover (ooc) move, we simply increase the $out\_of\_cover\_inc$ counters in the following way. Clause (3.42) expresses that if the game is still on, White chooses the out-of-cover (ooc) move, and $out\_of\_cover\_inc_{t-2,i-1}$, then $out\_of\_cover\_inc_{t,i}$. Note that the highest value of i attained will be equal to the number of times White has played outside of the cover. Furthermore, the only requirement on the binary value $[ooc]$ is that it differs from all $[x, y]$.

$$\neg time_t \vee \bigvee_{j;[ooc](j)=1} \neg moveW_{t,j} \vee \bigvee_{j;[ooc](j)=0} moveW_{t,j} \vee \neg out\_of\_cover\_inc_{t-2,i-1} \vee out\_of\_cover\_inc_{t,i} \tag{3.42}$$

This counting must be initialized with clause (3.43) to start at 0 and then clause (3.44) ensures that i can only increase as t increases. Note that in all these clauses i ranges up to the number of cells in the target shape. This is sufficient since there is no point in counting for more moves than there are cells in the target shape, as we will see in the winning condition.

$$out\_of\_cover\_inc_{t,0} \tag{3.43}$$

$$\neg out\_of\_cover\_inc_{t-2,i} \vee out\_of\_cover\_inc_{t,i} \tag{3.44}$$

Winning Condition. Clause (3.45) first expresses that win_e is true exactly when Black has completed all the cells of e. Then (3.46) ensures the first part of the winning condition, namely that Black must complete at least one target shape.

$$win_e \iff \bigwedge_{(x,y)\in e} black_{t_{end},x,y} \tag{3.45}$$

$$\bigvee_{e\in E} win_e \tag{3.46}$$

We must now ensure that White did not win. We consider for any shape e and number i of out-of-cover White moves all subsets p of shape e containing i elements and express with clause (3.47) that having, all at once, i out-of-cover White moves, all of $e \setminus p$ (all cells of e except those of p) white, and all cells of p outside the cover, is impossible. Therefore, whatever moves that White could have done outside the cover, this could never have completed a shape. This ensures that White clearly could not have completed the target shape.

$$\neg out\_of\_cover\_inc_{t_{end},i} \vee \bigvee_{(x,y)\in e\setminus p} \neg white_{t_{end},x,y} \vee \bigvee_{(x,y)\in p} incoverP_{x,y} \tag{3.47}$$

3.4. **Justification**

That our encoding is correct is essentially straightforward, except maybe for the clauses (3.41), (3.42) that can be justified in a way similar to that of [17, 20]. The justification therefore follows from the fact that the only universally quantified variables are the $moveW_{t,j}$ variables. More precisely, in the QBF game semantics, where the existential player aims at making all clauses true, for clause (3.41) the existential player will have to set $white_{t,x,y}$ only if all other literals are false and White has indeed made a move inside the cover. Similarly, for clause (3.42) the existential player will have to set $out\_of\_cover\_inc_{t,i}$, i.e., increase the count, only if White made an ooc move.

In other words, the existential player lay all stones, black and white, and sets the $out\_of\_cover\_inc_{t,i}$ counter. The clauses therefore ensure that the existential player acts on White's behalf only if White plays as expected.

4. **Experimental Results**

We compare our approach with the current best QBF encoding for HTTT, which is the COR+ encoding [20], an improved version of the COR encoding of [17]. Data and complete code to reproduce our results are available at gitlab.info.uqam.ca/boucher.steve/cover-sources.git.

Although the SN encoding of [26] has also been applied to the maker-breaker variant of HTTT, this is a weaker version of HTTT in which Black wins if she completes the target shape, regardless of the fact that White could have completed the target earlier.

Furthermore, that work only reports results on 11 polyominos on a 5x5 regular board that our method processes within the same time, and this for the more involved full game where Black loses as soon as White completes the target shape.

We evaluate our encoding with the following QBF solvers: CAQE [22], DepQBF [23], QESTO [24], Qute [25], and with the QBF-preprocessors (simplifiers): bloqqer [27], HQSPre [28], and QRATPre+ [29]. All experiments are run on Dell OptiPlex 7050, 3.6 GHz Intel Core i7-7700 Quad-Core, 16GB of 2400 MHz DDR4 RAM.

Since [18] easily solves all games on 4x4 boards and that our approach is equally effective on these small boards, our experiments are run on 5x5 boards, both normal and torus. As in [16, 17], we also consider all polyominos of specific sizes. Namely, we experiment on all 47 polyominos that fit on a 4x4 board (in order to allow some extra space for nontrivial games on our 5x5 boards), with the exception of the one-cell Elam that is obviously a winner on Black's first move.

4.1. **Iterative deepening on 5x5 boards**

In this subsection, we compare iterative deepening with COR+/PAIRING with iterative deepening with COVER/PAIRING.

In this setting, iterative deepening yields 3177 QBF instances for COR, 3177 for COVER (and a similar additional number for PAIRING) in order to solve the game as explained previously. Results are summarized in Table 1.

solver	preprocessor	B/W/U	COR+/PAIRING	COVER/PAIRING
CAQE	none	13/62/19	48466.13	48397.52
	bloqqer	**14**/62/18	45665.90	45308.02
	HQSPre	13/62/19	50256.66	50248.72
	QRATPre+	**14**/62/18	46024.13	45530.75
DepQBF	none	**14**/62/18	46669.55	45903.58
	bloqqer	**14**/62/18	47591.13	46078.10
	HQSPre	**14**/62/18	47491.20	46655.83
	QRATPre+	**14**/62/18	47175.92	46455.84
QESTO	none	**14**/**64**/16	44035.47	44911.54
	bloqqer	**14**/**64**/16	41270.15	41196.53
	HQSPre	12/**64**/18	48757.67	48756.93
	QRATPre+	**14**/**64**/16	41868.17	41834.90
Qute	none	12/60/22	55356.24	55265.69
	bloqqer	12/60/22	57134.60	57095.74
	HQSPre	12/61/21	53129.73	53122.35
	QRATPre+	12/60/22	55470.14	55385.08

Table 1. Iterative deepening for COR+/PAIRING and COVER/PAIRING on 5x5 boards with 2500s timeout

In Table 1 the **B/W/U** column shows the number of shapes/board types (normal or torus) for which a winning strategy for Black is found with COR+ and COVER, a blocking strategy for White is found with PAIRING, and the remaining Unknown cases where no winning nor blocking strategy is found.

One first notes that there is a single value for **B** since the exact same number of winning strategies for Black is found with COR+ and COVER. Moreover, we checked that COR+ and COVER both find a winning strategy in the very same cases (shape/board) and at the same iteration number (t_{end} value). Therefore, both winning strategies have the same number of moves. This shows that while COVER restricts itself to a specific kind of winning strategy, this does not prevent a winning strategy from being found with COVER each time that a winning strategy is found with COR+.

As to solver/preprocessor performance, one notes from Table 1 that QESTO is the solver that attains the highest number of winning/blocking strategies, namely 14 and 64, respectively. Furthermore, from the last two columns of the table, one sees that for every preprocessor but HQSPre, QESTO yields the best total run time, with QESTO/bloqqer and QESTO/QRATPre+ in the first and second positions. Finally, run time for COR+/PAIRING and COVER/PAIRING are similar. Therefore, overall, Table 1 shows that COVER is as effective as COR+ in this iterative deepening setting with PAIRING.

However, since iterative deepening runs up to timeout when no winning/blocking strategy is found and since Table 1 shows the total time including that for running PAIRING, this Table does not allow to correctly compare the performance of COR+ and COVER. To this end, we will now turn to a direct comparison of COR+ and COVER run times.

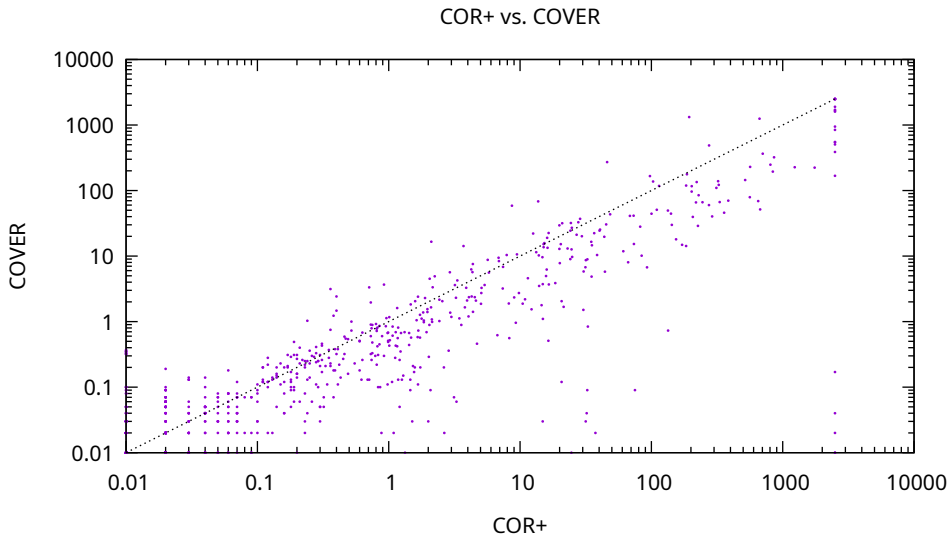

Figure 2. Run time for COVER/COR+ on 5x5 boards with 2500s timeout for all preprocessors and solvers

Figure 2 compares COVER and COR+ runtimes. More precisely, for each board type, shape, iteration (t_{end} value) and preprocessor/solver pair of the experiments presented in Table 1, Figure 2 shows the pairs formed of the runtimes for COVER and COR+. This therefore allows to compare COVER and COR+ on the same task (finding a winning strategy for Black) for a broad range of game settings (board type, shape, iteration) and this over many preprocessor/solver pairs.

One observes from this figure that the points appear largely under the diagonal, and that COVER therefore outperforms COR+ on most instances. Moreover, a detailed analysis of the data shows that COVER outperforms COR+ on 61% (336/547) of the instances taking more than 0.01s, 70% (248/355) of the instances taking more than 0.1s, 74% (136/183) of the instances taking more than 1.0s, and 76% (74/98) of the instances taking more than 10s. Therefore, COVER is furthermore even more beneficial as the difficulty the instance increases.

5. **Conclusion**

QBF is a flexible representation that enables the definition of many properties in a variety of ways. In order to solve a game such as HTTT, one must determine either that there is

a winning strategy for Black or a blocking strategy for White. However, QBF allows us to also look for specific kinds of strategies, as we did in this paper. This has many benefits. First, as we showed in Section 4 there are strategies which are easier to ascertain while still as frequent as a general strategy, under current QBF technologies.

There is, however, a second more fundamental benefit to our approach. By focusing on a specific kind of strategy, QBF technology is used to reason about the game, and learn new facts as the existence of such a specific strategy. This is clearly of interest to the study of combinatorial games. Indeed, this is reminiscent, for instance, of theoretical results such as [30] that, for HTTT with the Snaky target, shows that there is no winning strategy where Black only plays on a cell that share an edge with another black cell or [31] that shows that Snaky is a loser on a 6×6 board but that Black has a winning strategy if White is restricted to domino paving strategies. Our approach therefore has the potential to be successfully applied to further types of strategies and games, broadening our understanding of the essential properties of these games.

Acknowledgements

We gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2023-04557].

References

[1] E. D. Demaine and R. A. Hearn. "Playing games with algorithms: Algorithmic combinatorial game theory". In: *Games of No Chance 3*. Mathematical Sciences Research Institute Publications. Cambridge University Press, 2009, 3–56.

[2] M. A. Huggan, S. Huntemann, and B. Stevens. "The combinatorial game Nofil played on Steiner triple systems". In: *Journal of Combinatorial Designs* 30.1 (2022), pp. 19–47.

[3] E. D. Demaine and Y. Diomidov. "Strings-and-coins and nimstring are PSPACE-complete". In: *Combinatorial Game Theory: A Special Collection in Honor of Elwyn Berlekamp, John H. Conway and Richard K. Guy*. De Gruyter, 2022, 109 – 120.

[4] J. Guo and P.-C. Yu. "Puzzle and Dragons is hard". In: *Theoretical Computer Science* 994 (2024), p. 114477. ISSN: 0304-3975.

[5] F. Harary. "Achieving the Skinny Animal". In: *Eureka* 42 (1982), pp. 8–14.

[6] F. Harary, H. Harborth, and M. Seemann. "Handicap Achievement for Polyominoes". In: *Congressus Numerantium* 145 (2000), pp. 65–80.

[7] J. P. Bode and H. Harborth. "Hexagonal polyomino achievement". In: *Discrete Mathematics* 212 (2000), pp. 5–18.

[8] N. Sieben. "Hexagonal polyomino weak $(1, 2)$-achievement games". In: *Acta Cybernetica* 16 (2004), pp. 579–585.

[9] N. Sieben and E. Deabay. "Polyomino weak achievement games on 3-dimensional rectangular boards". In: *Discrete Mathematics* 290.1 (2005), pp. 61–78.

[10] E. Fisher and N. Sieben. "Rectangular polyomino set weak $(1, 2)$-achievement games". In: *Theoretical Computer Science* 409 (2008), pp. 333–340.

[11] N. Sieben. "Polyominoes with minimum site-perimeter and full set achievement games". In: *European Journal of Combinatorics* 29.1 (2008), pp. 108–117. ISSN: 0195-6698.

[12] J. Cardinal, S. Collette, H. Ito, M. Korman, L. S., H. Sakaidani, and T. P. "Cannibal Animal Games: a new variant of Tic-Tac-Toe". In: *Journal of Information Processing* 23.3 (2015), pp. 265–271.

[13] Diptarama, K. Narisawa, and A. Shinohara. "Drawing strategies for generalized tic-tac-toe (p, q)". In: *AIP Conference Proceedings* 1705.1 (2016), p. 020021.

[14] I. Gent and A. R. Rowley. "Encoding Connect-4 Using Quantified Boolean Formulae". In: *Modelling and Reformulating Constraint Satisfaction Problems*. 2003, pp. 78–93.

[15] C. Ansótegui, C. P. Gomes, and B. Selman. "The Achilles' Heel of QBF". In: *Proceedings of the Twentieth National Conference on Artificial Intelligence, AAAI 2005*. AAAI Press / The MIT Press, 2005, pp. 275–281.

[16] Diptarama, R. Yoshinaka, and A. Shinohara. "QBF Encoding of Generalized Tic-Tac-Toe". In: *Quantified Boolean Formulas, QBF 2016*. Vol. 1719. CEUR Workshop Proceedings. 2016, pp. 14–26.

[17] V. Mayer-Eichberger and A. Saffidine. "Positional Games and QBF: The Corrective Encoding". In: *Theory and Applications of Satisfiability Testing, SAT 2020*. Vol. 12178. LNCS. Springer, 2020, pp. 447–463.

[18] S. Boucher and R. Villemaire. "Quantified Boolean Solving for Achievement Games". In: *German Conference on AI,based KI 2021*. Vol. 12873. Lecture Notes in Computer Science. Springer, 2021, pp. 30–43.

[19] A. Shukla, A. Biere, L. Pulina, and M. Seidl. "A Survey on Applications of Quantified Boolean Formulas". In: *International Conference on Tools with Artificial Intelligence, ICTAI 2019*. IEEE, 2019, pp. 78–84.

[20] V. Mayer-Eichberger and A. Saffidine. *Positional Games and QBF: A Polished Encoding*. 2023. arXiv: 2005.05098 [cs.LO].

[21] L. J. Stockmeyer and A. R. Meyer. "Word Problems Requiring Exponential Time: Preliminary Report". In: *ACM Symposium on Theory of Computing (STOC)*. ACM, 1973, pp. 1–9.

[22] L. Tentrup. "CAQE and QuAbS: Abstraction Based QBF Solvers". In: *Journal on Satisfiability, Boolean Modeling and Computation* 11.1 (2019), pp. 155–210.

[23] F. Lonsing and U. Egly. "DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL". In: *Automated Deduction - CADE 26*. Vol. 10395. LNCS. Springer, 2017, pp. 371–384.

[24] M. Janota and J. Marques-Silva. "Solving QBF by Clause Selection". In: *International Joint Conference on Artificial Intelligence, IJCAI*. AAAI Press, 2015, pp. 325–331.

[25] T. Peitl, F. Slivovsky, and S. Szeider. "Qute in the QBF Evaluation 2018". In: *Journal on Satisfiability, Boolean Modeling and Computation* 11.1 (2019), pp. 261–272.

[26] I. Shaik, V. Mayer-Eichberger, J. van de Pol, and A. Saffidine. "Implicit QBF Encodings for Positional Games". In: *Advances in Computer Games, ACG 2023*. Vol. 14528. LNCS. Springer, 2023, pp. 133–145.

[27] A. Biere, F. Lonsing, and M. Seidl. "Blocked Clause Elimination for QBF". In: *Automated Deduction - CADE-23*. Vol. 6803. LNCS. Springer, 2011, pp. 101–115.

[28] R. Wimmer, S. Reimer, P. Marin, and B. Becker. "HQSpre - An Effective Preprocessor for QBF and DQBF". In: *Tools and Algorithms for the Construction and Analysis of Systems TACAS*. Vol. 10205. LNCS. 2017, pp. 373–390.

[29] F. Lonsing and U. Egly. "QRATPre+: Effective QBF Preprocessing via Strong Redundancy Properties". In: *Theory and Applications of Satisfiability Testing - SAT*. Vol. 11628. LNCS. Springer, 2019, pp. 203–210.

[30] H. Harborth and M. Seemann. "Snaky is an edge-to-edge loser". In: *Geombinatorics* V.4 (1996), pp. 132–136.

[31] H. Harborth and M. Seemann. "Snaky is a paving winner". In: *Bulletin of the Institute of Combinatorics and its Applications* 19 (1997), pp. 71–78.