QBF Encoding of Generalized Tic-Tac-Toe

Diptarama, Ryo Yoshinaka, Ayumi Shinohara

Graduate School of Information Sciences, Tohoku University, Japan
{diptarama@shino.,ry@,ayumi@}ecei.tohoku.ac.jp

Abstract. Harary's generalized Tic-Tac-Toe is an achievement game for polyominoes, where two players alternately put a stone on a grid board, and the player who first achieves a given polyomino wins the game. It is known whether the first player has a winning strategy in the generalized Tic-Tac-Toe for almost all polyominoes except the one called *Snaky*. $\mathsf{GTTT}(p, q)$ is an extension of the generalized Tic-Tac-Toe, where the first player places q stones in the first move and then the players place q stones in each turn. In this paper, in order to attack $\mathsf{GTTT}(p, q)$ by QBF solvers, we propose a QBF encoding for $\mathsf{GTTT}(p, q)$. Our encoding is based on Gent and Rowley's encoding for Connect-4. We modify three parts of the encoding: initial condition, move rule and winning condition of the game. The experimental results show that some QBF solvers can be used to solve $\mathsf{GTTT}(p, q)$ on 4×4 or smaller boards.

1 Introduction

In recent years it has been getting a more common approach to use SAT solvers to tackle some sort of hard problems for which no polynomial-time algorithm is known. Those problems include cryptanalysis [22, 26] and mathematical problems such as the Erdős discrepancy conjecture [17]. Still there are even harder problems in the real world to which SAT solvers cannot apply. The successful development of solvers for the QBF satisfiability problem (QBF solvers) [11, 14, 21] such as DepQBF [19, 18], RAReQS [25] and GhostQ [16] allows us to use them to attack PSPACE problems. Typical PSPACE problems are to decide whether a player has a winning strategy in two-player perfect information games. Particular instances of these games can be solved by reducing the game rules and a game position into a QBF and giving it to a QBF solver.

A trivial example of a two-player perfect information game is Tic-Tac-Toe (TTT), where two players alternately put a stone on a cell in the 3×3 board and the player who occupies three consecutive cells constituting a line will win. A generalization of TTT is known as *mnk*-Game, where players aim at achieving a line of length k on the $m \times n$-board. While the original TTT can be solved easily, it is known that to decide the winner of $mn5$-Game for a given position is PSPACE-complete [23]. Even when the game position of an instance is restricted to be blank, it has been open if the first player has a winning strategy in $mn5$-Game except for limited values of m and n [28].

Harary's generalized Tic-Tac-Toe (HTTT) is another variant [8]. The goal of HTTT is to achieve a given *polyomino*, i.e., a group of cells connected to

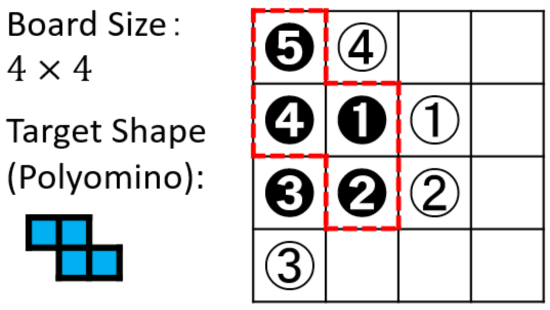

Fig. 1. Position example of Harary's generalized Tic-Tac-Toe on 4×4 board.

each other with edges, rather than a line. Fig. 1 gives an example of a game position of HTTT. The first and second players' stones are colored black and white, respectively, and the first player has won the game, because the four black stones numbered 1, 2, 4 and 5 form the target shape shown in the bottom left corner. Note that any rotation of the target polyomino and its reflection are admitted. HTTT has already been solved for all polyominoes on an arbitrary size board except cases of *Snaky* (Fig. 2) [8, 9]. Snaky is the name of a polyomino consisting of 6 cells. When the target is Snaky and the board dimension is 8×8 or smaller, it is known that no player has a winning strategy [7]. However, it is open for bigger boards. On the other hand, if the first player gets one additional stone at the initial position, he/she can certainly win [7, 10].

Diptarama *et al.* [4, 5] have proposed a further generalization, which we denote by $\mathsf{GTTT}(p, q)$, where the first player puts q stones at the first turn and then the two play p stones at each of their turns afterwards. Hence, $\mathsf{GTTT}(1, 1)$ is HTTT and $\mathsf{GTTT}(1, 2)$ corresponds the situation where the first player has a handicap stone at the beginning of a game. They showed that the first player will win in $\mathsf{GTTT}(2, 2)$ and the second player will win in $\mathsf{GTTT}(2, 1)$. Diptarama *et al.* [3] reported that in some cases QBF solvers can solve $\mathsf{GTTT}(p, q)$ that encoded by a tool called Toss [15, 27] faster than the proof number search [1]. Therefore, the QBF approach seems to be hopeful to tackle $\mathsf{GTTT}(p, q)$, including HTTT for Snaky. In order to evaluate the potential of the QBF approach against $\mathsf{GTTT}(p, q)$, we propose a QBF encoding of $\mathsf{GTTT}(p, q)$ and apply QBF solvers to it in this paper. Our QBF encoding is based on Gent and Rowley's encoding for the game called Connect-4. We modify their encoding for the initial position, players' move, and winning condition. We submitted 180 instances from $\mathsf{GTTT}(p, q)$ as benchmarks to QBFEVAL'16. This paper shows and analyses experimental results on those instances, where only instances with small board and target shape were solved. It still remains open whether the first player has a winning strategy in HTTT for Snaky on the 9×9 board. The instances are available at `www.shino.ecei.tohoku.ac.jp/~diptarama/gttt_qbf.html`.

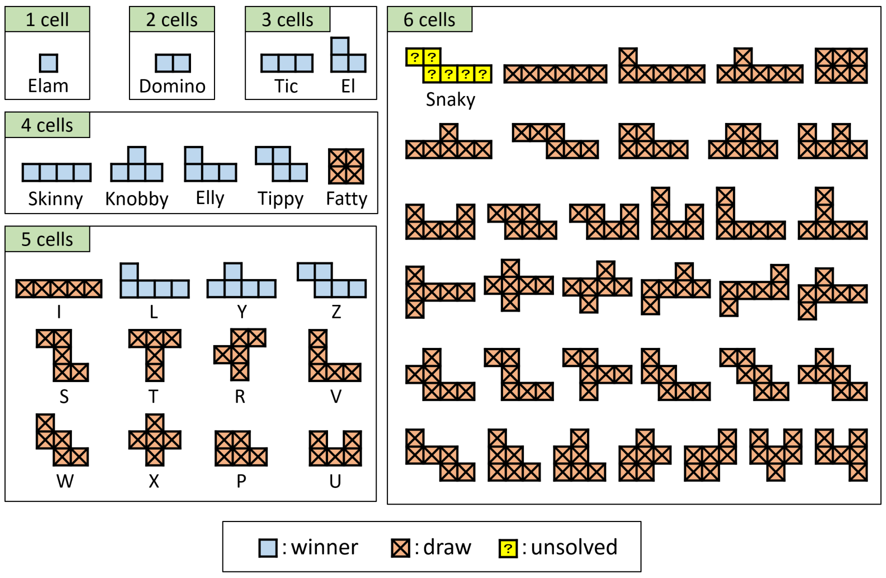

Fig. 2. List of 1–6 cell polyominoes. The first player can win HTTT for polyominoes colored blue, while no one can win for polyominoes with crossed cells and all of polyominoes with at least 7 cells.

2 QBF encoding

In this section, we will describe how to encode $\mathsf{GTTT}(p, q)$ into a QBF which holds true if and only if the first player has a winning strategy. It is easy to modify our formula for checking whether the second player can win. We modify the QBF encoding of the Connect-4 game proposed by Gent and Rowley [6], and define a new encoding for $\mathsf{GTTT}(p, q)$. First, we describe notation that used in variables and clauses. Next, we describe variables and quantifiers that are used in the $\mathsf{GTTT}(p, q)$ encoding. Last, we show clauses that are used in the $\mathsf{GTTT}(p, q)$ encoding. Throughout this paper we call the first player *Black* and the second player *White*.

2.1 Notation

Let W and H be the *width* and *height* of the game board, respectively. The maximum number Z of turns of the game in $\mathsf{GTTT}(p, q)$ is bounded by $Z = \lfloor (WH - q)/p \rfloor + 1$, because the first player places q stones at the first move and then both players place p stones respectively.

Next, similarly to the Connect-4 encoding, we define illegal moves in the game as cheats. We classify illegal moves into 4 categories; (1) the player places more stones than the number of stones that the player should place in one move, (2) the player places less stones than the number of stones that the player should

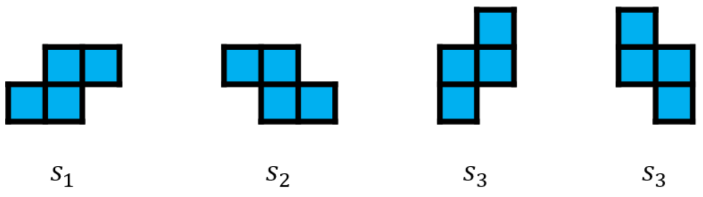

Fig. 3. Tippy as a target shape $S = \{s_1, s_2, s_3, s_4\}$, where $s_1 = ((0,0),(1,0),(1,1),(2,1))$, $s_2 = ((0,1),(1,0),(1,1),(2,0))$, $s_3 = ((0,0),(0,1),(1,1),(1,2))$, and $s_4 = ((0,1),(0,2),(1,0),(1,1))$.

put in one move, (3) the player places a stone on an occupied cell, and (4) the first player puts his/her first stone outside a specific area. The fourth kind of a cheat is not regarded illegal in a usual game play, but in our encoding, we force the first stone to be put in a special area to reduce the number of moves to be considered by breaking symmetries. The number of cheats in category a is denoted by C_a for $a \in \{1,2,3,4\}$. Then, $C_1 = \binom{WH}{p+1}$ at the first turn and $C_1 = \binom{WH}{q+1}$ otherwise. $C_2 = \binom{WH}{p-1}$ at the first turn and $C_2 = \binom{WH}{q-1}$ otherwise. Moreover, $C_3 = WH$ and $C_4 = 1$.

Last, the target shapes that both players try to achieve in a game are defined as follows. We represent a polyomino's shape as a tuple of relative coordinates of the cells $s = ((s^{x_1}, s^{y_1}), (s^{x_2}, s^{y_2}), \ldots, (s^{x_k}, s^{y_k}))$, where $\min_i s^{x_i} = \min_i s^{y_i} = 0$ and k is the number of cells of the polyomino. We define $|s|_W = \max_i s^{x_i} + 1$ and $|s|_H = \max_i s^{y_i} + 1$, which denote the width and height of the target shape, respectively. Since all of the 90, 180, 270 degree rotations and their reflections of a polyomino are admitted as a target, the target shapes of $\mathsf{GTTT}(p,q)$ are represented as a set $S = \{s_1, s_2, \ldots, s_{|S|}\}$, where $|S| \leq 8$. If the polyomino is symmetric, $|S|$ can be smaller than 8. For example, Fig. 3 shows the target shapes when the target polyomino is Tippy (see Fig. 2).

2.2 Variables and Quantifiers

By using the above parameters, we define variables used in the QBF encoding of $\mathsf{GTTT}(p,q)$ as follows. The range of each of the parameters below is as follows: $1 \leq x \leq W$, $1 \leq y \leq H$, $1 \leq a \leq 4$, $1 \leq c \leq C_a$, and $1 \leq s \leq |S|$. Moreover, $1 \leq z \leq Z+1$ for $\mathit{gameover}_z$ and $\mathit{occupied}_{z,x,y}$, and $1 \leq z \leq Z$ for the rest.

1. $\mathit{blackwin}$, $\mathit{whitewin}$, and draw: true iff Black wins, White wins, or draws, respectively.
2. $\mathit{blackwin}_z$ and $\mathit{whitewin}_z$: true iff Black or White wins at turn z, respectively.
3. $\mathit{gameover}_z$: true iff game is over at turn z.
4. $\mathit{occupied}_{z,x,y}$: true iff the cell at coordinates (x,y) is occupied by either a black or white stone before move z has been made.
5. $\mathit{black}_{z,x,y}$ and $\mathit{white}_{z,x,y}$: true iff there is a black or white stone on the cell at coordinates (x,y) after move z has been made, respectively.

6. $\mathit{blackcheats}_{z,a,c}$ and $\mathit{whitecheats}_{z,a,c}$: true iff Black or White respectively does an illegal move numbered c in category a at turn z.
7. $\mathit{blackcheat}_z$ and $\mathit{whitecheat}_z$: true iff Black or White respectively does any of illegal moves at turn z.
8. $\mathit{blackshape}_{z,i,x,y}$ and $\mathit{whiteshape}_{z,i,x,y}$: true iff Black or White respectively achieves a target shape s_i on coordinates (x, y) as origin at turn z.
9. $\mathit{blackmove}_{z,x,y}$ and $\mathit{whitemove}_{z,x,y}$: true iff Black or White respectively places a stone on a cell at coordinates (x, y) at turn z.

Quantifiers in QBF encoding of $\mathsf{GTTT}(p, q)$ are defined for each turn.

$$\begin{array}{l}
\exists.(\mathit{blackwin}, \mathit{whitewin}, \mathit{draw}, \mathit{gameover}_1, \mathit{occupied}_{1,1,1}, \ldots, \mathit{occupied}_{1,W,H}) \\
\exists.(\mathit{blackmove}_{1,1,1}, \mathit{blackmove}_{1,1,2}, \ldots, \mathit{blackmove}_{1,W,H}) \\
\exists.(\mathit{gameover}_2, \mathit{occupied}_{2,x,y}, \mathit{black}_{1,x,y}, \mathit{white}_{1,x,y}, \mathit{blackcheat}_1, \mathit{whitecheat}_1, \ldots) \\
\forall.(\mathit{whitemove}_{2,1,1}, \mathit{whitemove}_{2,1,2}, \ldots, \mathit{whitemove}_{2,W,H}) \\
\exists.(\mathit{gameover}_3, \mathit{occupied}_{3,x,y}, \mathit{black}_{2,x,y}, \mathit{white}_{2,x,y}, \mathit{blackcheat}_2, \mathit{whitecheat}_2, \ldots) \\
\qquad \vdots \\
\exists.(\mathit{blackmove}_{Z,1,1}, \mathit{blackmove}_{Z,1,2}, \ldots, \mathit{blackmove}_{Z,W,H}) \\
\exists.(\mathit{gameover}_{Z+1}, \mathit{occupied}_{Z+1,x,y}, \mathit{black}_{Z,x,y}, \mathit{white}_{Z,x,y}, \ldots)
\end{array}$$

First we introduce variables $\mathit{blackwin}$, $\mathit{whitewin}$, draw, $\mathit{gameover}_1$, and $\mathit{occupied}_{1,x,y}$ with the existential quantifier, which will be forced to be 1, 0, 0, 0, and 0, respectively, in the body CNF. Then for each turn z with $1 \leq z \leq Z$, we place "move" variables $\mathit{blackmove}_{z,i,j}$ and $\mathit{whitemove}_{z,i,j}$ with $\exists$ for $\mathit{blackmove}_{z,i,j}$ and $\forall$ for $\mathit{whitemove}_{z,i,j}$. After those move variables, we use $\exists$ for "state" variables $\mathit{gameover}_{z+1}$, $\mathit{occupied}_{z+1}$, $\mathit{blackwin}_z$, $\mathit{whitewin}_z$, $\mathit{black}_{z,x,y}$, $\mathit{white}_{z,x,y}$, $\mathit{blackcheats}_{z,a,c}$, $\mathit{whitecheats}_{z,a,c}$, $\mathit{blackcheat}_z$, $\mathit{whitecheat}_z$, $\mathit{blackshape}_{z,s,x,y}$ and $\mathit{whiteshape}_{z,s,x,y}$. These variables are used to express the board state after each player has ended his turn.

Note that if we want to determine whether White wins or not, we switch quantifiers for the move variables; $\forall$ for $\mathit{blackmove}_{z,i,j}$ and $\exists$ for $\mathit{whitemove}_{z,i,j}$.

2.3 Clauses

We now describe the body CNF used in our $\mathsf{GTTT}(p, q)$ encoding. Some of the formulas presented below are not conjunctions of clauses, which is only for readability. Converting them into a CNF is easy and does not increase the size of the formula significantly.

There are mainly three differences from the encoding of Connect-4.

(1) All cells are empty at the initial condition in $\mathsf{GTTT}(p, q)$.
(2) A move in $\mathsf{GTTT}(p, q)$ is defined as placing stones on any empty cell on the board, instead of dropping stones in any column.
(3) The winning condition in $\mathsf{GTTT}(p, q)$ is to achieve any of 0, 90, 180, 270 degrees rotation or reflection of the given polyomino.

We also implement the winning condition for torus board as an extension of the game in our encoding.

The encoding for $\mathsf{GTTT}(p,q)$ is the conjunction of the formulas described below. First, we will describe the clauses that are different from the encoding of Connect-4. Note that we only show clauses for Black, because ones for White can be obtained symmetrically.

Initial Condition There is no stone on any cell at the initial condition.

$$\bigwedge_{x=1}^{W}\bigwedge_{y=1}^{H}(\neg \mathit{occupied}_{1,x,y})$$

Move rule There are two modifications for the move rule in $\mathsf{GTTT}(p,q)$ encoding. First, the clauses for the move rule are modified from dropping the stone to the board, into placing the stone on the board. Second, we extend cheat clauses so that the player can place more than one stone for one move in $\mathsf{GTTT}(p,q)$. We also add cheat rules when a player places a stone on an occupied cell.

1. The move in $\mathsf{GTTT}(p,q)$ is defined as placing a stone on an empty cell of the board. Therefore, if a cell is empty then a black stone will be on the cell iff Black places a stone on the cell.

$$\bigwedge_{z=1}^{\lceil \frac{Z}{2} \rceil}\bigwedge_{x=1}^{W}\bigwedge_{y=1}^{H}(\mathit{gameover}_{2z-1} \vee (\neg \mathit{occupied}_{2z-1,x,y} \implies (\mathit{blackmove}_{2z-1,x,y} \iff \mathit{black}_{2z-1,x,y})))$$

2. If a cell is empty then a black stone cannot appear on that cell at the White turn.

$$\bigwedge_{z=1}^{\lfloor \frac{Z}{2} \rfloor}\bigwedge_{x=1}^{W}\bigwedge_{y=1}^{H}(\mathit{gameover}_{2z} \vee (\neg \mathit{occupied}_{2z,x,y} \implies \neg \mathit{black}_{2z,x,y}))$$

3. Let B be the set of all cells in the board, and $\binom{B}{v}$ be the set of all v-combinations of B. Let n_z be the number of stones that Black can place at turn z. Then $M_z = \binom{B}{n_z+1}$ is the collection of n_z+1 cells that are filled by black stones when Black places n_z+1 stones at turn z. Let $f : M_z \to \{1,\ldots,|M_z|\}$ be an injection which maps each $m \in M_z$ to an unique id $f(m)$ of the move rule in category 1.

$$\bigwedge_{z=1}^{\lceil \frac{Z}{2} \rceil}\bigwedge_{m\in M_z}(\mathit{gameover}_{2z-1} \vee (\bigwedge_{b\in m} \mathit{blackmove}_{2z-1,b_x,b_y} \iff \mathit{blackcheats}_{2z-1,1,f(m)}))$$

4. $M'_z = \binom{B}{WH-(n_z-1)}$ is the collection of $WH-(n_z-1)$ cells that are *not* filled by black stones when Black places $n_z - 1$ stones at turn Z. Let $g : M'_z \to \{1, \ldots, |M'_z|\}$ be an injection which maps each $m \in M'_z$ to an unique id $g(m)$ of the move rule in category 2.

$$\bigwedge_{z=1}^{\lceil \frac{Z}{2} \rceil} \bigwedge_{m \in M'_z} (gameover_{2z-1} \vee (\bigwedge_{b \in m} \neg blackmove_{2z-1,b_x,b_y} \iff blackcheats_{2z-1,1,g(m)}))$$

5. Black does illegal moves of category 3 iff Black places a stone on an occupied cell, where the function $h(x,y) = W{\cdot}(y-1)+x$ maps each coordinates (x,y) to an unique id of the move rule in category 3.

$$\bigwedge_{z=1}^{\lceil \frac{Z}{2} \rceil} \bigwedge_{x=1}^{W} \bigwedge_{y=1}^{H} (gameover_{2z-1} \vee ((occupied_{2z-1,x,y} \wedge blackmove_{2z-1,x,y}) \iff blackcheats_{2z-1,3,h(x,y)}))$$

6. Black has cheated iff he did one of the illegal moves.

$$(gameover_{2z-1} \vee ((\bigvee_{a=1}^{4} \bigvee_{c=1}^{C_a} blackcheats_{2z-1,a,c}) \iff blackcheat_{2z-1}))$$

Winning condition For a given polyomino, we calculate all rotations and reflections of the polyomino and then use them as target shapes for winning condition. Not only a polyomino, but any shape can also be used in this encoding, such as straight line in connect(m,n,k,p,q) [29] or wild polyomino [24]. Furthermore, we also implement winning condition for torus board in our encoding.

1. Black achieves a target shape iff there are black stones on the board that form the target shape.

$$\bigwedge_{z=1}^{Z} \bigwedge_{i=1}^{|S|} \bigwedge_{x=1}^{W-|s_i|_W+1} \bigwedge_{y=1}^{H-|s_i|_H+1} (gameover_z \vee ((\bigwedge_{j=1}^{k} black_{z,x+s_i^{x_j},y+s_i^{y_j}}) \iff blackshape_{z,i,x,y}))$$

For torus board, let $mod(u,v) = ((u-1) \mod v)+1$. We connect leftmost cells with rightmost cells and top cells with bottom cells when we check whether or not a target shape is achieved by Black.

$$\bigwedge_{z=1}^{Z} \bigwedge_{i=1}^{|S|} \bigwedge_{x=1}^{W} \bigwedge_{y=1}^{H} (gameover_z \vee ((\bigwedge_{j=1}^{k} black_{z,mod(x+s_i^{x_j},W),mod(y+s_i^{y_j},H)}) \iff blackshape_{z,i,x,y}))$$

2. If Black has not cheated, then Black wins the game at turn z iff he achieves a target shape or White cheats. We use $1 \leq x \leq W$, $1 \leq y \leq H$ for torus board.

$$\bigwedge_{z=1}^{Z} (\mathit{gameover}_z \vee (\neg \mathit{blackcheats}_z \implies$$
$$((\bigvee_{i=1}^{S} \bigvee_{x=1}^{W-|s_i|_W+1} \bigvee_{y=1}^{H-|s_i|_H+1} \mathit{blackshape}_{z,i,x,y} \vee \mathit{whitecheats}_z) \iff \mathit{blackwin}_z)))$$

3. Black cannot win if he has cheated, and he cannot cheat at White turn.

$$\bigwedge_{z=1}^{Z} (\mathit{gameover}_z \vee (\mathit{blackcheats}_z \implies \neg \mathit{blackwin}_z)) \wedge \bigwedge_{z=1}^{\lfloor \frac{Z}{2} \rfloor} (\neg \mathit{blackcheats}_{2z})$$

4. Black wins if he wins the game at any turn.

$$(\bigvee_{z=1}^{Z} (\neg \mathit{gameover}_z \wedge \mathit{blackwin}_z) \iff \mathit{blackwin})$$

Symmetry breaking We use three types of clauses for symmetry breaking, depending on the type of the board. The first clauses are symmetry breaking for the general board, the second clauses for square board ($W = H$), and the third clauses for torus board.

1. On general board, Black must place his first stone on the top left side rectangle of the board.

$$(\mathit{gameover}_1 \vee (\bigwedge_{x=1}^{\lceil \frac{W}{2} \rceil} \bigwedge_{y=1}^{\lceil \frac{H}{2} \rceil} \neg \mathit{blackmove}_{1,x,y} \iff \mathit{blackcheat}_{1,4,1}))$$

On square board, Black must place his first stone on the top left side triangle of the board.

$$(\mathit{gameover}_1 \vee (\bigwedge_{x=1}^{\lceil \frac{W}{2} \rceil} \bigwedge_{y=x}^{\lceil \frac{H}{2} \rceil} \neg \mathit{blackmove}_{1,x,y} \iff \mathit{blackcheat}_{1,4,1}))$$

On torus board, Black must place his first stone on the center of the board.

$$(\mathit{gameover}_1 \vee (\neg \mathit{blackmove}_{1,\lceil \frac{W}{2} \rceil,\lceil \frac{H}{2} \rceil} \iff \mathit{blackcheat}_{1,4,1}))$$

Next, we will describe the clauses of the $\mathsf{GTTT}(p, q)$ encoding that are the same as the Connect-4 encoding.

Board management We manage the cells of the board by the following clauses.

1. Any cell of the board cannot be occupied by both black stone and white stone simultaneously.
$$\bigwedge_{z=1}^{Z}\bigwedge_{x=1}^{W}\bigwedge_{y=1}^{H}(\mathit{gameover}_z \lor \neg \mathit{black}_{z,x,y} \lor \neg \mathit{white}_{z,x,y})$$
2. Any cell of the board is occupied if either a black stone or a white stone is placed at the previous turn.
$$\bigwedge_{z=1}^{Z}\bigwedge_{x=1}^{W}\bigwedge_{y=1}^{H}(\mathit{gameover}_z \lor ((\mathit{black}_{z,x,y} \lor \mathit{white}_{z,x,y}) \iff \mathit{occupied}_{z+1,x,y}))$$
3. If a cell is occupied by a black stone, then the cell is also occupied by black stone at the next turn.
$$\bigwedge_{z=1}^{Z-1}\bigwedge_{x=1}^{W}\bigwedge_{y=1}^{H}(\mathit{gameover}_z \lor (\mathit{black}_{z,x,y} \implies \mathit{black}_{z+1,x,y}))$$

Others

1. For any turn z, if the game has been already over, then the game is also over at the next turn. Also if the game is not over, then the game is over at next turn iff Black or White wins at this turn.
$$\bigwedge_{z=1}^{Z-1}(\mathit{gameover}_z \implies \mathit{gameover}_{z+1})$$
$$\land \bigwedge_{z=1}^{Z-1}(\neg\mathit{gameover}_z \implies ((\mathit{blackwin}_z \lor \mathit{whitewin}_z) \iff \mathit{gameover}_{z+1}))$$
2. Black cannot win after the game has over.
$$\bigwedge_{z=1}^{Z-1}(\mathit{gameover}_z \implies \neg\mathit{blackwin}_z)$$
3. We set *blackwin* as a clause if we want to check whether Black can win the game or not. Exactly one of *blackwin*, *whitewin*, and *draw* must be true. The game finishes draw iff the game is not over at turn $Z + 1$.
$$\begin{aligned}&(\mathit{blackwin})\\ &\land(\neg\mathit{gameover}_1)\\ &\land(\mathit{blackwin} \lor \mathit{whitewin} \lor \mathit{draw})\\ &\land(\neg\mathit{blackwin} \lor \neg\mathit{whitewin}) \land (\neg\mathit{blackwin} \lor \neg\mathit{draw}) \land (\neg\mathit{whitewin} \lor \neg\mathit{draw})\\ &\land(\neg\mathit{gameover}_{Z+1} \iff \mathit{draw})\end{aligned}$$

Table 1. The statistics of the instances in $\mathsf{GTTT}(p,q)$: average number of variables (vars*), average number of clauses (clauses*), the number of quantifier alternations (alt), and average numbers of existentials (∃*) and universals (∀*).

Board size	p	q	vars*	clauses*	alt.	∃*	∀*
3×3	1	1	1591	4694	8	1547	40
	2	1	1339	4028	4	1316	23
	2	2	1164	3507	3	1146	18
4×4	1	1	6748	18478	15	6620	128
	2	1	9753	27254	7	9689	64
	2	2	10663	29828	7	10599	64

Table 2. Number of solved instances by each solver within 10 seconds on 3×3 board and 1000 seconds on 4×4 board.

Board size	Result	Without bloqqer				With bloqqer			
		depqbf	ghostq	rareqs	mpidepqbf	depqbf	ghostq	rareqs	mpidepqbf
3×3	SAT	24	24	18	22	24	23	24	17
	UNSAT	60	60	39	58	60	60	60	52
	Total	84	84	57	80	84	83	84	79
4×4	SAT	28	23	2	13	28	15	11	29
	UNSAT	53	43	5	14	50	28	24	57
	Total	81	66	7	27	78	43	35	86

3 Experiments

We conducted experiments by solving $\mathsf{GTTT}(p,q)$ by using QBF solvers. We generated 84 ($= 3 \times 7 \times 2 \times 2$) instances from $\mathsf{GTTT}(p,q)$ on 3×3 board, which is the combination of the following parameters. The values of p and q for $1 \leq q \leq p \leq 2$, 7 polyominoes of 2–4 cells aside Skinny (see Fig. 2), 2 types of board; regular and torus board, and instances for Black and White. We also generated 96 ($= 3 \times 8 \times 2 \times 2$) instances from $\mathsf{GTTT}(p,q)$ on 4×4, with the same parameters as the 3×3 board, but we used all 2–4 cell polyominoes including Skinny. Table 1 shows some statistics of the instances that are used in the experiments.

We used a computer with Intel Xeon CPU E5-2609 8 cores 2.40GHz, 256GB memory, and Debian Wheezy OS as experimental environment. We used DepQBF [18], RAReQS [12], GhostQ [16], and MPIDepQBF [13] as solvers and bloqqer [2] for preprocessing. We use 8 cores when running the MPIDepQBF.

Fig. 4 and Fig. 5 show the running time of each solver on solving the instances of $\mathsf{GTTT}(p,q)$ on 3×3 and 4×4 boards, respectively. Notice that the point on 10 seconds on Fig. 4 (resp. 1000 seconds on Fig. 5) means that the solver solves the instances in 10 (resp. 1000) seconds or more. From Fig. 4 we can see that DepQBF solves the instances of $\mathsf{GTTT}(p,q)$ on 3×3 board faster than the other solvers. We can also see that preprocessing is not effective for DepQBF and

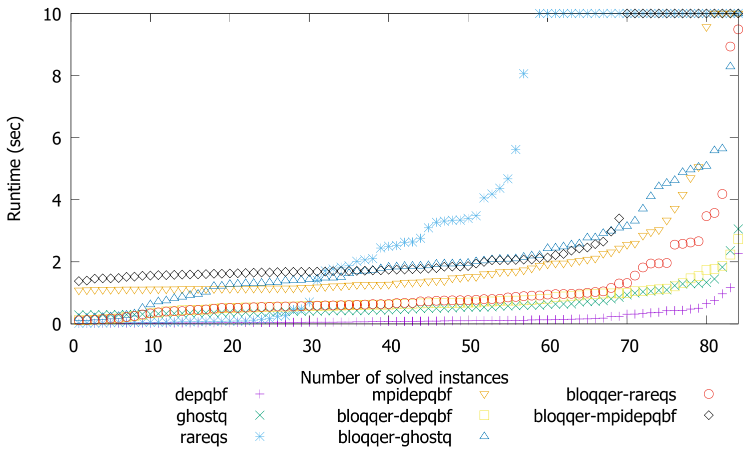

Fig. 4. Running time of QBF solvers on QBF instances of $\mathsf{GTTT}(p, q)$ on 3×3 board.

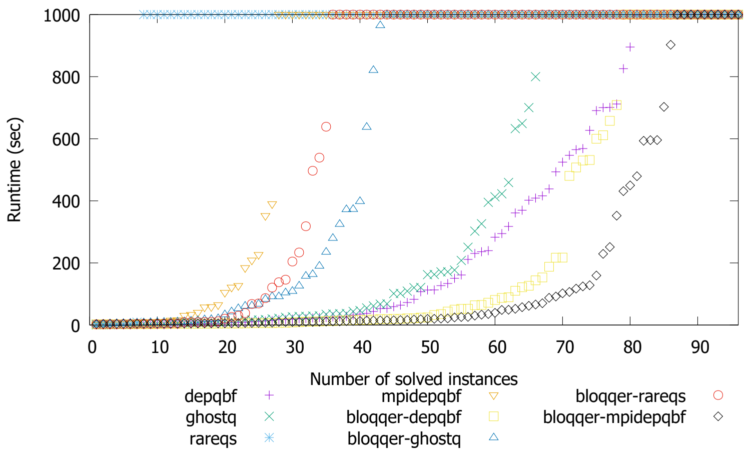

Fig. 5. Running time of QBF solvers on QBF instances of $\mathsf{GTTT}(p, q)$ on 4×4 board.

GhostQ from this result. On the other hand, from Fig. 5, MPIDepQBF with preprocessing is the fastest solver on solving the instances of $\mathsf{GTTT}(p, q)$ on 4×4 board. The preprocessing by using bloqqer is effective on DepQBF and MPIDepQBF in this case. The detailed number of instances and its result (SAT or UNSAT) are shown in Table. 2.

From these experimental results, we can conclude that the QBF solvers can solve most instances of $\mathsf{GTTT}(p, q)$ on 3×3 and 4×4 within 1000 seconds. Different to the results in [11, 21], DepQBF and MPIDepQBF can solve most of the instances faster than other solvers. This indicates that the QBF solving algorithm of DepQBF is more suitable to solve $\mathsf{GTTT}(p, q)$ encoding than other solvers. We can also see that parallelization is more effective when solving $\mathsf{GTTT}(p,q)$ encoding on 4×4 board. However, a better encoding and faster solver are needed in order to solve the instances of $\mathsf{GTTT}(p,q)$ on larger board and polyomino in less than 1000 seconds. Therefore, improving the encoding and the solvers are needed in order to solve Snaky on HTTT by using QBF solver.

4 Conclusion

We have proposed a QBF encoding of an extension $\mathsf{GTTT}(p,q)$ of Harary's generalized Tic-Tac-Toe. However, by our encoding, existing QBF solvers could not solve the open problem on HTTT, whether the first player has a winning strategy to achieve a Snaky on the 9×9 board, that has 616578 variables, 1557653 clauses, and 80 quantifier alternations in its instance, which are much bigger than the instances on 4×4 board. Our encoding technique is rather naive and easily understandable, so there remains a lot of room to improve. For example, our encoding contains many redundant clauses. Some of them would help QBF solvers search, but we did not examine which of them have a real positive effect. Moreover, although our encoding did not consider an incremental QBF solving [20], the encoding might be solved in incremental way because we can use the knowledge from the result on solving an instance of smaller board to solve an instance of larger board. Therefore, the encoding optimization for incremental QBF solving can be considered as a future work. We hope our experimental results and analysis help to improve QBF solvers and the open problem will be solved by a QBF approach by an even more elaborated QBF solver with a more sophisticated encoding in the future.

Acknowledgment

This work is supported by Tohoku University Divison For Interdisciplinary Advance Research and Education. We thank Charles Jordan for his advises on writing this paper.

References

1. Allis, L., van der Meulen, M., van den Herik, H.J.: Proof-number search. Artificial Intelligence **66**(1) (1994) 91–124
2. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: CADE'11. (2011) 101–115
3. Diptarama, Ishiguro, Y., Narisawa, K., Shinohara, A., Jordan, C.: Solving generalized tic-tac-toe by using QBF solver. In: Proceedings of Game Programming Workshop 2015. Volume 2015. (oct 2015) 154–161 (In Japanese).
4. Diptarama, Narisawa, K., Shinohara, A.: Extension of generalized tic-tac-toe: p stones for one move. IPSJ Journal **55**(11) (nov 2014) 2344–2352 (In Japanese).
5. Diptarama, Narisawa, K., Shinohara, A.: Drawing strategies for generalized tic-tac-toe (p, q). AIP Conference Proceedings **1705** (2016)

6. Gent, I.P., Rowley, A.: Encoding connect-4 using quantified boolean formulae. 2nd Intl. Work. Modelling and Reform. CSP (2003) 78–93
7. Halupczok, I., Schlage-Puchta, J.C.: Achieving snaky. Electronic Journal of Combinatorial Number Theory **7** (2007) G02
8. Harary, F.: Achievement and avoidance games for graphs. Ann. Discrete Math **13** (1982) 111–120
9. Harary, F.: Achieving the skinny animal. Eureka **42** (1982) 8–14
10. Ito, H., Miyagawa, H.: Snaky is a winner with one handicap. In: Proceedings of 8th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA 2007). (2007) 25–26
11. Janota, M., Jordan, C., Klieber, W., Lonsing, F., Seidl, M., Van Gelder, A.: The QBFGallery 2014: The QBF competition at the FLoC olympic games. Journal on Satisfiability, Boolean Modeling and Computation **9** (2016) 187–206
12. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.: Solving QBF with counterexample guided refinement. In: SAT'12. (2012) 114–128
13. Jordan, C., Kaiser, L., Lonsing, F., Seidl, M.: MPIDepQBF: Towards parallel qbf solving without knowledge sharing. In: Theory and Applications of Satisfiability Testing–SAT 2014. Springer (2014) 430–437
14. Jordan, C., Seidl, M.: QBF gallery 2014 (2014)
15. Kaiser, L., Stafiniak, L.: Playing structure rewriting games. In: AGI'10. (2010)
16. Klieber, W., Sapra, S., Gao, S., Clarke, E.: A non-prenex, non-clausal QBF solver with game-state learning. In: Theory and Applications of Satisfiability Testing–SAT 2010. Springer (2010) 128–142
17. Konev, B., Lisitsa, A.: A SAT attack on the Erdős discrepancy conjecture. In: Theory and Applications of Satisfiability Testing–SAT 2014. Springer (2014) 219–226
18. Lonsing, F., Bacchus, F., Biere, A., Egly, U., Seidl, M.: Enhancing search-based QBF solving by dynamic blocked clause elimination. In: Logic for Programming, Artificial Intelligence, and Reasoning, Springer (2015) 418–433
19. Lonsing, F., Biere, A.: DepQBF: A dependency-aware QBF solver. Journal on Satisfiability, Boolean Modeling and Computation **7** (2010) 71–76
20. Lonsing, F., Egly, U.: Incremental QBF solving. In: Principles and Practice of Constraint Programming-CP 2014. Springer (2014) 514–530
21. Lonsing, F., Seidl, M., Van Gelder, A.: The QBF gallery: Behind the scenes. arXiv preprint arXiv:1508.01045 (2015)
22. Mironov, I., Zhang, L.: Applications of SAT solvers to cryptanalysis of hash functions. In: Theory and Applications of Satisfiability Testing-SAT 2006. Springer (2006) 102–115
23. Reisch, S.: Gobang ist PSPACE-vollständig. Acta Inf. **13** (1980) 59–66
24. Sieben, N.: Wild polyomino weak $(1, 2)$-achievement games. Geombinatorics **13**(4) (2004) 180–185
25. Sieben, N.: Polyominoes with minimum site-perimeter and full set achievement games. European Journal of Combinatorics **29**(1) (2008) 108–117
26. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Theory and Applications of Satisfiability Testing-SAT 2009. Springer (2009) 244–257
27. Toss Team: Toss `http://toss.sourceforge.net/`.
28. Uiterwijk, J.W.H.M., van den Herik, H.J.: The advantage of the initiative. Inf. Sci. **122**(1) (2000) 43–58
29. Wu, I.C., Huang, D.Y.: A new family of k-in-a-row games. In: Proceedings of Advances in Computer Games. (2006) 180–194