2^3 QUANTIFIED BOOLEAN FORMULA GAMES AND THEIR COMPLEXITIES

Kyle Burke
Department of Computer Science and Technology, Plymouth State University, Plymouth, New Hampshire
kgburke@plymouth.edu

Received: 1/16/14, Revised: 5/7/15, Accepted: 5/22/15, Published: 6/15/15

Abstract

Consider QBF, the Quantified Boolean Formula problem, as a combinatorial game ruleset. The problem is rephrased as determining the winner of the game where two opposing players take turns assigning values to Boolean variables. In this paper, three variations of games are applied to create seven new rulesets: whether each player is restricted to where they may play, which values they may set variables to, or whether conditions they are shooting for at the end of the game differ. The complexity for determining which player can win is analyzed for all games. Of the seven, two are trivially in P and the other five are PSPACE-complete. Two of these hard games are impartial, (the only known impartial formula rulesets incorporating unassigned variables), and two are hard for 2-CNF formulas.

1. Introduction

1.1. Combinatorial Game Theory

Two-player games with alternating turns, perfect information, and no random elements are known as *combinatorial games*. Combinatorial Game Theory determines which of the players has a winning move from any *position* (game state). Many of the elegant aspects of CGT, such as game sums, are ignored here. The interested reader is encouraged to browse [4] and [1].

1.2. Quantified 3SAT

In computational complexity, the quantified Boolean satisfiability problem, QBF, consists of determining whether formulas of the following form are true. $\exists x_0 : \forall x_1 : \exists x_2 : \forall x_3 : \cdots Q_{n-1} x_{n-1} : f(x_0, x_1, x_2, \ldots, x_{n-1})$, where f is a Boolean formula and $Q_i = \exists$ if i is even, and $\forall$ when i is odd. QBF is commonly considered the fundamental problem for the complexity class PSPACE, the set of true-false

problems that can be solved using workspace polynomial in the size of the input. In other words, PSPACE is the set of problems that can be efficiently reduced to QBF.

Each QBF instance can be reconsidered as a combinatorial game between two players: Even/True and Odd/False. The initial position consists of a list of the indexed literals and the unquantified formula, f. Even/True moves first, choosing a value for x_0. Odd/False goes next, choosing a value for x_1. The players continue taking turns in this manner until all variables have been assigned. At this point, the value of f is determined. If f is true, Even/True wins, otherwise Odd/False wins.

1.2.1. Sample QBF Game

For example, consider the position with formula: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$ and no assignments. Then the following would be a legal sequence of turns:

- Even/True chooses True ($\mathbf{T}$) for x_0. The players are likely keeping track of the assignments and updating the formula:
  $(\mathbf{F} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee \mathbf{T}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= (x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Next, Odd/False chooses $\mathbf{T}$ for x_1. Formula:
  $(x_3 \vee \mathbf{F}) \wedge (x_2 \vee \mathbf{T} \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= x_3 \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Even/True (not feeling very confident at this point), chooses $\mathbf{F}$ for x_2:
  $x_3 \wedge (\mathbf{T} \vee \overline{x_4} \vee x_3)$
  $= x_3$
- Odd/False triumphantly chooses $\mathbf{F}$ for x_3. With the assignments, the evaluated formula will be False in the end. Despite this, the players may continue taking their turns:
- Even/True chooses $\mathbf{T}$ for x_4.
- Odd/False chooses $\mathbf{F}$ for x_5. (Notice that no instances of x_5 exist in the formula.)
- Even/True chooses $\mathbf{F}$ for x_6.
- The assignments cause the formula to be false; Odd/False wins.

1.3. Algorithmic Combinatorial Game Theory

Notice that determining, from the initial position, whether the Even/True player has a winning strategy is exactly the same problem as determining whether the QBF instance is true. Due to this equivalence, we will abuse notation slightly and refer to both this ruleset and the computational problem as QBF. A position in this ruleset is the unquantified formula, f, and the list of n indexed Boolean variables, with assignments for the first $k-1$ ($0 \leq k \leq n$).

The lack of distinction between computational problems and combinatorial rulesets is not specific to QBF. It is common to use a ruleset's name to refer to the computational complexity of a ruleset by the induced problem of determining which player can win. For example, we say that NIM is in P. (To be even more specific, NIM is in $O(n)$ where n is the number of heaps of sticks.) The study of algorithms and computational complexity to determine the winner is known as algorithmic combinatorial game theory [5]. This paper covers the computational hardness for new games based on formula satisfiability.

1.4. Another SAT Game: Positive CNF

Many other Boolean formula satisfiability games have been defined [6]. We go into more detail for one of them, G_{POS}(POS CNF)—here referred to as POSITIVE CNF—as it will be used in later proofs. Other satisfiability games from [6] are highly recommended to the interested reader.

Definition 1 (Positive CNF). POSITIVE CNF is the ruleset for games played on a Boolean 3-CNF formula, f, using n variables without including any negations. The players are indicated True and False. Each turn, the current player chooses any one unassigned variable and assigns it the value corresponding to their name. When all variables are assigned, True wins if the value of the formula is true, otherwise False wins.

POSITIVE CNF is known to be PSPACE-complete [6], which will make two of the five reductions (Sections 5 and 8) very simple.

2. Three Ruleset Toggles

Some PSPACE-hard combinatorial games are difficult to reduce to from QBF. These target games often have properties—such as being able to play anywhere on the "board"—very dissimilar from QBF. There are three common properties of games that we vary to modify QBF. We refer to each property as a ruleset *toggle*, since each has exactly two possible values.

2.1. Play Location

In QBF, on the i^{th} turn, the current player assigns a value to variable x_i. Many combinatorial games are more flexible, allowing the next player to play wherever on the board they would like. We can model that by allowing a variant of QBF where the current player may choose to play at any unassigned variable. We refer to this toggle as *locality* with possible values *local* and *anywhere*.

- **local**: The current player assigns a value to the unassigned variable, x_i, with lowest index.
- **anywhere**: The current player assigns a value to any one of the unassigned variables.

2.2. Boolean Choice

In QBF, each player chooses to set a variable either true or false. However, in rulesets such as SNORT, a player is identified by their color (either Red or Blue) and may only play pieces of that color. We model this with the toggle *Boolean identity*, with possible values *either* and *by player*.

- **either**: The current player assigns either true or false to a variable.
- **by player**: One player only assigns variables to true, the other player only assigns false.

2.3. Goal

In QBF, one player is trying to force the formula to have the value true, while the other one is vying for false. With impartial games, both players have the same goal; the one who makes the last move to reach that goal wins. We model that with the *goal* toggle, with values *different* and *same*. When the goal is the same, players avoid creating a formulas that are casually false (defined below). Unfortunately, we cannot simply use contradictions (or tautologies) as the forbidden formulas.

2.3.1. Contradictions are not Enough

There are two reasons it is impractical to enforce avoidance of contradictions:

- Determining whether a Boolean formula is satisfiable is NP-hard. Thus, it would be computationally difficult to determine when the formula is a contradiction and the game has already ended. Enforcing this would be extra onerous.

- If contradictions are not legal positions, then each position's formula must be satisfiable. That means all unset variables can be used, and the number of remaining turns is simply the number of unset variables. Determining who can win from a legal position is trivial.

Instead, a lazy form of evaluation is used to avoid only "obvious" contradictions. This allows for the first impartial Boolean formula games that use unassigned variables and are playable on all first-order formulas.

2.3.2. Casual Evaluation

To solve the problems, the evaluation scheme must be able to handle formulas with unassigned variables without trying all possible assignments. When x_0 is assigned to $false$, both $(x_0 \wedge x_1)$ and $(x_1 \wedge \overline{x_1})$ are contradictions. This scheme, however, will only detect a contradiction in the first case. The second case is detectable by trying both possibilities for x_1, but no such case analysis is used here. We refer to this as *casual evaluation* and define it for the *not*, *or*, and *and* operators:

- **Variables**: variables can be *true*, *false*, or *unassigned*.
- **Expressions**: expressions can be a *tautology*, *contradiction*, or *unknown*.
- **Literals**: true literals are tautologies, false literals are contradictions, and unassigned literals are unknown.
- **Fully-Known Operators**: Operations where none of the inputs are unknown act exactly as normal.
- **Not**: The negation of an unknown expression is also unknown.
- **And**: A conjunction with at least one unknown input is a contradiction if any of those inputs is a contradiction. If none of the inputs are contradictions, it is unknown.
- **Or**: A disjunction with at least one unknown input is a tautology if any of those inputs is a tautology. If none of the inputs are tautologies, it is unknown.

As mentioned before, $(contradiction \wedge x_5)$ casually evaluates to a contradiction, while $((x_4 \wedge \overline{x_4}) \wedge x_5)$ remains unknown. It becomes natural to refer to formulas that casually evaluate to true and false as *casually true* and *casually false*, respectively.

2.3.3. Goals: Different or Same

With this notion, we can define the two possible values for the goal toggle:

- **different**: One player is trying to set the formula to true, the other to false. After all variables are assigned, the formula is evaluated. Whichever player has reached their target value wins.

- **same**: Players are not allowed to assign a variable such that the resulting formula casually evalutes to false. If a player cannot move, they lose the game—the usual end-of-game condition for combinatorial games. This means that there are no legal moves on the formula: $x_0 \wedge \overline{x_0}$; the first player would automatically lose. (If the formula is a tautology at the end of the game, the player to choose the value of the last unassigned variable wins.)

2.4. QBF Categorized

QBF, then, is the EITHER-LOCAL-DIFFERENT ruleset: players have to play on the next variable, are allowed to set the value to either true or false, and one is shooting to make the formula true, while the other tries to make it false. There are seven total other rulesets generated by changing these variables. The following sections cover each of these and analyze their computational complexity. These results are summarized in table 1. For some rulesets, different minimum numbers of literals per clause are needed for formulas that are known to be hard. For example, for EITHER-LOCAL-SAME, hard instances are only known for formulas in Conjunctive Normal Form with at least four literals per clause. (DNF stands for Disjunctive Normal Form.)

Boolean choice	play location	goal	section	computational complexity	hard instances
by player	local	same	3	In P	never
by player	local	different	3	In P	never
by player	anywhere	same	4	PSPACE-complete	2-CNF
by player	anywhere	different	5	PSPACE-complete	11-CNF 11-DNF
either	local	same	6	PSPACE-complete	4-CNF
either	local	different	none	PSPACE-complete	3-CNF
either	anywhere	same	7	PSPACE-complete	2-CNF
either	anywhere	different	8	PSPACE-complete	11-CNF 11-DNF

Table 1: QBF Ruleset Complexities

3. By-Player-Local-X Rulesets

Two of the rulesets generated are trivial for determining the winner: BY-PLAYER-LOCAL-SAME and BY-PLAYER-LOCAL-DIFFERENT. In the games where players have neither a choice of the Boolean value nor the location to play, there is only one possible sequence of moves. To figure out which player will win, a program needs only simulate the moves, adhering to the goal condition.

For example, consider the BY-PLAYER-LOCAL-SAME initial position with the formula described in Section 1.2.1: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$. Then the following would be the sequence of turns:

3.1. Sample Game

- Even/True must assign $\mathbf{T}$ to x_0. We keep track by updating the formula:
  $(\mathbf{F} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee \mathbf{T}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= (x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Next, Odd/False assigns $\mathbf{F}$ to x_1. Formula:
  $(x_3 \vee \mathbf{T}) \wedge (x_2 \vee \mathbf{F} \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= (x_2 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Even/True assigns $\mathbf{T}$ to x_2:
  $(\mathbf{T} \vee \overline{x_6}) \wedge (\mathbf{F} \vee \overline{x_4} \vee x_3)$
  $= \overline{x_4} \vee x_3$
- Odd/False assigns $\mathbf{F}$ to x_3:
  $\overline{x_4} \vee \mathbf{F}$
  $= \overline{x_4}$
- Even/True now cannot make a move. (Their only normal option: assigning $\mathbf{T}$ to x_4, would make the formula casually evaluate to false, so they cannot move to that position.) Odd/False wins!

In BY-PLAYER-LOCAL-DIFFERENT with this same formula, Odd/False also wins because the formula evaluates to false by the end.

4. By-Player-Anywhere-Same

The BY-PLAYER-ANYWHERE-SAME QBF ruleset consists of the games with two players, True and False, each avoiding creating a casually false formula, while allowed to play anywhere on the board.

4.1. Sample Game

For example, consider the initial position with the same formula given in Section 1.2.1: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$. The following is a legal sequence of plays from the initial position:

- True chooses x_3. The partially-evaluated formula now looks like:
  $(\overline{x_0} \vee \mathbf{T} \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee \mathbf{T})$
  $= (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
- False chooses x_1. (Notice that choosing x_6 would cost False the game since there would be an odd number of moves remaining.)
  $(x_2 \vee \mathbf{F} \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
  $= (x_2 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
- True chooses x_2:
  $(\mathbf{T} \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
  $= x_4 \vee \overline{x_6} \vee x_0$
- False chooses x_4:
  $\mathbf{F} \vee \overline{x_6} \vee x_0$
  $\overline{x_6} \vee x_0$
- True chooses x_0:
  $\overline{x_6} \vee \mathbf{T}$
  $= \mathbf{T}$

  The formula now always evaluates to True. (There are an even number of moves left, so True wins.)

4.2. PSPACE-completeness

Theorem 1 (by-player-anywhere-same PSPACE-completeness). *Ruleset* BY-PLAYER-ANYWHERE-SAME *is* PSPACE*-complete on 2-CNF.*

Proof. To show hardness, we reduce from the well-known PSPACE-complete ruleset SNORT [6]. A SNORT position consists of a graph, with some vertices painted blue, some red, and the rest uncolored. The two players, Blue and Red, take turns choosing an uncolored vertex and painting it their own color. Players are not allowed to paint vertices adjacent to the opposite color. The first player who cannot play loses.

The reduction is as follows. Let $G = (V, E)$ be the SNORT graph and let $V = \{0, \ldots, n-1\}$. The set of literals for our formula is $x_0, x_1, \ldots, x_{n-1}$. Blue corresponds to True and Red corresponds to False. For any edge, $(i, j) \in E$, we include two clauses: $(x_i \vee \overline{x_j}) \wedge (\overline{x_i} \vee x_j)$. In the case where any of i and j are painted, the resulting formula is casually false only when they have opposing colors.

Thus, the only moves a player is not allowed to make correspond exactly to illegal moves in SNORT.

To get the overall formula, we conjoin all pieces together: $\bigwedge_{(i,j)\in E} (x_i \vee \overline{x_j}) \wedge (\overline{x_i} \vee x_j)$

Since the winnability of the two games are equal, the BY-PLAYER-ANYWHERE-SAME ruleset is PSPACE-hard. Moreover, these formulas are in conjunctive normal form using only two literals per clause, so hard games are in 2-CNF. □

5. By-Player-Anywhere-Different

The BY-PLAYER-ANYWHERE-DIFFERENT ruleset consists of games between two players, True and False, who may choose to play on any unassigned variable on their turn. Each player wins if the value of the formula matches their identity.

5.1. Sample Game

Consider the initial position with the same formula given in Section 1.2.1: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$. The following is a legal sequence of plays from the initial position:

- True chooses x_3:
  $(\overline{x_0} \vee \mathbf{T} \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee \mathbf{T})$
  $= (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
- False chooses x_2:
  $= (\mathbf{F} \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
  $= (x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
- True chooses x_1:
  $= (\mathbf{T} \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
  $= (x_4 \vee \overline{x_6} \vee x_0)$
- False chooses x_4:
  $= \mathbf{F} \vee \overline{x_6} \vee x_0$
  $= \overline{x_6} \vee x_0$
- True chooses x_0:
  $= \overline{x_6} \vee \mathbf{T}$
  $= \mathbf{T}$
- No matter which order x_5 and x_6 are chosen, True has won the game.

5.2. PSPACE-completeness

Theorem 2 (by-player-anywhere-different PSPACE-completeness). *Ruleset* BY-PLAYER-ANYWHERE-DIFFERENT *is* PSPACE*-complete on 11-CNF and 11-DNF.*

Proof. To show hardness, we reduce from POSITIVE CNF (see Section 1.4).

The rules for POSITIVE CNF are exactly the same as BY-PLAYER-ANYWHERE-DIFFERENT, except that not all formulas are allowed. Thus, each instance of POSITIVE CNF is also an instance of BY-PLAYER-ANYWHERE-DIFFERENT, meaning the new game is "automatically" PSPACE-hard.

The same reduction follows from the analagous POSITIVE DNF. Both of these rulesets are known to be hard on formulas with at least 11 literals per clause (11-CNF and 11-DNF)[6], so the same is true for BY-PLAYER-ANYWHERE-DIFFERENT. □

6. Either-Local-Same

The EITHER-LOCAL-SAME ruleset consists of the games between two players forced to play on a specific literal each turn while avoiding creating a casually false formula.

6.1. Sample Game

As an example, consider the initial position with the same formula given in Section 1.2.1: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$. The following is a legal sequence of plays from the initial position:

- Even assigns **F** to x_0:
  $(\mathbf{T} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee \mathbf{F}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Odd assigns **F** to x_1:
  $(x_2 \vee \mathbf{F} \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= (x_2 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Even assigns **T** to x_2:
  $(\mathbf{T} \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6}) \wedge (\mathbf{F} \vee \overline{x_4} \vee x_3)$
  $= (x_4 \vee \overline{x_6}) \wedge (\overline{x_4} \vee x_3)$
- Odd assigns **F** to x_3:
  $(x_4 \vee \overline{x_6}) \wedge (\overline{x_4} \vee \mathbf{F})$
  $= (x_4 \vee \overline{x_6}) \wedge \overline{x_4}$
- Even assigns **F** to x_4:
  $(\mathbf{F} \vee \overline{x_6}) \wedge \mathbf{T}$
  $\overline{x_6}$
- Odd assigns **T** to x_5. The formula does not change.
- Even assigns **F** to x_6. The formula will now always evaluate to True. There are no more variables, so Even wins!

6.2. PSPACE-completeness

Theorem 3 (either-local-same PSPACE-completeness). *Ruleset* EITHER-LOCAL-SAME *is* PSPACE*-complete on 4-CNF.*

Proof. To show hardness, we reduce from QBF.

Assume our QBF formula is written in conjunctive normal form (this subset of positions is still PSPACE-complete [2]) using n variables $x_0, \ldots, x_{n-1}$. Let $\bigwedge_{i\in[c]} \varphi_i$ be the formula, with clauses $\varphi_0, \ldots, \varphi_{c-1}$.

For each φ_i, we create a new clause, γ_i, in the following way. Let l be the largest index of the literals in φ_i.

Define $\gamma_i = \begin{cases} \varphi_i & , l \text{ is even} \\ (\varphi_i \vee (x_{l+1} \wedge \overline{x_{l+1}})) = (\varphi_i \vee x_{l+1}) \wedge (\varphi_i \vee \overline{x_{l+1}}) & , l \text{ is odd} \end{cases}$. Also, let $m = \begin{cases} n+1 & , n \text{ is even} \\ n+2 & , n \text{ is odd} \end{cases}$. In this way, m will always be odd.

The resulting position for EITHER-LOCAL-SAME consists of a formula with c clauses, $\bigwedge_{i\in[c]} \gamma_i$, and has m variables $x_0, \ldots, x_{m-1}$. In some cases, no literal with index $m-1$ will appear in the formula.

It remains to be shown that the winnability of the EITHER-LOCAL-SAME position is equivalent to the QBF position. Notice that in QBF, if any one clause is casually false, the even/true player loses. The γ-clauses simulate this in the reduced formula: the last assignment to a literal in a clause is always made by the even player. If the clause would become casually false, even cannot move and loses.

Alternatively, in order for odd to lose the game, the variables must have been set so that all clauses are true. To maintain this condition, we just make sure the last variable index is even. Thus, if all variables are successfully set, the even player will win the game after having made the last move. To accomplish this, we enforce that m must be odd, even if this means the literal x_{m-1} does not appear in the reduced formula.

Since the winnability of the two games are equal, the EITHER-LOCAL-SAME ruleset is PSPACE-hard. QBF is hard on 3-CNF formulas[6]. Since our reduction extends some of the clauses by a literal, EITHER-LOCAL-SAME is hard for 4-CNF. □

7. Either-Anywhere-Same

The EITHER-ANYWHERE-SAME ruleset consists of the games between two players who can play either value on any unassigned variable each turn while avoiding creating an casually false formula.

7.1. Sample Game

As an example, consider the initial position with the same formula given in Section 1.2.1: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$. The following is a legal sequence of plays from the initial position:

- Even assigns **F** to x_6:
  $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \mathbf{T}) \wedge (x_4 \vee \mathbf{T} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
  $= (\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$
- Odd assigns **T** to x_2:
  $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (\mathbf{F} \vee \overline{x_4} \vee x_3)$
  $= (\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (\overline{x_4} \vee x_3)$
- Even assigns **T** to x_3:
  $= (\overline{x_0} \vee \mathbf{T} \vee \overline{x_1}) \wedge (\overline{x_4} \vee \mathbf{T})$
  T
  From this point on, the formula will always evaluate to True. With four variables remaining, Even takes the last turn and wins.

7.2. PSPACE-completeness

Theorem 4 (either-anywhere-same PSPACE-completeness). *Ruleset* EITHER-ANYWHERE-SAME *is* PSPACE*-complete on 2-CNF.*

Proof. The reduction here is similar to the reduction from SNORT to BY-PLAYER-ANYWHERE-SAME in Section 4. Instead of SNORT, we'll reduce from PROPER 2-COLORING, a different graph game where players take turns painting uncolored vertices so that no two neighboring vertices have the same color. In this game, both players can choose either color on their turn. PROPER 2-COLORING is impartial and PSPACE-complete [3].

The reduction is as follows. Let $G = (V, E)$ be the PROPER 2-COLORING graph and let $V = \{0, \ldots, n-1\}$. The literals for the formula are: $x_0, x_1, \ldots, x_{n-1}$. Blue corresponds to True and Red corresponds to False. Now, for each edge, $(i, j) \in E$, we include: $(x_i \vee x_j) \wedge (\overline{x_i} \vee \overline{x_j})$. This subformula will only become casually false if both variables are given the same value, corresponding to them being painted the same color in PROPER 2-COLORING.

The overall formula is the conjunction of these pieces: $\bigwedge_{(i,j)\in E} (x_i \vee x_j) \wedge (\overline{x_i} \vee \overline{x_j})$

Since the two games are equivalent, EITHER-ANYWHERE-SAME is PSPACE- complete. Just as with BY-PLAYER-ANYWHERE-SAME, these formulas are all in 2-CNF, thus the game is hard on 2-CNF formulas. □

8. Either-Anywhere-Different

The EITHER-ANYWHERE-DIFFERENT ruleset consists of the games between the two players who can play either value on any unassigned variable each turn. The players have separate goals: the even player is attempting to set the entire formula to true, the odd player is shooting for false.

8.1. Sample Game

As an example, consider the initial position with the same formula given in Section 1.2.1: $(\overline{x_0} \vee x_3 \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee x_3)$. One legal sequence of plays from the initial position could be:

- Even/True assigns $\mathbf{T}$ to x_3:
  $(\overline{x_0} \vee \mathbf{T} \vee \overline{x_1}) \wedge (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0) \wedge (\overline{x_2} \vee \overline{x_4} \vee \mathbf{T})$
  $= (x_2 \vee x_1 \vee \overline{x_6}) \wedge (x_4 \vee \overline{x_6} \vee x_0)$
- Odd/False assigns $\mathbf{T}$ to x_6:
  $(x_2 \vee x_1 \vee \mathbf{F}) \wedge (x_4 \vee \mathbf{F} \vee x_0)$
  $= (x_2 \vee x_1) \wedge (x_4 \vee x_0)$
- Even/True confidently assigns $\mathbf{T}$ to x_4:
  $(x_2 \vee x_1) \wedge (\mathbf{T} \vee x_0)$
  $= x_2 \vee x_1$
- Odd/False assigns $\mathbf{F}$ to x_1:
  $x_2 \vee \mathbf{F}$
  $= x_2$
- Even/True assigns $\mathbf{T}$ to x_2 and wins.

8.2. PSPACE-completeness

Although the reduction for this completeness proof is quite simple, we first introduce an intermediate ruleset for clarity:

Definition 2 (toy positive cnf). TOY POSITIVE CNF is exactly the same as POSITIVE CNF, except that each player may choose to assign either True or False to the variable on their turn.

Lemma 1 (toy positive cnf). TOY POSITIVE CNF *is* PSPACE*-complete.*

To prove this, we reduce from POSITIVE CNF (see Section 1.4).

Proof. To begin this proof, we notice that it never improves a player's strategy to choose to assign a variable with the value opposite their identity. Since no negations

exist in the formula, f, the True player never benefits by assigning False and the False player never benefits by choosing True.

Thus, any winning strategy cooresponds directly to a winning strategy in POSITIVE CNF. Our reduction is trivial; no transformation is needed to reduce from POSITIVE CNF to TOY POSITIVE CNF. □

Theorem 5 (either-anywhere-different PSPACE-completeness). *The ruleset* EITHER-ANYWHERE-DIFFERENT *is* PSPACE*-complete on 11-CNF and 11-DNF.*

To prove this, we will reduce from TOY POSITIVE CNF.

Proof. Note that the set of TOY POSITIVE CNF positions is exactly a subset of the set of EITHER-ANYWHERE-DIFFERENT positions. Thus, we can use the trivial (identity) reduction to show that EITHER-ANYWHERE-DIFFERENT is also PSPACE-complete. POSITIVE CNF is known to be hard for 11-CNF[6], so the same is true of TOY POSITIVE CNF and EITHER-ANYWHERE-DIFFERENT. The same reduction holds from POSITIVE DNF, hard on 11-DNF[6]. Thus, EITHER-ANYWHERE-DIFFERENT is also hard on 11-DNF. □

9. Conclusions

This work defines seven new combinatorial game rulesets based on satisfying Boolean formulas. Two of these are trivial in that the players do not have any choices to make. The other five are all computationally difficult (PSPACE-complete) to determine which player has a winning strategy in the worst case.

Of the five new hard rulesets, both EITHER-LOCAL-SAME and EITHER-ANYWHERE-SAME are impartial and incorporate unassigned variables into gameplay: the first games of this kind. They are hard for CNF formulas with only 4 and 2 literals per clause, respectively.

Also among the five new hard rulesets, BY-PLAYER-ANYWHERE-SAME is hard on 2-CNF formulas. The low size is surprising for this and EITHER-ANYWHERE-SAME because QBF is easy to solve on 2-CNF.

A summary of the results, by ruleset, can be found in Table 1.

10. Future Work

10.1. Additional Toggles

This work can be expanded on by introducing more toggle properties for satisfiability games. Schaefer defines many related games, including games with partitions

on the variables between the two players [6]. Certainly all relevant properties of rulesets are not covered here and further attributes can be considered.

10.2. Open: Tightness of Hard Instances

Although it is known that QBF is easy on 2-CNF and hard on 3-CNF, the same tight bounds are not known for the other games. It would be very useful to find the other boundaries. Hardness of the DNF versions are also unknown for most of the games.

References

[1] M. H. Albert, R. J. Nowakowski, and D. Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory.* A. K. Peters, Wellesley, Massachusetts, 2007.

[2] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. *A linear-time algorithm for testing the truth of certain quantified boolean formulas.* Inform. Process. Lett., 8(3):121–123, 1979.

[3] Gabriel Beaulieu, Kyle G. Burke, and Éric Duchêne. *Impartial coloring games.* Theoret. Comput. Sci., 485:49–60, 2013.

[4] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for your Mathematical Plays*, volume 1. A K Peters, Wellesley, Massachsetts, 2001.

[5] Erik D. Demaine and Robert A. Hearn. *Playing games with algorithms: Algorithmic combinatorial game theory.* Games of No Chance 3, volume 56 of Math. Sci. Res. Inst. Publ., pages 3–56. Cambridge Univ. Press, 2009.

[6] Thomas J. Schaefer. *On the complexity of some two-person perfect-information games.* J. Comput. System Sci., 16(2):185–225, 1978.