Tractable Unordered 3-CNF Games

Md Lutfar Rahman(✉) and Thomas Watson

University of Memphis, Memphis, TN, USA
mrahman9@memphis.edu

Abstract. The classic TQBF problem can be viewed as a game in which two players alternate turns assigning truth values to a CNF formula's variables in a prescribed order, and the winner is determined by whether the CNF gets satisfied. The complexity of deciding which player has a winning strategy in this game is well-understood: it is NL-complete for 2-CNFs and PSPACE-complete for 3-CNFs.

We continue the study of the *unordered* variant of this game, in which each turn consists of picking any remaining variable and assigning it a truth value. The complexity of deciding who can win on a given CNF is less well-understood; prior work by the authors showed it is in L for 2-CNFs and PSPACE-complete for 5-CNFs. We conjecture it may be efficiently solvable on 3-CNFs, and we make progress in this direction by proving the problem is in P, indeed in L, for 3-CNFs with a certain restriction, namely that each width-3 clause has at least one variable that appears in no other clause. Another (incomparable) restriction of this problem was previously shown to be tractable by Kutz.

Keywords: 3-CNF · Games · Unordered · Logarithmic space

1 Introduction

Two-player games play an important role in complexity theory, particularly in the study of space-bounded computations. For example, the seminal PSPACE-complete problem TQBF—in which the goal is to determine whether a given quantified boolean formula $\exists x_1\ \forall x_2\ \exists x_3\ \forall x_4 \cdots \varphi(x_1, \ldots, x_n)$ is true—can be viewed as deciding who has a winning strategy in the following two-player game: player 1 picks a bit value to assign to x_1, then player 2 assigns x_2, then player 1 assigns x_3, then player 2 assigns x_4, etc., with player 1 winning iff φ is satisfied.

Most commonly, φ is a conjunctive normal form (CNF) formula, which consists of a conjunction of clauses where each clause is a disjunction of literals. A w-CNF has at most w literals in each clause, and this *width* parameter w often governs the complexity of problems involving CNFs. For 2-CNFs, TQBF is NL-complete [2,4] (in particular, in P), while for 3-CNFs it is PSPACE-complete [12]. We call the corresponding game the *ordered CNF game* because the players are required to "play" the variables in a particular order prescribed in the input.

This work was supported by NSF grant CCF-1657377.

Y. Kohayakawa and F. K. Miyazawa (Eds.): LATIN 2020, LNCS 12118, pp. 360–372, 2020.
https://doi.org/10.1007/978-3-030-61792-9_29

Complexity of the Unordered CNF Game. In contrast, many real-world games have greater flexibility in terms of the set of moves available in each turn: the current player may be allowed to pick any of the remaining possible moves to do. We can define a variant of TQBF, called the *unordered CNF game*, which has this format: The input is again a CNF φ, and in each turn the current player picks a remaining (unassigned) variable and picks a bit value to assign it. The winner is determined by whether φ gets satisfied; we let T denote the player who wins when every clause of φ is true, and F denote the player who wins when some clause of φ is false. For 2-CNFs, deciding who has a winning strategy in this game is known to be in L [7], while PSPACE-completeness was shown for 11-CNFs [10,11], then for 6-CNFs [1], and then for 5-CNFs [7]. It remains a mystery what happens for widths 3 and 4.

We boldly conjecture that, in stark contrast to its ordered counterpart, *the unordered* 3*-CNF game may actually be tractable.* Progress toward confirming this conjecture can be made by considering certain restrictions on the input CNF, and showing that the game is tractable under these restrictions. The contribution of this paper is such a result. Before stating our result, for comparison we review other restrictions that have been studied.

One natural restriction is CNFs that are positive (a.k.a. monotone), meaning that all literal occurrences are unnegated variables; in this case, the unordered CNF game is equivalent to the so-called Maker–Breaker game (which is widely-studied in the combinatorics literature). In fact, [10,11] proved that the unordered CNF game is PSPACE-complete even for positive 11-CNFs (and a simplified proof for unbounded-width positive CNFs appears in [3]). Kutz [5,6] proved that for positive 3-CNFs, the unordered CNF game is tractable (in P) under an additional restriction on the hypergraph structure of the CNF, namely that no two clauses have more than one variable in common. This is the only previous result in the direction of confirming our conjecture.

It would be interesting to lift either the "positive" restriction or the "only one common variable" restriction in Kutz's result. We prove that *both* can be lifted if we instead impose a different (incomparable) restriction on the CNF's hypergraph structure. Specifically, we can view the variables in a clause as nodes, which are places where the clause can "connect" to other clauses (by sharing the variable). One difficulty in Kutz's analysis was handling width-3 clauses that use each of their 3 nodes to connect to other clauses. By restricting this difficulty away, we are able to address both limitations of Kutz's result, by handling general (not positive) CNFs that can have more than one common variable between pairs of clauses. (Our analysis does not end up resembling Kutz's very much, though.)

Thus our theorem can be stated as: the unordered 3-CNF game is in P, in fact in L, when each width-3 clause has at least one "spare" variable that appears in no other clauses. In the context of satisfiability, this restriction (each width-3 clause has a spare variable) is not very interesting since it would reduce to 2-SAT (the width-3 clauses could automatically be satisfied). Similarly, under this restriction, 3-TQBF would reduce to 2-TQBF since each clause with a spare variable belonging to T ($\exists$) would get satisfied (and thus disappear), and each

clause whose spare variable belongs to F (∀) would shrink to a width-2 clause. However, for the unordered 3-CNF game there is no clear way to reduce this restricted version to a 2-CNF game, since both players can vie for any spare variable. As we show in this paper, combinatorially characterizing the winner of such a restricted unordered 3-CNF game turns out to be drastically more involved than for unordered 2-CNF games [7].

Proof Outline. To prove our theorem, there are multiple cases depending on who has the first move and who has the last move. The case where T goes first reduces to the case where F goes first (by trying all possibilities for T's opening move, and seeing whether any of them lead to a win for T in the residual game where F moves first), so we focus on the latter. Our proof separately handles the cases where F has both the first and last moves (Sect. 3) and where F has the first move and T has the last move (omitted due to space constraints).

The case where F has both the first and last moves (so the number of variables is odd) is somewhat simpler to analyze. We state and prove a characterization of who has a winning strategy in this case, in terms of certain features of the input formula; an efficient algorithm follows straightforwardly from this. To obtain the characterization, we begin by identifying various types of subformulas whose presence in the input formula would enable F to win. It is an elementary but non-trivial case analysis to verify that in any of these subformulas, F indeed has a strategy to ensure some clause gets falsified (Sect. 3.1). The more interesting part of the proof is to show that not only do these subformulas constitute "obstacles" to T winning, but in a sense they are the *only* obstacles (Sect. 3.2). Although it is not true that F can win iff at least one of those subformulas exists in the original formula, we prove something just as good: F can win iff he has an opening move that ensures at least one of those subformulas will exist in the residual formula at the end of the first round. (A round consists of an F move followed by a T move.)

In other words, if T can fend off all the obstacles for one round, then he will be able to fend them off for the entire game. This non-obvious fact is key to taming the combinatorial structure of the game. The proof of this fact involves a subtle induction that modifies the game rules to allow F to "pass" (forgo his turn) whenever he wishes—this can only make it harder for T to win, but it is needed for the induction to go through. After a round, we can prove that for each of the smaller components that were created in the residual formula: either we can design a direct winning strategy for T in that component by exploiting the absence of the obstacle subformulas, or T can fend off obstacles for one more round in that component, enabling us to apply the induction hypothesis. Finally, to combine the "sub-strategies" for the separate components into an overall strategy for T, we exploit the resilience of the sub-strategies against pass moves by F.

The case where F goes first and T goes last follows a similar structure but is more involved. Some of the above argument can be recycled, but the parts that relied on F moving last need to be changed. Now the "complete" set of obstacles is larger and more complicated. The inductive argument for T's winning strategy

requires a more detailed analysis and uses a further modification of the game: the new rule says that a certain subformula gets immediately removed from the game (its variables become unplayable) whenever it is created in the residual formula. The deleted copies of this subformula are then dealt with "outside of" the induction, to recover a proof for the unmodified game.

Summary. One motivation for studying the unordered CNF game is that it is naturally analogous to a variety of real-world games where the same moves are available to both players. Indeed, the original result of Schaefer [10,11] has been used in many reductions to show PSPACE-completeness of other natural games with an unordered flavor (see [7] for a list). At a more fundamental level, the problem we study is very simple to define, and our result reveals new insights about CNFs, which are among the most ubiquitous representations of boolean functions.

A potential big payoff for this research direction is to show that the general unordered 3-CNF game is tractable. That may sound outlandish since arbitrary 3-CNFs are typically thought of as "too unstructured" to admit efficient algorithms for interesting problems. Our result together with the complementary result by Kutz [5,6] provides a glimpse into why the bold conjecture may be true, and a plausible roadmap for proving it: by combining our techniques, which handle negated literals and clauses that share two variables, with Kutz's techniques, which handle clauses without spare variables. Short of handling the general game, there are other open and interesting special cases to which our techniques may be germane, such as the Maker–Breaker game on general 3-uniform hypergraphs.

The proof of our result reveals a novel structural property: it is impossible for F to mount a "long-range" attack for creating a simple "obstacle" after a super-constant number of rounds—it is a "now or never" situation for F. We conjecture the same phenomenon holds for the game on unrestricted 3-CNFs, since we are unaware of any counterexamples. If a counterexample is found, it might be turned into a gadget for proving hardness of the general game. Even NL-hardness would be fundamentally interesting since our algorithm—based on detecting a simple obstacle after constantly many rounds—only uses logarithmic space. (As a side result—not included in this paper—we can show that the unordered 4-CNF game is NL-hard.)

Although our requirement that every width-3 clause has a spare variable seems to be a very strong restriction, and may not naturally show up in other contexts, we feel it is an important stepping stone for understanding more general games. It already adds a very significant layer of complexity over the unordered 2-CNF game, and it represents a reasonable way of suppressing some of the difficulties posed by the hypergraph structure of 3-CNFs (which Kutz's proof works hard to address), en route to a more general result.

Furthermore, our proof contributes some innovative techniques for analyzing games, including: modifying the game to facilitate an induction; our framework for showing how T can extend his good fortune from one round to all subsequent

rounds; and a method for simplifying gameplay analysis by imagining that the moves happened in a different order.

2 Preliminaries

We define a **formula** as a pair (φ, X) where φ is a CNF and $X = \{x_1, \ldots, x_n\}$ contains all the variables that appear in φ (and possibly more). In the unordered CNF game there are two players, denoted T (for "true") and F (for "false"), who alternate turns. Each turn consists of picking a remaining (unassigned) variable from X and assigning it a value 0 or 1. The game ends when all variables of X have been assigned, and T wins if φ is satisfied, and F wins if it is not. We let G (for "game") denote the problem of deciding which player has a winning strategy, given the formula (φ, X) and a specification of which player goes first. We let G_w denote the restriction of G to instances where each clause has at most w literals (φ has width w). We define a **spare variable** as occurring in only one clause, and we assume without loss of generality that a spare variable appears as a positive literal. Then we let G_3^* denote the restriction of G_3 to instances where each width-3 clause in φ has at least one spare variable.

Theorem 1. *G_3^* is in polynomial time, in fact, in logarithmic space.*

We introduce subscripts to distinguish the different patterns for "who goes first" and "who goes last". For $a, b \in \{\mathrm{T}, \mathrm{F}\}$, the subscript $a \cdots b$ means player a goes first and player b goes last, $a \cdots$ means a goes first, and $\cdots b$ means b goes last. Thus $\mathrm{G}^*_{3,\mathrm{T}\cdots}$ corresponds to the game where T goes first, which (as noted in Sect. 1) reduces to $\mathrm{G}^*_{3,\mathrm{F}\cdots}$ by brute-forcing T's first move. So, we just prove Theorem 1 for $\mathrm{G}^*_{3,\mathrm{F}\cdots}$, which is split into the cases $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{F}}$ (F goes first and last, so $n = |X|$ must be odd) and $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$ (F goes first and T goes last, so $n = |X|$ must be even). We use the terms **move**, **turn**, or **play** interchangeably to mean T or F assigning a bit value to one variable. A **round** consists of two consecutive moves, and since we only need to consider F having the first move, each round will consist of one F move followed by one T move (except in $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{F}}$, the last round will have only one move).

A **subformula** (φ', X') of a formula (φ, X) is defined as φ' having a subset of clauses from φ and $X' \subseteq X$ containing all the variables that appear in φ' (and possibly more). After a move, the formula changes to a **residual formula** where the variable that got played is removed from X, and each clause containing the variable either disappears (since it is satisfied by a true literal) or shrinks (since a false literal might as well not be there). F wins if the residual formula has a width-0 clause, and T wins if it has no clauses. The residual formula after a move may or may not be a subformula of the formula before the move.

When we say F **can ensure** some property **within** k rounds, we formally mean that either

- the original formula has the property, or
- ($\exists$ F move) ($\forall$ T move) the residual formula has the property, or

- (∃ F move) (∀ T move) (∃ F move) (∀ T move) the residual formula has the property, or ⋯⋯
- (∃ F move) (∀ T move) ⋯ (∃ F move in k^{th} round) (∀ T move in k^{th} round) the residual formula has the property.

Note that the property is only checked at the boundary between rounds (and not after F's move but before T's move inside of a round).

A positive CNF is equivalent to a hypergraph where nodes are variables and hyperedges are clauses. In this paper, we use a hypergraph representation of general (not necessarily positive) CNFs. As shown in Fig. 1, a clause is a hyperedge where nodes represent variables, and signs are annotations representing variables' literal appearances. When we omit the sign of a variable on a diagram, it could be either + or − but it is not relevant.

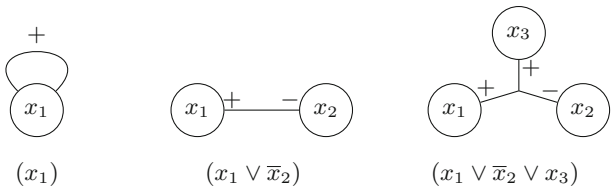

Fig. 1. Example clauses and their hypergraph representations

Two clauses in a general CNF can share any number of same signed or opposite signed literals. We think of a shared variable as a connection between two clauses, and we define two types of connections:

- **Pure Connection:** A variable that appears with the same sign in two clauses. For example, in $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5)$ there is a pure connection at x_2. See Fig. 2 on the left. Another example: in $(x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee x_4 \vee x_5)$ there is again a pure connection at x_2.
- **Mixed Connection:** A variable that appears with the opposite sign in two clauses. For example, in $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee x_4 \vee x_5)$ there is a mixed connection at x_2. See Fig. 2 on the right. Another example: in $(x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5)$ there is again a mixed connection at x_2.

A formula (φ, X) is called **connected** if the associated hypergraph is connected (with the signs being irrelevant); i.e., it is possible to get from any variable to any other variable by a sequence of clauses, each having a connection to the next. A formula is thus naturally partitioned into connected components, each of which is a subformula. An **isolated variable** is one that is in X but not in any clause of φ, and thus forms a connected component by itself since the associated node is incident to no hyperedges. A variable in a width-1 clause is not considered isolated.

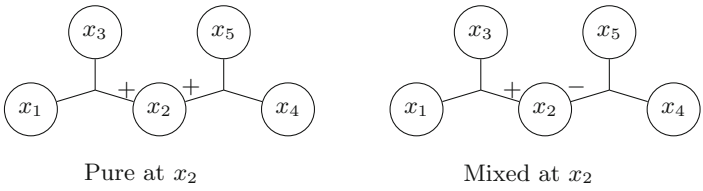

Fig. 2. Clause connections

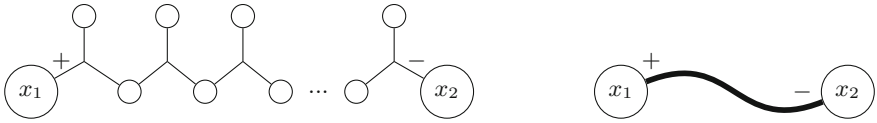

Fig. 3. A chain between x_1 and x_2

A **chain** is a sequence of distinct width-3 clauses each sharing exactly one variable with the next, and with no shared variables between two non-consecutive clauses. The length L of the chain is the number of clauses. An arbitrary chain between x_1 and x_2 is illustrated in Fig. 3 on the left. On the right, we show how the chain can be depicted by a thick line. If $L = 0$ then $x_1 = x_2$. If $L = 1$ then the only clause in the chain contains both x_1 and x_2.

A **cycle** is like a chain with $L > 2$ and $x_1 = x_2$. A **diamond** happens when two width-3 clauses share exactly two variables. Intuitively, a diamond is like the smallest case of a cycle, with $L = 2$.

3 $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{F}}$

We henceforth assume that in a formula (φ, X), φ is always a 3-CNF where each width-3 clause has at least one spare variable.

Lemma 1. F *has a winning strategy in a* $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{F}}$ *game iff* F *can ensure within one round at least one of the following subformulas exists.*

*(1) A width-*0 *or width-*1 *clause.*
*(2) Two width-*2 *clauses sharing both variables.*
*(3) Two width-*2 *clauses and a chain (of length* ≥ 0*) between them.*
*(4) A width-*2 *clause and a chain (of length* ≥ 1*) between its two variables with at least one mixed connection between the chain and the width-*2 *clause.*
*(5) A width-*2 *clause, a cycle or diamond containing at most one width-*2 *clause variable, and a chain (of length* ≥ 0*) between them.*

Moreover, if subformula (4) or (5) exists at the beginning of a round then F *can ensure subformula (1) or (2) or (3) exists within two more rounds.*

The proof of Lemma 1 is in Sect. 3.1 and Sect. 3.2.

Corollary 1. F *has a winning strategy in a* $G^*_{3,F\cdots F}$ *game iff* F *can ensure subformula (1) or (2) or (3) exists within the first three rounds.*

Corollary 1 yields a direct approach to devise an algorithm for $G^*_{3,F\cdots F}$:

> Try all possible sequences of 6 moves for the first 3 rounds. Check whether (∃ F move) (∀ T move) (∃ F move) (∀ T move) (∃ F move) (∀ T move): subformula *(1)* or *(2)* or *(3)* exists in the residual formula.

This can be implemented in log space, because keeping track of a sequence of the first six moves takes log space, searching for subformula *(1)* or *(2)* takes log space, and searching for subformula *(3)* also takes log space since it can be expressed as an undirected s–t connectivity problem [8,9]: for each pair of width-2 clauses, check whether there exists a chain between them.

We conjecture the same algorithm (possibly with a different number of brute-force rounds) actually solves $G_{3,F\cdots F}$; we are not aware of any counterexamples.

3.1 Right-to-left Implication of Lemma 1

Suppose at least one of the subformulas *(1–5)* exists when it is F's turn to play. We will handle each subformula in separate claims. For concreteness, we illustrate the arguments using literals with particular signs, but all the arguments work even if we negate all occurrences of any variable.

Claim 1. *If subformula (1) exists,* F *has a winning strategy.*

Proof. If a width-0 clause exists then T has no chance to satisfy it, so F wins. If a width-1 clause exists, say (x_1), then F can play $x_1 = 0$ and win. □

Claim 2. *If subformula (2) exists,* F *has a winning strategy.*

Proof. There are two possible ways that can happen:

- Case 1: *The clauses have opposite signs for one variable (mixed connection).* For example, in $(x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2)$ only x_1 has opposite signs. Then F can play $x_2 = 0$, and whatever the value of x_1, F will win.
- Case 2: *The clauses have opposite signs for both variables.* For example, in $(x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2)$ both x_1 and x_2 have opposite signs. Since F moves last, F can wait by playing other variables until T has to play x_1 or x_2. Then F makes $x_1 = x_2$ and wins. □

Claim 3. *If subformula (3) exists,* F *has a winning strategy.*

Proof. We call this situation a manriki (a Japanese ninja weapon). The two width-2 clauses are like two handles and the chain in the middle can be arbitrarily long. We prove this claim by induction on the length of the chain.

Base case: The length of the chain is zero, i.e., the two handles directly share a variable. We can assume the two handles do not share both variables since otherwise that falls under Claim 2. There are two possible ways the handles can have one common variable:

- Case 1: *Pure Connection.* For example, in $(x_1 \vee x_2) \wedge (x_2 \vee x_3)$, x_2 forms a pure connection. F can play $x_2 = 0$. Then whatever T does, F plays $x_1 = 0$ or $x_3 = 0$ and wins.
- Case 2: *Mixed Connection.* For example, in $(x_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3)$, x_2 forms a mixed connection. F can play $x_1 = 0$. If T plays $x_2 = 1$ then F plays $x_3 = 0$ and wins. If T plays $x_2 = 0$ then F wins. If T does not play x_2, F wins by playing $x_2 = 0$.

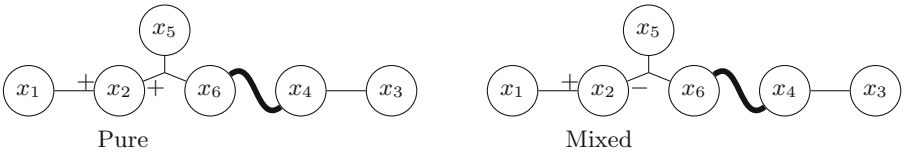

Fig. 4. Subformula *(3)* (Claim 3)

Induction Step: There are two cases depending on the type of connection at the common variable between one of the handles and the chain:

- Case 1: *Pure Connection.* For example, in Fig. 4 on the left, x_2 forms a pure connection between handle $(x_1 \vee x_2)$ and the chain. F can play $x_2 = 0$. If T plays $x_1 = 1$ then we have a smaller manriki from $(x_5 \vee x_6)$ to $(x_4 \vee x_3)$ where F can win by the induction hypothesis. If T plays $x_1 = 0$ then F wins. If T does not play x_1 then F wins by playing $x_1 = 0$.
- Case 2: *Mixed Connection.* For example, in Fig. 4 on the right, x_2 forms a mixed connection between handle $(x_1 \vee x_2)$ and the chain. F can play $x_1 = 0$. If T plays $x_2 = 1$ then we have a smaller manriki from $(x_5 \vee x_6)$ to $(x_4 \vee x_3)$ where F can win by the induction hypothesis. If T plays $x_2 = 0$ then F wins. If T does not play x_2 then F wins by playing $x_2 = 0$. □

Claim 4. *If subformula (4) exists,* F *has a winning strategy.*

Proof. There are three cases depending on how the width-2 clause is connected to the chain. For example, in Fig. 5, $(x_1 \vee x_2)$ is the width-2 clause and x_2 is a mixed connection. In the smallest versions, the chain (the bold line illustrated in the general versions) has length 1 for cases 1 and 2 and length 0 for case 3.

- Case 1: *Pure at* x_1. F can play $x_1 = 0$. If T plays $x_2 = 1$ then in the smallest case F wins by $x_3 = 0$ and in the general case F wins by Claim 3 by a manriki created from x_1's left end to x_2's right end. If T plays $x_2 = 0$ then F wins. If T does not play x_2 then F wins by $x_2 = 0$.
- Case 2: *Mixed at* x_1 *but pure at* x_4 *(the next non-spare variable on the chain).* F can play $x_4 = 0$. If T plays $x_1 = 0$ or $x_2 = 0$ then F wins by $x_2 = 0$ or $x_1 = 0$. If T plays $x_1 = 1$ then F wins by $x_3 = 0$. If T plays $x_2 = 1$ then

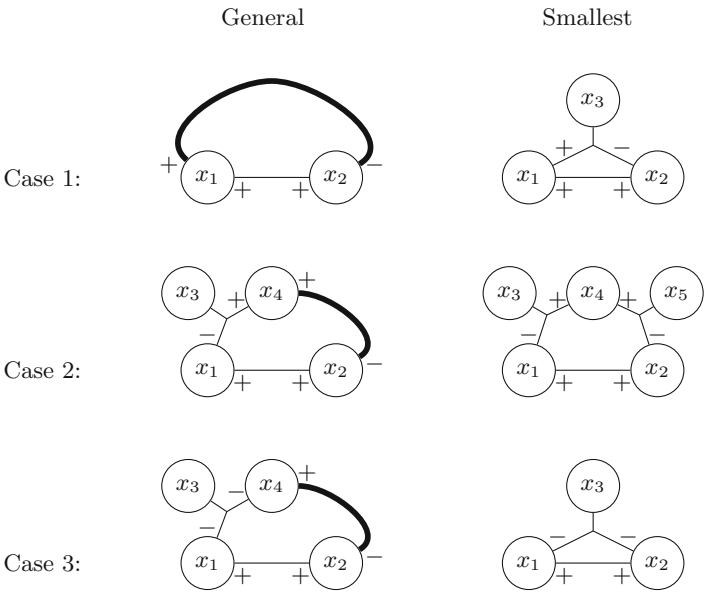

Fig. 5. Subformula *(4)* (Claim 4)

in the smallest case F wins by $x_5 = 0$ and in the general case F wins by a manriki created from x_4's right end to x_2's right end. If T plays x_3 then in the smallest case F wins by the manriki $(x_5 \vee \overline{x}_2) \wedge (x_2 \vee x_1)$ and in the general case F wins by a manriki created from x_4's right end to $(x_2 \vee x_1)$. If T plays any other variable then F wins by the manriki $(x_3 \vee \overline{x}_1) \wedge (x_1 \vee x_2)$.

- Case 3: *Mixed at both x_1 and x_4.* F can play $x_3 = 0$. In the smallest case, since F moves last, F can wait by playing other variables until T has to play x_1 or x_2, and then F can win by making $x_1 = x_2$. Now consider the general case. If T plays $x_1 = 0$ or $x_2 = 0$ then F wins by $x_2 = 0$ or $x_1 = 0$. If T plays $x_1 = 1$ then F wins by $x_4 = 1$. If T plays $x_2 = 1$ then F wins by a manriki created from x_2's right end to $(\overline{x}_4 \vee \overline{x}_1)$. If T plays $x_4 = 0$ then F wins by a manriki created from x_4's right end to $(x_2 \vee x_1)$. If T plays $x_4 = 1$ then F wins by $x_1 = 1$. If T plays any other variable then F wins by the manriki $(x_2 \vee x_1) \wedge (\overline{x}_1 \vee \overline{x}_4)$. □

Claim 5. *If subformula (5) exists,* F *has a winning strategy.*

The proof of Claim 5 is omitted due to space constraints.

Moreover, in all cases, there exists a subformula *(1)* or *(2)* or *(3)* within one round for Claim 4 and within two rounds for Claim 5.

3.2 Left-to-right Implication of Lemma 1

Definition 1. *A **cobweb** is a formula where none of the subformulas (1–5) exist (and each width-*3 *clause has at least one spare variable). Note that any subformula in a cobweb is also a cobweb.*

Observation 1. *A cobweb has a variable that occurs in at most one clause.*

The proof of Observation 1 is omitted due to space constraints.

Suppose F cannot ensure that at least one of the subformulas *(1–5)* exists within one round. So at the beginning the formula is a cobweb and in the first round, for every move by F there exists a move for T such that the residual formula is again a cobweb. In other words, T can ensure that the beginning cobweb remains a cobweb after a round. We will argue that T has a winning strategy. The proof will be by induction on the number of variables. In order for the induction to go through, we need to prove something stronger: "T can win even if F is allowed to use pass moves." This means F has the option of forgoing any turn, thus forcing T to play multiple variables in a row. In this case it does not make sense to consider which player has the last move, so we consider the game $\mathrm{G}^*_{3,\mathrm{F}\cdots}$ in this section.

First we consider a special case of cobweb that we call a jellyfish.

Definition 2. *A **jellyfish** is a connected cobweb with a width-*2 *clause. Its **eyes** are the variables in the width-*2 *clause.*

Lemma 2. *If the formula is a jellyfish then* T *has a winning strategy in* $\mathrm{G}^*_{3,\mathrm{F}\cdots}$ *even if* F *can use pass moves.*

The proof of Lemma 2 is omitted due to space constraints.

Definition 3. *A **winweb** is a cobweb such that* T *can ensure that it remains a cobweb after a round (where* F *is not allowed to use pass moves).*

Lemma 3. *Every subformula of a winweb is also a winweb.*

The proof of Lemma 3 is omitted due to space constraints.

The following lemma proves something stronger than the left-to-right implication of Lemma 1, because F can use pass moves.

Lemma 4. *If the formula is a winweb then* T *has a winning strategy in* $\mathrm{G}^*_{3,\mathrm{F}\cdots}$ *even if* F *can use pass moves.*

Proof. We prove this by induction on the number of variables.

Base case: The formula is a cobweb with one or two variables. In case of one variable the only possibility is an isolated variable with no clauses since subformula *(1)* does not exist. T has already won in this case. In case of two variables there exists either two isolated variables where T has already won or a width-2 clause which T can satisfy in one move.

Induction step: The formula (φ, X) is a winweb with at least three variables.

Suppose F played a pass move. There exists an isolated or spare variable since the formula is a cobweb (Observation 1). T can play that isolated/spare variable to remove the isolated variable or satisfy a clause. The residual formula is a subformula, which is a winweb by Lemma 3. Thus T can win the rest of the game by the induction hypothesis.

Now suppose F did not play a pass move. By the definition of winweb, T has a response such that the residual formula is a cobweb. Call this residual formula (φ', X') and let (φ_1, X_1), (φ_2, X_2), ..., (φ_k, X_k) be its connected components (so $\varphi' = \bigwedge_i \varphi_i$ and $X' = \bigcup_i X_i$). We claim that for each component individually, T has a winning strategy even if F can use pass moves:

- If (φ_i, X_i) has a width-2 clause then it is a jellyfish (since it is a connected cobweb) so by Lemma 2, T can win even if F can use pass moves.
- Suppose (φ_i, X_i) has no width-2 clause. Then it has only width-3 clauses since subformula *(1)* does not exist, and so it is a subformula of the winweb (φ, X) since no new width-3 clause can be created during the game. By Lemma 3, (φ_i, X_i) is also a winweb and hence by the induction hypothesis, T can win even if F can use pass moves.

We now explain how to combine T's winning strategies for the separate components to get a winning strategy for the rest of the game on (φ', X'). After F plays a variable in some X_i, T simply responds according to his winning strategy for component (φ_i, X_i), unless F played the last remaining variable in X_i. In the latter case, or if F played a pass move, T picks any other component (φ_j, X_j) with remaining variables and continues according to his winning strategy in that component, as if F had just played a pass move in that component. □

References

1. Ahlroth, L., Orponen, P.: Unordered constraint satisfaction games. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 64–75. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32589-2_9
2. Aspvall, B., Plass, M., Tarjan, R.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Inf. Process. Lett. **8**(3), 121–123 (1979)
3. Byskov, J.: Maker-maker and maker-breaker games are PSPACE-complete. Technical report RS-04-14, BRICS, Department of Computer Science, Aarhus University (2004)
4. Calabro, C.: 2-TQBF is in P (2008). https://cseweb.ucsd.edu/~ccalabro/essays/complexity_of_2tqbf.pdf. Unpublished
5. Kutz, M.: The angel problem, positional games, and digraph roots. Ph.D. thesis, Freie Universität Berlin (2004). Chapter 2: Weak Positional Games
6. Kutz, M.: Weak positional games on hypergraphs of rank three. In: Proceedings of the 3rd European Conference on Combinatorics, Graph Theory, and Applications (EuroComb), pp. 31–36. Discrete Mathematics & Theoretical Computer Science (2005)
7. Rahman, M.L., Watson, T.: Complexity of unordered CNF games. In: Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC), pp. 9:1–9:12. Schloss Dagstuhl (2018)

8. Reingold, O.: Undirected connectivity in log-space. J. ACM **55**(4), 17:1–17:24 (2008)
9. Rozenman, E., Vadhan, S.: Derandomized squaring of graphs. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX/RANDOM -2005. LNCS, vol. 3624, pp. 436–447. Springer, Heidelberg (2005). https://doi.org/10.1007/11538462_37
10. Schaefer, T.: Complexity of decision problems based on finite two-person perfect-information games. In: Proceedings of the 8th Symposium on Theory of Computing (STOC), pp. 41–49. ACM (1976)
11. Schaefer, T.: On the complexity of some two-person perfect-information games. J. Comput. Syst. Sci. **16**(2), 185–225 (1978)
12. Stockmeyer, L., Meyer, A.: Word problems requiring exponential time. In: Proceedings of the 5th Symposium on Theory of Computing (STOC), pp. 1–9. ACM (1973)