Tractable Unordered 3-CNF Games

Md Lutfar Rahman Thomas Watson

University of Memphis

November 9, 2019

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.

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 [APT79, Cal08] (in particular, in P), while for 3-CNFs it is PSPACE-complete [SM73]. 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.

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 [RW18], while PSPACE-completeness was shown for 11-CNFs [Sch76, Sch78], then for 6-CNFs [AO12], and then for 5-CNFs [RW18]. 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, [Sch76, Sch78] 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 [Bys04]). Kutz [Kut04, Kut05] 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 ($\forall$) 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 [RW18].

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 (Section 3) and where F has the first

move and T has the last move (Section 4).

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 (Section 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 (Section 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 [Sch76, Sch78] has been used in many reductions to show PSPACE-completeness of other natural games with an unordered flavor (see [RW18] 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 [Kut04, Kut05] 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 Section 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

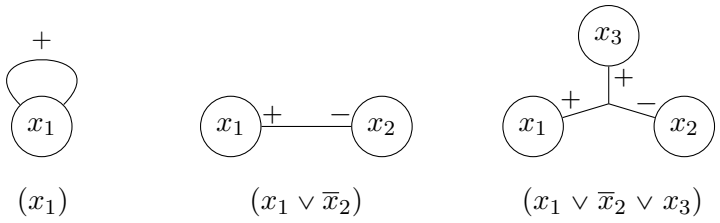


Figure 1: Example clauses and their hypergraph representations

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
- ($\exists$ F move) ($\forall$ T move) ($\exists$ F move) ($\forall$ T move) the residual formula has the property, or $\cdots\cdots$
- ($\exists$ F move) ($\forall$ T move) $\cdots$ ($\exists$ F move in k^{th} round) ($\forall$ 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 Figure 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.

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 Figure 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 Figure 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

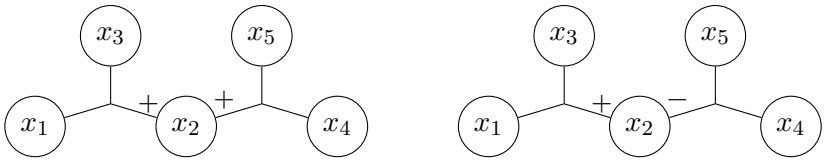


Figure 2: Clause connections

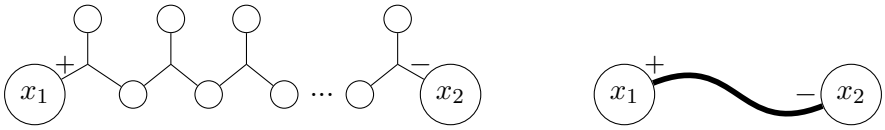


Figure 3: A chain between x_1 and x_2

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.

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 Figure 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* $\geqslant 0$*) between them.*

*(4) A width-*2 *clause and a chain (of length* $\geqslant 1$*) 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* $\geqslant 0$*) 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 Section 3.1 and Section 3.2.

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

Proof. $\Leftarrow$: Once subformula *(1)* or *(2)* or *(3)* exists, applying Lemma 1 to the residual formula shows that F has a winning strategy for the rest of the game.
$\Rightarrow$: According to Lemma 1, when F has a winning strategy, F can ensure that at the beginning or after the first round there exists at least one of the subformulas *(1–5)*. If subformula *(4)* or *(5)* exists then F can ensure that within two more rounds there exists subformula *(1)* or *(2)* or *(3)* by the "moreover" part. □

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

> 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 [Rei08, RV05]: for each pair of width-2 clauses, check whether there exists a chain between them. (It is possible to design a faster algorithm by appealing to Lemma 1 and using only one round of brute force, but we are not concerned with optimizing the running time.)

We conjecture the same algorithm (possibly with a different number of brute-force rounds) actually solves $\mathrm{G}_{3,\mathrm{F}\cdots\mathrm{F}}$; 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. □

Figure 4: Manriki (source: karatemart.com)

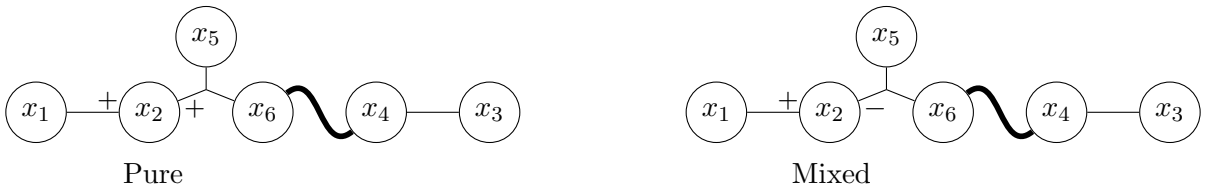


Figure 5: Subformula *(3)* (Claim 3)

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

Proof. We call this situation a manriki (a Japanese ninja weapon, see Figure 4). 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 $x_2 = 0$.

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 Figure 5 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 Figure 5 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 $x_2 = 0$. □

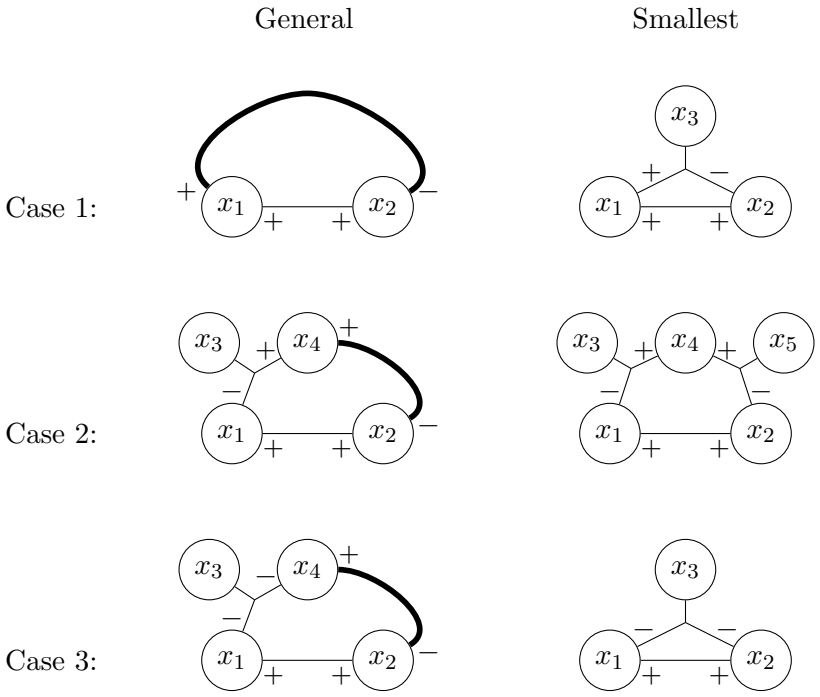


Figure 6: Subformula *(4)* (Claim 4)

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 Figure 6, $(x_1 \vee x_2)$ 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 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 $(x_2 \vee x_1) \wedge (\overline{x}_1 \vee \overline{x}_4)$. □

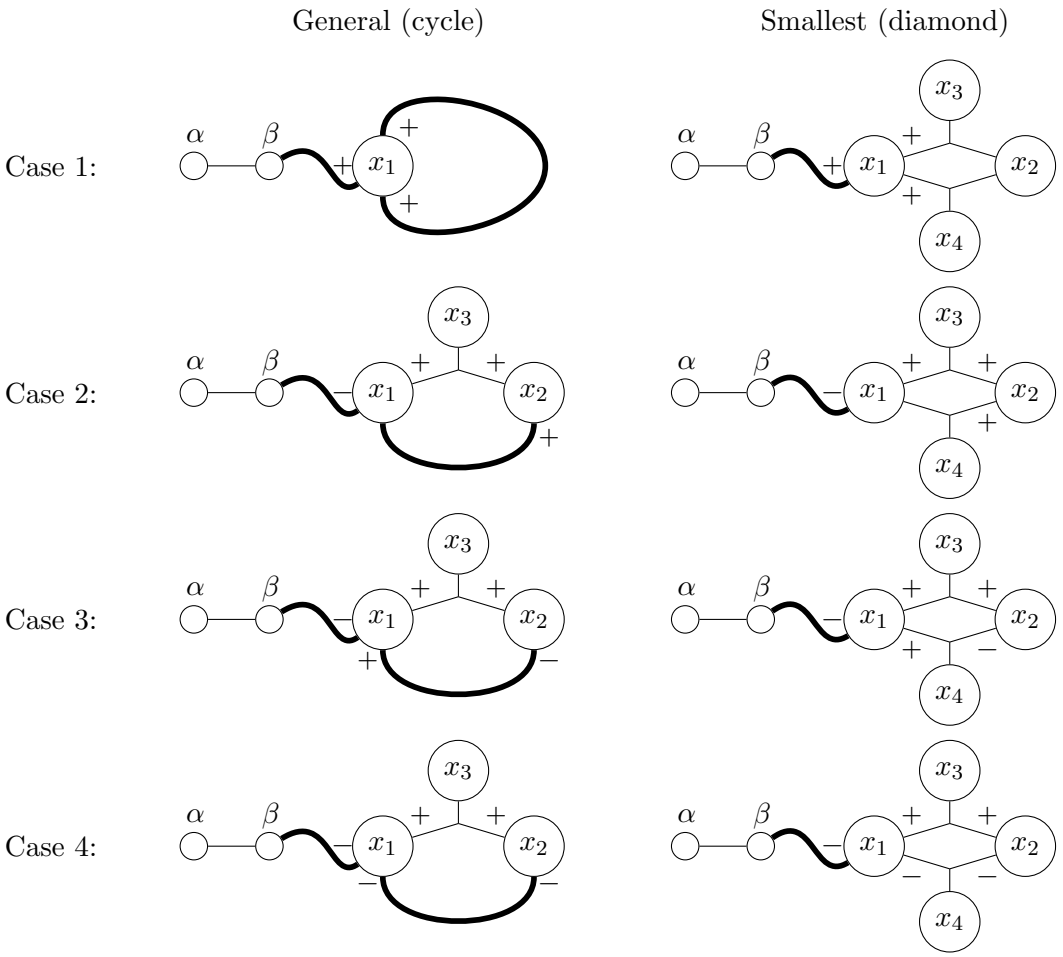


Figure 7: Subformula *(5)* (Claim 5)

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

Proof. We may assume the chain and the cycle/diamond have no common variable other than the connection variable, by just considering the initial segment of the chain up to the first point where it intersects the cycle/diamond.

There are four cases depending on how the "outer chain" (between the width-2 clause and the cycle/diamond) and the "inner chain" (inside the cycle/diamond) are connected. For example, in Figure 7, x_1 is the connection variable and $C = (\alpha \vee \beta)$ is the width-2 clause. The outer chain has length $\geqslant 0$.

- Case 1: *Three-way pure at* x_1. F can play $x_1 = 0$. If the outer chain had length $\geqslant 1$, then this move creates two disjoint manrikis, one inside the cycle or diamond and the other outside. T cannot destroy two disjoint manrikis in a single move, so one manriki will remain untouched and F wins by Claim 3. If the outer chain had length 0 (so x_1 is β), then the width-1 clause (α) takes the place of one of the two manrikis, and the same argument works.
- Case 2: *Mixed at* x_1 *and pure at* x_2 *(the next non-spare variable on the same side of the cycle as a mixed connection to the outer chain).* F can play $x_2 = 0$. If T plays $x_1 = 0$ then F wins

by $x_3 = 0$. If T plays $x_1 = 1$ then F wins by a manriki created from x_1's left end to C (or just by playing $\alpha = 0$ if the outer chain had length 0). If T plays x_3 then F wins by a manriki created from x_2's lower end to C. If T plays any other variable in the cycle or diamond then F wins by a manriki created from $(x_3 \vee x_1)$ to C. If T plays any other variable outside the cycle or diamond then F wins by a manriki created from $(x_3 \vee x_1)$ to x_2's lower end.

- Case 3: *Mixed from the outer chain to both sides of the cycle or diamond, and mixed at x_2.* F can play $x_3 = 0$. If T plays $x_1 = 0$ or $x_2 = 0$ then F can win by $x_2 = 0$ or $x_1 = 0$. If T plays $x_1 = 1$ then F wins by a manriki created from x_1's left end to C (or just by playing $\alpha = 0$ if the outer chain had length 0). If T plays $x_2 = 1$ then F wins by a manriki created from x_2's lower end to C. If T plays any other variable in the cycle or diamond then F wins by a manriki created from $(x_2 \vee x_1)$ to C. If T plays any other variable outside the cycle or diamond then F wins by Claim 4 (case 1).
- Case 4: *Mixed from the outer chain to only one side of the cycle or diamond, and mixed at x_2 (the next non-spare variable on the same side of the cycle as a mixed connection to the outer chain).* Similarly to case 3, F can play $x_3 = 0$, and F wins if T plays x_1 or x_2 or another variable inside the cycle or diamond. If T plays any other variable outside the cycle or diamond then F wins in the cycle by Claim 4 (case 2 or case 3 general) and in the diamond by Claim 4 (case 3 smallest). □

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.*

Proof. If the cobweb has a width-3 clause then the spare variable in it occurs in only one clause. Suppose there is no width-3 clause but there exists a width-2 clause. Then every width-2 clause must be isolated (have no connections to other clauses) since subformula *(1)* does not exist and two connected width-2 clauses would form either subformula *(2)* or subformula *(3)*. So any variable in any width-2 clause occurs only once. If there are no width-3 or width-2 clauses then the cobweb has only isolated variables, which occur in no clauses. □

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.

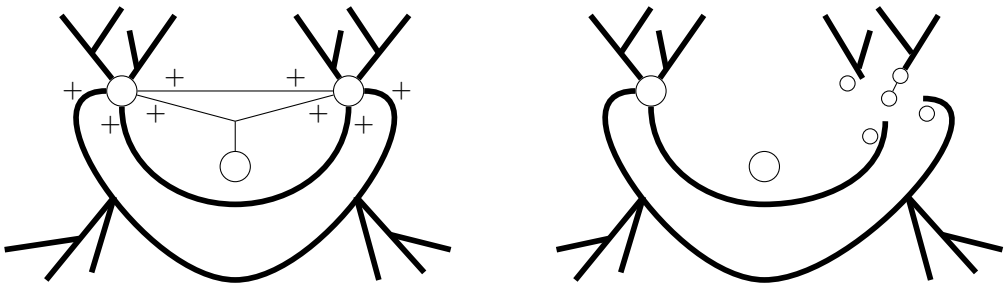


Figure 8: Jellyfish (left) and T's move on the right eye = 1 (right)

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.*

Proof. First we argue what a jellyfish must look like. A jellyfish does not have any width-0 or width-1 clause because that would form subformula *(1)*. Since a jellyfish is connected and contains a width-2 clause (the eyes), there cannot be another width-2 clause because that would form subformula *(2)* or *(3)*. Since a jellyfish is connected there cannot be any cycle or diamond (of width-3 clauses) containing at most one eye because that would form subformula *(5)* with the help of the eyes. The two eyes can be connected by chains but those connections to the width-2 clause must be pure because any mixed connection at the eyes would form subformula *(4)*. All chains connecting the two eyes must be disjoint from each other except at the eyes themselves, otherwise it would form subformula *(5)* in one of three ways depending on how the two chains overlap: If the overlaps create a diamond then the diamond is chain-connected to an eye. If the overlaps do not have any diamond then it will create either a cycle containing an eye or a cycle chain-connected to an eye.

In summary: A jellyfish has exactly one width-2 clause (eyes). There can be any number of disjoint chains of length $\geqslant 1$ between the two eyes, and those chains must have pure connections to the width-2 clause. Without loss of generality we assume those connections are all positive, i.e., the eyes are positive literals in the width-2 clause and in clauses on chains connecting the eyes. There can be arbitrary trees of width-3 clauses hanging off at any non-spare variables. Trees hanging off of an eye can have pure or mixed connections at the eye. This characterization of jellyfish is illustrated in Figure 8 (left).

Now we will show that T can win on such a formula like Figure 8 (left) that we call jellyfish. The argument has two parts. In the first part, we argue that T can ensure that within one round on a jellyfish, it looks like a forest where trees can be of two types as in Figure 9.

Definition 3. *A **single tree** is a connected formula with only width-3 clauses where there is no cycle or diamond (and each clause has a spare variable). A **married tree** is a connected formula where two disjoint single trees got "married" by a width-2 clause with one endpoint in each of the single trees (and these endpoints are considered roots of the single trees). A **win-forest** is a formula where each connected component is either a single tree or a married tree.*

In the second part, we argue that T can ensure that once a win-forest, always a win-forest after each round. For convenience, we work on the second part first.

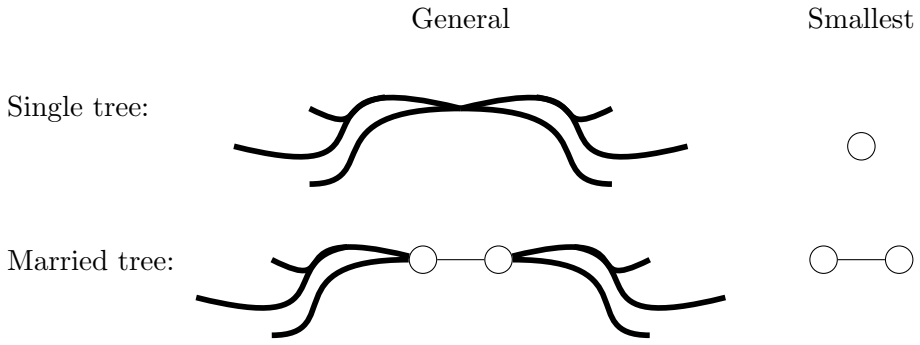


Figure 9: Single tree and married tree

Claim 6. T *can ensure that a win-forest remains a win-forest after a round even if* F *can use pass moves.*

Proof. The argument will show that whatever F plays, whether a pass move or in a single tree or married tree, T has a response such that each component of the residual formula is again either a single tree or a married tree; therefore the residual formula is again a win-forest. Any move by T or F can occur in three different scenarios as illustrated in Figure 10 (where we suppose the move is $x_1 = 0$). Specifically, Scenario 1 is a move on a non-spare variable, Scenario 2 is a move on a spare variable that does not satisfy the clause, and Scenario 3 is a move on a spare variable that satisfies the clause. (Normally we assume a spare variable is a positive literal, but to illustrate satisfying a clause with $x_1 = 0$ we let it be a negative literal in Scenario 3 in Figure 10.)

Suppose F played a pass move. Since a win-forest is a cobweb, there exists an isolated or spare variable (Observation 1). T can play that isolated/spare variable to remove the isolated variable or satisfy a clause by Scenario 3. If T plays a spare variable in a single tree then it creates two single trees. If T plays a spare variable in a width-3 clause in a married tree then it creates one single tree and one married tree. If T plays a spare variable in a married tree's width-2 clause then it creates one single tree. This preserves the win-forest property.

Suppose F played in a single tree. F's move could occur in the three different scenarios. In Scenario 1, F's move can create some isolated variables (which are single trees), some other single trees, and some married trees (in which one spouse is just a single variable). In Scenario 2, F's move creates one married tree. In Scenario 3, F's move creates two single trees. In all scenarios it still remains a win-forest after F's move, so T can now just play as if F had used a pass move on this win-forest, as explained in the previous paragraph.

Suppose F played in a married tree. F's move happened in one of the two single trees that got married. T can play the root of the other single tree (where F has not played) and satisfy the width-2 clause. This means the two single trees get separated by T's move and it also breaks T's single tree at the root by Scenario 1. On the other hand F's move in his/her single tree also preserves the win-forest property because, whatever F played in that single tree, it must follow one of the three scenarios as explained in the previous paragraph. □

Claim 7. T *can ensure that a jellyfish becomes a win-forest within one round even if* F *can use pass moves.*

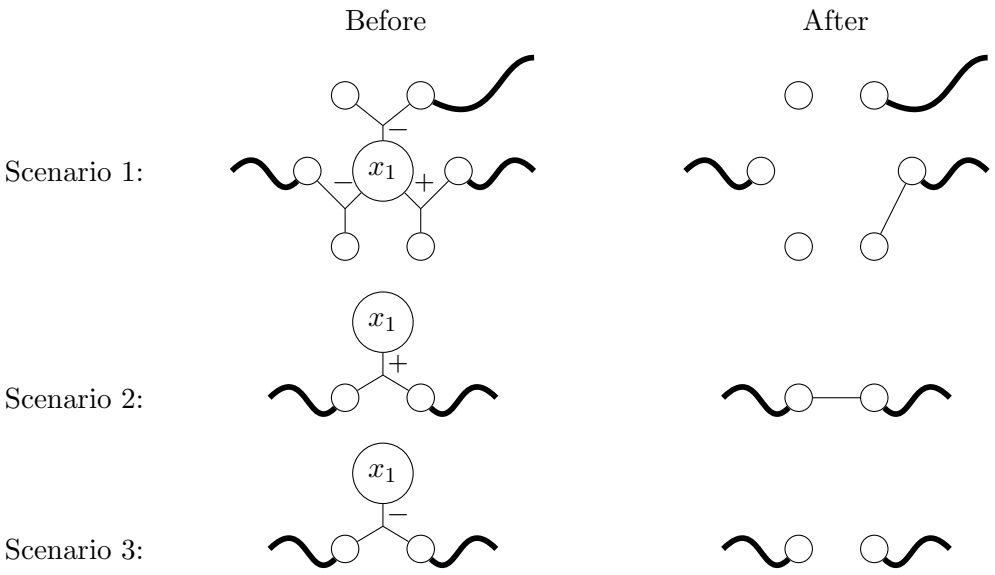


Figure 10: Move $x_1 = 0$ and its effect on formulas

Proof. The jellyfish can have a single tree rooted at each eye, so together with the width-2 clause these form a married tree which is a subformula of the jellyfish. Let us call those single trees "left-eyed spouse" and "right-eyed spouse". There can be chains connecting the two eyes. Let us call these chains "body-chains".

Whatever F's move is, it does not touch at least one of the two spouse trees. Without loss of generality we assume F did not play in the right-eyed spouse. Then T responds by playing the right eye to satisfy the width-2 clause. To prove the residual formula is a win-forest, we pretend T's move happened before F's move, and consider the formula after T's move but before F's move.

The right eye move by T is shown in Figure 8 (right). The width-2 clause gets satisfied. The right-eyed spouse can be broken into some single trees and married trees by Scenario 1. The rest of the connections at the right eye were only pure and they all came from the body-chains. So when the body-chains get broken at the right eye, they only create some isolated variables but no width-2 clauses. Then the broken body-chains together with the left-eyed spouse and any other trees hanging off the broken body-chains form one big single tree. The residual formula as illustrated in Figure 8 (right) is now a win-forest.

Now we consider how F's move could have affected this win-forest. If F used a pass move, then no harm done, we already got a win-forest. If F did not use a pass move then any move in a single tree preserves the win-forest property (as explained in the proof of Claim 6). F's move could not happen in any of the current married trees since those only came from the right-eyed spouse and F did not play there. □

Now putting it all together: If the formula is a jellyfish then T can ensure that it becomes a win-forest within one round by Claim 7. Then after each subsequent round, T can always ensure that the formula remains a win-forest by Claim 6. In the last round, if there exists only one remaining variable then the only possibility is an isolated variable with no clauses since subformula *(1)* does not exist. T has already won in this case. If there are two remaining variables in the last round

then there exists either two isolated variables where T has already won or a width-2 clause which T can satisfy in one move. This finishes the proof of Lemma 2. □

Definition 4. *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.*

Proof. Suppose for contradiction (φ, X) is a winweb but there exists a subformula (φ', X') that is not a winweb. Now (φ', X') is a cobweb and there exists a move for F in (φ', X') such that for every move for T the residual formula is not a cobweb anymore, i.e., at least one of the subformulas *(1–5)* exists. We claim that F's move in (φ', X') must already create at least one of the subformulas *(1–5)*, because otherwise it would remain a cobweb where there exists an isolated or spare variable (Observation 1) which T could safely play, and removing an isolated variable or a clause never creates a new subformula *(1–5)*. Now the assumption becomes: F's move in (φ', X') creates at least one of the subformulas *(1–5)* and then after any possible T's move there again exists at least one of the subformulas *(1–5)*. F can use the same strategy in (φ, X). If T responds in X' then there exists at least one of the subformulas *(1–5)*. If T responds outside X' then it is a futile move since there already exists at least one of the subformulas *(1–5)* with variables in X'. That means (φ, X) is not a winweb since T cannot ensure that it remains a cobweb after a round. □

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), $\ldots$, (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. □

4 $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$

In this section, we consider the game where F has the first move but T has the last move. The key difference is, since T moves last F cannot win in the subformula *(2)* case 2 that appeared in Lemma 1 (Claim 2) because moving last was necessary for F to win. In fact, neither player has an incentive to be the first to play in such a subformula: if T plays one of the variables first then F can immediately win by playing the other variable, and if F plays one of the variables first then T can immediately play the other variable so that both clauses are satisfied. Intuitively, both players would prefer to delay playing in such a subformula until the other touches it first. Henceforth we use the term "delay" to capture this and related situations. Specifically, subformula *(4)* case 3 smallest (Claim 4) and subformula *(5)* case 4 smallest (Claim 5) also do not yield a win for F when T moves last, since the winning strategies relied on subformula *(2)* case 2 when F had the last move. Analogous to Lemma 1, Lemma 5 will characterize when F can win a game with T moving last. In Lemma 5 the first five subformulas are almost identical to Lemma 1, except these three excluded cases. For convenience we give names to these excluded cases.

A **mirror** is two width-2 clauses sharing both variables. A **delay mirror** happens when both connections are mixed. For example $(x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2)$ and $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_2)$ are both delay mirrors. An **isolated delay mirror** has no connection to any other clauses. In Lemma 1, delay mirror appeared as subformula *(2)* case 2. In Lemma 5, delay mirror is excluded from subformula *(2)* but reappears in subformula *(6)*.

A **pyramid** is one width-2 and one width-3 clause, sharing both of the width-2 clause's variables. A **delay pyramid** happens when both connections are mixed. For example $(x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3)$ and $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_2 \vee x_3)$ are both delay pyramids. In Lemma 1, delay pyramid appeared as subformula *(4)* case 3 smallest. In Lemma 5, delay pyramid is excluded from subformula *(4)* but reappears in subformulas *(6)*, *(7)*, and *(8)*.

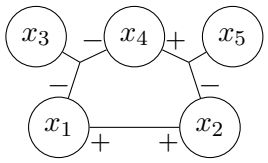
Recall a **diamond** is two width-3 clauses sharing two variables. A **delay diamond** happens when both connections are mixed. For example $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_4)$ and $(x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_4)$ are both delay diamonds. A **double delay diamond** happens when two delay diamonds share exactly one variable. In Lemma 1, delay diamond appeared in subformula *(5)* case 4 smallest. In Lemma 5, delay diamond's case is excluded from subformula *(5)* but reappears in subformulas *(6)*, *(8)*, and *(9)*.

In summary for Lemma 5: subformula *(2)* will exclude delay mirror but it reappears in subformula *(6)*; subformula *(4)* will exclude delay pyramid but it reappears in subformulas *(6)*, *(7)*, and *(8)*; and subformula *(5)* will exclude delay diamond but it reappears in subformulas *(6)*, *(8)*, and *(9)*.

Lemma 5. F *has a winning strategy in a* $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$ *game iff* F *can ensure within one round at least one of the following subformulas exists.*

*(1) A width-*0 *or width-*1 *clause.*

Subformula *(4)* case 3 smallest:



Subformula *(5)* case 4 smallest:

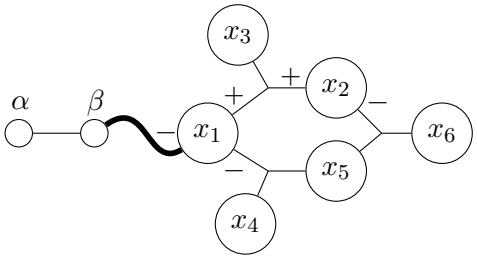


Figure 11: Changed cases of subformula *(4)* and subformula *(5)*

(2) Two width-2 clauses sharing both variables (excluding delay mirror).

(3) Two width-2 clauses and a chain (of length $\geqslant 0$) between them.

(4) A width-2 clause and a chain (of length $\geqslant 1$) between its two variables with at least one mixed connection between the chain and the width-2 clause (excluding delay pyramid).

(5) A width-2 clause, a cycle or diamond (excluding delay diamond) containing at most one width-2 clause variable, and a chain (of length $\geqslant 0$) between them.

(6) A delay diamond sharing exactly one variable with either a delay mirror or a delay pyramid.

(7) A delay pyramid and between its two shared variables: either a chain of length $\geqslant 3$, or a chain of length 2 with a pure connection in the middle of this chain.

(8) A delay pyramid and between its two shared variables a chain of length 2 with a mixed connection in the middle of this chain, and furthermore: either another such chain, or another delay pyramid with the same width-2 clause, or a delay diamond containing the mixed connection variable in the middle of the chain.

(9) A width-2 clause, a double delay diamond containing at most one width-2 clause variable, and a chain (of length $\geqslant 0$) between them.

Moreover, if subformula (4), (5), (6), (7), (8), or (9) 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 5 is in Section 4.1 and Section 4.2. Analogously to Corollary 1, we have:

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

Corollary 2 yields a direct approach to devise a log-space algorithm for $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$: brute force the first three rounds and check whether subformula *(1)* or *(2)* or *(3)* exists in the residual formula. We conjecture the same algorithm (possibly with a different number of brute-force rounds) actually solves the general game $\mathrm{G}_{3,\mathrm{F}\cdots\mathrm{T}}$; we are not aware of any counterexamples.

4.1 Right-to-left implication of Lemma 5

Suppose at least one of the subformulas *(1–9)* 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.

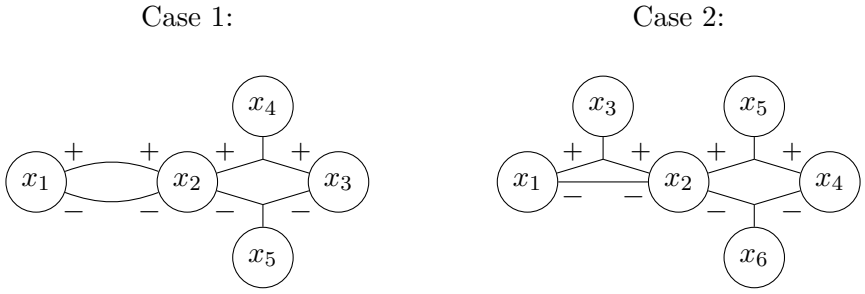


Figure 12: Subformula *(6)* (Claim 8)

The proofs of Claim 1 to Claim 5 from Lemma 1 apply to subformulas *(1–5)* for Lemma 5 except the excluded cases in subformulas *(2)*, *(4)*, and *(5)* (in all other cases, F's winning strategy did not rely on F moving last). In Figure 11, the new smallest in case 3 of subformula *(4)* and the new smallest in case 4 of subformula *(5)* are shown.

Claim 8. *If subformula (6) exists,* F *has a winning strategy.*

Proof. There are two cases, shown in Figure 12.

- Case 1: *The delay diamond shares a variable with a delay mirror.* F can play $x_4 = 0$. This creates manrikis $(x_1 \vee x_2) \wedge (x_2 \vee x_3)$ and $(\overline{x}_1 \vee \overline{x}_2) \wedge (x_2 \vee x_3)$. If T plays x_1 or x_2, F wins by making $x_1 = x_2$. If T plays $x_3 = 0$, F wins by $x_2 = 0$. If T plays $x_3 = 1$, F wins in either of the new manrikis $(x_1 \vee x_2) \wedge (\overline{x}_2 \vee x_5)$ or $(\overline{x}_1 \vee \overline{x}_2) \wedge (\overline{x}_2 \vee x_5)$. If T plays any other variable, F wins in either of the two existing manrikis.
- Case 2: *The delay diamond shares a variable with a delay pyramid.* F can play $x_5 = 0$. This creates a manriki $(\overline{x}_1 \vee \overline{x}_2) \wedge (x_2 \vee x_4)$. If T plays $x_1 = 1$ or $x_2 = 1$ or $x_2 = 0$ or $x_4 = 0$ then F wins by $x_2 = 1$ or $x_1 = 1$ or $x_4 = 0$ or $x_2 = 0$. If T plays $x_1 = 0$ then F wins in the new manriki $(x_3 \vee x_2) \wedge (x_2 \vee x_4)$. If T plays $x_4 = 1$ then F wins in the new manriki $(\overline{x}_1 \vee \overline{x}_2) \wedge (\overline{x}_2 \vee x_6)$. If T plays any other variable, F wins in the existing manriki. □

Claim 9. *If subformula (7) exists,* F *has a winning strategy.*

Proof. We reformulate subformula *(7)* with the following two cases, shown in Figure 13.

- Case 1: *There is a chain of length ⩾ 2 with a pure connection somewhere in the interior of the chain.* In the figure, x_4 forms a pure connection. F can play $x_4 = 0$. This creates manrikis from $(x_1 \vee x_2)$ to x_4's left end, and from $(x_1 \vee x_2)$ to x_4's right end. If T plays $x_1 = 0$ or $x_2 = 0$ then F wins by $x_2 = 0$ or $x_1 = 0$. If T plays $x_1 = 1$ or $x_2 = 1$ then it creates a manriki involving x_3, where F can win. If T plays any other variable, at least one of the two existing manrikis survives where F can win.
- Case 2: *There is a chain of length ⩾ 3 and all its interior connections are mixed.* F can play $x_6 = 0$. This creates two manrikis from $(x_1 \vee x_2)$ to $(x_4 \vee x_5)$. If T plays $x_1 = 0$ or $x_2 = 0$ or $x_4 = 0$ or $x_5 = 0$ then F wins by $x_2 = 0$ or $x_1 = 0$ or $x_5 = 0$ or $x_4 = 0$. If T plays $x_1 = 1$ or $x_2 = 1$ or $x_4 = 1$ or $x_5 = 1$ then it creates another manriki, where F can win. If T plays any other variable, at least one of the two existing manrikis survives where F can win. □

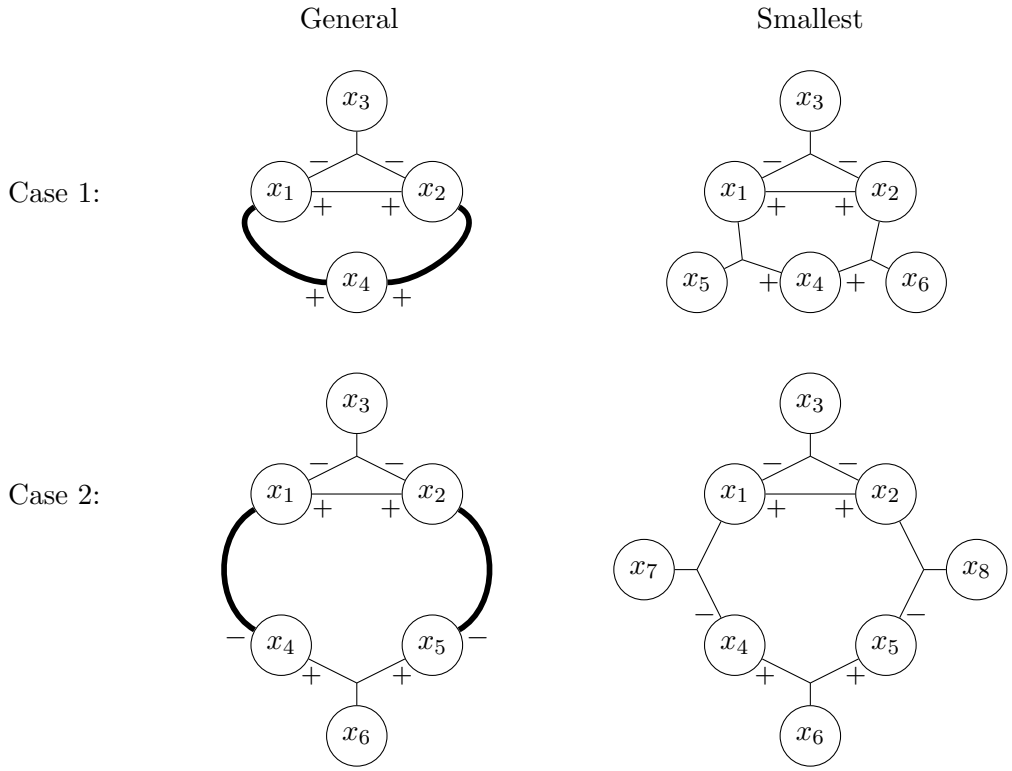


Figure 13: Subformula *(7)* (Claim 9)

Claim 10. *If subformula (8) exists,* F *has a winning strategy.*

Proof. There are three cases, shown in Figure 14.

- Case 1: *Another such chain.* We may assume these chains have pure connections to the width-2 clause, since otherwise that would form subformula *(4)*. We may also assume the chains do not have the same interior connection variable as each other, since otherwise that would form subformula *(5)* or *(6)*. F can play $x_3 = 0$. This creates two instances of subformula *(4)* case 3 (new smallest). If T plays x_1 or x_2, F wins by making $x_1 = x_2$. If T plays any other variable, at least one instance of subformula *(4)* case 3 survives where F can win by Claim 4.
- Case 2: *Another delay pyramid with the same width-*2 *clause.* F can play $x_6 = 0$. This creates a manriki $(x_1 \vee x_2) \wedge (x_2 \vee \overline{x}_4)$. If T plays $x_1 = 0$ or $x_2 = 0$ or $x_4 = 1$ then F wins by $x_2 = 0$ or $x_1 = 0$ or $x_2 = 0$. If T plays $x_1 = 1$ then it creates a manriki $(x_3 \vee \overline{x}_2) \wedge (\overline{x}_2 \vee x_7)$ where F can win. If T plays $x_2 = 1$ then it creates a manriki $(x_3 \vee \overline{x}_1) \wedge (\overline{x}_1 \vee x_7)$ where F can win. If T plays $x_4 = 0$ then it creates a manriki $(x_5 \vee x_1) \wedge (x_1 \vee x_2)$ where F can win. If T plays any other variable, F wins in the existing manriki.
- Case 3: *A delay diamond containing the mixed connection variable in the middle of the chain.* We may assume the delay diamond contains neither x_1 nor x_2 since otherwise that would form subformula *(6)*. F can play $x_8 = 0$. This creates two manrikis from $(x_7 \vee x_4)$ to $(x_1 \vee x_2)$. If T plays $x_1 = 0$ or $x_2 = 0$ or $x_4 = 0$ or $x_7 = 0$ then F wins by $x_2 = 0$ or $x_1 = 0$ or $x_7 = 0$ or $x_4 = 0$. If T plays $x_1 = 1$ then it creates a manriki from $(x_3 \vee \overline{x}_2)$ to $(x_4 \vee x_7)$ where F can

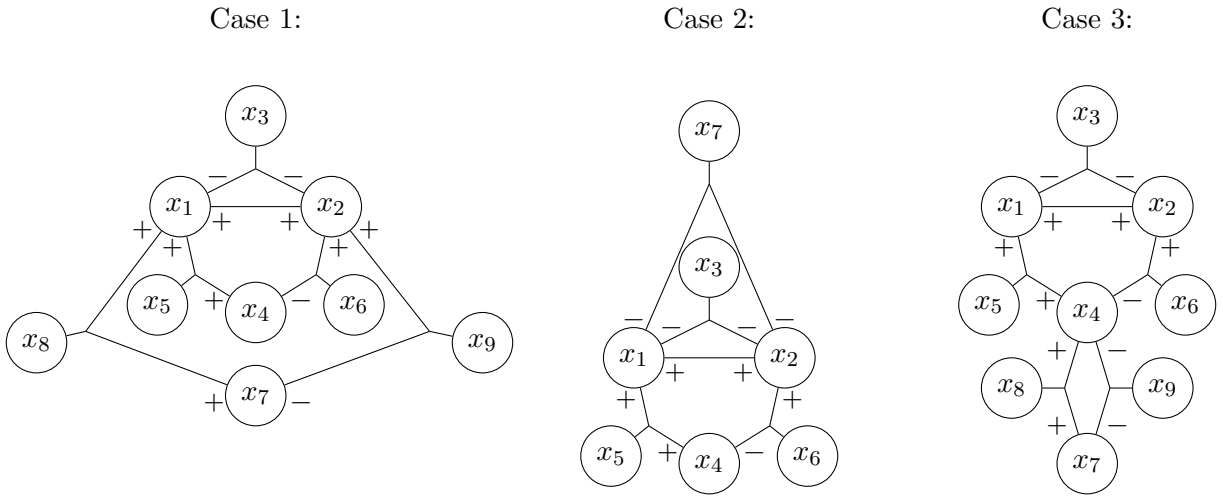


Figure 14: Subformula *(8)* (Claim 10)

win. If T plays $x_2 = 1$ then it creates a manriki from $(x_3 \vee \overline{x}_1)$ to $(x_4 \vee x_7)$ where F can win. If T plays $x_4 = 1$ then it creates a manriki $(x_6 \vee x_2) \wedge (x_2 \vee x_1)$ where F can win. If T plays $x_7 = 1$ then it creates a manriki from $(x_9 \vee \overline{x}_4)$ to $(x_1 \vee x_2)$ where F can win. If T plays any other variable, at least one of the two existing manrikis survives where F can win. □

Claim 11. *If subformula (9) exists,* F *has a winning strategy.*

Proof. There are two cases depending on whether the width-2 clause is chain-connected to a side (case 1) or midpoint (case 2) of the double delay diamond, shown in Figure 15.

For both cases F's winning strategy is: F can play $x_4 = 0$. This creates a manriki from $(\alpha \vee \beta)$ to $(\overline{x}_1 \vee \overline{x}_2)$. If T plays outside the manriki then F can win the manriki. If T plays $x_1 = 1$ or $x_2 = 1$ then F wins by $x_2 = 1$ or $x_1 = 1$. If T plays $x_1 = 0$, then it creates another manriki from $(\alpha \vee \beta)$ to x_1's left end (case 1) or upper end (case 2) where F can win—or if the chain had length 0 (so x_1 is β), then F can win the width-1 clause (α). If T plays $x_2 = 0$, then it creates another manriki from $(\alpha \vee \beta)$ to $(x_1 \vee x_3)$ where F can win. If T plays outside the double delay diamond then it leaves subformula *(6)* case 2 where F can win by Claim 8. □

Moreover, in all cases, there exists a subformula *(1)* or *(2)* or *(3)* within two rounds.

4.2 Left-to-right implication of Lemma 5

Many concepts in this section are analogous to concepts from Section 3.2. To distinguish them from the versions in Section 3.2, we generally prefix the terms with "d-" (standing for "delay") to emphasize that the various delay subformulas (mirrors, pyramids, diamonds) are the key difference.

Definition 5. *A **d-cobweb** is a formula where none of the subformulas (1–9) exist (and each width-3 clause has at least one spare variable). Note that any subformula in a d-cobweb is also a d-cobweb.*

Definition 6. *We say a formula is **dead** if it consists only of isolated delay mirrors (having no clauses or variables beyond those involved in the delay mirrors) and **live** otherwise. Note that a dead formula is a d-cobweb with an even number of variables.*

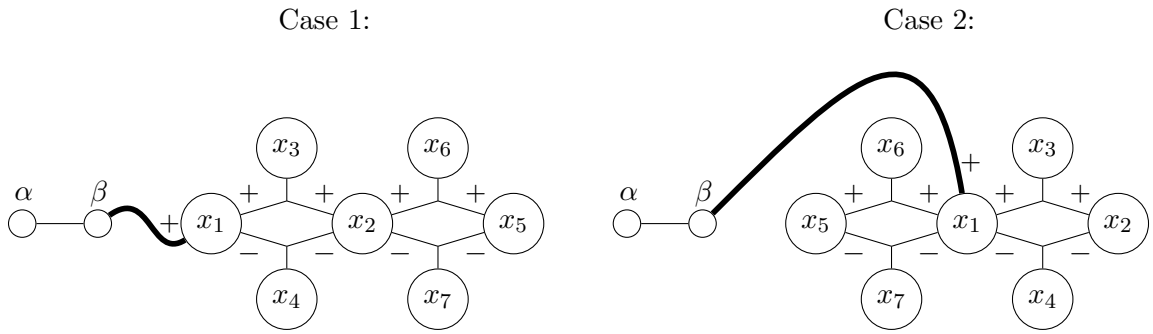


Figure 15: Subformula *(9)* (Claim 11)

Observation 2. *If the formula is dead then* T *has a winning strategy in* $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$.

Proof. Whenever F plays, it is in an isolated delay mirror and satisfies one of the two clauses. T can respond by playing the other variable in the same delay mirror to satisfy the other clause. All clauses get satisfied in this way, so T wins. □

Observation 2 says T's goal should be to make the formula dead, since that ensures T can win.

Observation 3. *A live d-cobweb has a variable that occurs in at most one clause.*

Proof. If the d-cobweb has a width-3 clause then the spare variable in it occurs in only one clause. Suppose there is no width-3 clause but there exists a width-2 clause. Then every width-2 clause must be either isolated (have no connections to other clauses) or part of an isolated delay mirror, since none of subformulas *(1)* or *(2)* or *(3)* exist. Since the d-cobweb is live, there must be either an isolated width-2 clause (any variable in it occurs only once) or an isolated variable (which occurs in no clauses). If there are no width-3 or width-2 clauses then the d-cobweb has only isolated variables. □

To prove the left-to-right implication of Lemma 5, suppose F cannot ensure that at least one of the subformulas *(1–9)* exists within one round. So at the beginning the formula is a d-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 d-cobweb. In other words, T can ensure that the beginning d-cobweb remains a d-cobweb after a round. We will argue that T has a winning strategy. The proof will be by induction on the number of variables. As in Section 3.2, for the induction to go through we need to allow F to use pass moves (meaning F has the option of forgoing any turn, thus forcing T to play multiple variables in a row). However, this means T is no longer guaranteed to have the last move, which causes an issue with delay mirrors, where T may wish to rely on moving last in order to satisfy those clauses. To deal with this, we consider a further modification of the game, which we call "$\mathrm{G}^*_{3,\mathrm{F}\cdots}$ ignoring isolated delay mirrors": the new rule is that whenever an isolated delay mirror appears in the residual formula (or exists at the beginning of the game), it immediately vanishes—its two variables are no longer available to play. (An isolated delay mirror is moved to a special forbidden "graveyard" comprising the dead part of the formula.) This means the game ends and T wins when the residual formula is dead (it consists entirely of isolated delay mirrors, or no variables remain). We design a winning strategy for T assuming F can use pass moves and ignoring isolated delay

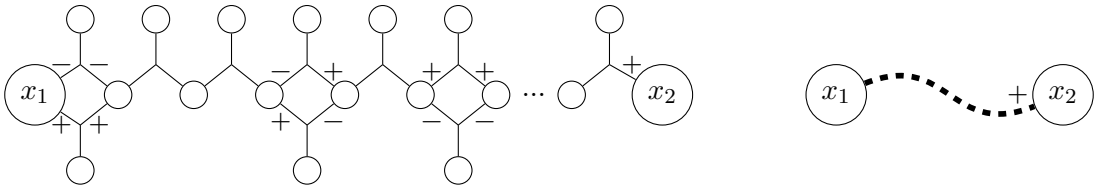


Figure 16: A d-chain between x_1 and x_2

mirrors. At the end of this section, we will recover from this a proof of the left-to-right implication of Lemma 5 for the unmodified $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$ game, using the fact that T can win the graveyard if he has the last move, like in Observation 2.

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

Definition 7.

- *A **d-jellyfish** is a connected d-cobweb with a width-2 clause. Its **eyes** are the variables in the width-2 clause.*
- *A **d-chain** is a sequence of distinct "links", where each link is a width-3 clause or a delay diamond (and each clause has a spare variable), such that: consecutive links share exactly one variable, non-consecutive links share no variables, and no consecutive links are both delay diamonds (i.e., no double delay diamond). An arbitrary d-chain between x_1 and x_2 is illustrated in Figure 16 on the left. On the right, we show how the d-chain can be depicted by a thick dotted line. (d-Chains play a role analogous to chains in Section 3 shown in Figure 3.)*
- *A **d-tree** is a connected formula with only width-3 clauses where there is no cycle, no non-delay diamond, and no double delay diamond (and each clause has a spare variable). Informally a d-tree is like a tree where each path is a d-chain, so we also depict arbitrary d-trees using thick dotted lines. The smallest case of d-tree is a single variable. (d-Trees play a role analogous to single trees from Definition 3.)*
- *A **d-ocean** is a formula where each connected component is either a d-jellyfish or a d-tree.*

Lemma 6. *If the formula is a d-jellyfish then* T *has a winning strategy in* $\mathrm{G}^*_{3,\mathrm{F}\cdots}$ *even if* F *can use pass moves, ignoring isolated delay mirrors.*

Proof. The proof of the analogous Lemma 2 had a simple structure: T could ensure that a jellyfish became a win-forest after one round, and that a win-forest remained a win-forest after each subsequent round. This simple structure does not work now—after many rounds there may still be d-jellyfish in the residual formula. Instead, we will argue that T can maintain that after each round, each connected component is either a d-jellyfish or a d-tree (i.e. the formula is a d-ocean).

First we argue what a d-jellyfish must look like. A d-jellyfish does not have any width-0 or width-1 clause because that would form subformula *(1)*. Now there are four cases.

- Case 1: *There is another width-2 clause.* The only possibility is that there are exactly two width-2 clauses and they form a delay mirror (in particular, the eyes are uniquely determined) because otherwise that would form subformula *(2)* or *(3)* (since a d-jellyfish is connected). There cannot be any chain of length $\geqslant 2$ between the eyes because that would form subformula *(4)* with at least one of the width-2 clauses, and similarly there cannot be any chain of length 1 between the eyes that does not form a delay pyramid with one of the width-2 clauses. There

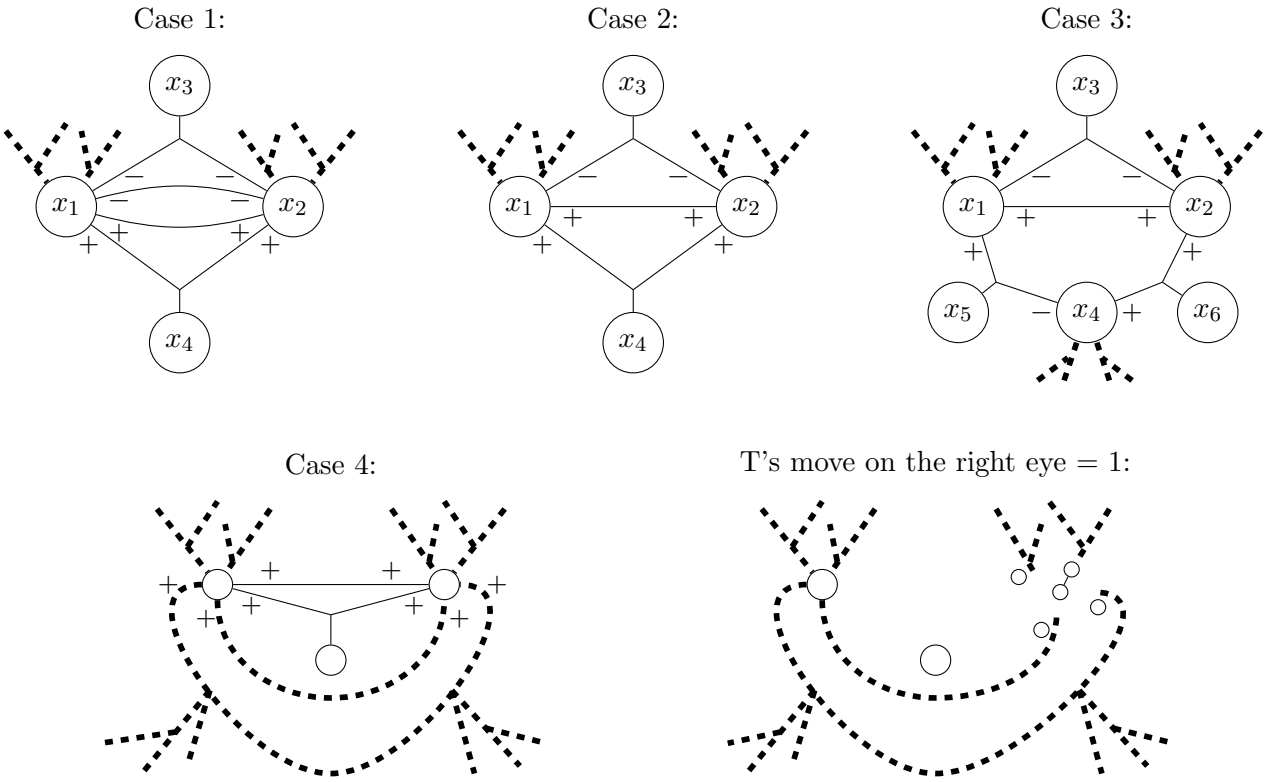


Figure 17: d-Jellyfish

cannot be any cycle or non-delay diamond containing at most one eye, or any double delay diamond containing at most one eye, because that would form subformula *(5)* or *(9)* (since a d-jellyfish is connected); i.e., the part hanging off of an eye must be a d-tree. Furthermore, there cannot be any delay diamond containing exactly one eye because that would form subformula *(6)*.

In summary: A d-jellyfish case 1 has one delay mirror (eyes). Both of the width-2 clauses may participate in any number of delay pyramids. Rooted at each eye can be a d-tree, but this d-tree must have no delay diamond touching the eye. This characterization is illustrated in Figure 17 case 1.

In the remaining cases, there is only one width-2 clause, and we assume without loss of generality that both eyes are positive literals in it.

- Case 2: *There is only one width-*2 *clause, it is in a delay pyramid, and there is no chain of length* $\geqslant 2$ *between the eyes.* There cannot be any width-3 clause with one pure and one mixed connection to the width-2 clause because that would form subformula *(4)*. Like case 1, the part hanging off of an eye must be a d-tree and there cannot be any delay diamond containing exactly one eye.

  In summary: A d-jellyfish case 2 has one width-2 clause (eyes). This clause participates in at least one delay pyramid, and in any number of pyramids with pure connections at both eyes. Rooted at each eye can be a d-tree, but this d-tree must have no delay diamond touching

the eye. In other words, case 2 looks similar to case 1 except one of the delay mirror's clauses is now missing, and the presence of a delay pyramid is now mandatory. This characterization is illustrated in Figure 17 case 2.

- Case 3: *There is only one width-2 clause, it is in a delay pyramid, and there is a chain of length $\geqslant 2$ between the eyes.* There cannot be any width-3 clause with one pure and one mixed connection to the width-2 clause because that would form subformula *(4)*. For any chain of length $\geqslant 2$ between the eyes, it cannot have any mixed connection to the width-2 clause because otherwise that would form subformula *(4)*, and it cannot have length $\geqslant 3$ or have length 2 with a pure connection in the middle because otherwise that would form subformula *(7)*; i.e., the chain must have length 2 with pure connections to the width-2 clause at both eyes and a mixed connection in the middle. There can only be one such chain, and the width-2 clause can only be in one delay pyramid, because otherwise that would form subformula *(8)*. We call the mixed connection variable in this unique chain the **nose**. The part hanging off of the nose must be disjoint from the parts hanging off of the eyes because otherwise that would form subformula *(5)*. Like cases 1 and 2, the part hanging off of an eye or off of the nose must be a d-tree and there cannot be any delay diamond containing exactly one eye. Furthermore, there cannot be any delay diamond containing the nose because otherwise that would form subformula *(8)*.

  In summary: A d-jellyfish case 3 has one width-2 clause (eyes). This clause participates in exactly one delay pyramid, and in any number of pyramids with pure connections at both eyes. Between the eyes there is one chain of length 2, which has pure connections to the width-2 clause and a mixed connection in the middle (nose). Rooted at each eye and at the nose can be a d-tree, but this d-tree must have no delay diamond touching the eye or nose. This characterization is illustrated in Figure 17 case 3 (though for clarity, this illustration happens not to show any pyramids with pure connections to the width-2 clause).

- Case 4: *There is only one width-2 clause, and it is not in a delay pyramid.* The two eyes can be connected by chains (of length $\geqslant 1$) but those connections to the width-2 clause must be pure because any mixed connection at the eyes would form subformula *(4)* or delay pyramid. These chains may overlap but we claim that the only way these chains can overlap is by forming d-chains between the eyes, and that these d-chains must be disjoint from each other except at the eyes themselves. This is because supposing the claim is not true, i.e., two distinct such chains overlap (share a non-eye variable) but are not consistent with a d-chain, then it would form subformula *(5)* or *(9)* in one of several ways: Following the two chains from an eye, either they form a double delay diamond chain-connected to the eye, or we consider the first place they diverge other than delay diamonds: It either forms a non-delay diamond chain-connected to the eye, or it opens a cycle chain-connected to the eye but not containing the other eye, or it is part of a cycle containing the other eye but not containing the first eye. This shows the claim. Finally, like the other cases, the part hanging off of an eye or off of a non-spare variable on one of the d-chains must be a d-tree, but these attachments cannot create any double delay diamond because otherwise that would form subformula *(9)*.

  In summary: A d-jellyfish case 4 has one width-2 clause (eyes). There can be any number of disjoint d-chains of length $\geqslant 1$ between the two eyes (body), and those d-chains must have only pure connections to the width-2 clause (in particular, none of them can have a delay diamond touching an eye). There can be d-trees hanging off at any non-spare variables as long as no double delay diamonds are present anywhere. A d-tree hanging off of an eye can

have pure or mixed connections at the eye and may have a delay diamond touching the eye. This characterization is illustrated in Figure 17 case 4.

Claim 12. T *can ensure that a d-ocean remains a d-ocean after a round even if* F *can use pass moves, ignoring isolated delay mirrors.*

Proof. The proof is similar to Claim 6 and Claim 7 except it is more involved. The argument will show that whatever F plays, whether a pass move or in a d-tree or any case of d-jellyfish, T has a response such that each connected component of the residual formula is again a d-tree or some case of d-jellyfish; therefore the residual formula is again a d-ocean. Any move by T or F in a d-tree can occur in five different scenarios as illustrated in Figure 18 (where we suppose the move is $x_1 = 0$). Specifically, Scenario 1 is a move on a non-spare variable, and the others are moves on a spare variable: in Scenarios 2 and 3 the clause is not in a delay diamond while in Scenarios 4 and 5 the clause is in a delay diamond, and in Scenarios 2 and 4 the clause is not satisfied by the move (it shrinks to a width-2 clause) while in Scenarios 3 and 5 the clause is satisfied. (Normally we assume a spare variable is a positive literal, but to illustrate satisfying a clause with $x_1 = 0$ we let it be a negative literal in Scenarios 3 and 5 in Figure 18.)

- **Pass move:** Suppose F played a pass move. Since a d-ocean is a d-cobweb (and we may assume it is live since the game ignores isolated delay mirrors), there exists an isolated or spare variable (Observation 3). T can play that isolated/spare variable to remove the isolated variable or satisfy a clause. If T plays an isolated variable then no harm in it and this preserves the d-ocean property. If T plays a spare variable in a d-tree then satisfying the clause produces either two smaller d-trees by Scenario 3 (not in a delay diamond) or one d-tree by Scenario 5 (in a delay diamond). Now suppose T plays a spare variable in a width-3 clause in a d-jellyfish. Satisfying the clause can produce at most two connected components. If it is still one connected component, this component must again be a d-jellyfish since it still has a width-2 clause and no subformula *(1–9)* has been created. If two connected components are produced, one of them must have all the original d-jellyfish's width-2 clauses (and is again a d-jellyfish like in the one-component case) while the other must be a d-tree since it has no width-2 clause and no cycle or non-delay diamond (otherwise the original d-jellyfish would have had subformula *(5)*) and no double delay diamond (otherwise the original d-jellyfish would have had subformula *(9)*). Finally, suppose T plays a spare variable in a d-jellyfish's width-2 clause, satisfying the clause. This must produce one component, which is a d-tree: there can be no more width-2 clauses (otherwise the original d-jellyfish would have had subformula *(3)*) and no cycle, non-delay diamond, or double delay diamond (otherwise the original d-jellyfish would have had subformula *(5)* or *(9)*). This preserves the d-ocean property.
- **d-Tree:** Suppose F played in a d-tree. F's move could occur in the five different scenarios. In Scenario 1, F's move can create some isolated variables (which are d-trees), some other d-trees, and some d-jellyfish case 4 (with no body d-chains and no d-tree rooted at one eye). In Scenario 2, F's move creates one d-jellyfish case 4 (with no body d-chains). In Scenario 3, F's move creates two d-trees. In Scenario 4, F's move creates one d-jellyfish case 2 (note that no delay diamond touches either eye, since there were no double delay diamonds in the original d-tree). In Scenario 5, F's move creates one d-tree. In all scenarios it still remains a d-ocean after F's move, so T can now just play as if F had used a pass move on this d-ocean.

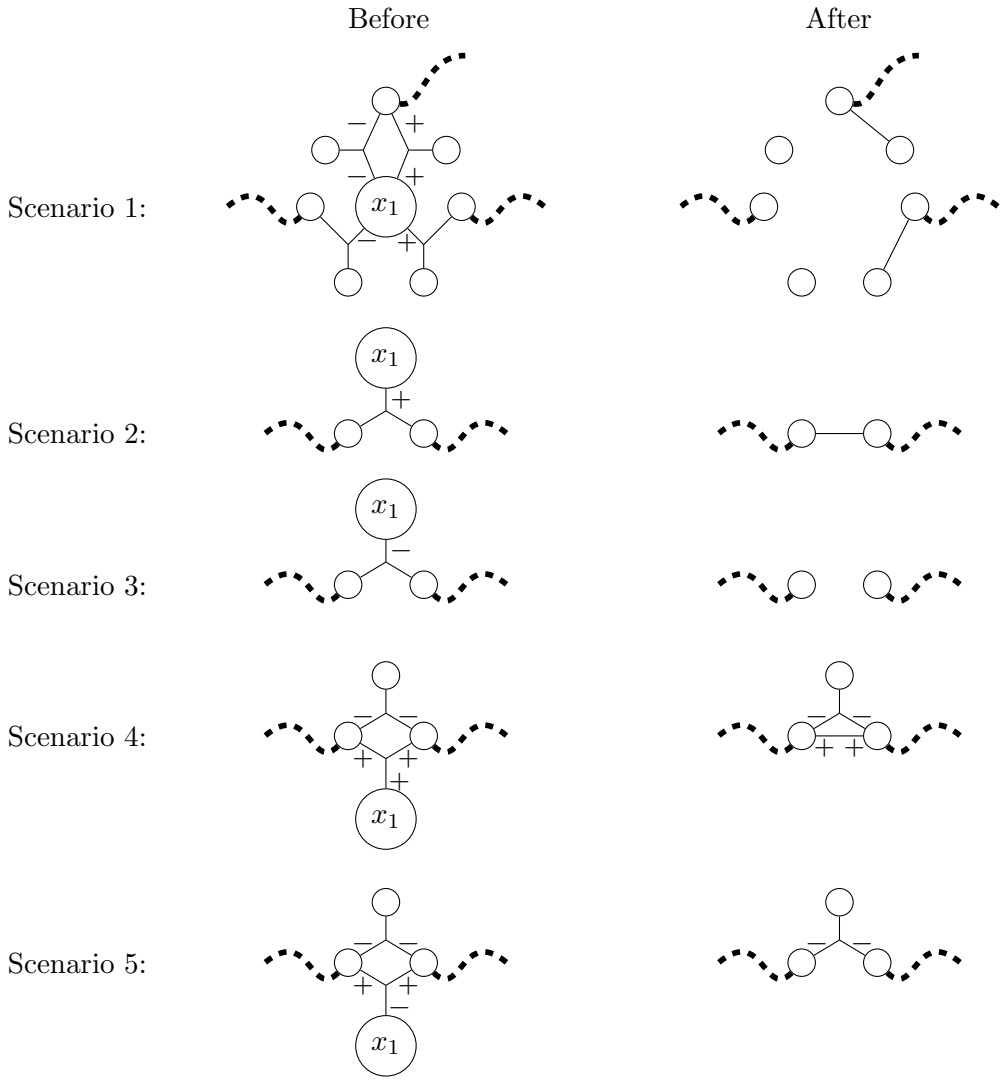


Figure 18: Move $x_1 = 0$ and its effect on d-trees

- **d-Jellyfish case 1:** Suppose F played in a d-jellyfish case 1.

  If F played the spare variable of a pyramid then this width-3 clause disappears (it either is satisfied or shrinks to recreate one of the delay mirror's clauses) so the component is still a d-jellyfish case 1, and the formula is still a d-ocean. T can play as if F had used a pass move.

  If F played either eye, T can play the other eye so that all delay mirror and pyramid clauses are now satisfied (by either F's move or T's move). The d-tree hanging off of either eye gets broken and separated by Scenario 1. As explained earlier, such a move on a d-tree results in components each of which is a d-tree or a d-jellyfish case 4, so this preserves the d-ocean property.

  If F played a spare variable in a non-pyramid clause containing an eye, then T can play this clause's third variable x_i (not the eye or the spare variable F played) to satisfy the clause. This breaks off part of the d-tree hanging off of the eye (namely the sub-d-tree rooted at x_i)

which gets separated by Scenario 1 into components each of which is a d-tree or a d-jellyfish case 4. Because the clause containing the eye was not part of a delay diamond, the rest of the original component remains a d-jellyfish case 1 with a smaller d-tree hanging off of one eye. This preserves the d-ocean property.

Otherwise, F played a variable x_i in the d-tree hanging off of an eye, but not a spare variable in a clause containing the eye. T can locate the d-chain from x_i to the eye and play the spare variable in the clause (on this d-chain) containing the eye, to satisfy the clause. (This clause is unique since the d-tree had no delay diamond containing the eye.) To see that the d-ocean property is preserved, we pretend that T's move happened before F's move, and consider the formula after T's move but before F's move. T's move breaks off a sub-d-tree (containing x_i) and leaves the original component as a d-jellyfish case 1, so this formula is a d-ocean. As explained earlier, any move by F on the sub-d-tree preserves the d-ocean property.

- **d-Jellyfish case 2:** Suppose F played in a d-jellyfish case 2. If F played the spare variable of a pyramid then the component becomes a d-jellyfish, either case 1 (if a delay mirror is created) or case 2 (if there remains a delay pyramid) or case 4 (if there is no delay pyramid). Since the formula is still a d-ocean, T can play as if F had used a pass move. If F played any other variable then T can use the exact same strategy explained for d-jellyfish case 1.
- **d-Jellyfish case 3:** Suppose F played in a d-jellyfish case 3.

  If F played in the d-tree hanging off of either eye or off of the nose (excluding playing the eyes or nose themselves) then the same strategy explained for d-jellyfish case 1 works in this situation (using the fact that these d-trees have no delay diamond touching the eye or nose).

  If F played the spare variable in a pyramid with pure connections to the width-2 clause, then this width-3 clause disappears (it either is satisfied or shrinks to recreate the width-2 clause) so the component is still a d-jellyfish case 3, and the formula is still a d-ocean. T can play as if F had used a pass move.

  The only remaining possibilities for F's move are on the variables $x_1, x_2, x_3, x_4, x_5, x_6$ shown in Figure 17 case 3.

  If F played either eye (x_1 or x_2) then T can play the opposite value for the other eye (x_2 or x_1). Now the width-2 clause and all pyramid clauses are satisfied, and as explained earlier, the d-trees hanging off of the eyes get broken by Scenario 1 into components each of which is a d-tree or a d-jellyfish case 4. One of the two clauses in the body chain gets satisfied (leaving its spare variable isolated) while the other shrinks to a width-2 clause which, together with the nose's d-tree, forms a d-jellyfish case 4 (where one eye has no d-tree, the other eye is the former nose, and there are no body d-chains). The d-ocean property is preserved.

  If F played the spare variable (x_3) of the delay pyramid then T can play $x_6 = 1$. If F's move was $x_3 = 0$ then after T's move the component is a d-jellyfish case 1 where the nose (x_4) and its d-tree, as well as x_5, have become part of the d-tree hanging off of the eye x_1. If F's move was $x_3 = 1$ then this similarly results in a d-jellyfish case 4. The d-ocean property is preserved.

  If F played the nose $x_4 = 0$ or $x_4 = 1$ then T can play $x_6 = 1$ or $x_5 = 1$. Now both body chain clauses are satisfied (and one of their spare variables is isolated), the component containing the delay pyramid becomes a d-jellyfish case 2, and as explained earlier, the d-tree hanging off of the nose gets broken by Scenario 1 into components each of which is a d-tree or a d-jellyfish case 4. The d-ocean property is preserved.

If F played $x_5 = 0$ then T can play $x_1 = 1$. This satisfies the width-2 clause and the body chain clause containing x_5, and as explained earlier, the d-tree hanging off of x_1 gets broken by Scenario 1 into components each of which is a d-tree or a d-jellyfish case 4. The rest of the component becomes a d-jellyfish case 4 with width-2 clause $(x_3 \vee \overline{x}_2)$, where the new eye x_3 has no d-tree, and the eye x_2's d-tree now includes the nose (x_4) and its d-tree, as well as x_6. Similarly, if F played $x_6 = 0$ then T can play $x_2 = 1$ and the d-ocean property is preserved.

If F played $x_5 = 1$ (satisfying the clause and breaking the body chain) then the component becomes a d-jellyfish case 2 where the nose (x_4) and its d-tree, as well as x_6, have become part of the d-tree hanging off of the eye x_2. After that, T faces a d-ocean and T can play as if F had used a pass move. Similarly, if F played $x_6 = 1$ then T can preserve the d-ocean property.

- **d-Jellyfish case 4:** Suppose F played in a d-jellyfish case 4. The strategy is analogous to Claim 7, but instead of producing single and married trees, it now produces some d-trees, at most one d-jellyfish case 2, and some d-jellyfish case 4. For completeness we rephrase the argument for the current setting.

  Whatever F's move is, it does not touch at least one of the two d-trees hanging off of the eyes. Without loss of generality we assume F did not play in the right eye's d-tree. Then T responds by playing the right eye to satisfy the width-2 clause. To prove the residual formula is a d-ocean, we pretend that T's move happened before F's move, and consider the formula after T's move but before F's move.

  The right eye move by T is shown in Figure 17 (after case 4). The width-2 clause gets satisfied. The right eye's d-tree can be broken into some d-trees and some d-jellyfish case 4 by Scenario 1 (even though the original d-tree may have had a delay diamond touching the eye). The rest of the connections at the right eye were only pure and they all came from the body d-chains. So when the body d-chains get broken at the right eye, they only create some isolated variables but no width-2 clauses. Then the broken body d-chains together with the left eye's d-tree and any other d-trees hanging off the broken body d-chains form one big d-tree. The residual formula as illustrated in Figure 17 (after case 4) is now a d-ocean.

  Now we consider how F's move could have affected this d-ocean. If F used a pass move, then no harm done, we already got a d-ocean. If F did not use a pass move then any move in a d-tree preserves the d-ocean property (as we already know). F's move could not happen in any of the current d-jellyfish components since those only came from the right eye's d-tree and F did not play there.

This finishes the proof of Claim 12. □

Now putting it all together: If the formula is a d-jellyfish then in particular it is a d-ocean, so T can always ensure that the residual formula is a d-ocean after each round by Claim 12. In the last round, if there exists only one remaining playable variable (ignoring isolated delay mirrors) then the only possibility is an isolated variable with no clauses since subformula *(1)* does not exist. T has already won in this case. If there are two remaining playable variables in the last round then either they are isolated and T has already won, or they are in an isolated width-2 clause which T can satisfy in one move. It is also possible that there are more than two remaining playable variables at the beginning of the last round and that the game ends due to the creation of isolated delay mirrors—T wins in this case as well. This finishes the proof of Lemma 6. □

Definition 8. *A **d-winweb** is a d-cobweb such that for every non-pass move by* F*, either the residual formula is dead or there exists a move for* T *after which it is again a d-cobweb. (Here we allow moves in isolated delay mirrors.)*

Lemma 7. *Every subformula of a d-winweb is also a d-winweb.*

Proof. Suppose for contradiction (φ, X) is a d-winweb but there exists a subformula (φ', X') that is not a d-winweb. Now (φ', X') is a d-cobweb and there exists a non-pass move by F in (φ', X') such that the residual formula is live and for every move for T the new residual formula is not a d-cobweb, i.e., at least one of the subformulas *(1–9)* exists. We claim that F's move in (φ', X') must already create at least one of the subformulas *(1–9)*, because otherwise it would remain a live d-cobweb where there exists an isolated or spare variable (Observation 3) which T could safely play, and removing an isolated variable or a clause never creates a new subformula *(1–9)*. Now the assumption becomes: F's move in (φ', X') creates at least one of the subformulas *(1–9)* and then after any possible T's move there again exists at least one of the subformulas *(1–9)*. F can use the same strategy in (φ, X). The residual formula after F's move in (φ, X) is live since it has a subformula *(1–9)*. If T responds in X' then there exists at least one of the subformulas *(1–9)*. If T responds outside X' then it is a futile move since there already exists at least one of the subformulas *(1–9)* with variables in X'. That means (φ, X) is not a d-winweb since T cannot ensure that it remains a d-cobweb after a round. □

Lemma 8. *If the formula is a d-winweb then* T *has a winning strategy in* $\mathrm{G}^*_{3,\mathrm{F}\cdots}$ *even if* F *can use pass moves, ignoring isolated delay mirrors.*

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

Base case: The formula is a d-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 an isolated width-2 clause which T can satisfy in one move, or an isolated delay mirror in which case T has won.

Induction step: The formula (φ, X) is a d-winweb with at least three variables. We assume (φ, X) is live since otherwise T has already won.

Suppose F played a pass move. There exists an isolated or spare variable since the formula is a live d-cobweb (Observation 3). 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 d-winweb by Lemma 7. 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 d-winweb, either the residual formula is dead in which case T has won, or T has a response such that the residual formula is a d-cobweb. T's move cannot have been in an isolated delay mirror since then subformula *(1)* would have been created. Call the residual formula after T's move (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, ignoring isolated delay mirrors:

- If (φ_i, X_i) has a width-2 clause then it is a d-jellyfish (since it is a connected d-cobweb) so by Lemma 6, T can win even if F can use pass moves, ignoring isolated delay mirrors.

- 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 d-winweb (φ, X) since no new width-3 clause can be created during the game. By Lemma 7, (φ_i, X_i) is also a d-winweb and hence by the induction hypothesis, T can win even if F can use pass moves, ignoring isolated delay mirrors.

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 there is no remaining playable variable in X_i because it has become dead. In the latter case, or if F played a pass move, T picks any other component (φ_j, X_j) with remaining playable variables and continues according to his winning strategy in that component, as if F had just played a pass move in that component.□

We are finally ready to prove the left-to-right implication of Lemma 5:

Lemma 9. *In a* $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$ *game, if* F *cannot ensure within one round at least one of the subformulas (1–9) exists, then* T *has a winning strategy.*

Proof. Our assumption is that at the beginning the formula is a d-cobweb and in the first round, for every (non-pass) move by F there exists a move for T such that the residual formula is again a d-cobweb. (Since there are an even number of variables, there must be a remaining variable for T to play after F's move in the first round.) In particular, the original formula is a d-winweb. By Lemma 8, T has a winning strategy even if F can use pass moves, ignoring isolated delay mirrors; in particular, T has a winning strategy when F cannot use pass moves but still ignoring isolated delay mirrors. To get a winning strategy in the unmodified $\mathrm{G}^*_{3,\mathrm{F}\cdots\mathrm{T}}$ game (not ignoring isolated delay mirrors), T can follow the strategy for the modified game, but whenever F plays a variable in an isolated delay mirror (necessarily satisfying one of its clauses), T immediately responds by playing the other variable to satisfy the other clause in the delay mirror. Because T has the last move, he will never be forced to make the first move in an isolated delay mirror—when the modified game ends and all that remains are isolated delay mirrors, it will be F's turn. □

Acknowledgments

This work was supported by NSF grant CCF-1657377.

References

[AO12] Lauri Ahlroth and Pekka Orponen. Unordered constraint satisfaction games. In *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 64–75. Springer, 2012.

[APT79] Bengt Aspvall, Michael Plass, and Robert Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[Bys04] Jesper Byskov. Maker-maker and maker-breaker games are PSPACE-complete. Technical Report RS-04-14, BRICS, Department of Computer Science, Aarhus University, 2004.

[Cal08] Chris Calabro. 2-TQBF is in P, 2008. Unpublished. URL: https://cseweb.ucsd.edu/~ccalabro/essays/complexity_of_2tqbf.pdf.

[Kut04] Martin Kutz. *The Angel Problem, Positional Games, and Digraph Roots*. PhD thesis, Freie Universität Berlin, 2004. Chapter 2: Weak Positional Games.

[Kut05] Martin Kutz. Weak positional games on hypergraphs of rank three. In *Proceedings of the 3rd European Conference on Combinatorics, Graph Theory, and Applications (EuroComb)*, pages 31–36. Discrete Mathematics & Theoretical Computer Science, 2005.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):17:1–17:24, 2008.

[RV05] Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 436–447. Springer, 2005.

[RW18] Md Lutfar Rahman and Thomas Watson. Complexity of unordered CNF games. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC)*, pages 9:1–9:12. Schloss Dagstuhl, 2018.

[Sch76] Thomas Schaefer. Complexity of decision problems based on finite two-person perfect-information games. In *Proceedings of the 8th Symposium on Theory of Computing (STOC)*, pages 41–49. ACM, 1976.

[Sch78] Thomas Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

[SM73] Larry Stockmeyer and Albert Meyer. Word problems requiring exponential time. In *Proceedings of the 5th Symposium on Theory of Computing (STOC)*, pages 1–9. ACM, 1973.