Complexity of Unordered CNF Games

Md Lutfar Rahman
The University of Memphis, Memphis, TN, USA
mrahman9@memphis.edu

Thomas Watson
The University of Memphis, Memphis, TN, USA
Thomas.Watson@memphis.edu

Abstract

The classic TQBF problem is to determine who has a winning strategy in a game played on a given CNF formula, where the two players alternate turns picking truth values for the variables in a given order, and the winner is determined by whether the CNF gets satisfied. We study variants of this game in which the variables may be played in any order, and each turn consists of picking a remaining variable and a truth value for it.

- For the version where the set of variables is partitioned into two halves and each player may only pick variables from his/her half, we prove that the problem is PSPACE-complete for 5-CNFs and in P for 2-CNFs. Previously, it was known to be PSPACE-complete for unbounded-width CNFs (Schaefer, STOC 1976).
- For the general unordered version (where each variable can be picked by either player), we also prove that the problem is PSPACE-complete for 5-CNFs and in P for 2-CNFs. Previously, it was known to be PSPACE-complete for 6-CNFs (Ahlroth and Orponen, MFCS 2012) and PSPACE-complete for positive 11-CNFs (Schaefer, STOC 1976).

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography, Theory of computation → Problems, reductions and completeness, Theory of computation

Keywords and phrases CNF, Games, PSPACE-complete, SAT, Linear Time

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.9

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2018/039/`.

Funding This work was supported by NSF grant CCF-1657377.

1 Introduction

Conjunctive normal form formulas (CNFs) are among the most prevalent representations of boolean functions. All sorts of computational problems concerning CNFs – such as satisfying them, minimizing them, learning them, refuting them, fooling them, and playing games on them – play central roles in complexity theory. A CNF is a conjunction of clauses, where each clause is a disjunction of literals; a w-CNF has at most w literals per clause. The *width* w is often the most important parameter governing the complexity of problems concerning CNFs. The following are three classical games played on a CNF $\varphi(x_1, \ldots, x_n)$:

- In the *ordered* game, player 1 assigns a bit value for x_1, then player 2 assigns x_2, then player 1 assigns x_3, and so on, and the winner is determined by whether φ gets satisfied. Note that the variables must be played in the prescribed order $x_1, x_2, x_3, \ldots$. Deciding

who has a winning strategy – better known as TQBF or QSAT – is PSPACE-complete for 3-CNFs [11] and in P for 2-CNFs [2, 6]. Many PSPACE-completeness results have been shown by reducing from the ordered 3-CNF game.

- In the *unordered* game, each player is allowed to pick which remaining variable to play next (as well as which bit value to assign it), and again the winner is determined by whether φ gets satisfied. Deciding who has a winning strategy is PSPACE-complete for 6-CNFs [1] and for 11-CNFs with only positive literals [9, 10]. The unordered game on positive CNFs is also known as the maker–breaker game, and a simplified proof of PSPACE-completeness for unbounded-width positive CNFs appears in [5]. Many PSPACE-completeness results have been proven by reducing from the unordered positive CNF game [7, 5, 8]. For the general unordered CNF game, nothing was known for width < 6; in particular, the complexity of the unordered 2-CNF game was not studied in the literature before.
- In the *partitioned* game, the set of variables is partitioned into two halves and each player may only pick variables from his/her half. This is, in a sense, intermediate between ordered and unordered: the ordered game restricts the set of variables available to each player *and* the order they must be played; the unordered game restricts neither; the partitioned game restricts only the former. Deciding who has a winning strategy was shown to be PSPACE-complete for unbounded-width CNFs in [9, 10], where it was explicitly posed as an open problem to show PSPACE-completeness with any constant bound on the width. This game has been used for PSPACE-completeness reductions [3], and a variant with a matching between the two players' variables has also been studied [4]. The partitioned 2-CNF game was not studied in the literature before.

We prove that the unordered and partitioned games are both PSPACE-complete for 5-CNFs; the former improves the width 6 bound from [1], and the latter resolves the 42-year-old open problem from [9, 10]. We also prove that the unordered and partitioned games are both in P for 2-CNFs. The complexity for width 3 and 4 remains open. In the following section we give the precise definitions and theorem statements.

1.1 Statement of results

The *unordered CNF game* is defined as follows. There are two players, denoted T (for "true") and F (for "false"). The input consists of a CNF φ, a set of variables $X = \{x_1, \ldots, x_n\}$ containing all the variables that appear in φ (and possibly more), and a specification of which player goes first. The players alternate turns, and each turn consists of picking a remaining variable from X and assigning it a value 0 or 1. Once all variables have been assigned, the game ends 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 φ, X, and who goes first.

The *partitioned CNF game* is similar to the unordered CNF game, except that X is partitioned into two halves X_{T} and X_{F}, and each player may only pick variables from his/her half. If n is even we require $|X_{\mathrm{T}}| = |X_{\mathrm{F}}|$, and if n is odd we require $|X_{\mathrm{T}}| = |X_{\mathrm{F}}| + 1$ if T goes first, and $|X_{\mathrm{F}}| = |X_{\mathrm{T}}| + 1$ if F goes first. We let $\mathrm{G}^{\%}$ denote the problem of deciding which player has a winning strategy, given φ, the partition $X = X_{\mathrm{T}} \cup X_{\mathrm{F}}$, and who goes first.

We let G_w and $\mathrm{G}^{\%}_w$ denote the restrictions of G and $\mathrm{G}^{\%}$, respectively, to instances where φ has width w, i.e., each clause has at most w literals. Now, we state our results as the following theorems:

▶ **Theorem 1.** G_5 *is* PSPACE-complete.

▶ **Theorem 2.** $G_5^{\%}$ *is* PSPACE-complete.

▶ **Theorem 3.** G_2 *is in* P*, in fact, in* Linear Time.

▶ **Theorem 4.** $G_2^{\%}$ *is in* P*, in fact, in* Linear Time.

We prove Theorem 1 and Theorem 2 in Section 2 by showing reductions from the PSPACE-complete games G and $G^{\%}$ respectively. For Theorem 3 and Theorem 4 in Section 3 we prove characterizations in terms of the graph representation from the classical 2-SAT algorithm – who has a winning strategy in terms of certain graph properties – and we design linear time algorithms to check these properties.[1]

In the proofs, it is helpful to distinguish four patterns for "who goes first" and "who goes last", we introduce new subscripts. 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. These may be combined with the width w subscript. For example, $G^{\%}_{\mathrm{T}\cdots\mathrm{F}}$ (which was denoted $G_{\%\,\mathrm{free}}(\mathrm{CNF})$ in [9, 10], by the way) corresponds to the partitioned game where T goes first and F goes last (so $n = |X|$ must be even), and $G_{5,\cdots\mathrm{T}}$ corresponds to the unordered game with width 5 where T goes last (so either n is even and F goes first, or n is odd and T goes first).

2 5-CNF

We prove Theorem 1 in Section 2.1 and Theorem 2 in Section 2.2.

2.1 G_5

In this section we prove Theorem 1. It is trivial to argue that $G_5 \in$ PSPACE. We prove PSPACE-hardness by showing a reduction $G_{\mathrm{T}\cdots\mathrm{F}} \leqslant G_{5,\mathrm{T}\cdots\mathrm{F}}$ in Section 2.1.2. $G_{\mathrm{T}\cdots\mathrm{F}}$ is already known to be PSPACE-complete [9, 10, 5, 1]. We will talk about the other three patterns $G_{\mathrm{F}\cdots\mathrm{F}}$, $G_{\mathrm{T}\cdots\mathrm{T}}$, $G_{\mathrm{F}\cdots\mathrm{T}}$ in the full version. Before the formal proof we develop the intuition in Section 2.1.1.

2.1.1 Intuition

In NP-completeness, recall the following simple reduction from SAT with unbounded width to 3-SAT. Suppose a SAT instance is given by φ over set of variables X. If $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_k)$ is a clause in φ with width $k > 3$, then the reduction introduces fresh variables $z_1, z_2, \ldots, z_{k-1}$ and generates a chain of clauses in φ' as follows:

$$(\ell_1 \vee z_1) \wedge (\overline{z}_1 \vee \ell_2 \vee z_2) \wedge \cdots \wedge (\overline{z}_{i-1} \vee \ell_i \vee z_i) \wedge \cdots \wedge (\overline{z}_{k-2} \vee \ell_{k-1} \vee z_{k-1}) \wedge (\overline{z}_{k-1} \vee \ell_k)$$

Each clause of φ gets a separate set of fresh variables for its chain, and we let $Z = \{z_1, z_2, \ldots\}$ be the set of all fresh variables for all chains. The reduction claims that φ is satisfiable if and only if φ' is satisfiable. We are going to have a stronger property in Claim 1.

▶ **Claim 1.** *For every assignment x to X: $\varphi(x)$ is satisfied iff there exists an assignment z to Z such that $\varphi'(x, z)$ is satisfied.*

[1] We remark that it is not automatic that two-player games on 2-CNFs are solvable in polynomial time; e.g., the game played on a 2-CNF with only negative literals in which players alternate turns assigning variables of their choice to 0 and where the loser is the first to falsify the 2-CNF, as well as the partitioned variant of this game, are PSPACE-complete [9, 10].

Proof. Suppose x satisfies φ. If x satisfies $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_k)$ in φ by $\ell_i = 1$, then in the corresponding chain of clauses in φ', the clause having ℓ_i also gets satisfied by $\ell_i = 1$ and the rest of the clauses in that chain can get satisfied by assigning all z's on the left side of ℓ_i as 1 and right side of ℓ_i as 0.

Now suppose x does not satisfy φ. Then at least one of the clauses of φ has all literals assigned as 0. The corresponding chain of clauses in φ' essentially becomes:

$$(z_1) \wedge (\overline{z}_1 \vee z_2) \wedge \cdots \wedge (\overline{z}_{i-1} \vee z_i) \wedge \cdots \wedge (\overline{z}_{k-2} \vee z_{k-1}) \wedge (\overline{z}_{k-1})$$

In order to satisfy the above chain, $z_1 = 1$ and $z_{k-1} = 0$. It also introduces the following chain of implications: $z_1 \Rightarrow z_2 \Rightarrow z_3 \Rightarrow \cdots \Rightarrow z_{k-1}$. Following the chain we get $(z_1 \Rightarrow z_{k-1})$ $= (1 \Rightarrow 0)$. Therefore, we conclude that $\varphi'(x, z)$ cannot be satisfied for any assignment z. ◀

Now this reduction does not show $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{3,\mathrm{T}\cdots\mathrm{F}}$ since the games on φ and φ' are not equivalent. We show a simple example to make our point. Consider the following $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}}$ game over variables $\{x_0, x_1, \ldots, x_k\}$.

$$\varphi = x_0 \wedge (x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_k), \text{ where } k > 1$$

In the above $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}}$ game, T has a winning strategy: On the first move T plays $x_0 = 1$. Then whatever F plays, T plays one of the $k-1$ many unassigned x_i from $\{x_1, x_2, \ldots, x_k\}$ as 1. T wins.

But if we introduce fresh variables $\{z_1, z_2, z_3, \ldots\}$ as in the NP-completeness reduction then we get a game over variables $\{x_0, x_1, x_2, \ldots, x_k\} \cup \{z_1, \ldots, z_{k-1}\}$:

$$\varphi' = x_0 \wedge (x_1 \vee z_1) \wedge \cdots \wedge (\overline{z}_{i-1} \vee x_i \vee z_i) \wedge \cdots \wedge (\overline{z}_{k-1} \vee x_k)$$

In the above $\mathrm{G}_{3,\mathrm{T}\cdots\mathrm{F}}$ game, F has a winning strategy: On the first move T must play $x_0 = 1$, otherwise F wins by $x_0 = 0$. Then F plays $x_1 = 0$ and T must reply by $z_1 = 1$, otherwise F wins by $z_1 = 0$. Then F plays $x_2 = 0$ and T must reply by $z_2 = 1$, otherwise F wins by $z_2 = 0$. The strategy goes on like this until the last clause and F wins by $x_k = 0$.

The $\mathrm{G}_{3,\mathrm{T}\cdots\mathrm{F}}$ game is disadvantageous for T compared to the $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}}$ game. The disadvantage arises from F having the beginning move in a fresh chain of clauses.

Now the intuition is to design a game version of the NP-completeness reduction by fixing the imbalance. We design ψ in such a way that the games on φ and ψ stay equivalent. In order to counter the unfairness for T due to fresh variables $\{z_1, z_2, z_3, \ldots\}$, we replace z_i by a pair of variables (a_i, b_i) which gives T more opportunities to satisfy the clauses. The construction of a chain of clauses in ψ from a clause $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_k)$ in φ goes as follows:

$$(\ell_1 \vee a_1 \vee b_1) \wedge \cdots \wedge (\overline{a}_{i-1} \vee \overline{b}_{i-1} \vee \ell_i \vee a_i \vee b_i) \wedge \cdots \wedge (\overline{a}_{k-1} \vee \overline{b}_{k-1} \vee \ell_k)$$

We just constructed a 5-CNF ψ. Let us consider the $\mathrm{G}_{5,\mathrm{T}\cdots\mathrm{F}}$ game on ψ. In an optimal gameplay, no player should play a's or b's before playing x's. Intuitively, this is because, if F plays any a_i or b_i, then T can reply by making $a_i \neq b_i$ and both clauses involving a_i and b_i will be satisfied, which benefits T. If T plays any a_i or b_i, F can reply by making $a_i = b_i$, which satisfies one clause involving a_i and b_i but the other clause gets two 0 literals. Since only one of the two clauses gets satisfied by a_i, b_i, T would like to wait for more information before deciding which one to satisfy with a_i, b_i: it depends on whether they are on the right side or left side of a satisfied ℓ_i in a chain, which in turn depends on the assignment x.

So, an optimal gameplay consists of two phases. In the first phase, players should play only x's. Whoever deviates from this optimal strategy does not have the upper hand. The second phase begins when all the x's have been played and someone must start playing a's and b's. Since the number of fresh variables is even ($2|Z|$) and F plays last, T must be the one to start the second phase, which is essential since if F started the second phase then T could satisfy all the clauses regardless of what happened in the first phase. This observation also allows us to show PSPACE-completeness of $\mathrm{G}_{5,\mathrm{F}\cdots\mathrm{F}}$, discussed in the full version.

In the second phase, after T plays any a_i or b_i, it is optimal for F to reply by making $a_i = b_i$. Assuming this optimal gameplay by F, we can consider a pair (a_i,b_i) as a single variable z_i which can be assigned only by T. Effectively, the second phase just consists of T choosing an assignment z to φ' from the NP-completeness reduction. Thus $\psi(x, a, b)$ is satisfied iff $\varphi'(x, z)$ is satisfied, which by Claim 1 is possible iff $\varphi(x)$ is satisfied, where x is the assignment from the first phase.

2.1.2 Formal Proof

We show $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{5,\mathrm{T}\cdots\mathrm{F}}$. Suppose an instance of $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}}$ is given by (φ, X) where φ is a CNF with unbounded width over set of variables X. We show how to construct an instance (ψ, Y) for $\mathrm{G}_{5,\mathrm{T}\cdots\mathrm{F}}$ where ψ is a 5-CNF over set of variables Y. Suppose $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_k)$ is a clause in φ. If $k \leqslant 3$, the same clause remains in ψ. If $k > 3$, we show how to construct a chain of clauses in ψ. We introduce two sets of fresh variables $\{a_1, a_2, a_3, \ldots, a_{k-1}\}$ and $\{b_1, b_2, b_3, \ldots, b_{k-1}\}$ as follows:

$$(\ell_1 \vee a_1 \vee b_1) \wedge \cdots \wedge (\overline{a}_{i-1} \vee \overline{b}_{i-1} \vee \ell_i \vee a_i \vee b_i) \wedge \cdots \wedge (\overline{a}_{k-1} \vee \overline{b}_{k-1} \vee \ell_k)$$

Each clause of φ gets separate sets of fresh variables for its chain, and we let $A = \{a_1, a_2, a_3, \ldots\}$ and $B = \{b_1, b_2, b_3, \ldots\}$ be the sets of all fresh variables for all chains. Finally we get a 5-CNF ψ over set of variables $Y = X \cup A \cup B$.

We claim that T has a winning strategy in (φ, X) iff T has a winning strategy in (ψ, Y).

Suppose T has a winning strategy in (φ, X). We describe T's winning strategy in (ψ, Y) as Algorithm 1. To see that the strategy works, note that the winning strategy in (φ, X) ensures that $\varphi(x)$ is satisfied by the assignment x to X in the first phase, so according to 1, there is an assignment z to Z such that $\varphi'(x, z)$ is satisfied. T can ensure that for each i, either $a_i = z_i$ or $b_i = z_i$ (since $a_i = z_i$ or $b_i = z_i$ due to line 8, or $a_i \neq b_i$ due to line 4 or line 7) and thus $\psi(x, a, b)$ gets satisfied, since $\varphi'(x, z)$ is satisfied and each clause of ψ is identical to a clause from φ' but with each z_i replaced with $a_i \vee b_i$ and $\overline{z}_i$ replaced with $\overline{a}_i \vee \overline{b}_i$.

Suppose F has a winning strategy in (φ, X). We describe F's winning strategy in (ψ, Y) as Algorithm 2. To see that the strategy works, note that the winning strategy in (φ, X) ensures that $\varphi(x)$ is unsatisfied by the assignment x to X, so according to Claim 1, for all assignments z to Z, $\varphi'(x, z)$ is unsatisfied. F can ensure that for each i, $a_i = b_i$; let us call this common value z_i. Thus $\psi(x, a, b)$ is unsatisfied, since $\varphi'(x, z)$ is unsatisfied and $\psi(x, a, b) = \varphi'(x, z)$.

2.2 $\mathrm{G}^{\%}_{\mathbf{5}}$

In this section we prove Theorem 2. It is trivial to argue that $\mathrm{G}^{\%}_5 \in$ PSPACE. We prove PSPACE-hardness by showing a reduction $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{5,\mathrm{T}\cdots\mathrm{F}}$ in Section 2.2.2. $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}}$ is already known to be PSPACE-complete [9, 10]. We will talk about the other three patterns $\mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}}$, $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{T}}$, $\mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{T}}$ in the full version. Before the formal proof we develop the intuition in Section 2.2.1.

Algorithm 1: T's winning strategy in (ψ, Y) when T has a winning strategy in (φ, X).

while there is a remaining X-variable **do**
  if (first move) or (F played an X-variable in the previous move) **then**
    play according to the same winning strategy as in (φ, X)
  else if F played a_i or b_i in the previous move **then** play the other one to make $a_i \neq b_i$
while there is a remaining A-variable or B-variable **do**
  if (F played a_i or b_i in the previous move) and (one of a_i or b_i remains unplayed) **then**
    play the other one to make $a_i \neq b_i$
  else pick a remaining a_i or b_i and assign it z_i's value from Claim 1

Algorithm 2: F's winning strategy in (ψ, Y) when F has a winning strategy in (φ, X).

while there is a remaining variable **do**
  if T played an X-variable in the previous move **then**
    play according to the same winning strategy as in (φ, X)
  else if T played a_i or b_i in the previous move **then** play the other one to make $a_i = b_i$

2.2.1 Intuition

This intution is a continuation of Section 2.1.1. The reduction is the same as $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{5,\mathrm{T}\cdots\mathrm{F}}$ reduction except giving A-variables to T and B-variables to F. In the general unordered game if any player plays a_i or b_i, then the other player can immediately play the other one from a_i, b_i in a certain advantageous way. In the partitioned version they can do the same thing if a_i belongs to T and b_i belongs to F.

2.2.2 Formal Proof

We show $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{5,\mathrm{T}\cdots\mathrm{F}}$. Suppose an instance of $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}}$ is given by $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$ where φ is a CNF with unbounded width over sets of variables X_{T} and X_{F}. We show how to construct an instance $(\psi, Y_{\mathrm{T}}, Y_{\mathrm{F}})$ for $\mathrm{G}^{\%}_{5,\mathrm{T}\cdots\mathrm{F}}$ where ψ is a 5-CNF over sets of variables Y_{T} and Y_{F}. Suppose $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_k)$ is a clause in φ. If $k \leqslant 3$, the same clause remains in ψ. If $k > 3$, we show how to construct a chain of clauses in ψ. We introduce two sets of fresh variables $\{a_1, a_2, a_3, \ldots, a_{k-1}\}$ for T and $\{b_1, b_2, b_3, \ldots, b_{k-1}\}$ for F as follows:

$$(\ell_1 \vee a_1 \vee b_1) \wedge \cdots \wedge (\overline{a}_{i-1} \vee \overline{b}_{i-1} \vee \ell_i \vee a_i \vee b_i) \wedge \cdots \wedge (\overline{a}_{k-1} \vee \overline{b}_{k-1} \vee \ell_k)$$

Each clause of φ gets separate sets of fresh variables for its chain, and we let $A = \{a_1, a_2, a_3, \ldots\}$ for T and $B = \{b_1, b_2, b_3, \ldots\}$ for F be the sets of all fresh variables for all chains. Finally we get a 5-CNF ψ over sets of variables $Y_{\mathrm{T}} = X_{\mathrm{T}} \cup A$ and $Y_{\mathrm{F}} = X_{\mathrm{F}} \cup B$.

We claim that T has a winning strategy in $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$ iff T has a winning strategy in $(\psi, Y_{\mathrm{T}}, Y_{\mathrm{F}})$.

Algorithm 3: T's winning strategy in (ψ, Y_T, Y_F) when T has a winning strategy in (φ, X_T, X_F).

```
1 while there is a remaining X_T-variable do
2     if (first move) or (F played an X_F-variable in the previous move) then
3         play according to the same winning strategy as in (φ, X_T, X_F)
4     else if F played b_i in the previous move then play a_i to make a_i ≠ b_i
5 while there is a remaining A-variable do
6     if (F played b_i in the previous move) and (a_i remains unplayed) then
7         play a_i to make a_i ≠ b_i
8     else pick a remaining a_i and assign it z_i's value from Claim 1
```

Algorithm 4: F's winning strategy in (ψ, Y_T, Y_F) when F has a winning strategy in (φ, X_T, X_F).

```
1 while there is a remaining variable do
2     if T played an X_T-variable in the previous move then
3         play according to the same winning strategy as in (φ, X_T, X_F)
4     else if T played a_i in the previous move then play b_i to make a_i = b_i
```

Suppose T has a winning strategy in (φ, X_T, X_F). We describe T's winning strategy in (ψ, Y_T, Y_F) as Algorithm 3. To see that the strategy works, note that the winning strategy in (φ, X_T, X_F) ensures that $\varphi(x)$ is satisfied by the assignment x to $X_T \cup X_F$ in the first phase, so according to Claim 1, there is an assignment z to Z such that $\varphi'(x, z)$ is satisfied. T can ensure that for each i, either $a_i = z_i$ or $b_i = z_i$ (since $a_i = z_i$ due to line 8, or $a_i \neq b_i$ due to line 4 or line 7) and thus $\psi(x, a, b)$ gets satisfied, since $\varphi'(x, z)$ is satisfied and each clause of ψ is identical to a clause from φ' but with each z_i replaced with $a_i \vee b_i$ and $\overline{z}_i$ replaced with $\overline{a}_i \vee \overline{b}_i$.

Suppose F has a winning strategy in (φ, X_T, X_F). We describe F's winning strategy in (ψ, Y_T, Y_F) as Algorithm 4. To see that the strategy works, note that the winning strategy in (φ, X_T, X_F) ensures that $\varphi(x)$ is unsatisfied by the assignment x to $X_T \cup X_F$, so according to Claim 1, for all assignments z to Z, $\varphi'(x, z)$ is unsatisfied. F can ensure that for each i, $a_i = b_i$; let us call this common value z_i. Thus $\psi(x, a, b)$ is unsatisfied, since $\varphi'(x, z)$ is unsatisfied and $\psi(x, a, b) = \varphi'(x, z)$.

3 2-CNF

In order to analyze the complexity of the games G_2 and $G_2^{\%}$, we construct a directed graph $g(\varphi, X)$ by the classical technique for 2-SAT:

- For each variable $x_i \in X$, form two nodes x_i and $\overline{x}_i$. Let ℓ_i refer to either x_i or $\overline{x}_i$.[2]
- For each clause $(\ell_i \vee \ell_j)$, add two directed edges $\overline{\ell}_i \to \ell_j$ and $\ell_i \leftarrow \overline{\ell}_j$. In case of a single variable clause (ℓ_i), consider the clause as $(\ell_i \vee \ell_i)$ and add one directed edge $\overline{\ell}_i \to \ell_i$.

[2] In Section 2, ℓ_i represented an arbitrary literal; in Section 3, ℓ_i always represents either x_i or $\overline{x}_i$.

In the graph, every path $\ell_i \rightsquigarrow \ell_j$ has a mirror path $\overline{\ell}_i \leftsquigarrow \overline{\ell}_j$. If there exist two paths $\ell_i \rightsquigarrow \ell_j$ and $\ell_i \leftsquigarrow \ell_j$, we express this as $\ell_i \leftrightsquigarrow \ell_j$. We are interested in strongly connected components, which we call strong components for short.

The 2-CNF game analogy on this graph is, if any variable x_i is assigned a bit value in φ, then in the graph both nodes x_i and $\overline{x}_i$ are assigned. Conversely, if say a player assigns a bit value to a node ℓ_i, then the complement node $\overline{\ell}_i$ simultaneously gets assigned the opposite value. If ℓ_i refers to x_i, then x_i gets assigned the same value as ℓ_i and similarly for ℓ_i referring to $\overline{x}_i$. Thus we can describe strategies as assigning bit values to nodes in the graph.

In a satisfying assignment for φ, there must not exist any false implication edge $(1 \to 0)$ in the graph. In fact, the graph must not have any path $(1 \rightsquigarrow 0)$ since the path will contain at least one $(1 \to 0)$ edge. Player F's goal is to create a false implication and player T will try to make all implications true.

We prove Theorem 3 in Section 3.1 and Theorem 4 in the full version.

3.1 $\mathrm{G}_{\mathbf{2}}$

G_2 is the unordered analogue of the 2-TQBF game. We prove Theorem 3 by separately considering the cases $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{F}}$ in Section 3.1.1, $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{T}}$ in Section 3.1.2, and $\mathrm{G}_{2,\mathrm{T}\cdots}$ in Section 3.1.3.

3.1.1 $\mathrm{G}_{\mathbf{2},\mathrm{F}\cdots\mathrm{F}} \in$ Linear Time

▶ **Lemma 5.** F *has a winning strategy in* $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{F}}$ *iff at least one of the following statements holds in the graph* $g(\varphi, X)$*:*

(1) *There exists a node* ℓ_i *such that* $\overline{\ell}_i \rightsquigarrow \ell_i$*.*

(2) *There exist three nodes* ℓ_i*,* ℓ_j*,* ℓ_k *such that* $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$*.*

(3) *There exist two nodes* ℓ_i*,* ℓ_j *such that* $\ell_i \leftrightsquigarrow \ell_j$*.*

Proof. Suppose at least one of the statements holds.

If statement *(1)* holds, F can win by $\ell_i = 0$ as the very first move.

If statement *(2)* holds but statement *(1)* does not, there can be two cases:

- In the first case, ℓ_i, ℓ_j, ℓ_k represent three distinct variables. At the beginning, F can play $\ell_i = 0$, then whatever T plays, F still has at least one of ℓ_j or ℓ_k to play. F can assign ℓ_j or ℓ_k as 1 and wins.
- In the second case, ℓ_i, ℓ_j, ℓ_k do not represent three distinct variables. The only possibility is that ℓ_k is $\overline{\ell}_j$, i.e., $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \overline{\ell}_j$. F can play $\ell_i = 0$, then whatever the value of ℓ_j, F wins.

If statement *(3)* holds but statement *(1)* does not, F can wait by playing variables other than x_i, x_j with arbitrary values until T plays x_i or x_j. Then F can immediately respond by making $\ell_i \neq \ell_j$ and win. As F moves last, he/she can always wait for that opportunity.

Conversely, suppose none of the statements hold. Then we claim the graph has no two edges that share an endpoint. Otherwise, two edges that share an endpoint would cause statement *(2)* or statement *(3)* to be satisfied. We show this by considering all possible ways of two edges sharing an endpoint:

- $\ell_i \leftrightarrow \ell_j$: Satisfies statement *(3)*.
- $\ell_j \to \ell_i \leftarrow \ell_k$ or its mirror $\overline{\ell}_j \leftarrow \overline{\ell}_i \to \overline{\ell}_k$: Satisfies statement *(2)*.
- $\ell_k \to \ell_j \to \ell_i$: Satisfies statement *(2)*.

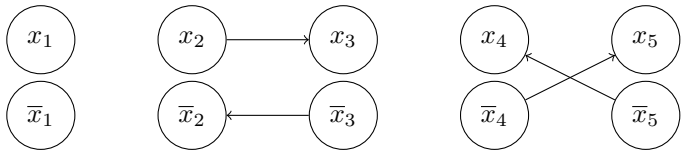

Figure 1 T has a winning strategy in $G_{2,F\cdots F}$ for $(\overline{x}_2 \vee x_3) \wedge (x_4 \vee x_5)$.

Algorithm 5: Linear Time Algorithm for $G_{2,F\cdots F}$.

construct $g(\varphi, X)$
foreach $x_i \in X$ **do**
  if $(x_i \to \overline{x}_i)$ or $(x_i \leftarrow \overline{x}_i)$ or (x_i has at least two incident edges) **then** output F
output T

So, the graph can only have some isolated nodes and isolated edges. Since statement *(1)* does not hold, there are no edges between complementary nodes. An example of such a graph looks like Figure 1. Conversely, in any such graph (like Figure 1) none of statements *(1)*, *(2)*, *(3)* holds.

Now, we describe a winning strategy for T on such a graph. If F plays ℓ_i or ℓ_j of any fresh (both endpoints unassigned) edge $\ell_i \to \ell_j$, T plays in the same edge by the same bit value for the other node, i.e., making $\ell_i = \ell_j$. Otherwise, T picks any remaining node ℓ_i. If ℓ_i is isolated, T assigns any arbitrary bit value. If ℓ_i has an incoming edge, T plays $\ell_i = 1$. If ℓ_i has an outgoing edge, T plays $\ell_i = 0$.

The strategy works, since all the edges $\ell_i \to \ell_j$ will be satisfied, by either $\ell_i = \ell_j$ or $\ell_i = 0$ or $\ell_j = 1$. ◀

The characterization of such a graph in the proof of Lemma 5 can be verified in linear time, and that yields a Linear Time algorithm for $G_{2,F\cdots F}$. Details of the idea have been described as Algorithm 5.

3.1.2 $G_{2,F\cdots T} \in$ Linear Time

The characterization is the same as for $G_{2,F\cdots F}$ but without statement *(3)*.

▶ **Lemma 6.** F *has a winning strategy in* $G_{2,F\cdots T}$ *iff at least one of the following statements holds in the graph* $g(\varphi, X)$*:*

(1) *There exists a node* ℓ_i *such that* $\overline{\ell}_i \rightsquigarrow \ell_i$*.*

(2) *There exist three nodes* ℓ_i*,* ℓ_j*,* ℓ_k *such that* $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$*.*

Proof. Suppose one of the statements holds. In Lemma 5, we have already seen that statement *(1)* and statement *(2)* allow player F to win at the beginning.

Conversely, suppose none of the statements hold. The graph can have strong components of size 2. Other than that, there are no two edges sharing an endpoint because statement *(2)* does not hold. So, the graph can only have some isolated nodes, isolated edges, and isolated strong components of size 2. Since statement *(1)* does not hold, there are no edges between complementary nodes. An example of such a graph looks like Figure 2. Conversely, in any such graph (like Figure 2) none of statements *(1)*, *(2)* holds.

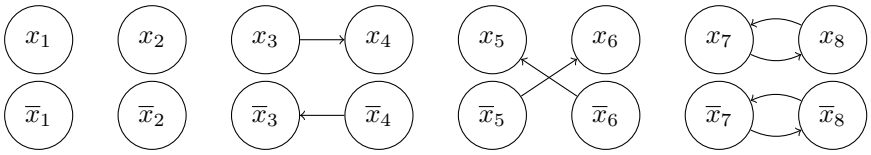

Figure 2 T has a winning strategy in $G_{2,F\cdots T}$ for $(\overline{x}_3 \vee x_4) \wedge (x_5 \vee x_6) \wedge (\overline{x}_7 \vee x_8) \wedge (x_7 \vee \overline{x}_8)$.

Algorithm 6: Linear Time Algorithm for $G_{2,F\cdots T}$.

```
construct g(φ, X)
foreach x_i ∈ X do
    if (x_i → x̄_i) or (x_i ← x̄_i) or (x_i has at least two neighbors) then output F
output T
```

Now, we describe a winning strategy for T on such a graph. If F plays ℓ_i or ℓ_j of any fresh (both endpoints unassigned) edge $\ell_i \to \ell_j$ or strong component $\ell_i \leftrightarrow \ell_j$, T plays in the same edge or strong component by the same bit value for the other node, i.e., making $\ell_i = \ell_j$. Otherwise, T picks any remaining isolated node and gives it any arbitrary bit value. Since $|X|$ is even, T can always play such a node.

The strategy works, since all the edges $\ell_i \to \ell_j$ will be satisfied by $\ell_i = \ell_j$. ◀

The characterization of such a graph in the proof of Lemma 6 can be verified in linear time, and that yields a Linear Time algorithm for $G_{2,F\cdots T}$. Details of the idea have been described as Algorithm 6.

3.1.3 $G_{2,T\cdots} \in$ Linear Time

In order to win $G_{2,T\cdots}$, at the beginning T must locate a node ℓ_i such that after playing it, the game is reduced to a $G_{2,F\cdots}$ game such that T still has a winning strategy in it. So, T's success depends on finding such a node ℓ_i. On the other hand, F's success depends on there not existing such a node ℓ_i.

▶ **Lemma 7.** T *has a winning strategy in* $G_{2,T\cdots}$ *iff there exists an* ℓ_i *with no outgoing edges such that after deleting* ℓ_i, $\overline{\ell}_i$ *and their incident edges, in the rest of the graph* T *has a winning strategy in* $G_{2,F\cdots}$.

Proof. Suppose T has a winning strategy in $G_{2,T\cdots}$. Let T's first move in the winning strategy be $\ell_i = 1$ (or $\overline{\ell}_i = 0$). Then ℓ_i must not have any outgoing edge, otherwise either that edge goes to $\overline{\ell}_i$ or F could play the other endpoint node of that edge as 0 and win.

Conversely, suppose there exists such an ℓ_i. At the beginning, T can play $\ell_i = 1$, and all the incoming edges to ℓ_i and outgoing edges from $\overline{\ell}_i$ get satisfied. Then T can continue the game according to the winning strategy in $G_{2,F\cdots}$ for the rest of the graph and win. For example, in Figure 3, T's winning strategy is to play $\ell_i = 1$ at the beginning then continue the winning strategy for $G_{2,F\cdots}$. ◀

We define L as the set of all nodes that have no outgoing edges. If $|L| = 0$, then according to Lemma 7, T has no winning strategy in $G_{2,T\cdots}$. If $|L| > 0$, then the trivial algorithm for $G_{2,T\cdots}$ is, checking for each node $\ell_i \in L$, whether or not after playing $\ell_i = 1$ the rest of the graph becomes a winning graph for T in $G_{2,F\cdots}$, i.e., running Algorithm 5 or Algorithm 6 for $O(|L|)$ times, which is a quadratic time algorithm. We argue that we can do better than that.

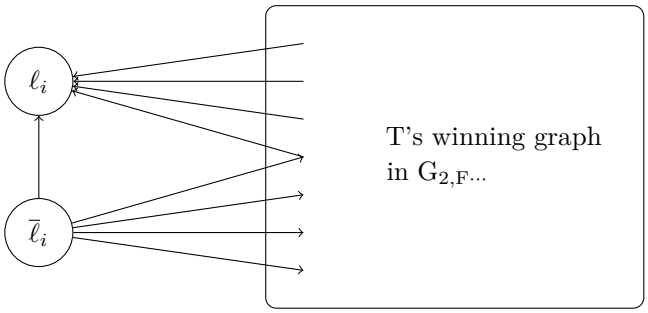

Figure 3 T's winning graph in $G_{2,T\cdots}$ (all edges incident to ℓ_i or $\overline{\ell}_i$ are optional).

We filter the possibilities in L and show that there are only three cases to consider:

- There exists a node $\ell_i \in L$ such that statement *(1)* from Lemma 5 and Lemma 6 holds. We consider this case in Claim 2.
- There exists a node $\ell_i \in L$ such that statement *(2)* from Lemma 5 and Lemma 6 holds. We consider this case in Claim 3.
- There exists no node $\ell_i \in L$ such that statement *(1)* or statement *(2)* from Lemma 5 and Lemma 6 holds. We consider this case in Claim 4.

Then in Claim 5 and Claim 6 we analyze the efficiency of this approach. We will provide proofs of Claim 2 to Claim 6 in the full version.

▶ **Claim 2.** *If there exists $\ell_i \in L$ such that $\overline{\ell}_i \rightsquigarrow \ell_i$ and* T *has a winning strategy in $G_{2,T\cdots}$, then* T*'s first move must be $\ell_i = 1$.*

▶ **Claim 3.** *If there exists $\ell_i \in L$ such that $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$ for two other nodes ℓ_j, ℓ_k and* T *has a winning strategy in $G_{2,T\cdots}$, then* T*'s first move must be $\ell_i = 1$ or $\overline{\ell}_j = 1$ or $\overline{\ell}_k = 1$.*

▶ **Claim 4.** *If there exists no $\ell_i \in L$ such that $\overline{\ell}_i \rightsquigarrow \ell_i$ or $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$ for two other nodes ℓ_j, ℓ_k and* T *has a winning strategy in $G_{2,T\cdots}$, then for all $\ell_i \in L$,* T *has a winning strategy in $G_{2,T\cdots}$ beginning with $\ell_i = 1$.*

The overall idea is: If we can find an ℓ_i for which statement *(1)* or statement *(2)* from Lemma 5 and Lemma 6 holds, then Claim 2 and Claim 3 allow us to narrow down T's first move to $O(1)$ possibilities. If we cannot find such an ℓ_i, then Claim 4 allows T to play any arbitrary $\ell_i \in L$ as the first move because all of them are equivalent as the first move. We define L^* as the $O(1)$ possibilities in L. Then we can run Algorithm 5 or Algorithm 6 for $|L^*| = O(1)$ times.

In the following two claims, we show how we can efficiently verify whether or not there exists such an ℓ_i for which statement *(1)* or statement *(2)* from Lemma 5 and Lemma 6 holds.

▶ **Claim 5.** *There exists a constant-time algorithm for: given ℓ_i, find two other nodes ℓ_j, ℓ_k such that $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$ or determine they do not exist.*

▶ **Claim 6.** *There exists a constant-time algorithm for: given ℓ_i for which there are no ℓ_j, ℓ_k as in Claim 5, decide whether there exists a path $\overline{\ell}_i \rightsquigarrow \ell_i$.*

Now combining the whole idea from Claim 2 to Claim 6 we can develop an algorithm for $G_{2,T\cdots}$. Details of the idea have been described as Algorithm 7.

Algorithm 7: Linear Time Algorithm for $G_{2,T\cdots}$.

construct $g(\varphi, X)$
let $L = \{\}$, $L^* = \{\}$
foreach node ℓ_i **do**
  if ℓ_i has no outgoing edges **then** $L = L \cup \{\ell_i\}$
if $|L| = 0$ **then** output F
foreach $\ell_i \in L$ **do**
  if $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$ for two other nodes ℓ_j, ℓ_k (using Claim 5) **then**
    $L^* = L \cap \{\ell_i, \overline{\ell}_j, \overline{\ell}_k\}$ (Claim 3), break loop
  else if $\overline{\ell}_i \rightsquigarrow \ell_i$ (using Claim 6) **then** $L^* = \{\ell_i\}$ (Claim 2), break loop
if $|L^*| = 0$ **then** $L^* = \{\ell_i\}$ for an arbitrary $\ell_i \in L$ (Claim 4)
foreach $\ell_i \in L^*$ **do**
  form graph g' from $g(\varphi, X)$ by deleting nodes ℓ_i, $\overline{\ell}_i$ and their incident edges
  run Algorithm 5 or Algorithm 6 on g' as the $G_{2,F\cdots}$ game
  if T has a winning strategy in $G_{2,F\cdots}$ **then** output T
output F

References

1 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.

2 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.

3 Kyle Burke, Erik Demaine, Harrison Gregg, Robert Hearn, Adam Hesterberg, Michael Hoffmann, Hiro Ito, Irina Kostitsyna, Jody Leonard, Maarten Löffler, Aaron Santiago, Christiane Schmidt, Ryuhei Uehara, Yushi Uno, and Aaron Williams. Single-Player and Two-Player Buttons & Scissors Games. In *Proceedings of the 18th Japan Conference on Discrete and Computational Geometry and Graphs (JCDCGG)*, pages 60–72. Springer, 2015.

4 William Burley and Sandy Irani. On Algorithm Design for Metrical Task Systems. *Algorithmica*, 18(4):461–485, 1997.

5 Jesper Byskov. Maker-Maker and Maker-Breaker Games Are PSPACE-Complete. Technical Report RS-04-14, BRICS, Department of Computer Science, Aarhus University, 2004.

6 Chris Calabro. 2-TQBF Is in P, 2008. Unpublished. URL: `https://cseweb.ucsd.edu/~ccalabro/essays/complexity_of_2tqbf.pdf`.

7 Aviezri Fraenkel and Elisheva Goldschmidt. PSPACE-hardness of some combinatorial games. *Journal of Combinatorial Theory, Series A*, 46(1):21–38, 1987.

8 Robert Hearn. Amazons, Konane, and Cross Purposes are PSPACE-complete. In *Games of No Chance 3*, Mathematical Sciences Research Institute Publications, pages 287–306. Cambridge University Press, 2009.

9 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.

10 Thomas Schaefer. On the Complexity of Some Two-Person Perfect-Information Games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

11 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.