Complexity of Unordered CNF Games

Md Lutfar Rahman Thomas Watson

University of Memphis

February 5, 2020

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).

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. The CNF format is so prevalent because it can represent all Boolean functions and can do so in a succinct way for many functions of interest. 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; this is because problems often turn out to be tractable for small width (e.g., satisfiability of 2-CNFs) and intractable for larger width (e.g., satisfiability of 3-CNFs). The following are three classical two-player 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 [SM73] and in P for 2-CNFs [APT79, Cal08]. Many PSPACE-completeness results have been shown by reducing from the ordered 3-CNF game; classic examples include Generalized Geography [Sch76, Sch78] and Node Kayles [Sch76, Sch78].

- 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 [AO12] and for 11-CNFs with only positive literals [Sch76, Sch78]. 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 [Bys04]. Many PSPACE-completeness results have been proven by reducing from the unordered positive CNF game [FG87, Sla00, Sla02, AS03, Bys04, Hea09, TDU11, vV13, FGM+15, BDK+16]. 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. An experimental evaluation of heuristics for the unordered CNF game appears in [ZM04].
- 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 [Sch76, Sch78], 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 [BDG+15], and a variant with a matching between the two players' variables has also been studied [BI97]. The partitioned 2-CNF game was not studied in the literature before.

Study of the unordered and partitioned games is motivated by their resemblance to real-world two-player games that also lack a prescribed "order" for possible moves. For example: The game of Hex has an unordered flavor since any cell can potentially be played by either player at any time. The game of Checkers has a partitioned flavor since for any configuration of pieces, the set of moves one player is allowed to make is disjoint from the set of moves the other player is allowed to make, and each player may pick any of their available moves. Hardness results for the unordered and partitioned CNF games may translate via reduction more easily (than the ordered game) to other games of interest.

We prove that the unordered and partitioned games are both PSPACE-complete for 5-CNFs; the former improves the width 6 bound from [AO12], and the latter resolves the 42-year-old open problem[1] from [Sch76, Sch78]. 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

[1] From [Sch78, top of p.216]: "We have not been able to produce any fixed integer bound k such that $L_{\%\text{free}}(\text{CNF})$ is complete when restricted to formulas with at most k disjuncts in each conjunct."

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. *$\mathrm{G}_5^{\%}$ is* PSPACE*-complete.*

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

Theorem 4. *$\mathrm{G}_2^{\%}$ is in* P*, in fact, in* Linear Time*.*

Here, linear time means $O(|X| + \text{number of clauses in } \varphi)$.

We prove Theorem 1 and Theorem 2 in Section 2 by showing reductions from the PSPACE-complete games G and $\mathrm{G}^{\%}$ 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.[2]

In the proofs, it is helpful to distinguish four patterns for "who goes first" and "who goes last", so 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, $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}}$ (which was denoted $L_{\%\,\mathrm{free}}(\mathrm{CNF})$ in [Sch76, Sch78]) corresponds to the partitioned game where T goes first and F goes last (so $n = |X|$ must be even), and $\mathrm{G}_{5,\cdots\mathrm{T}}$ 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. We use the $\leqslant$ symbol to indicate the existence of a polynomial-time mapping (Karp) reduction from one problem to another.

2.1 G_5

In this section we prove Theorem 1. 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 2.1.2. $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}}$ is already known to be PSPACE-complete [Sch76, Sch78, Bys04, AO12]. 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 Section 2.1.3. Before the formal proof we develop the intuition in Section 2.1.1.

[2] 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 positive 2-CNF 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 [Sch76, Sch78].

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 will make use of the following specific property of the reduction.

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.*

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. $\square$

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:

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

Let us consider a $\mathrm{G}_{5,\mathrm{T}\cdots\mathrm{F}}$ 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. 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.

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 $\{b_1, b_2, b_3, \ldots, b_{k-1}\}$ and clauses 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 Claim 1, there is an assignment z to Z (the set of fresh variables introduced in the definition of φ') 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 $\overline{a}_i \vee \overline{b}_i$.

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 $a_i = b_i$

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 $\psi(x, a, b) = \varphi'(x, z)$.

2.1.3 $\mathrm{G}_{\mathrm{F}\cdots\mathrm{F}}$, $\mathrm{G}_{\mathrm{T}\cdots\mathrm{T}}$, $\mathrm{G}_{\mathrm{F}\cdots\mathrm{T}}$

Corollary 1. $\mathrm{G}_{5,\mathrm{F}\cdots\mathrm{F}}$ *is* PSPACE-*complete.*

Proof. The reduction is $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{\mathrm{F}\cdots\mathrm{F}} \leqslant \mathrm{G}_{5,\mathrm{F}\cdots\mathrm{F}}$. First we show $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{\mathrm{F}\cdots\mathrm{F}}$. Suppose $\varphi = c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_m$ over set of variables X is an instance of $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}}$. We introduce a fresh variable z and construct $\psi = (c_1 \vee z) \wedge (c_2 \vee z) \wedge (c_3 \vee z) \wedge \cdots \wedge (c_m \vee z)$ over set of variables $Y = X \cup \{z\}$. Now in the $\mathrm{G}_{\mathrm{F}\cdots\mathrm{F}}$ game on (ψ, Y), F's first move must be $z = 0$ otherwise T wins by $z = 1$ as the first move. Then the rest of the winning strategy for T or F is the same as in (φ, X). This completes the reduction $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{\mathrm{F}\cdots\mathrm{F}}$.

Now the reduction $\mathrm{G}_{\mathrm{F}\cdots\mathrm{F}} \leqslant \mathrm{G}_{5,\mathrm{F}\cdots\mathrm{F}}$ is identical to Section 2.1.2 except it is F's move first. ☐

To handle the patterns where T moves last, we do not rely on our proof of Theorem 1 but rather derive corollaries of the result from [Sch76, Sch78]:

Corollary 2. $\mathrm{G}_{11,\mathrm{T}\cdots\mathrm{T}}$ *is* PSPACE-*complete.*

Proof. The reduction is $\mathrm{G}^{+}_{11,\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}^{+}_{11,\mathrm{T}\cdots\mathrm{T}} \leqslant \mathrm{G}_{11,\mathrm{T}\cdots\mathrm{T}}$, where G^{+}_{11} is the restriction of G_{11} to instances with only positive literals (and $\mathrm{G}^{+}_{11,\mathrm{T}\cdots\mathrm{F}}$ is known to be PSPACE-complete [Sch76, Sch78]). Given a positive 11-CNF φ^{+} over set of variables X, we simply introduce a dummy variable z that does not appear in φ^+ and use $Y = X \cup \{z\}$. We claim that T has a winning strategy in $\mathrm{G}^+_{11,\mathrm{T}\cdots\mathrm{F}}$ on (φ^+, X) iff T has a winning strategy in $\mathrm{G}^+_{11,\mathrm{T}\cdots\mathrm{T}}$ on (φ^+, Y).

Suppose T has a winning strategy on (φ^+, X). We show T's winning strategy on (φ^+, Y). T can start by the same strategy as in (φ^+, X) and continue as long as F does not play z. If F never plays z, then T plays z at the end and wins as in (φ^+, X). If F plays z then T can respond by playing any remaining variable $x_i = 1$, then T resumes his strategy from (φ^+, X) until that strategy tells him to play x_i. At this time, T again picks any other remaining variable and assigns it 1. Then T again resumes his strategy from (φ^+, X). The game goes on like this in phases. At the end, T has played all the variables he would have played in the (φ^+, X) game and possibly one more. Since φ^+ is positive it must still be satisfied when one of the variables is 1 instead of 0.

A similar winning strategy works for F as well (making $x_i = 0$). This completes the reduction. Trivially, $\mathrm{G}^+_{11,\mathrm{T}\cdots\mathrm{T}} \leqslant \mathrm{G}_{11,\mathrm{T}\cdots\mathrm{T}}$. □

Corollary 3. $\mathrm{G}_{12,\mathrm{F}\cdots\mathrm{T}}$ *is* PSPACE*-complete.*

Proof. The reduction is $\mathrm{G}_{11,\mathrm{T}\cdots\mathrm{T}} \leqslant \mathrm{G}_{12,\mathrm{F}\cdots\mathrm{T}}$ (similar to $\mathrm{G}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}_{\mathrm{F}\cdots\mathrm{F}}$ in Corollary 1): Introduce a fresh variable z to every clause in φ. Then F must play $z = 0$ as the first move otherwise T wins by $z = 1$ as the first move. Like in Corollary 2, this in fact shows PSPACE-completeness of $\mathrm{G}^+_{12,\mathrm{F}\cdots\mathrm{T}}$. □

2.2 $\mathrm{G}^{\%}_5$

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 [Sch76, Sch78]. 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 Section 2.2.3. Before the formal proof we develop the intuition in Section 2.2.1.

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 and clauses 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 $Y_\mathrm{F} = X_\mathrm{F} \cup B$.

Algorithm 3: T's winning strategy in $(\psi, Y_\mathrm{T}, Y_\mathrm{F})$ when T has a winning strategy in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$

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 $(\varphi, X_\mathrm{T}, X_\mathrm{F})$
4   **else if** F played b_i in the previous move **then** play a_i to make $a_i \neq b_i$
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 $a_i \neq b_i$
8   **else** pick a remaining a_i and assign it z_i's value from Claim 1

Algorithm 4: F's winning strategy in $(\psi, Y_\mathrm{T}, Y_\mathrm{F})$ when F has a winning strategy in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$

while there is a remaining variable **do**
    if T played an X_T-variable in the previous move **then**
        play according to the same winning strategy as in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$
    else if T played a_i in the previous move **then** play b_i to make $a_i = b_i$

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

Suppose T has a winning strategy in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$. We describe T's winning strategy in $(\psi, Y_\mathrm{T}, Y_\mathrm{F})$ as Algorithm 3. To see that the strategy works, note that the winning strategy in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$ ensures that $\varphi(x)$ is satisfied by the assignment x to $X_\mathrm{T} \cup X_\mathrm{F}$ in the first phase, so according to Claim 1, there is an assignment z to Z (the set of fresh variables introduced in the definition of φ') 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 $(\varphi, X_\mathrm{T}, X_\mathrm{F})$. We describe F's winning strategy in $(\psi, Y_\mathrm{T}, Y_\mathrm{F})$ as Algorithm 4. To see that the strategy works, note that the winning strategy in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$ ensures that $\varphi(x)$ is unsatisfied by the assignment x to $X_\mathrm{T} \cup X_\mathrm{F}$, 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 $\psi(x, a, b) = \varphi'(x, z)$.

2.2.3 $\mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}}$, $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{T}}$, $\mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{T}}$

Corollary 4. $\mathrm{G}^{\%}_{5,\mathrm{F}\cdots\mathrm{F}}$ *is* PSPACE*-complete.*

Proof. The reduction is $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{5,\mathrm{F}\cdots\mathrm{F}}$. First we show $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}}$. Suppose $(\varphi, X_\mathrm{T}, X_\mathrm{F})$ is an instance of $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}}$. We introduce a dummy variable z that does not appear in φ and give it to F: $Y_\mathrm{T} = X_\mathrm{T}$, $Y_\mathrm{F} = X_\mathrm{F} \cup \{z\}$. Thus $(\varphi, Y_\mathrm{T}, Y_\mathrm{F})$ is an instance of $\mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}}$. The reduction works since: When F has a winning strategy in $(\varphi, X_\mathrm{T}, X_\mathrm{F})$, F can play z as the first move, then continue the winning strategy as in $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$. Conversely, when T has a winning strategy in $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$, T can use the same strategy from $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$ if F plays z as the starting move. If F plays x_i instead of playing z at the beginning then T can ignore F's first move and start playing with the same strategy from $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$. The game can continue as usual until F plays z, then T can pretend that F just played x_i and continue the usual strategy from there. At the end, T and F have both played the same assignment as they would have in $(\varphi, X_{\mathrm{T}}, X_{\mathrm{F}})$, so T still wins.

This completes the reduction $\mathrm{G}^{\%}_{\mathrm{T}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}}$. Now the reduction $\mathrm{G}^{\%}_{\mathrm{F}\cdots\mathrm{F}} \leqslant \mathrm{G}^{\%}_{5,\mathrm{F}\cdots\mathrm{F}}$ is identical to Section 2.2.2 except it is F's move first. □

Observation 1. $\mathrm{G}^{\%}_{3,\mathrm{T}\cdots\mathrm{F}}$, $\mathrm{G}^{\%}_{3,\mathrm{F}\cdots\mathrm{F}}$, $\mathrm{G}^{\%}_{3,\mathrm{T}\cdots\mathrm{T}}$, $\mathrm{G}^{\%}_{3,\mathrm{F}\cdots\mathrm{T}}$ *are* NP*-hard.*

Proof. First we show that 3-SAT $\leqslant \mathrm{G}^{\%}_{3,\mathrm{T}\cdots\mathrm{F}}$. Suppose (φ, X) is an instance of 3-SAT. We construct the instance $(\varphi, Y_{\mathrm{T}}, Y_{\mathrm{F}})$ of $\mathrm{G}^{\%}_{3,\mathrm{T}\cdots\mathrm{F}}$ where $Y_{\mathrm{T}} = X$ and Y_{F} is a new set of fresh variables such that $|Y_{\mathrm{F}}| = |X|$. F's moves do not matter. If φ is satisfiable then T can play a satisfying assignment, otherwise T cannot satisfy φ.

The reductions for the other patterns are similar. Only the number of dummy variables $|Y_{\mathrm{F}}|$ changes: $|Y_{\mathrm{F}}| = |X| + 1$ for $\mathrm{G}^{\%}_{3,\mathrm{F}\cdots\mathrm{F}}$, $|Y_{\mathrm{F}}| = |X| - 1$ for $\mathrm{G}^{\%}_{3,\mathrm{T}\cdots\mathrm{T}}$, and $|Y_{\mathrm{F}}| = |X|$ for $\mathrm{G}^{\%}_{3,\mathrm{F}\cdots\mathrm{T}}$. □

3 2-CNF

In order to analyze the complexity of the games G_2 and $\mathrm{G}^{\%}_2$, 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 $\overline{x}_i$.[3]
- 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 $\overline{\ell}_i \to \ell_i$.

In our arguments, we write $\ell_i \rightsquigarrow \ell_j$ to mean there exists a path from node ℓ_i to node ℓ_j. 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. We say an edge is incident to a node if the node is an endpoint of the edge (head or tail). We say two nodes are neighbors if there exists an edge between them (in either direction).

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 Section 3.2. In terms of the graph representation, linear time means $O(n + m)$ where n = number of nodes and m = number of edges.

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

3.1 G_2

G_2 is the unordered analogue of the 2-TQBF game. We prove Theorem 3 by separately considering the cases $\mathrm{G}_{2,\mathrm{F}\ldots\mathrm{F}}$ in Section 3.1.1, $\mathrm{G}_{2,\mathrm{F}\ldots\mathrm{T}}$ in Section 3.1.2, and $\mathrm{G}_{2,\mathrm{T}\ldots}$ in Section 3.1.3. Our algorithm for G_2 is to run either Algorithm 5 or Algorithm 6 or Algorithm 7, depending on the pattern of who goes first and who goes last.

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

Lemma 1. F *has a winning strategy in* $\mathrm{G}_{2,\mathrm{F}\ldots\mathrm{F}}$ *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., $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \overline{\ell}_j$ (because otherwise ℓ_i would represent the same variable as either ℓ_j or ℓ_k, in which case we would have $\overline{\ell}_i \rightsquigarrow \ell_i$, which is covered by statement *(1)*). 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)*.
- $\ell_j \rightarrow \ell_i \leftarrow \ell_k$ or its mirror $\overline{\ell}_j \leftarrow \overline{\ell}_i \rightarrow \overline{\ell}_k$: Satisfies statement *(2)*.
- $\ell_k \rightarrow \ell_j \rightarrow \ell_i$: Satisfies statement *(2)*.

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 $\ell_i \rightarrow \ell_j$, 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

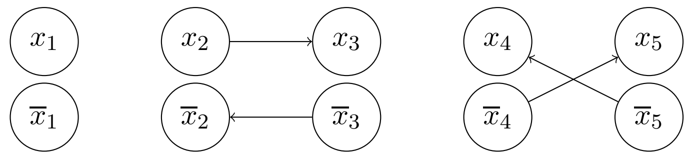

Figure 1: T has a winning strategy in $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{F}}$ for $(\overline{x}_2 \vee x_3) \wedge (x_4 \vee x_5)$

Algorithm 5: Linear Time Algorithm for $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{F}}$

Input: φ, X **Output:** which player has a winning strategy

```
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 incident edges) then output F
output 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 $\ell_j = 1$. $\square$

The characterization of such a graph in the proof of Lemma 1 can be verified in linear time, and that yields a Linear Time algorithm for $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{F}}$. Details of the idea have been described as Algorithm 5.

3.1.2 $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{T}} \in$ Linear Time

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

Lemma 2. F *has a winning strategy in* $\mathrm{G}_{2,\mathrm{F}\cdots\mathrm{T}}$ *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 1, 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.

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

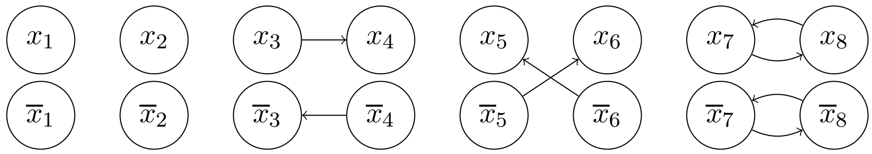

Figure 2: T has a winning strategy in $\mathrm{G}_{2,\mathrm{F}\ldots\mathrm{T}}$ for $(\overline{x}_3 \vee x_4) \wedge (x_5 \vee x_6) \wedge (\overline{x}_7 \vee x_8) \wedge (x_7 \vee \overline{x}_8)$

Algorithm 6: Linear Time Algorithm for $\mathrm{G}_{2,\mathrm{F}\ldots\mathrm{T}}$

Input: φ, X **Output:** which player has a winning strategy

```
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
```

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 2 can be verified in linear time, and that yields a Linear Time algorithm for $\mathrm{G}_{2,\mathrm{F}\ldots\mathrm{T}}$. Details of the idea have been described as Algorithm 6.

3.1.3 $\mathrm{G}_{2,\mathrm{T}\ldots} \in$ Linear Time

In order to win $\mathrm{G}_{2,\mathrm{T}\ldots}$, at the beginning T must locate a node ℓ_i such that after playing it, the game is reduced to a $\mathrm{G}_{2,\mathrm{F}\ldots}$ 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 3. T *has a winning strategy in* $\mathrm{G}_{2,\mathrm{T}\ldots}$ *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* $\mathrm{G}_{2,\mathrm{F}\ldots}$.

Proof. Suppose T has a winning strategy in $\mathrm{G}_{2,\mathrm{T}\ldots}$. 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 $\mathrm{G}_{2,\mathrm{F}\ldots}$ 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 $\mathrm{G}_{2,\mathrm{F}\ldots}$. ☐

We define L as the set of all nodes that have no outgoing edges. If $|L| = 0$, then according to Lemma 3, T has no winning strategy in $\mathrm{G}_{2,\mathrm{T}\ldots}$. If $|L| > 0$, then the trivial algorithm for $\mathrm{G}_{2,\mathrm{T}\ldots}$ is, checking for each node $\ell_i \in L$, whether or not after playing $\ell_i = 1$ the rest of the graph becomes a

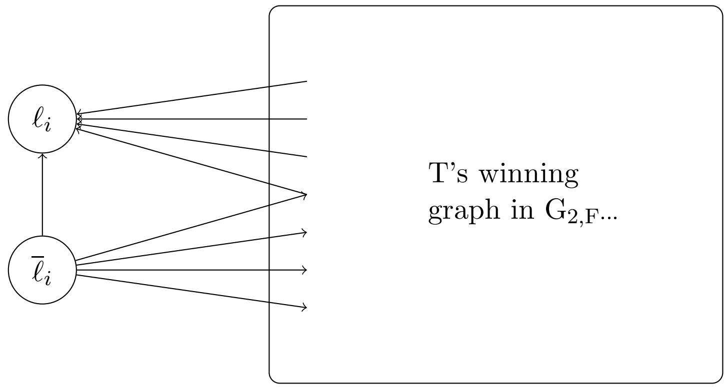

Figure 3: T's winning graph in $G_{2,T\ldots}$ (all edges incident to ℓ_i or $\overline{\ell}_i$ are optional)

winning graph for T in $G_{2,F\ldots}$, 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.

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 1 and Lemma 2 holds. We consider this case in Claim 2.
- There exists a node $\ell_i \in L$ such that statement *(2)* from Lemma 1 and Lemma 2 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 1 and Lemma 2 holds. We consider this case in Claim 4.

Then in Claim 5 and Claim 6 we analyze the efficiency of this approach.

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

Proof. Suppose T's first move is not $\ell_i = 1$. If T's first move assigns 1 to a node with an outgoing edge, then T loses as in Lemma 3. Otherwise, T's first move must not involve any variable on the path $\overline{\ell}_i \rightsquigarrow \ell_i$ (since if it assigns 1 to a node on the path other than ℓ_i then that node has an outgoing edge, and if it assigns 0 to a node on the path other than $\overline{\ell}_i$ then that node's complement has an outgoing edge). In this case, in the rest of the game T loses by statement *(1)* from Lemma 1 and Lemma 2. $\square$

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* $G_{2,T\ldots}$*, then* T*'s first move must be $\ell_i = 1$ or $\overline{\ell}_j = 1$ or $\overline{\ell}_k = 1$.*

Proof. Suppose T's first move is not $\ell_i = 1$ or $\overline{\ell}_j = 1$ or $\overline{\ell}_k = 1$. If T's first move assigns 1 to a node with an outgoing edge, then T loses as in Lemma 3. Otherwise, T's first move must not involve any variable on the paths $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$ (since if it assigns 1 to a node on the paths other than ℓ_i then that node has an outgoing edge, and if it assigns 0 to a node on the paths other than ℓ_j or ℓ_k then that node's complement has an outgoing edge). In this case, in the rest of the game T loses by statement *(2)* from Lemma 1 and Lemma 2. $\square$

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* $\mathrm{G}_{2,\mathrm{T}\ldots}$*, then for all $\ell_i \in L$,* T *has a winning strategy in* $\mathrm{G}_{2,\mathrm{T}\ldots}$ *beginning with $\ell_i = 1$.*

Proof. For all nodes $\ell_i \in L$, statement *(1)* and statement *(2)* from Lemma 1 and Lemma 2 do not hold. So all nodes $\ell_i \in L$ are either isolated single nodes or have only one isolated incoming edge, from another variable's node outside L. (The argument is similar to the situation when statement *(1)* and statement *(2)* do not hold in Lemma 1 and Lemma 2.) If T plays any $\ell_i \in L$ as $\ell_i = 1$, then it does not affect whether or not statements *(1)*, *(2)*, *(3)* from Lemma 1 and Lemma 2 hold on the rest of the graph. So, if T indeed has a winning strategy then it does not matter which $\ell_i \in L$ is assigned as 1 as the first move. □

The overall idea is: If we can find an ℓ_i for which statement *(1)* or statement *(2)* from Lemma 1 and Lemma 2 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 1 and Lemma 2 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.*

Proof. It is sufficient to check three cases:

- ℓ_i has indegree > 1: Then we can find $\ell_j \to \ell_i \leftarrow \ell_k$.
- ℓ_i has indegree $= 1$: There exists ℓ_j with $\ell_j \to \ell_i$. Then look for ℓ_k with $\ell_k \to \ell_j$.
- ℓ_i has indegree < 1: Such ℓ_j, ℓ_k 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 $\overline{\ell}_i \rightsquigarrow \ell_i$.*

Proof. Since $\ell_j \rightsquigarrow \ell_i \leftsquigarrow \ell_k$ does not hold, ℓ_i has indegree $\leqslant 1$ and any incoming neighbor has indegree 0. It is sufficient to check two cases:

- ℓ_i has indegree $= 1$: Then check if $\overline{\ell}_i \to \ell_i$.
- ℓ_i has indegree < 1: Such a path does not exist. □

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

3.2 $\mathrm{G}_2^{\%}$

In this section we prove Theorem 4 by separately considering the cases $\mathrm{G}^{\%}_{2,\ldots\mathrm{F}}$ in Section 3.2.1 and $\mathrm{G}^{\%}_{2,\ldots\mathrm{T}}$ in Section 3.2.2. Our algorithm for $\mathrm{G}_2^{\%}$ is to run either Algorithm 8 or Algorithm 9, depending on the pattern of who goes first and who goes last. We let V_T and V_F be the sets of nodes created from X_T and X_F respectively. Also, let $V = V_\mathrm{T} \cup V_\mathrm{F}$ be the set of all nodes.

Algorithm 7: Linear Time Algorithm for $\mathrm{G}_{2,\mathrm{T}\ldots}$

Input: φ, X **Output:** which player has a winning strategy

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 $\mathrm{G}_{2,\mathrm{F}\ldots}$ game
  if T has a winning strategy in $\mathrm{G}_{2,\mathrm{F}\ldots}$ **then** output T
output F

3.2.1 $\mathrm{G}^{\%}_{2,\ldots\mathrm{F}} \in$ Linear Time

Lemma 4. F *has a winning strategy in* $\mathrm{G}^{\%}_{2,\ldots\mathrm{F}}$ *iff at least one of the following statements holds in the graph* $g(\varphi, X)$*:*

(1) There exists a node $\ell_i \in V$ *such that* $\overline{\ell}_i \leftrightsquigarrow \ell_i$.

(2) There exist two nodes ℓ_i, $\ell_j \in V_\mathrm{F}$ *such that* $\ell_i \rightsquigarrow \ell_j$.

(3) There exist two nodes $\ell_i \in V_\mathrm{F}$ *and* $\ell_j \in V_\mathrm{T}$ *such that* $\ell_i \leftrightsquigarrow \ell_j$.

Proof. Suppose at least one of the statements holds.

If statement *(1)* holds, F can win by any strategy since φ is unsatisfiable.

If statement *(2)* holds, F can play $\ell_i = 1$, and either ℓ_j is $\overline{\ell}_i = 0$ or F can play $\ell_j = 0$ and win.

If statement *(3)* holds, F can wait by playing variables other than x_i with arbitrary values until T plays x_j. Then F can 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. Since statement *(1)* does not hold, the graph has a satisfying assignment [APT79]. Since statement *(2)* does not hold, there is no edge or path between any two nodes of V_F. Since statement *(3)* also does not hold, there is no node in V_F that belongs to a strong component of size > 1. Intuitively, if V_F is reachable from a node $\ell_i \in V_\mathrm{T}$ then F can force T to assign $\ell_i = 0$ by assigning the other endpoint of the path as 0, and similarly if $\ell_i \in V_\mathrm{T}$ is reachable from V_F then F can force T to assign $\ell_i = 1$. All other nodes in V_T are intuitively free from the influence of F's strategy, meaning T is free to assign any bit value he/she likes. This motivates us to partition V_T into three sets $V_{\mathrm{T},0}$, $V_{\mathrm{T},1}$, $V_{\mathrm{T},\mathrm{free}}$, defined as follows:

- $V_{\mathrm{T},0} = \{\ell_j \in V_\mathrm{T} : \ell_j \rightsquigarrow \ell_i \text{ for some } \ell_i \in V_\mathrm{F}\}$

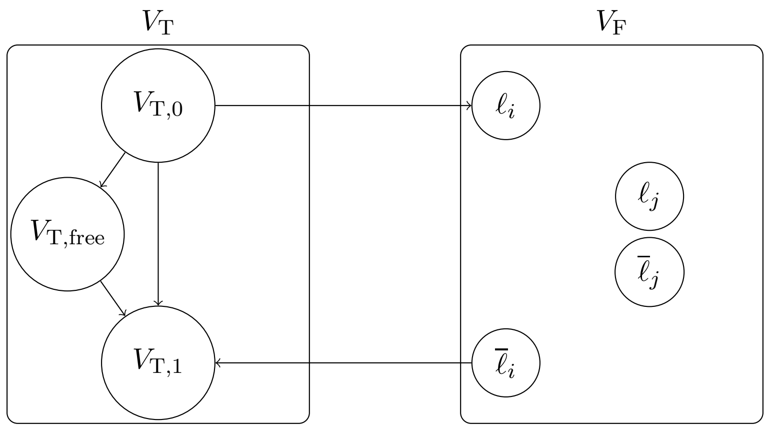

Figure 4: T has a winning strategy in $\mathrm{G}^{\%}_{2,\cdots\mathrm{F}}$

- $V_{\mathrm{T},1} = \{\ell_j \in V_\mathrm{T} : \ell_j \leftsquigarrow \ell_i \text{ for some } \ell_i \in V_\mathrm{F}\}$
- $V_{\mathrm{T,free}} = V_\mathrm{T} \setminus (V_{\mathrm{T},0} \cup V_{\mathrm{T},1})$

This is indeed a partition: there must not be any common node that is in both $V_{\mathrm{T},0}$ and $V_{\mathrm{T},1}$, because this would create either a path between two nodes of V_F (satisfying statement *(2)*) or a cycle touching both V_T and V_F (satisfying statement *(3)*). Note that there cannot be any edge entering $V_{\mathrm{T},0}$, leaving $V_{\mathrm{T},1}$, or between $V_{\mathrm{T,free}}$ and V_F. In general V_F may have many isolated nodes. A general case of the graph looks like Figure 4.

Now we describe a winning strategy for T on such a graph. Whatever F plays, T picks any remaining node to play. If the node is in $V_{\mathrm{T},0}$, T assigns it 0. If the node is in $V_{\mathrm{T},1}$, T assigns it 1. If the node is in $V_{\mathrm{T,free}}$, T assigns it according to a satisfying assignment that exists since statement *(1)* does not hold.

The strategy works since each edge $\ell_i \to \ell_j$ has either $\ell_i \in V_{\mathrm{T},0}$ in which case it gets satisfied by $\ell_i = 0$, or $\ell_j \in V_{\mathrm{T},1}$ in which case it gets satisfied by $\ell_j = 1$, or $\ell_i, \ell_j \in V_{\mathrm{T,free}}$ in which case it gets satisfied by the satisfying assignment. ☐

Now, we develop a linear time algorithm to check statements *(1)*, *(2)*, *(3)* in Lemma 4. We start by creating a topologically sorted DAG of strong components for the whole graph. The DAG construction can be done in linear time [Tar72]. We can check statements *(1)*, *(3)* by directly inspecting the strong components. In order to check statement *(2)* we do dynamic programming over the topological order of strong components to see whether any strong component containing a node in V_F is reachable from any other such strong component. The idea has been described as Algorithm 8.

3.2.2 $\mathrm{G}^{\%}_{2,\cdots\mathrm{T}} \in$ Linear Time

The characterization is the same as for $\mathrm{G}^{\%}_{2,\cdots\mathrm{F}}$ except statement *(3)*.

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

(1) There exists a node $\ell_i \in V$ *such that* $\overline{\ell}_i \leftrightsquigarrow \ell_i$*.*

Algorithm 8: Linear Time algorithm for $\mathrm{G}^{\%}_{2,\cdots\mathrm{F}}$

Input: φ, X **Output:** which player has a winning strategy

construct $g(\varphi, X)$
construct g^* as the DAG of strong components from $g(\varphi, X)$
let S = set of all strong components of $g(\varphi, X)$ (nodes of g^*)
foreach $\ell_i \in V$ **do**
    let $s = \ell_i$'s strong component
    if $(\overline{\ell}_i \in s)$ or $(\ell_i \in V_\mathrm{F}$ and $|s| > 1)$ **then** output F
let S_F = set of strong components containing nodes from V_F
let S_T = set of strong components containing nodes from V_T
mark all $s \in S_\mathrm{F}$ as "reachable from S_F"
topologically order $s_1, s_2, s_3, \ldots \in S$ so edges of g^* go from lower to higher indices
foreach $i = 1, 2, 3, \ldots, |S|$ **do**
    if $\exists j < i$ such that $s_j \to s_i$ and s_j is marked **then**
        if $s_i \in S_\mathrm{T}$ **then** mark s_i as "reachable from S_F"
        else output F
output T

(2) There exist two nodes ℓ_i, $\ell_j \in V_\mathrm{F}$ such that $\ell_i \rightsquigarrow \ell_j$.

(3) There exist three nodes $\ell_i \in V_\mathrm{F}$ and ℓ_j, $\ell_k \in V_\mathrm{T}$ such that $\ell_j \leftrightsquigarrow \ell_i \leftrightsquigarrow \ell_k$.

Proof. Suppose at least one of the statements holds. In Lemma 4, we have already seen that statement *(1)* and statement *(2)* allow player F to win.

If statement *(3)* holds, F can wait by playing variables other than x_i with arbitrary values until T plays x_j or x_k. Then F can respond by making $\ell_i \neq \ell_j$ or $\ell_i \neq \ell_k$ and win.

Conversely, suppose none of the statements hold. The graph structure remains the same as we had for $\mathrm{G}^{\%}_{2,\cdots\mathrm{F}}$, except it is allowed to have shared strong components of size 2 which form a matching between some nodes of V_T and V_F. Intuitively, F can force T to assign $V_{\mathrm{T,sc}}$ nodes as any bit values he/she likes, by assigning the corresponding matching endpoints, and T must wait to find out what those values are. We partition V_T into four sets $V_{\mathrm{T,sc}}$, $V_{\mathrm{T},0}$, $V_{\mathrm{T},1}$, $V_{\mathrm{T,free}}$, defined as follows:

- $V_{\mathrm{T,sc}} = \{\ell_j \in V_\mathrm{T} : \ell_j \leftrightarrow \ell_i \text{ for some } \ell_i \in V_\mathrm{F}\}$
- $V_{\mathrm{T},0} = \{\ell_j \in V_\mathrm{T} : \ell_j \rightsquigarrow \ell_i \text{ for some } \ell_i \in V_\mathrm{F}\} \setminus V_{\mathrm{T,sc}}$
- $V_{\mathrm{T},1} = \{\ell_j \in V_\mathrm{T} : \ell_j \leftsquigarrow \ell_i \text{ for some } \ell_i \in V_\mathrm{F}\} \setminus V_{\mathrm{T,sc}}$
- $V_{\mathrm{T,free}} = V_\mathrm{T} \setminus (V_{\mathrm{T,sc}} \cup V_{\mathrm{T},0} \cup V_{\mathrm{T},1})$

This is indeed a partition: there must not be any common node that is in both $V_{\mathrm{T},0}$ and $V_{\mathrm{T},1}$, because this would create either a path between two nodes of V_F (satisfying statement *(2)*) or a cycle of length > 2 that touches both V_T and V_F (satisfying statement *(2)* or statement *(3)*). Note that there cannot be any edge entering $V_{\mathrm{T},0}$, leaving $V_{\mathrm{T},1}$, between $V_{\mathrm{T,free}}$ and $V_{\mathrm{T,sc}} \cup V_\mathrm{F}$, between nodes of $V_{\mathrm{T,sc}}$, or between $V_{\mathrm{T,sc}}$ and V_F except the matching edges. In general V_F may have many isolated nodes. A general case of the graph looks like Figure 5.

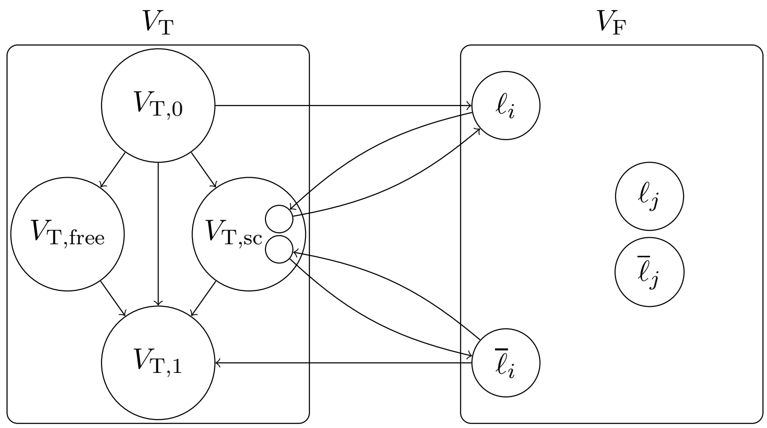

Figure 5: T has a winning strategy in $\mathrm{G}^{\%}_{2,\cdots\mathrm{T}}$

Now we describe a winning strategy for T on such a graph. If F's previous move was in a shared strong component $\ell_j \leftrightarrow \ell_i$ then make $\ell_j = \ell_i$. Otherwise, T picks any remaining node not in $V_{\mathrm{T,sc}}$. If the node is in $V_{\mathrm{T},0}$, T assigns it 0. If the node is in $V_{\mathrm{T},1}$, T assigns it 1. If the node is in $V_{\mathrm{T,free}}$, T assigns it according to a satisfying assignment that exists since statement *(1)* does not hold.

The strategy works since T has the last move, so T will always be able to respond when F plays in a shared strong component to ensure these edges gets satisfied. Each other edge $\ell_i \to \ell_j$ has either $\ell_i \in V_{\mathrm{T},0}$ in which case it gets satisfied by $\ell_i = 0$, or $\ell_j \in V_{\mathrm{T},1}$ in which case it gets satisfied by $\ell_j = 1$, or $\ell_i, \ell_j \in V_{\mathrm{T,free}}$ in which case it gets satisfied by the satisfying assignment. □

The algorithm for checking the characterization of such a graph is almost identical to Algorithm 8, except in line 6 it is necessary to check for the size of strong components being greater than 2 instead of 1. The idea has been described as Algorithm 9.

4 Conclusion

In this paper, we have determined the ordered and partitioned game complexities for 2-CNFs and 5-CNFs, thereby providing new algorithmic techniques for solving games and new starting points to prove hardness of other games. Interestingly, any completeness result for 3-CNFs or 4-CNFs, for either the unordered or partitioned version, remains open. In this direction, we boldly conjecture that the unordered game on 3-CNFs is tractable. So far we have already proven this conjecture is indeed true for 3-CNFs under a certain restriction—that each width-3 clause has a variable that occurs in no other clauses [RW19]. We have also proven that the unordered 4-CNF game is at least NL-hard. Future work could also explore hardness of approximation for the unordered and partitioned CNF games.

Acknowledgments

We thank anonymous reviewers for helpful comments. This work was supported by NSF grant CCF-1657377.

Algorithm 9: Linear Time algorithm for $\mathrm{G}^{\%}_{2,\ldots\mathrm{T}}$

Input: φ, X **Output:** which player has a winning strategy

construct $g(\varphi, X)$
construct g^* as the DAG of strong components from $g(\varphi, X)$
let S = set of all strong components of $g(\varphi, X)$ (nodes of g^*)
foreach $\ell_i \in V$ **do**
  let $s = \ell_i$'s strong component
  if ($\overline{\ell}_i \in s$) or ($\ell_i \in V_{\mathrm{F}}$ and $|s| > 2$) **then** output F
let S_{F} = set of strong components containing at least one node from V_{F}
let S_{T} = set of strong components containing only nodes from V_{T}
mark all $s \in S_{\mathrm{F}}$ as "reachable from S_{F}"
topologically order $s_1, s_2, s_3, \ldots \in S$ so edges of g^* go from lower to higher indices
foreach $i = 1, 2, 3, \ldots, |S|$ **do**
  if $\exists j < i$ such that $s_j \to s_i$ and s_j is marked **then**
    if $s_i \in S_{\mathrm{T}}$ **then** mark s_i as "reachable from S_{F}"
    else output F
output T

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.

[AS03] Argimiro Arratia and Iain Stewart. A note on first-order projections and games. *Theoretical Computer Science*, 290(3):2085–2093, 2003.

[BDG+15] 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.

[BDK+16] Boštjan Brešar, Paul Dorbec, Sandi Klavžar, Gašper Košmrlj, and Gabriel Renault. Complexity of the game domination problem. *Theoretical Computer Science*, 648:1–7, 2016.

[BI97] William Burley and Sandy Irani. On algorithm design for metrical task systems. *Algorithmica*, 18(4):461–485, 1997.

[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.

[FG87] Aviezri Fraenkel and Elisheva Goldschmidt. PSPACE-hardness of some combinatorial games. *Journal of Combinatorial Theory, Series A*, 46(1):21–38, 1987.

[FGM+15] Stephen Fenner, Daniel Grier, Jochen Messner, Luke Schaeffer, and Thomas Thierauf. Game values and computational complexity: An analysis via black-white combinatorial games. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC)*, pages 689–699. Springer, 2015.

[Hea09] 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.

[RW19] Md Lutfar Rahman and Thomas Watson. Tractable unordered 3-CNF games. Technical Report TR19-160, Electronic Colloquium on Computational Complexity (ECCC), 2019. URL: https://eccc.weizmann.ac.il/report/2019/160/.

[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.

[Sla00] Wolfgang Slany. The complexity of graph Ramsey games. In *Proceedings of the 2nd International Conference on Computers and Games (CG)*, pages 186–203. Springer, 2000.

[Sla02] Wolfgang Slany. Endgame problems of Sim-like graph Ramsey avoidance games are PSPACE-complete. *Theoretical Computer Science*, 289(1):829–843, 2002.

[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.

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[TDU11] Sachio Teramoto, Erik Demaine, and Ryuhei Uehara. The Voronoi game on graphs and its complexity. *Journal of Graph Algorithms and Applications*, 15(4):485–501, 2011.

[vV13] Jan van Rijn and Jonathan Vis. Complexity and retrograde analysis of the game Dou Shou Qi. In *Proceedings of the 25th Benelux Conference on Artificial Intelligence (BNAIC)*, 2013.

[ZM04] Ling Zhao and Martin Müller. Game-SAT: A preliminary report. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 357–362, 2004.