Complexity of Unbounded Boolean Formula Games

by

Lilith Chung

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2026

Authored by: Lilith Chung
Department of Electrical Engineering and Computer Science
January 23, 2026

Certified by: Erik D. Demaine
Professor of Electrical Engineering and Computer Science, Thesis Supervisor

Certified by: Ronitt Rubinfeld
Professor of Electrical Engineering and Computer Science, Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Complexity of Unbounded Boolean Formula Games

by

Lilith Chung

Submitted to the Department of Electrical Engineering and Computer Science
on January 23, 2026 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ABSTRACT

Chandra and Stockmeyer's influential 1979 work "Provably Difficult Combinatorial Games" [1] introduced six EXPTIME-complete games, numbered G_1 through G_6, which are played by setting Boolean variables in accordance with certain Boolean formulae. In this thesis, we systematically define a class of games generalizing $G_1 \dots G_6$, and determine the computational complexities of a variety of games in this classification. We also characterize the computational complexity of several variants of the games $G_1 \dots G_6$ obtained by restricting the types of CNF or DNF formulae used in their definitions.

Thesis supervisor: Erik D. Demaine
Title: Professor of Electrical Engineering and Computer Science

Thesis supervisor: Ronitt Rubinfeld
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

This work grew out of the open problem sessions of MIT's 6.892 and 6.5440 classes on Algorithmic Lower Bounds, taught by Erik in 2019 and 2023. Thank you to the staff and participants of that class for many inspiring discussions. Special thanks to my collaborators Josh, Erik, Jenny, and Hayashi who worked on these problems with me.

Thank you to my advisors Erik and Ronitt for your invaluable guidance and steadfast patience.

Thank you to my friends and family, inside and outside of academia, for your endless support. Lastly, thank you Riley and Ashlyn, for everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unbounded Boolean formula games are a class of two-player perfect-information games. An example of such a game is the game G_4 defined by Chandra and Stockmeyer [1].

In G_4, each of two players controls a finite set of Boolean variables, whose initial values are specified. On each player's turn they may change the value of any single one of their own variables. The goal of the game is to cause a certain Boolean formula in 12-DNF form to become satisfied: the first player to satisfy the formula wins.

A key property of G_4 is that it is *unbounded*: There is, a priori, no limitation on the number of turns which may take place. This distinguishes G_4 from *bounded* games, such as those studied by Schaefer [2] in 1978. A typical bounded game played with Boolean formulae is QSAT, in which players take turns setting the values of Boolean variables in a set order, and compete over whether a formula turns out true or false. In QSAT, each variable is set only once, so the number of turns in the game is limited by the number of variables. Bounded games are naturally contained in PSPACE, and indeed QSAT is a prototypical example of a PSPACE-complete problem.

In contrast, unbounded games are naturally contained in EXPTIME. For instance, in [1], the problem of deciding the outcome of a G_4 game, given the Boolean formula and the initial values of the variables, was shown to be EXPTIME-complete, showing that G_4 is qualitatively more complex than QSAT, assuming PSPACE $\neq$ EXPTIME. The same result was shown for five other Boolean formula games (numbered $G_1, \ldots, G_6$). These results on unbounded Boolean formula games have been used to prove that many other types of unbounded games, including Checkers [3], Chess [4], and Go [5], are also EXPTIME-hard. Boolean formula games have the common feature that they are played by changing the values of Boolean variables, and Boolean formulae are used to evaluate the winner. They vary on other aspects such as the structure of a legal move, or whether the goal is to satisfy or avoid the formulae in question.

This thesis sets out a systematic taxonomy of unbounded Boolean formula games which includes the games $G_1, \ldots, G_6$ as well as many others, and analyzes the computational complexity of several such games. We show EXPTIME-completeness both of variations of $G_1, \ldots, G_6$ with various restrictions on the formulae, as well as several novel games. We also exhibit a few unbounded games whose outcome can be computed in PSPACE for quite nontrivial reasons. Our results are summarized in Table 1.1.

Formula ownership	Flip	Outcome	a.k.a.	Results
Partizan	Any	Win		Σ_2^{P} [Theorem 7]
Partizan	> 0	Win		PSPACE [Corollary 15]
Partizan	Any	Lose	G_1	3DNF EXPTIME-c [1]
Partizan	> 0	Lose		3DNF EXPTIME-c [↑, Lemma 3]
Partizan	Any	Win/Lose		Σ_3^{P} [Theorem 8]
Partizan	> 0	Win/Lose		
Partizan	0 or 1	Win	G_2	12DNF EXPTIME-c [1] 2DNF P [Theorem 9] 6DNF EXPTIME-c [Theorem 17]
Partizan	1	Win		6DNF EXPTIME-c [Theorem 17]
Partizan	0 or 1	Lose	$\tilde{G}_3$	5DNF EXPTIME-c [Theorem 16] 5DNF/Constant EXPTIME-c [Theorem 21]
Partizan	1	Lose	G_3	12DNF EXPTIME-c [1] 5DNF EXPTIME-c [Theorem 16] 5DNF/Constant EXPTIME-c [Theorem 21]
Partizan	0 or 1	Win/Lose		12DNF EXPTIME-c [Theorem 22]
Partizan	1	Win/Lose		12DNF EXPTIME-c [Theorem 22]
Impartial	Any	Win		Σ_2^{P} [Theorem 7]
Impartial	> 0	Win		PSPACE [Corollary 15]
Impartial	Any	Lose		NP [Theorem 6]
Impartial	> 0	Lose		
Impartial	Any	Win/Lose		Σ_3^{P} [Theorem 8]
Impartial	> 0	Win/Lose		
Impartial	0 or 1	Win	G_4	13DNF EXPTIME-c [1] 2DNF P [Theorem 9] 6DNF EXPTIME-c [Theorem 17]
Impartial	1	Win		6DNF EXPTIME-c [Theorem 17]
Impartial	0 or 1	Lose		NP [Theorem 6]
Impartial	1	Lose		5DNF EXPTIME-c [Theorem 18]
Impartial	0 or 1	Win/Lose		
Impartial	1	Win/Lose		
Single	Any	Win		Σ_3^{P} [Theorem 8]
Single	> 0	Win		PSPACE [Corollary 15]
Single	Any	Lose		Σ_3^{P} [Theorem 8]
Single	> 0	Lose		
Single	0 or 1	Win	G_5/G_6	CNF EXPTIME-c [1]
Single	1	Win		CNF EXPTIME-c [↑, Lemma 3]
Single	0 or 1	Lose		12DNF EXPTIME-c [Theorem 22]
Single	1	Lose		12DNF EXPTIME-c [Theorem 22]

Table 1.1: Table of results already known or obtained in this thesis. See Definition 1 for an explanation of the terms used.

The remainder of this thesis is structured as follows. Chapter 2 will lay out the classification

of Boolean formula games. Chapter 3 will prove algorithmic (upper-bound) results. Chapter 4 will prove hardness (lower-bound) results. Finally, Chapter 5 will discuss open problems and future directions.

This thesis includes joint work with Josh Brunner, Erik D. Demaine, Jenny Diomidova, and Hayashi Layers.

Chapter 2

Classification of Boolean formula games

The games $G_1, \ldots, G_6$ of [1] all follow the same basic structure, which we outline here. They are two-player, perfect-information, turn-based games played with two disjoint sets of Boolean variables X and Y, and two formulae over these variables, which we refer to as ϕ_1 and ϕ_2. We consider the variables X and the formula ϕ_1 as "belonging" to Player I; similarly the variables Y and the formula ϕ_2 belong to Player II. Each player's turn consists of changing the values of some or all of their own variables, followed by checking whether their own formula is true. If it is then the game ends immediately; otherwise it becomes the other player's turn.

We classify the variations in rules between the games as follows. This classification also includes some rule variations not seen in $G_1 \ldots G_6$, but which are useful for defining other games to be discussed.

Definition 1: Classification of unbounded formula games

Formula ownership.

In ***Partizan-formula*** games, each player has their own formula as described above.

In ***Impartial-formula*** games, the two players' formulae are required to be identical, $\phi_1 = \phi_2$.

In ***Single-formula*** games, only Player I has an associated formula ϕ_1. Player II's moves cannot end the game.

Turn structure.

In ***Flip-Any-Number*** games, a player may set the values of all their variables on their turn, without restriction. The ***Flip-Nonzero*** designation is similar but prohibits passing: at least one variable's value must change.

In ***Flip-Zero-Or-One*** games, a player may flip the value of a single variable on their turn, or pass. The ***Flip-Exactly-One*** designation is similar but prohibits passing.

Outcome.

If a player's formula is satisfied after their turn, they either ***Win*** or ***Lose*** according to the designation.

The designation ***Win/Lose*** means that Player I will win if they satisfy their formula, whereas Player II will lose if they satisfy their formula. In this case Player II cannot hope to win the game but can instead aim to prolong it indefinitely.

Formula type.

The ***CNF*** or ***DNF*** designations indicate that the formulae are specified in conjunctive or disjunctive normal forms. We can also set further restrictions on the formulae, such as "k-CNF" or "k-DNF".

The games $G_1 \dots G_6$ proved to be EXPTIME-complete in [1] can be seen to fit into the above classification by minor manipulations of their definitions.

For instance, the game G_1 is defined in [1] using a single formula $F(X, Y, \{\tau\})$, where τ is a *turn variable* which is defined to be 1 after Player I's move and 0 after Player II's move. A player moves by setting all of their own variables and the turn variable, and they lose if F is false after their move. In [1] this game is proved EXPTIME-complete when F is restricted to be a 4CNF formula where each clause includes the turn variable τ or its negation.

Defining $\phi_1(X, Y) = \neg F(X, Y, 1)$ and $\phi_2(X, Y) = \neg F(X, Y, 0)$, we can fit this game into the above classification as a "Lose 3DNF" game. The formulae ϕ_1 and ϕ_2 are in 3DNF because substituting 1 or 0 for τ in F will either reduce each clause in size by 1 or eliminate it altogether.

By this method, the games $G_1 \dots G_6$ proved EXPTIME-complete in [1] are classified as follows:

G_1: Partizan-formula Flip-Any-Number Lose 3DNF

G_2: Partizan-formula Flip-Zero-Or-One Win 12DNF

G_3: Partizan-formula Flip-Exactly-One Lose 12DNF

G_4: Impartial-formula Flip-Zero-Or-One Win 13DNF

G_5: Single-formula Flip-Zero-Or-One Win

G_6: Single-formula Flip-Zero-Or-One Win CNF

The games defined above are all ***Partizan-variable*** games, meaning that the sets of variables X, Y were disjoint. In ***Impartial-variable*** games, there is instead just one set of variables $X = Y$ used by both players. The classification and analysis of Impartial-variable games is made more complicated by the fact that in order to be nontrivial, a "ko" rule forbidding players from simply undoing their opponent's previous moves must be added. In this thesis we will focus on games without ko rules, so all formula games discussed are Partizan-variable.

Each game in the classification defines a corresponding decision problem: Given sets of variables X, Y, initial values X_0, Y_0 for the variables, and formulae ϕ_1, ϕ_2, does perfect play lead to a forced win for Player I? One detail is that perfect play might lead to a draw (i.e., infinite play) rather than a win for either player. In cases where it is relevant, we will specify whether draws are counted as a win for Player I or Player II. For $G_1 \dots G_4$, the games can be shown EXPTIME-complete via reductions which prevent draws from occurring; that is, one player or the other always has a strategy which wins in finite time. For G_5 and G_6, draws are counted as wins for Player II.

Now we write down a few standard observations about the games in Definition 1, which will be used throughout.

First, a standard argument shows that these games are all contained in EXPTIME.

Lemma 2. *The games of Definition 1 are all in* EXPTIME.

Proof. For each of these games there are only exponentially many possible reachable positions. The outcome of the game can be computed by recursively exploring the entire game tree, which takes exponential time. □

Second, games which allow passing can be seen as special cases of games which forbid it. We will use the following lemma to prove results for both the passing-allowed and passing-forbidden versions of games.

Lemma 3. *For any Flip-Any-Number game defined in Definition 1, there is a reduction to the Flip-Nonzero version. For any Flip-Zero-Or-One game, there is a reduction to the Flip-Exactly-One version.*

Proof. The reduction simply adds a new variable for each player which is not mentioned in the formulae. Any time a player would like to pass, they may instead flip the new variable to achieve the same effect. □

Lastly, Impartial-formula and Single-formula games are special cases of Partizan-formula games.

Lemma 4. *For any Impartial-formula game defined in Definition 1, there is a reduction to the Partizan-formula version.*

Proof. Set ϕ_1 and ϕ_2 equal. □

Lemma 5. *For any Single-formula Win game defined in Definition 1, there are reductions to both the Partizan-formula Win version and the Partizan-formula Win/Lose version. For any Single-formula Lose game, there are reductions to both the Partizan-formula Lose version and to the complement of the Partizan-formula Win/Lose version.*

Proof. For the first statement, set $\phi_2 = \bot$. For the second statement, set $\phi_1 = \bot$ and interchange Players I and II. □

Chapter 3

Algorithmic results

In this chapter, we show that certain of the games in Definition 1 are "easy", at least compared to EXPTIME. Specifically, we show that several of them are contained in PSPACE and one of them is contained in P.

Our first three algorithmic results observe that several games are easy to decide because they will inevitably end in a draw unless one of the players can win within the first few turns.

Theorem 6. *The games Impartial-formula Flip-Any-Number Lose and Impartial-formula Flip-Zero-Or-One Lose are in* NP.

Proof. If the formula is unsatisfied, then the game is a draw since both players can simply pass to not lose. The only way a player can lose is if the formula is satisfied by the initial assignment (X_0, Y_0), in which case Player I needs to find a way to unsatisfy it. So Player I draws if and only if there exists an assignment X_1 such that $\phi(X_1, Y_0)$ is false, which is a SAT problem contained in NP. □

Theorem 7. *The games Partizan-formula Flip-Any-Number Win and Impartial-formula Flip-Any-Number Win are in* Σ_2^{P}.

Proof. By Lemma 4 it suffices to consider the Partizan-formula version.

If Player I can win immediately from the starting position (X_0, Y_0), they will do so. Otherwise they must find an assignment X_1 so that Player II cannot win immediately. If Player I can do so, then the game ends in a draw since both players can pass to avoid losing after that. Thus the game is won by Player I if the formula $\exists X_1 : \phi_1(X_1, Y_0)$ is satisfied; else it is a draw if the formula $\exists X_1 : \forall Y_1 : \neg\phi_2(X_1, Y_1)$ is satisfied; else Player II wins.

Thus the question of whether Player I wins is in Σ_2^{P} if we count draws as wins for Player I; or it is in NP if we count draws as wins for Player II. □

Theorem 8. *The games Partizan-formula Flip-Any-Number Win/Lose and Impartial-formula Flip-Any-Number Win/Lose are in* Σ_3^{P}. *Draws are counted as wins for Player II.*

The game Single-formula Flip-Any-Number Win is in Σ_3^{P}. *Draws are counted as wins for Player II.*

The game Single-formula Flip-Any-Number Lose is in Π_3^{P}. *Draws are counted as wins for Player I.*

Proof. The Impartial-formula and Single-formula games are special cases of the Partizan-formula one (Lemmas 4 and 5). Thus it suffices to show that Partizan-formula Win/Lose Flip-Any-Number games are in Σ_3^{P}.

Consider an instance of a Partizan-formula Win/Lose Flip-Any-Number game, and suppose Player I has a forced win. Suppose further that Player II plays in such a way as to maximize the number of turns it takes before they lose. Then there are three possibilities for how Player I wins:

- If Player I wins on the very first turn, it means they chose a variable setting X_1 so that together with the initial setting Y_0 the formula $\phi_1(X_1, Y_0)$ was satisfied.
- If Player I wins because Player II satisfies the formula ϕ_2, it must be the case that on the previous turn Player I set X_1 so that every possible setting Y_2 causes $\phi_2(X_1, Y_2)$ to be satisfied.
- If Player I wins by satisfying their own formula ϕ_1, it must be the case that on their previous turn Player I set X_1 so that every possible setting Y_2 not satisfying $\phi_2(X_1, Y_2)$ has a third assignment X_3 which satisfies $\phi_1(X_3, Y_2)$.

In other words, Player I wins because the quantified formula

$$\exists X_1 : [\phi_1(X_1, Y_0) \vee \forall Y_2 : [\phi_2(X_1, Y_2) \vee \exists X_3 : \phi_1(X_3, Y_2)]]$$

is true. Conversely if this formula is true then Player I has a winning strategy by starting with X_1, and then choosing X_3 according to Player II's choice of Y_2. Thus Player I has a forced win if and only if the above formula is true, which gives a reduction to Σ_3^{P}. □

The third algorithmic result shows that G_2 and G_4 with very small clause widths are decidable in polynomial time.

Theorem 9. *The game G_2 (Partizan-formula Flip-Zero-Or-One Win) with 2DNF formulae is in* P. *The same holds for G_4 (Impartial-formula Flip-Zero-Or-One Win) with 2DNF formulae.*

Proof. Since G_4 is a special case of G_2 (Lemma 4), the second statement follows from the first.

Consider an instance (X, Y, ϕ_1, ϕ_2) of G_2 2DNF and assume for now that $\min(|X|, |Y|) \geq 3$; we can start to characterize optimal play as follows. First, if you can win the game immediately, you should do so; otherwise, you should make a move that prevents your opponent from winning immediately. Building a game tree according to the above rules to a depth of four turns shows that computing the outcome of the game reduces to computing the outcome of positions in which neither player could win immediately if it were their turn. This is because after Player I's first turn, if they are still threatening an immediate win it is because they have a half-satisfied clause involving two variables in X, which guarantees the game will end in two more turns.

Consider a position in which neither player could win immediately if it were their turn. By inverting variables in the formulae we can assume without loss of generality that all variables are initially set to 0.

Taking Player I's perspective, we label the variables in X according to the following categories:

Unsafe variables: Variables x such that a clause of the form $x \wedge y$ or $x \wedge \neg y$ appears in ϕ_2.

Matched variables: Pairs of variables x_1, x_2 such that one of $x_1 \wedge x_2$ or $\neg x_1 \wedge x_2$ or $x_1 \wedge \neg x_2$ appears in ϕ_2.

Since passing is allowed, we can assume neither player will flip an unsafe variable unless doing so wins the game. So the only way for Player I to force a win is by satisfying a clause of the form $x_i \wedge x_j$ in ϕ_1, which requires flipping one of x_i or x_j without either flipping an unsafe variable or giving the opponent a win when flipping a matched variable. We can view matched variables as forming a dependency graph: for instance, if the clause $\neg x_1 \wedge x_2$ appears in ϕ_2 then x_1 must be flipped before x_2; we say x_2 depends on x_1. If the clause $x_1 \wedge x_2$ appears in ϕ_2 we say x_2 depends on $\neg x_1$ and x_1 depends on $\neg x_2$.

Let $D(x_i)$ be the set of literals on which x_i transitively depends; we say x_i is *viable* if $D(x_i)$ doesn't include any unsafe variables (including x_i itself), doesn't include both a variable and its negation, and doesn't include any dependency cycles. Winning the game requires identifying a clause of the form $x_i \wedge x_j$ in ϕ_1, choosing one of x_i or x_j which is viable, and flipping all positive literals in $D(x_i)$ in order by dependency.

Since this is the case for both players, the winner can be computed as the player whose winning formula has a clause containing a viable variable z, such that $D(z)$ has the smallest number of positive literals, with ties broken by whoever is to move. If neither player can win in this way then the game is a draw. This allows computing the winner in polynomial time. □

Our last algorithmic result concerns a class of games where the "interesting" moves are those which work only because the opponent cannot pass.

Theorem 10. *Consider the following class of games. Player I has a finite set A and Player II has a finite set B. A position in the game is an element of $A \times B$. The* winning positions *for I and II are subsets $\phi_1, \phi_2 \subset A \times B$. A move for Player I consists of changing the position (a, b) to some (a', b) where $a \neq a'$; then Player I wins if $(a', b) \in \phi_1$. Similarly Player II moves by changing (a, b) to (a, b') where $b \neq b'$, and wins if $(a, b') \in \phi_2$.*

Suppose that we are given an oracle which answers, given a position (a, b), the question of whether (a, b) is in ϕ_1 and/or ϕ_2. Then the winner of the game can be determined in space $O(\log |A| + \log |B|)$.

To prove Theorem 10, we introduce some definitions.

Definition 11. A move $b \in B$ ***k-counters*** $a \in A$ if Player II has a strategy which wins in at most k turns after Player II moves to (a, b). In particular b 0-counters a if $(a, b) \in \phi_2$.

A move $a \in A$ is ***uniquely k-countered*** by $b \in B$ if b is the only move which k-counters a.

A move $a \in A$ is ***k-safe*** if there does not exist b which k-counters a.

A move $a \in A$ is ***k-legal*** for $b \in B$ if either $(a, b) \in \phi_1$ or a is k-safe or a is uniquely k-countered by b.

The above definitions apply to both sides' moves by interchanging the roles of Players I and II.

Using these definitions we prove some basic facts:

Lemma 12. *Suppose b' is k-illegal for a, and Player II moves from (a, b) to (a, b'). Then Player I wins in at most $k + 1$ more moves.*

Proof. By definition: Player II did not just win the game, b' is not k-safe, and b' is not uniquely k-countered by a. Thus there exists some $a' \neq a$ which k-counters b'. So after Player I moves to (a', b'), they win in k turns. □

Lemma 13. *Suppose a move $b \in B$ is 1-safe. Then in fact b is ∞-safe.*

Proof. Suppose Player I moves to (a, b); we give a strategy for Player II which never loses. By the definition of 1-safety, there exists $b' \neq b$ which either wins immediately (meaning $(a, b') \in \phi_2$) or (a, b') is 0-safe; in either case Player II moves to b'. Then by 0-safety Player I cannot win immediately, and then Player II moves back to b on the subsequent turn and repeats the above process forever. □

Lemma 14. *Let k be a constant such that there exist no k-safe moves for Player II in B, and suppose Player I has a winning strategy starting from (a_0, b_0) which always wins as quickly as possible. Then this strategy does not make any 0-safe moves except possibly in the following cases:*

- *When moving from a position of the form (a_0, b).*
- *After Player II has played a k-illegal move.*

Proof. Suppose for contradiction that this is not true. Then there exists (a, b) where $a \neq a_0$ reachable via the strategy such that b is k-legal for a, and the strategy moves from (a, b) to (a', b) where a' is 0-safe. We claim that the strategy could have instead moved from (a_0, b_0) to (a', b_0) as the very first move, and this would guarantee a faster win, contradicting the definition of the strategy. Because b is k-legal and we only reach (a', b) after at least 2 turns, it takes Player I at least $k + 3$ turns to win overall.

After the hypothesized move to (a', b_0) there are two possibilities. If Player II moves to (a', b) then Player I can reply with a to win in k moves because b is k-legal for a but not k-safe, hence uniquely k-countered by a. Overall the win takes at most $k + 1$ turns.

If Player II moves to some other (a', b') then Player I can reply using the original winning strategy; this takes fewer turns to win since we reached (a', b') faster.

In both cases the win is faster than the original strategy. □

Now we can give the algorithm.

Proof of Theorem 10. We give an algorithm deciding whether Player I can force a win starting from (a_0, b_0); this suffices to compute the outcome of the game. The algorithm is as follows:

- Check if b_0 is 1-illegal for a_0; if so then Player I wins.
- Check if Player II has a 1-safe move; if so then Player II draws by Lemma 13.

- By Lemma 12 we can restrict our attention to moves which are 1-legal for the other half of the position. Since Player II has no 1-safe moves, then by Lemma 14 Player I will not make any 0-safe moves except when moving from a position of the form (a_0, b) or if Player II makes a 1-illegal move. Thus we can safely ignore all moves other than those which immediately win, those which move out of (a_0, b) positions, and those which are uniquely k-countered by the other half of the position.

- Consider the tree of moves rooted at (a_0, b_0) satisfying the above conditions. We claim that we can recursively traverse this tree to determine whether Player I has a forced win using only $O(\log |A| + \log |B|)$ space. First, moves which immediately win terminate the recursion immediately. Second, we can terminate the recursion if there has been more than one move to a_0, since in this case Player II can draw by repeating. Third, for moves which are uniquely k-countered by the other half of the position, we do not need to keep track of the parent in the tree. This is because the parent can be recomputed by searching for the unique k-counter. Finally, we can detect draws by noticing when more than $2|A| \cdot |B|$ turns have been played, guaranteeing that a repetition has occurred.

  Thus, the recursive algorithm only needs to store the current position, a counter of how many turns have been played, a counter of how many moves to a_0 there have been, and any moves out of (a_0, b) positions (at most a constant number).

  Each position and the turn counter takes $O(\log |A| + \log |B|)$ bits to store, and we only need to store constantly many positions, so this uses $O(\log |A| + \log |B|)$ space overall.

□

Corollary 15. *The games Partizan-formula Flip-Nonzero Win, Impartial-formula Flip-Nonzero Win, and Single-formula Flip-Nonzero Win are in* PSPACE.

Proof. By Theorem 10, where A and B are the sets of possible variable assignments. □

Chapter 4

Hardness results

In this chapter we analyze several games classified by Definition 1 and prove that they are EXPTIME-complete. Many of these games are variants of G_3 with restrictions on the types of formulae used.

Recall that G_3 is classified according to Definition 1 as the game Partizan-formula Flip-Exactly-One Lose 12DNF. This means that Players I and II have separate variables X, Y and separate 12DNF formulae ϕ_1, ϕ_2, and a player's turn consists of flipping exactly one of their own variables without satisfying their own formula.

It will be convenient for us to prove our hardness results for the game $\tilde{G}_3$, which is G_3 with the "Flip-Exactly-One" rule changed to "Flip-Zero-Or-One". Lemma 3 gives a reduction from $\tilde{G}_3$ to G_3, so all our hardness results carry over to the version without passing.

We start by showing that $\tilde{G}_3$ remains EXPTIME-complete when the clause width is restricted to 5.

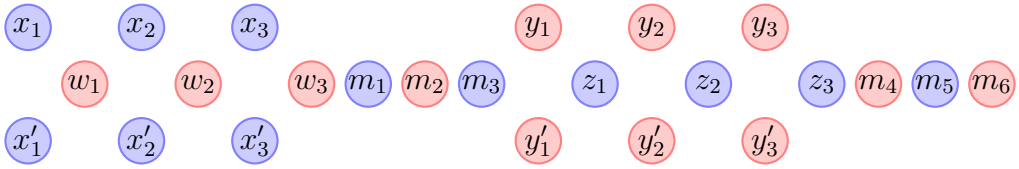

Figure 4.1: The list of expressions L used in the proof of Theorem 16. Blue variables are controlled by Player I, and red ones by Player II. Variables in the same vertical column are XOR'd together to form an entry of L.

Theorem 16. *The games G_3 (Partizan-formula Flip-Exactly-One Lose) and $\tilde{G}_3$ (Partizan-formula Flip-Zero-Or-One Lose) are* EXPTIME*-complete with 5DNF formulae. Draws are impossible.*

Proof. By Lemma 3 it suffices to consider $\tilde{G}_3$.

We reduce from G_1, considered as Partizan-formula Flip-Any-Number Lose 3DNF. Let $(X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_n\}, \phi_1, \phi_2)$ be an instance of G_1. We can assume without loss of generality that X and Y are the same size, that Player I moves first, and (by inverting occurrences in the formulae) that the initial value of each variable is 0.

We define a corresponding instance $(X', Y', \phi_1', \phi_2')$ of $\tilde{G}_3$ as follows. The sets X' and Y' are defined as

$$X' = \{x_i, x_i', z_i : 1 \le i \le n\} \cup \{m_1, m_3, m_5\}$$
$$Y' = \{y_i, y_i', w_i : 1 \le i \le n\} \cup \{m_2, m_4, m_6\},$$

and the initial values of all these variables shall be 0.

To describe the formulae ϕ_1' and ϕ_2' it will be convenient to define the list $L = [\ell_1, \ldots, \ell_{4n+6}]$ by

$$L = \begin{pmatrix} [x_1 \oplus x_1', w_1, x_2 \oplus x_2', w_2, \ldots, x_n \oplus x_n', w_n] \\ \| \, [m_1, m_2, m_3] \\ \| \, [y_1 \oplus y_1', z_1, y_2 \oplus y_2', z_2, \ldots, y_n \oplus y_n', z_n] \\ \| \, [m_4, m_5, m_6] \end{pmatrix},$$

where each entry ℓ_i of L is either a variable or the XOR of two variables. The entries of L alternate between I-controlled and II-controlled expressions. Refer to Figure 4.1 for a pictorial description.

We now define the formulae ϕ_1' and ϕ_2' as:

$$\phi_1' = \left(\bigvee_{\substack{i=3 \\ i \text{ odd}}}^{|L|-1} (\ell_i \neq \ell_{i-1}) \right) \vee (m_6 = x_1 \oplus x_1') \vee (m_1 \neq m_2 \wedge \phi_1)$$
$$\phi_2' = \left(\bigvee_{\substack{i=2 \\ i \text{ even}}}^{|L|} (\ell_i \neq \ell_{i-1}) \right) \vee (m_4 \neq m_5 \wedge \phi_2).$$

The idea behind these formulae is that on turn j, the player to move is forced to flip a variable appearing in ℓ_i where $i \equiv j \pmod{|L|}$. For instance, initially all the ℓ_i evaluate to zero, and so Player I must flip either x_1 or x_1' in order to unsatisfy the clause $m_6 = x_1 \oplus x_1'$. Then, Player II must flip w_1 to unsatisfy the clause $(\ell_2 \neq \ell_1)$. Following that, Player I must flip either x_2 or x_2' to unsatisfy the clause $\ell_3 \neq \ell_2$. This continues until every expression in L evaluates to one, at which point the cycle repeats.

Over the course of this cycle, the players can be seen as taking turns setting all of their own variables. Player I can freely choose the value of x_i by choosing whether to flip x_i or x_i' at the appropriate time, and similarly for Player II. Finally, the clauses $(m_1 \neq m_2 \wedge \phi_1)$ and $(m_4 \neq m_5 \wedge \phi_2)$ have the effect of causing each player to lose if their formula ϕ_1 or ϕ_2 is satisfied after they set all of their variables. Thus each cycle corresponds exactly to a turn for Player I followed by a turn for Player II in G_1.

It is straightforward to check that the formulae ϕ_1' and ϕ_2' can be expanded into 5DNF form, and that they can be computed in polynomial time. Since whoever has a winning strategy in the G_1 instance will also win in the G_3 instance, we have defined a valid reduction. □

Using this result, we can reduce the clause widths for G_2 and G_4.

Theorem 17. *The games G_2 (Partizan-formula Flip-Zero-Or-One Win) and G_4 (Impartial-formula Flip-Zero-Or-One Win) are* EXPTIME*-complete with 6DNF formulae. The same is true for the Flip-Exactly-One versions of both. Draws are impossible.*

Proof. By Lemmas 3 and 4, it suffices to prove that G_4 6DNF is EXPTIME-complete, which we do via a reduction from $\tilde{G}_3$.

Given an instance (X, Y, ϕ_1, ϕ_2) of G_3, define the instance (X', Y', ϕ') of G_4 as follows:

$$\begin{aligned} X' &= X \cup \{x\} \\ Y' &= Y \cup \{y\} \\ \phi' &= (x \wedge y) \vee (x \wedge \phi_2) \vee (y \wedge \phi_1), \end{aligned}$$

where x and y are initially set to 0. Then play proceeds as in $\tilde{G}_3$ since whoever flips x or y first will lose immediately, unless the other player just satisfied their losing formula ϕ_1 or ϕ_2, in which case the player who flips x or y wins. This reduction increases the width of the DNF clauses by 1. □

The reduction from the proof of Theorem 16 can also be used to prove EXPTIME-completeness of the Impartial-formula game with passing forbidden.

Theorem 18. *The game Impartial-formula Flip-Exactly-One Lose 5DNF is* EXPTIME*-complete.*

Proof. Follow the same reduction from G_1 as in the proof of Theorem 16, but instead of two separate formulae ϕ_1', ϕ_2' we instead define the shared formula

$$\phi' = \left(\bigvee_{i=1}^{|L|-1} (\ell_i \neq \ell_{i+1} \wedge \ell_i = m_6) \right) \vee (m_1 \neq m_2 \wedge \phi_1) \vee (m_4 \neq m_5 \wedge \phi_2).$$

This formula, together with the fact that passing is forbidden, forces the players to flip variables in the same order as before. The clauses checking ϕ_1 and ϕ_2 activate following Player I and Player II's turn respectively, which ensures that the loss from satisfying one of the formulae is attributed to the correct player. □

The next sequence of results starts with a new type of game in which only Player II has a win condition.

Definition 19: Latch game

The ***latch game*** is a game played with a set of variables $x_1, \ldots, x_n, \tilde{x}_1, \ldots, \tilde{x}_n, y$ and a 3DNF formula ϕ_1 over all the variables. A turn for Player I consists of setting $\tilde{x}_i \leftarrow x_i$ for all i, then setting the x_i however they choose. Player I loses if ϕ_1 is satisfied after their turn. A turn for Player II consists only of setting y.
Player I wins if the game goes on forever.

Theorem 20. *The latch game is* EXPTIME*-complete.*

Proof. We reduce from the halting problem for linear-bounded alternating Turing machines. Consider linear-bounded alternating Turing machines with the following properties:

- Each configuration has at most two successor configurations.
- Existential configurations and universal configurations alternate.
- Every universal configuration has at least one successor.
- The machine begins in a universal configuration with exactly one successor.

Then the question of whether such a machine rejects or fails to reject is EXPTIME-complete.[1]

Let M be such a machine. We can encode the transition relation of M into a Boolean formula $\mathsf{Next}(i, U, V)$ which evaluates to true if U and V encode configurations of M and either U has two successor configurations V_0, V_1 such that $V = V_i$, or V is the unique successor configuration to U. Furthermore we can transform Next into a 3CNF formula $\mathsf{Next}_3(i, U, V, Z)$ such that $\mathsf{Next}(i, U, V) \equiv \exists Z : \mathsf{Next}_3(i, U, V, Z)$.

Using this formula, we define an instance of the latch game as follows. Player I's variables consist of x, U_1, U_2, Z_1, Z_2, and the 3DNF formula $\phi_1(x, U_1, U_2, Z_1, Z_2, \tilde{x}, \tilde{U}_1, \tilde{U}_2, \tilde{Z}_1, \tilde{Z}_2, y)$ is defined by

$$\phi_1 = \neg\mathsf{Next}_3(y, \tilde{U}_1, U_2, Z_1) \vee \neg\mathsf{Next}_3(x, U_2, U_1, Z_2).$$

It's always in Player I's interest to set Z_1 and Z_2 to satisfy Next_3 if possible, so this formula is equivalent to $\neg\mathsf{Next}(y, \tilde{U}_1, U_2) \vee \neg\mathsf{Next}(x, U_2, U_1)$. Each turn of Player I simulates two steps of M, going from the universal configuration $\tilde{U}_1$ to U_2 according to Player II's choice y, and then from the existential configuration U_2 to U_1 according to Player I's own choice x. If the machine ever halts, Player I loses because they can't satisfy the Next formulae. □

Using this result, we can show EXPTIME-hardness of $\tilde{G}_3$ when one of the formulae is *constant*.

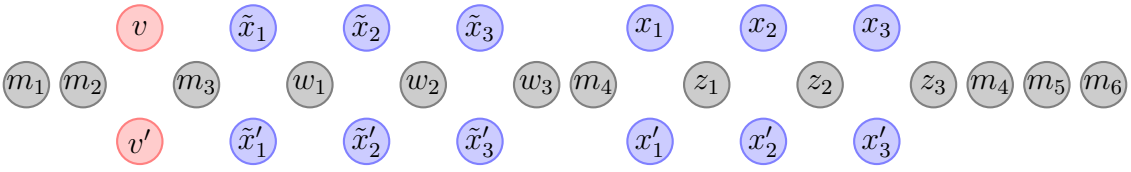

Figure 4.2: The list of expressions L used in the proof of Theorem 21. All variables are controlled by Player I. Variables in the same vertical column are XOR'd together to form an entry of L.

Theorem 21. *There exists a DNF formula ϕ_2, consisting of one clause of width 2, nine clauses of width 4, and four clauses of width 6, such that the games G_3 (Partizan-formula*

[1] An existential configuration of an alternating Turing machine ***rejects in k steps*** (where $k \in \mathbb{N}$) if and only if all successor configurations reject in at most $k-1$ steps; and a universal configuration rejects in k steps if and only if some successor configuration rejects in at most $k-1$ steps.

Flip-Exactly-One Lose) and $\tilde{G}_3$ (Partizan-formula Flip-Zero-Or-One Lose) are EXPTIME*-hard when ϕ_1 is a 5DNF formula and ϕ_2 is the constant formula specified. Draws are counted as wins for Player I.*

Proof. By Lemma 3 it suffices to consider $\tilde{G}_3$.

We reduce from the latch game (Definition 19). Let $(x_1, \ldots, x_n, \tilde{x}_1, \ldots, \tilde{x}_n, y, \phi_1)$ be an instance of the latch game. We can assume that the initial values of all variables are zero, since this can be accomplished for $x_1 \ldots x_n$ and y by inverting their occurrences in the formula, and because the initial values of $\tilde{x}_1 \ldots \tilde{x}_n$ don't matter.

We will make use of a gadget called a "four-clock". A four-clock is built from two Boolean variables, which we interpret as an element of $\mathbb{Z}/4$ according to the mapping $00 \mapsto 0, 01 \mapsto 1, 11 \mapsto 2, 10 \mapsto 3$. A single variable flip increments or decrements the four-clock by 1 (modulo 4).

The players' variables are

$$X = \{x_i, \tilde{x}_i, x'_i, \tilde{x}'_i, w_i, z_i : 1 \le i \le n\} \cup \{m_1, m_2, m_3, m_4, m_5, m_6, m_7, z, z', d\}$$
$$Y = \{y_1, y_2, c_1, c_2\},$$

where c_1, c_2, d are four-clocks. The initial values are all zero except that c_1, c_2, d are initially set to 1.

Similarly to the proof of Theorem 16, we define the list $L = [\ell_1, \ldots, \ell_{4n+8}]$ by

$$L = \begin{pmatrix} [m_1, m_2, v \oplus v', m_3] \\ \| \, [\tilde{x}_1 \oplus \tilde{x}'_1, w_1, \tilde{x}_2 \oplus \tilde{x}'_2, w_2, \ldots, \tilde{x}_n \oplus \tilde{x}'_n, w_n] \\ \| \, [m_4] \\ \| \, [x_1 \oplus x'_1, z_1, x_2 \oplus x'_2, z_2, \ldots, x_n \oplus x'_n, z_n] \\ \| \, [m_5, m_6, m_7] \end{pmatrix}.$$

Refer to Figure 4.2 for a pictorial description.

The formulae ϕ_1' and ϕ_2' are defined as:

$$\phi_1' = \left(\begin{array}{l} \bigvee_{i=1}^{|L|-1} (\ell_i \neq \ell_{i+1} \wedge \ell_i = m_7) \\ \vee \left(\bigvee_{i=1}^{|L|-1} (\ell_i \neq \ell_{i+1} \wedge c_2 \not\equiv i \mod 4) \right) \vee (m_7 = m_1 \wedge c_2 \not\equiv |L| \mod 4) \\ \vee (v \oplus v' \neq m_3 \wedge y_1 \neq v) \\ \vee \bigvee_{i=1}^{n} (w_n \neq m_4 \wedge \tilde{x}_i \neq x_i) \\ \vee (m_5 \neq m_6 \wedge \phi_1[v/y]) \\ \vee (c_1 = c_2 \wedge c_1 \neq d) \end{array} \right)$$

$$\phi_2' = \left(\begin{array}{l} (c_1 - c_2 \notin \{0, 1\}) \\ \vee (c_1 \neq c_2 \wedge c_1 = d) \\ \vee (y_1 = 1 \wedge y_2 = 0) \\ \vee (y_1 = 0 \wedge y_2 = 1 \wedge c_2 = 3) \end{array} \right).$$

It can be checked that these formulae can be expressed in the desired forms. We now explain the reasoning for these formulae.

We start with the clocks. The purpose of the clocks is to let Player II force Player I to progress across L from left to right, setting variables in order. This works as follows. Player I's losing clause

$$\bigvee_{i=1}^{|L|-1} (\ell_i \neq \ell_{i+1} \wedge \ell_i = m_7)$$

means that the expressions in L must comprise a contiguous block of zeroes followed by a contiguous block of ones; or vice versa. So at any time there are only two possible expressions in L which Player I is free to change.

The losing clause

$$\left(\bigvee_{i=1}^{|L|-1} (\ell_i \neq \ell_{i+1} \wedge c_2 \not\equiv i \mod 4) \right) \vee (m_7 = m_1 \wedge c_2 \not\equiv |L| \mod 4)$$

further restricts this choice: it ensures that the boundary between the zeroes and ones in L stays in sync with the four-clock c_2, which is controlled by Player II. (This mechanism requires $|L|$ to be a multiple of 4.) So by incrementing c_2, Player II causes Player I to toggle the expressions of L in order.

Player II is not free to manipulate c_2 as they please, however. In particular they must be prevented from running the clock backwards, which would force Player I to flip variables

in the opposite order from intended. The formulae force the two players to play as follows, assuming Player II wishes to advance the clocks at all.

Initially the clocks c_1, c_2, d are all equal, and Player II moves first. Due to their losing clauses $c_1 - c_2 \notin \{0, 1\}$ and $c_1 \neq c_2 \wedge c_1 = d$, the only change they can make to the clocks is to increment c_1. Then Player I takes a turn. Next, Player II increments c_2, and Player I must toggle the next expression ℓ_i where $i \equiv c_2 \mod 4$. At this point Player I's clause $c_1 = c_2 \wedge c_1 \neq d$ will cause them to lose unless their earlier turn was spent incrementing d.

In fact, after the first increment to c_1, the following three moves are all forced since when Player I increments d, Player II is forced to increment c_2. The only opportunity to deviate is at the very beginning, when Player II may choose to toggle y_1 or y_2 instead of incrementing c_1; they may only exercise this option when $c_2 \neq 3$. Whenever Player II does toggle y_1 or y_2 in this way, Player I is forced to pass.

Now the remaining clauses in the formula can be understood as forcing the following behavior from the players:

- When it is time for Player I to toggle $v \oplus v'$, they must set $v = y_1$. Because $v \oplus v'$ is the third element of L, this occurs when $c_2 = 3$, preventing Player II from toggling y_1 or y_2 immediately after Player I does this.
- When Player I toggles w_n, they must have $\tilde{x}_i = x_i$ for all i. This ensures Player I sets the $\tilde{x}_i$ according to the rules of the latch game.
- When Player I toggles m_5, they must satisfy ϕ_1 with v substituted for y. Since v was set to y_1 earlier, the value of y was ultimately chosen by Player II. The purpose of v is just to prevent Player II from changing their choice during or after Player I's move.

Altogether, this simulates the latch game, and thus defines a reduction. □

Using this result, we can prove EXPTIME-completeness for a new Single-formula game.

Theorem 22. *The games Single-formula Flip-Zero-Or-One Lose 12DNF (where draws count as wins for Player I) and Partizan-formula Flip-Zero-Or-One Win/Lose 12DNF (where draws count as wins for Player II) are* EXPTIME*-complete. The same is true for the Flip-Exactly-One versions.*

Proof. By Lemmas 3 and 5 it suffices to show EXPTIME-completeness of Single-formula Flip-Zero-Or-One Lose 12DNF.

We reduce from the $\tilde{G}_3$ variant of Theorem 21. Let X, Y, ϕ_1, ϕ_2 be an instance of that game. Let $P = \{P_1, \ldots, P_p\}$ be a partition (to be determined later) of the clauses of ϕ_2 into p parts, and define

$$
\begin{aligned}
X' &= X \cup \{w, x_1, \ldots, x_p\} \\
Y' &= Y \\
\phi_1' &= \left((\phi_1 \vee w) \wedge \bigwedge_{i=1}^{p} \neg x_i \right) \vee \bigvee_{i=1}^{p} \left(x_i \wedge \neg w \wedge \bigwedge_{c \in P_i} \neg c \right)
\end{aligned}
$$

The idea is that as long as ϕ_2 remains false, Player I must play to avoid ϕ_1 as usual, since they cannot toggle any of the x_i or w variables. However, if any clause of ϕ_2 becomes true, Player I can toggle the x_i corresponding to the part P_i containing that clause, which deactivates their requirement to avoid ϕ_1. On the subsequent turn, Player I can then toggle w, which immediately draws the game because Player I cannot lose while w and x_i are both true. Thus, Player I draws if and only if they can win or draw in the $\tilde{G}_3$ game.

The formula ϕ_1' can be expressed in k-DNF, where

$$k = \max\left\{5 + p, \max_{P_i \in P}\left(2 + \sum_{c \in P_i} |c|\right)\right\}.$$

Recall that the formula ϕ_2 in Theorem 21 consists of one clause of width 2, nine clauses of width 4, and four clauses of width 6. These can be partitioned into four parts with widths $\{4, 6\}$, two parts with widths $\{4, 4\}$, and one part with width $\{2, 4\}$, resulting in $k = 12$. □

Chapter 5

Future directions

We conclude this thesis with a discussion of directions for future work. First, there are several games in Definition 1 whose complexity relative to EXPTIME is left open; these include Impartial-formula Win/Lose games as well as the Flip-Nonzero versions of the games discussed by Theorems 6 and 8. Is it possible to extend the results of those theorems to the case where passing is forbidden? This would be analogous to how Corollary 15 can be viewed as a passing-forbidden extension of Theorem 7, where the resulting containment is weakened from Σ_2^{P} to PSPACE.

Another direction of interest concerns games with a "ko" rule, which forbids players from undoing their opponent's previous move, as featured in the board game Go. With such a rule in place, Impartial-variable games in which both players must contend over a shared set of variables become nontrivial. Some existing work in this direction includes the work of J.M. Robson, who proved EXPTIME-completeness of some Boolean formula-type games with ko rules [6,7], and applied these results to show that Go is EXPTIME-complete under the simple ko rule [7]. Robson also showed that variants of G_1, G_2, and G_3 become EXPSPACE-complete when a "superko" rule is instituted, preventing not only the repetition of the immediately prior position, but also the repetition of any previously-played position [8]. It would be interesting to investigate the effects of ko and superko rules on many types of Boolean formula games.

References

[1] L. J. Stockmeyer and A. K. Chandra. "Provably Difficult Combinatorial Games." *SIAM Journal on Computing*, **8**(2), 1979, pp. 151–174. DOI: 10.1137/0208013.

[2] T. J. Schaefer. "On the complexity of some two-person perfect-information games." *Journal of Computer and System Sciences*, **16**(2), 1978, pp. 185–225. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(78)90045-4. URL: https://www.sciencedirect.com/science/article/pii/0022000078900454.

[3] J. M. Robson. "N by N Checkers is Exptime Complete." *SIAM Journal on Computing*, **13**(2), 1984, pp. 252–267. DOI: 10.1137/0213018. URL: https://archive.org/details/informationproce0000ifip/page/413.

[4] A. S. Fraenkel and D. Lichtenstein. "Computing a perfect strategy for $n \times n$ chess requires time exponential in n." *Journal of Combinatorial Theory, Series A*, **31**(2), 1981, pp. 199–214. ISSN: 0097-3165. DOI: 10.1016/0097-3165(81)90016-9.

[5] J. M. Robson. "The Complexity of Go." In: *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*. Ed. by R. E. A. Mason. North-Holland/IFIP, 1983, pp. 413–417.

[6] J. Robson. *Another intractable game on boolean expressions*. Tech. rep. TR-CS-81-02. Australian National University Department of Computer Science, 1981.

[7] J. Robson. *Exponential time decision problems relating to KO-like transition rules*. Tech. rep. TR-CS-82-02. Australian National University Department of Computer Science, 1982.

[8] J. Robson. "Combinatorial games with exponential space complete decision problems." In: *Mathematical Foundations of Computer Science 1984*. Ed. by M. Chytil and V. Koubek. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 498–506. ISBN: 978-3-540-38929-3.