Complexité des jeux positionnels sur les graphes

Nacim Oijid

Université Claude Bernard Lyon 1

THESE de DOCTORAT DE
L'UNIVERSITE CLAUDE BERNARD LYON 1

Ecole Doctorale N° 512
ÉCOLE DOCTORALE INFORMATIQUE, MATHÉMATIQUES DE LYON (INFOMATHS)

Discipline : Informatique

Soutenue publiquement le 08/07/2024, par :
Nacim Oijid

Complexité des jeux positionnels sur les graphes
Complexity of positional games on graphs

Devant le jury composé de :

Lampis Michael	Maître de conférence, HDR, Université Paris Dauphine	Rapporteur
Todinca Ioan	Professeur des universités, Université d'Orléans	Rapporteur
Demaine Erik	Professeur, Massachusetts Institute of Technology	Examinateur
Guerin Lassous Isabelle	Professeure des universités, Université Lyon 1	Présidente
Duchêne Eric	Professeur des universités, Université Lyon 1	Directeur de thèse
Parreau Aline	Chargée de recherche CNRS, Lyon	Co-directrice de thèse

Abstract

This Ph.D. thesis deals with the complexity of positional games, i.e. games in which two players take turns claiming unclaimed vertices of a hypergraph. In the most famous convention, Maker-Breaker, Maker wins if she manages to claim all the vertices of a hyperedge, otherwise, Breaker wins. In these games, there is always one player who has a winning strategy, and we study here the algorithmic complexity of determining which player it is, in different conventions and on different structures.

This model of games is very general, and in the most recent studies, the hypergraph is an underlying structure of the game, which is played on a graph, called the board. In this context, the players take turns claiming edges of the graph and Maker wins if she manages to create some structure in the graph, such as a copy of some other graph, or a perfect matching. The focus of this manuscript is on the algorithmic complexity of different positional games in different conventions. In this manuscript, we present some results on different aspects of the complexity studies of positional games.

First, we focus on the general complexity of the Avoider-Enforcer and Client-Waiter conventions. In both cases, we prove their PSPACE-completeness, even restricting the input to hypergraphs with small hyperedges. Then, we present complexity results on some of the most studied positional games, namely, the H-game and the perfect matching game, played in Maker-Breaker convention on the edge set of graphs. In both cases, we prove their PSPACE-hardness, and we provide some polynomial and FPT results for the H-game. Next, we switch to games on vertices, and we focus on the parameterized complexity of the Maker-Breaker domination game. We prove that it is W[2]-hard to know whether Dominator can build a dominating set in k moves. However, when considering graph parameters, we provide FPT algorithms for the modular-width, the size of a minimum feedback edge set, and the distance to cluster. Keeping the focus on games on vertices, we then switch the convention and consider the Maker-Maker convention of the domination game. We provide a polynomial-time algorithm for computing the winner of the Maker-Maker domination game on forests. Finally, we introduce scoring positional games as a way to extend positional games to a larger framework. Here, Maker aims to fill up as many hyperedges as possible, rather than just a single one. We prove that computing Maker's score is PSPACE-complete, even for rank 2 hypergraphs. We then provide a linear-time algorithm for computing the score on unions of paths and an FPT algorithm parameterized by the neighborhood diversity.

Résumé

Cette thèse traite de la complexité des jeux positionnels qui sont des jeux joués sur des hypergraphes : Deux joueurs prennent à tour de rôle les sommets libres de l'hypergraphe. Les conditions de victoire dépendent alors des hyperarêtes. Selon un ensemble de règles appelé convention, les joueurs ont différents objectifs. Ils peuvent, par exemple, chercher à remplir une hyperarête avant leur adversaire (convention Maker-Maker), ou chercher à forcer leur adversaire à en remplir une avant eux (convention Avoider-Avoider) par exemple. Dans la convention la plus célèbre, Maker-Breaker, Maker gagne s'il parvient à prendre tous les sommets d'une hyperarête ; sinon, c'est Breaker qui l'emporte.

L'une des propriétés les plus importantes de ces jeux est qu'il s'agit de jeux finis, à information parfaite et sans hasard. Par conséquent, l'ensemble des suites de coups possible est également fini et peut-être connu des deux joueurs. On définit alors l'arbre des configurations comme un ensemble de sommet étiquetés par les différentes positions du jeu, et où un sommet y est enfant d'un sommet x si l'on peut passer de x à y en effectuant un coup dans le jeu. Par conséquent, il existe une stratégie optimale pour chaque joueur, qu'il est possible de calculer en sélectionnant chaque coup, en ayant connaissance de tout le sous-arbre issu du sommet où l'on se trouve. L'issue du jeu est alors définie comme le joueur qui l'emporte si le jeu est joué de manière optimale. S'il n'y a pas de gagnant, l'issue est nulle. Toutefois, le nombre de configurations à prendre en compte est généralement exponentielle, en la taille de l'hypergraphe de départ. C'est pourquoi nous nous intéressons généralement à des calculs de stratégies optimales sans passer par l'arbre des configurations.

Les jeux positionnels ont été introduits en 1963 par Hales et Jewett [HJ63], et les principaux résultats pionniers de cette étude ont été apportés par Erdős et Selfridge en 1973 [ES73]. En 1978, Schaefer [Sch78] prouve que déterminer quel joueur a une stratégie gagnante dans un jeu Maker-Breaker est PSPACE-complet, mais ce résultat a été le seul résultat de complexité connu pendant longtemps. L'étude de la complexité des jeux positionnels est cependant à nouveau d'actualité, puisque récemment, de nombreux résultats sont apparus tant en complexité paramétrée par Bonnet *et al.* [BGL$^+$17] qu'en complexité générale, par Burke et Hearn [BH19] pour la convention Avoider-Avoider ou encore Miltzow et Stojaković [MS22] pour la convention Avoider-Enforcer.

L'étude des jeux positionnels est motivée par ses nombreux liens avec d'autres domaines de l'informatique. Pour ne citer que les principaux, des liens immédiats avec la théorie des graphes et des hypergraphes, la logique et la théorie de la complexité justifient cette étude et certains résultats dans ces domaines proviennent de résultats en jeux positionnels. Par exemple, Erdős et Selfridge [ES73] donnent une borne inférieure sur le nombre chromatique des hypergraphes en fonction de l'issue d'un jeu positionnel. Plus tard, Bonnet *et al.* [BGL$^+$17] prouvent que la vérification de modèles d'une certaine famille de formules logiques, représentant des parties de jeux positionnels, est dans W[1] .

Ce modèle de jeu est très général et, dans les études les plus récentes, l'hypergraphe est une structure sous-jacente du jeu, qui se joue en réalité sur un graphe. Dans les études les plus courantes, les joueurs prennent à tour de rôle des arêtes du graphe et Maker gagne s'il parvient à créer une certaine structure dans le graphe, par exemple, une copie d'un autre graphe cible ou un couplage parfait. Ici, l'hypergraphe sous-jacent a donc une hyperarête par copie du graphe cible ou par couplage parfait du graphe. Le nombre de sous-graphes isomorphes au graphe cible ou de couplages parfaits d'un graphe pouvant être arbitrairement grand, on comprend donc qu'il est plus simple de jouer sur un graphe, et non sur l'hypergraphe sous-jacent.

Cette thèse s'intéresse à la complexité des jeux positionnels sous trois différents points de vue. D'abord, nous nous intéressons à leur complexité dans le cadre général de l'étude de différentes conventions jouées sur des hypergraphes. En effet, bien que de nombreuses conventions aient été introduites, seuls trois d'entre elles étaient connues comme étant PSPACE-complètes pour le calcul de l'issue. Puis, nous étudions des jeux positionnels plus particuliers, joués sur des graphes, où les ensembles gagnants coïncident avec des structures particulières des graphes considérés. Nous étudierons ces jeux à la fois dans un cadre général et en observant

leur complexité paramétrée. Enfin, nous présenterons une piste pour élargir le cadre des jeux positionnels en ajoutant un score qu'un joueur cherchera à maximiser et l'autre à minimiser, proposant ainsi un cadre d'étude pour les jeux positionnels à score.

Un résultat important de cette thèse, formulé ici pour la première fois de manière générique, est le Super Lemma. Ce lemme, applicable aussi bien dans les conventions Maker-Breaker et Avoider-Enforcer, permet de traiter les sommets ayant le même voisinage dans les hypergraphes considérés. Ce lemme affirme que si deux sommets ont même voisinage, il existe une stratégie optimale où chaque joueur prendra l'un de ces deux sommets. Bien que son énoncé se formule simplement, ce lemme a de nombreuses applications, notamment en complexité paramétrée, car c'est l'outil principal qui sera utilisé dans plusieurs chapitres de cette thèse afin d'obtenir des algorithmes FPT.

Dans la littérature, il existe six principales conventions dans lesquelles les jeux positionnels sont étudiés. Parmi elles, trois étaient déjà connues pour leur difficulté algorithmique : il est PSPACE-complet de déterminer si un joueur a une stratégie gagnante dans un jeu Maker-Breaker, Maker-Maker ou Avoider-Avoider. La complexité algorithmique des trois autres grandes conventions des jeux positionnels était encore inconnue. Nous prouvons pour commencer que parmi ces dernières, dans les conventions Avoider-Enforcer et Client-Waiter, il est également PSPACE-complet de déterminer quel joueur a une stratégie gagnante, enrichissant ainsi cette étude et en ne laissant que la complexité de la convention Waiter-Client ouverte.

Nous nous concentrerons ensuite sur les jeux positionnels joués sur des graphes. Les jeux les plus étudiés dans ce domaine se jouent en convention Maker-Breaker sur les arêtes des graphes considérés. Les jeux les plus connus étant le jeu de connectivité, le H-game, le jeu du couplage parfait et le jeu du cycle hamiltonien. La plupart des études de ces jeux ont été réalisées sur des graphes complets et visaient à obtenir des résultats asymptotiques en considérant des biais. Le seul résultat général de complexité algorithmique provient de Lehman [Leh64], prouvant que le jeu de connectivité peut être résolu en temps linéaire. Nous réalisons donc ici la première étude algorithmique du H-game et du jeu du couplage parfait en prouvant qu'il est PSPACE-complet de déterminer quel joueur a une stratégie gagnante dans ces deux jeux. Nous nous intéressons alors à des classes de graphes plus restreintes nous permettant d'obtenir des résultats polynomiaux ou FPT selon les classes considérées.

Le reste de cette thèse se concentre ensuite sur les jeux joués sur les sommets des graphes. Parmi eux, le plus connu est le jeu de domination, introduit en Maker-Breaker par Duchêne *et al.* [DGPR20]. Dans leur article, ils prouvent que le jeu est PSPACE-complet en général, et proposent des algorithmes polynomiaux lorsque le jeu est joué sur un cographe ou une forêt. Nous cherchons ici à étendre ces résultats en nous intéressant à la complexité paramétrée de ce jeu. D'une part, nous prouvons que ce jeu est $W[2]$-dur si nous limitons le nombre de coups autorisés pour que Dominator crée un ensemble dominant. Puis nous nous intéressons à des paramètres de graphes mesurant la distance entre un graphe et les classes de graphe où des algorithmes polynomiaux sont connus. Ainsi, nous proposons des algorithmes FPT pour la distance aux clusters, la taille d'un plus petit feedback edge set, et la largeur modulaire mesurant la distance aux unions de cliques, arbres et cographes respectivement.

Bien qu'ayant été introduit en convention Maker-Breaker, la façon la plus naturelle de jouer aux jeux positionnels est de considérer la version Maker-Maker. C'est pourquoi, nous continuons l'étude du jeu de domination en considérant cette fois-ci la convention Maker-Maker. Rapidement, des différences entre les deux conventions apparaissent et bien que Maker l'emporte facilement sur n'importe quel cycle en Maker-Breaker, nous prouvons qu'il existe une infinité de cycles en Maker-Maker sur lesquelles la partie se termine en une partie nulle. Nous nous intéressons alors aux forêts, et prouvons que, comme c'est le cas pour la version Maker-Breaker, il est possible de déterminer l'issue du jeu en temps polynomial, lorsque le jeu est joué sur une forêt. Cependant, en Maker-Maker, nous perdons l'équivalence entre l'existence d'une stratégie gagnante et l'existence d'un couplage parfait dans la forêt. L'algorithme proposé est donc plus subtil et nécessite de regarder précisément les différentes composantes connexes de la forêt ainsi que les coups des deux joueurs.

Enfin, nous introduisons les jeux positionnels à score comme un moyen d'étendre l'étude des jeux positionnels à un cadre plus large. Nous cherchons à obtenir des résultats plus généraux sur ces derniers et à obtenir à cadre ou différents résultats, connus à la fois des jeux à score et des jeux positionnels, peuvent s'appliquer. Ici, en Maker-Breaker, Maker vise à remplir autant d'hyperarêtes que possible, plutôt qu'une seule, et Breaker cherche à limiter ce nombre ; tandis qu'en Maker-Maker, chaque joueur tente de remplir plus d'hyperarêtes que son adversaire. Nous nous concentrons alors sur Incidence, le jeu positionnel à score joué sur un graphe, où prendre les deux extrémités d'une arête rapporte un point. Nous prouvons que le calcul du score optimal de Maker dans le jeu Maker-Breaker Incidence est PSPACE-complet. Mais que le jeu devient cependant polynomial, lorsque nous considérons la convention Maker-Maker, en donnant la formule exacte du calcul du score optimal, en fonction des degrés des différents sommets du graphe. La possibilité de calculer le score optimal en temps polynomial en Maker-Maker nous invite donc à chercher des résultats positifs en Maker-Breaker également. Ainsi, nous fournissons un algorithme FPT paramétré par la neighborhood diversity du graphe pour calculer ce score. Enfin, nous donnons la valeur exacte du score optimal sur les chemins, et fournissons un algorithme linéaire pour le calculer sur les cycles et les unions de chemins.

Contents

Introduction

Grab a coffee, take a seat. It's gonna be fun.

Definition of the subject

Have you ever wondered why some games are fun, while others are not? Let us try to answer this question. A few years ago, when I started studying computer science, my friend Truc-Muche, who enjoys coding algorithms to solve games, told me that the longer his algorithm take to solve a game, the more fun the game is. I think this definition is a good starting point. A game is fun if Truc-Muche is not able to code a fast algorithm that plays the game perfectly. However, since some of you won't know who Truc-Muche is, nor his coding skills, I decided to switch to a formal definition and compute the complexity of finding a strategy in these games. Classifying games according to their computational complexity is quite classical, as it was done for several Nintendo games by Aloupis *et al.* [ADGV15]. Sorry for the disappointment, but Pokemon will not be included in this manuscript.

Funding and supervisors

Now that I have a subject, I have to find a way to write a thesis about it. During my third year at the ENS de Lyon, after a previous internship in (economic) game theory, I wanted to discover combinatorial game theory. When I googled some names, recent papers on the subject often referred to Eric Duchêne and Aline Parreau. That's how I met them and started my Master 2 internship with them. During this internship, thanks to my skills in ~~cake baking,~~ graph theory and complexity, we discussed the possibility of continuing our work together. So I started a Ph.D. with them. At the same time, Aline got her funding for the ANR P-GASE (on positional games), which I joined right from the beginning. With this newly built team, we started to study the complexity of positional games.

Positional games ?

Oh yeah, sorry, I forgot to tell you what I decided to study. Do you know Tic-Tac-Toe ? The two-player game where we place crosses and noughts in a $3 * 3$ grid and we try to align three of them ? This is the most famous positional game. So famous that Beck named his book *Combinatorial games: Tic-Tac-Toe theory*. In practice, a positional game is played by two players on a set of points called vertices, on which we define some subsets of points called hyperedges (or sometimes winning sets). If you put the vertices and the hyperedges together, you get a hypergraph. The players take turns claiming the vertices, and the first player which manages to claim all the vertices of a hyperedge wins. In Tic-Tac-Toe, there are nine vertices and eight hyperedges corresponding to the eight lines on the grid, see Figure 1.

However, the title of my thesis refers to "positional games on graphs" and not "on hypergraphs". This is because hypergraphs are very general, and a good way to represent any positional game, but specific games are often played on graphs, which are sets of points called vertices, connected by lines called edges. Take your favorite graph problem. For example, finding a small set of vertices, called a dominating set, that is connected to all the other vertices. Finding such a set in a graph is a single-player problem. Now, we add an opponent who wants to prevent you from finding a dominating set in your graph by forbidding some vertices from being in it. This becomes a two-player game, called the Maker-Breaker domination game: one player,

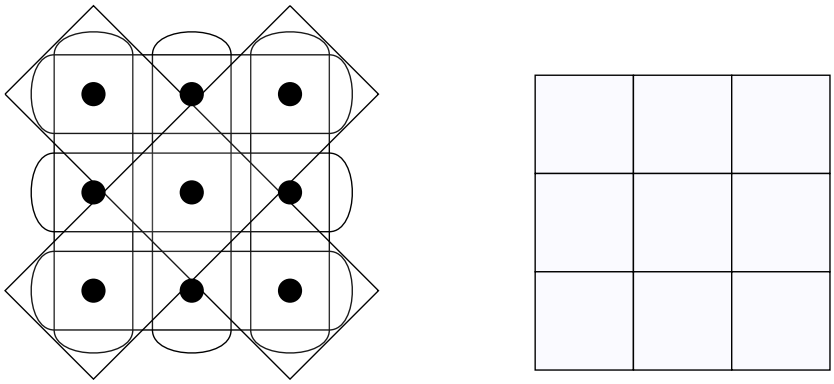


Figure 1: On the left, the hypergraph corresponding to Tic-Tac-Toe, on the right, a Tic-Tac-Toe grid.

Dominator adds vertices to a set. The second player, Staller, forbids some vertices from being in the set. When all the vertices are either in the set or forbidden to be in the set, Dominator wins if and only if the set she has created is a dominating set of the graph.

Difficulty of a game

"Okay, but what do you mean by "a game is difficult"? Because Tic-Tac-Toe is not what I call a difficult game."

We are dealing here with general complexity in computer science. In Tic-Tac-Toe, since there are 9 squares in the grid, there are roughly 9! possible games: any game corresponds to an order in which the squares are claimed. This phenomenon can be generalized to larger grids. For any finite grid, there is a finite (but very large) number of games in the grid. The fact that there is a finite number of games shows that optimal moves exists: by looking at all the possible games after each move, you can choose the one which makes you win (if any). This property is generally true in positional games: as there is a finite number of vertices, there is a finite number of moves. Thus, at any moment of the game, a player can choose an optimal move. This property shows that either one player has a winning strategy, or both players can ensure a draw. However, factorials grow very fast. Usually we cannot look at all the possible games. Therefore, we try to determine which player has a winning strategy faster than by looking at all the possibilities.

Decision problems, such as determining which player has a winning strategy in a certain game, can be divided into categories based on how much time and memory an algorithm needs to solve them. Without going into details, if an algorithm runs in polynomial time, it is fast. Otherwise, it is not. Here most problems are in PSPACE, a very large class of problems, that use polynomial space, but can take a long time to solve. For now, it is sufficient to know that if a problem is PSPACE-complete, Truc-Muche will like it.

Interest of the study

"Did you just spend three years just playing Tic-Tac-Toe and seeing if Truc-Muche would find some games funny or not?"

No. Positional games have several applications because they are related to other important areas of computer science. The most natural connection is with hypergraph theory, since positional games are played on hypergraphs. For instance, hypergraph coloring is a quite famous problem in hypergraph theory, and some results about non 2-colorable hypergraphs were obtained from positional game theory. The three other major links between positional games and other fields are with Ramsey theory, complexity theory and logic theory. Without going further into details as several notions are required to explain them properly, strategies in games can be explained using tools from logic. The study of positional games made it possible to reach some results about different classes of model checking in logic and their complexity. Even if some of these

links will be developed in Chapter 1, we refer the reader to the articles from Bonnet *et al.* [BJS16, BGL+17] for a more complete answer.

Study of different conventions

"Ok, so basically, your job is to take a game, prove that it is difficult, take another one, prove that it is difficult and so on ?"

Not exactly. The study of positional games started in 1963, when they were introduced by Hales and Jewett [HJ63], but it has become more popular recently, with the book of Hefetz *et al.* [HKSS14]. However, most of the studies were made in a very specific set of rules that we aim to extend in order to have a more general framework. We aim to find more general results about the structure of the hypergraph, and about other sets of rules, called conventions. Tic-Tac-Toe is what we call a game in the Maker-Maker convention, i.e. a game in which both players aim to fill up a hyperedge. But what happens if we change the rules ? Say for instance that now the first player to align three marks loses instead of winning. We define different conventions depending on whether the players want to fill up the hyperedges or not and on how the vertices are claimed. Since one can prove that, in Maker-Maker games, the second player cannot win, his goal is now to prevent the first player to win. This new ruleset is called Maker-Breaker and is the most studied convention since the introduction of positional games. This thesis goes through the different conventions and proves several complexity results on them.

Regarding the complexity aspect of the question, note that the complexity study of positional games is quite trendy. Except for a result by Schaefer [Sch78] in 1978, most of the complexity results are recent. Therefore, some aspects have not been studied much yet. An important part of our work has been the treatment of positional games under the parameterized complexity paradigm. Since it is generally difficult to compute the winner of the game, we aim to better understand the difficulty of the problem by considering some of its parameters and what happens when we can control them.

Organization of this thesis

This manuscript is divided into six chapters. Most of the results that will be presented here have already been published. This manuscript presents the articulation of all these results together and their contribution to the framework of positional games. For example, the Super Lemma is presented for the first time in its most general version, so that it can be applied in several chapters.

- In Chapter 1, we present the general framework of positional games, together with the six different conventions that will be studied here. Since several aspects of graph theory and complexity theory will also be required in this work, we introduce some of their definitions in order to make this manuscript self-contained. We then focus on positional games, and present the history of their introduction. Next, we focus on the complexity, both in the general case and when applied to positional games. Finally, we present some general results that have several applications in this thesis.

- In Chapter 2, we focus on the general complexity of positional games. When studying positional games, an important question is the computational complexity of computing the winner of a game. Among the six positional game conventions studied in this dissertation, three of them, Maker-Breaker, Maker-Maker and Avoider-Avoider, were already known to be PSPACE-complete. We prove that two others, Avoider-Enforcer and Client-Waiter are also PSPACE-complete, and we provide direct examples of these results, by proving that the domination game is also PSPACE-complete under these conventions.

- Chapter 3 focuses on classical Maker-Breaker games. The most studied ones are played on the edges of graphs, where Maker tries to build some structure in the graph with the edges she claims. Most of the previous studies of these games have not focused on the computational complexity. We present here the first hardness results when computing their winners. Depending on the structure, different complexity results are obtained. In this chapter, we prove that for most of the classical structures, the game is PSPACE-complete, and we provide algorithms for simpler structures.

- The most classical way to play positional games on graphs is to claim the edges, the study of games claiming the vertices is much more recent. Among the games studied in this way, one of the most studied is the domination game. It was already known that this game is PSPACE-complete. When this is the case, the next step in the study is to analyze the game more carefully to determine what makes it difficult. This branch of study is called parameterized complexity. In Chapter 4 we focus on the Maker-Breaker domination game from a parameterized complexity point of view, i.e., we look at its complexity as a function of other parameters. Thus, we focus on the game where the players need to win in a bounded number of moves, or on graphs with some bounded parameters, such as their modular-width, their feedback edge set number or their distance to cluster.

- In Chapter 5, we keep working on the domination game, but we switch to the Maker-Maker convention. Among the results in the Maker-Breaker convention, there is the fact that the winner on trees can be computed in linear time. In this chapter, we prove that this is also the case in the Maker-Maker convention, but the algorithm is much more difficult. This chapter will consist in providing the algorithm and proving its correctness.

- Chapter 6 introduces scoring positional games as a way to go further in the study of positional games. In fact, up to now, all the studies have focused on determining whether a hyperedge is filled up by a player or not. Here, we merge the study of positional games with the scoring game theory to study a quantitative version of Maker-Breaker games. In scoring positional games, Maker's goal is to fill up as many hyperedges as possible. We focus on this game played on graphs, where claiming any two adjacent vertices is worth one point.

Collaborations and productions

During my Ph.D., I have had the opportunity to work with many people and to travel to many places. (Did my advisors want to get rid of me ?). I have had the opportunity to travel to the Czech Republic, Germany, Portugal, Serbia, and Sweden, either for collaborations or conferences, and to several French cities to start new collaborations. I also had a total of 23 co-authors, coming from six different countries: Austria, France, Germany, Serbia, Sweden, and United States. Most of these collaborations were supported by the ANR P-GASE.

Chapter 1

Preliminaries and state of the art

Prove that you have already read some papers about games

This chapter introduces the different notions and objects that will be studied in this manuscript. It also contains the historical aspects of positional games and some well-known results about them. In order to make this thesis self-contained, some of these results will be proved here, since their exact source is often hard to find. In Section 1.1, we will introduce the different definitions of positional game theory. Section 1.2 will present its historical study. In Section 1.3, we will present the complexity context of positional game theory. Finally, in Section 1.4, we will introduce several statements that will be used in the next chapters.

1.1 Definitions

1.1.1 General definition of positional games

Positional games are a branch of combinatorial games. Consequently, they are perfect information and deterministic, i.e. the players know all the information about the state of the game, and there is no random event. First introduced by Hales and Jewett [HJ63] in 1963 as multiplayer games, positional games were popularized as two-player games by Erdős and Selfridge [ES73] in 1973. In their most general definition, positional games are played on the vertex set of a hypergraph, and the winning conditions depend on how the hyperedges are filled up.

The board

Definition 1.1 (Hypergraph)**.** *A* hypergraph $\mathcal{H}$ *is a pair* $(\mathcal{X}, \mathcal{F})$ *where* $\mathcal{X}$ *is a finite set of vertices, and* $\mathcal{F} \subset 2^{\mathcal{X}}$ *is a set of hyperedges.* $\mathcal{H}$ *is generally called the* board *of the game, and* $\mathcal{F}$ *the winning sets.*

The hypergraph is said to have rank k *(or to be a* k*-hypergraph) if any* $f \in \mathcal{F}$ *has order at most* k*.*

It is said to be k-uniform *if any* $f \in \mathcal{F}$ *has order exactly* k*.*

Since the winning conditions consist of filling hyperedges, the notion of transversal is quite important. Indeed, claiming a transversal consists of preventing the opponent from filling a hyperedge.

Definition 1.2 (Transversal)**.** *Let* $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ *be a hypergraph. A* transversal $f \subset \mathcal{X}$ *of* $\mathcal{H}$ *is a set of vertices that intersect all the hyperedges of* $\mathcal{H}$*.*

The transversal hypergraph *(or* dual hypergraph *of* $\mathcal{H}$*, denoted by* $\mathcal{H}^* = (\mathcal{X}^*, \mathcal{F}^*)$ *is defined by* $\mathcal{X}^* = \mathcal{X}$ *and* $\mathcal{F}^* = \{f \subset \mathcal{X} \mid f$ *transversal of* $\mathcal{H}\}$*.*

Note that the transversal of the transversal of a hypergraph $\mathcal{H}$ is $\mathcal{H}$ whenever we remove inclusion between hyperedges.

Sometimes, only a few elements of the hypergraph need to be considered in the game. Therefore, we also present the notion of sub-hypergraph.

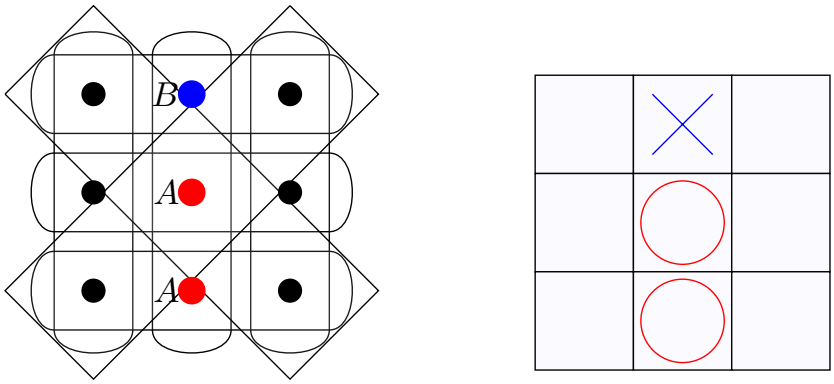


Figure 1.1: On the left, the hypergraph corresponding to Tic-Tac-Toe, on the right, a Tic-Tac-Toe grid.

Definition 1.3 (Sub-hypergraph)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* sub-hypergraph *of $\mathcal{H}$ is a hypergraph $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$ with $\mathcal{X}' \subset \mathcal{X}$ and $\mathcal{F}' \subset \mathcal{F}$.*

Note that as $\mathcal{H}'$ is a hypergraph, we have necessarily, $\mathcal{F}' \subset 2^{\mathcal{X}'}$.

The rules

The two players alternate claiming vertices of $\mathcal{X}$. The way the vertices are claimed, and the winner depend on the *convention* considered with the hypergraph. There are several conventions for positional games, but we will only focus on the six most studied.

Definition 1.4 (Maker-Maker game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Maker-Maker game *is played by two players, Alice and Bob. Alice and Bob take turns claiming an unclaimed vertex of $\mathcal{H}$, with Alice starting. The first player to claim all the vertices of a hyperedge $f \in \mathcal{F}$ wins. If all the vertices are claimed without any hyperedge being filled by a single player, the game ends in a draw.*

This convention is for example the one of the famous Tic-Tac-Toe, which can be represented by a hypergraph as depicted in Figure 1.1

Definition 1.5 (Maker-Breaker game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Maker-Breaker game *is played by two players, Maker and Breaker. Maker and Breaker take turns claiming each unclaimed vertex of $\mathcal{H}$. Maker wins if she claims all the vertices of a hyperedge $f \in \mathcal{F}$. Otherwise, Breaker wins.*

Definition 1.6 (Avoider-Avoider game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. An* Avoider-Avoider game *is played by two players, namely Alice and Bob. Alice and Bob take turns claiming an unclaimed vertex of $\mathcal{H}$, with Alice starting. The first player to claim all the vertices of a hyperedge $f \in \mathcal{F}$ loses. If all vertices are claimed without any hyperedge being filled by a single player, the game ends in a draw.*

Definition 1.7 (Avoider-Enforcer game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. An* Avoider-Enforcer game *is played by two players, Avoider and Enforcer. Avoider and Enforcer take turns claiming an unclaimed vertex of $\mathcal{H}$. Enforcer wins if Avoider claims all the vertices of a hyperedge $f \in \mathcal{F}$. Otherwise, Avoider wins.*

Definition 1.8 (Waiter-Client game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Waiter-Client game *is played by two players, Waiter and Client. Each turn Waiter chooses two unclaimed vertices of $\mathcal{H}$ and offers them to Client. Client selects and claims one of them, and Waiter claims the other. If only one vertex is left, it goes to Client. At the end of the game, if Waiter has claimed all the vertices of a hyperedge $f \in \mathcal{F}$, she wins. Otherwise, Client wins.*

Definition 1.9 (Client-Waiter game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Client-Waiter game *is played by two players, Waiter and Client. Each turn Waiter chooses two unclaimed vertices of $\mathcal{H}$ and offers them to Client. Client selects and claims one of them, and Waiter claims the other. If only one vertex is left, it goes to Client. At the end of the game, if Client has claimed all the vertices of a hyperedge $f \in \mathcal{F}$, he wins. Otherwise, Waiter wins.*

Note that it sometimes happens that the last unclaimed vertex goes to Waiter. However, we prefer to give it to Client here because it better fits the definition of positional games. Indeed, since Client claims the first vertex in a pair and Waiter the second, when there is only one vertex left, the last player to claim a vertex was Waiter, and it seems more natural to give the last vertex to Client.

The first four conventions presented here are related to each other and are often separated as follows:

- Maker-Maker games and Maker-Breaker games, are called *achievement games*. They are games in which the players try to claim vertices forming some structure.
- Avoider-Avoider and Avoider-Enforcer games, are called *avoidance games*. They are games in which the players try to avoid that the vertices they have claimed form some structure.
- Maker-Maker games and Avoider-Avoider games, are called *strong games*. They are games in which both players have the same goal.
- Maker-Breaker and Avoider-Enforcer games, are called *weak games*. They are games in which one player has a goal and the second wants to prevent the former to succeed in it. Thus, in these games, there is no draw.

Note that in weak games, it is possible to always claim all the vertices of the board, as if Maker or Avoider has filled up a hyperedge, she will have it filled up at the end, and the outcome will not change. In strong games, the game has to be stopped as soon as a hyperedge is filled up, otherwise, we cannot know from a position which player has won if both players have filled up a hyperedge.

Despite that Maker-Breaker and Avoider-Enforcer games look very asymmetric as the two players of each has different roles, by considering the transversal hypergraph, we obtain that the roles of either Maker and Breaker or Avoider and Enforcer has been exchanged. Indeed, claiming a transversal consists exactly in claiming one vertex in each hyperedge, and thus prevent the other player to fill up a hyperedge. Therefore, several results obtained for a player have a pendant for the second one.

Strategies

To keep track of the moves made by each player, we define a position of the game as a state of the game that can be reached by playing on the hypergraph.

Definition 1.10 (position)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* position *$\mathcal{P}$ in the game played on $\mathcal{H}$ is a triple $(\mathcal{H}, \mathcal{X}_1, \mathcal{X}_2)$ such that $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$ and $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. $\mathcal{X}_1$ and $\mathcal{X}_2$ corresponds to the set of vertices already claimed by the players.*

Intuitively, a strategy, is a way of responding to every possible move of the opponent, which can be translated by a move to play in every position of the game. It is a winning strategy if the player using this strategy always reaches a position of the game in which he has won.

Definition 1.11 (Strategy)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Strategie *on $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ is a function $\mathcal{S} : 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to \mathcal{X}$ such for any subset $\mathcal{X}_1, \mathcal{X}_2$ such that $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \neq \mathcal{X}$, we have $\mathcal{S}(\mathcal{X}_1, \mathcal{X}_2) \notin \mathcal{X}_1 \cup \mathcal{X}_2$.*

A strategy $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) is said to be a winning strategy *for the first (resp. second) player if, for any strategy $\mathcal{S}_2$ (resp. $\mathcal{S}_1$) and any sequence of claimed vertices $(\mathcal{X}_1^1, \mathcal{X}_2^1), \ldots, (\mathcal{X}_1^k, \mathcal{X}_2^k)$ such that $(\mathcal{X}_1^1, \mathcal{X}_2^1) = (\emptyset, \emptyset)$, and for any $1 \leq i < k$, we have $\mathcal{X}_1^{i+1} = \mathcal{X}_1^i \cup \mathcal{S}_1(\mathcal{X}_1^i, \mathcal{X}_2^i)$ and $\mathcal{X}_2^{i+1} = \mathcal{X}_2^i \cup \mathcal{S}_2(\mathcal{X}_1^{i+1}, \mathcal{X}_2^i)$, the first (resp. second) player reaches a position satisfying the winning conditions of the convention before his opponent.*

Note that several positions of the game are inaccessible, or a strategy can prevent them to appear, and therefore, a strategy does not necessarily need to be defined for all $(2^{\mathcal{X}} \times 2^{\mathcal{X}})$ to be applied.

Winners and outcomes

Since in each convention, the game ends in at most $|\mathcal{X}|$ moves, the set of all possible positions is finite. Thus, a strategy can be computed by looking at every position. Therefore, in every convention, it is possible to prove that either one player has a winning strategy or both players can ensure a draw. Since now, we will suppose that the players will play *optimally*, i.e. if a player has a winning strategy, he or she will play it.

Definition 1.12 (outcome)**.** *The* outcome *of a position P is defined as the player having a winning strategy, if any, or Draw otherwise. It is denoted by $o(P)$.*

The outcome *of a game on a hypergraph $\mathcal{H}$ is defined as the outcome of the position $(\mathcal{H}, \emptyset, \emptyset)$, and we can write $o(\mathcal{H})$ directly.*

- In Maker-Maker or Avoider-Avoider games:
  - $\mathcal{A}$ if Alice has a winning strategy.
  - $\mathcal{B}$ if Bob has a winning strategy.
  - $\mathcal{D}$ if both players have a draw strategy.
- In Maker-Breaker games:
  - $\mathcal{M}$ if Maker has a winning strategy.
  - $\mathcal{B}$ if Breaker has a winning strategy.
  - $\mathcal{N}$ if the next player to move has a winning strategy.
  - $\mathcal{P}$ if the second player to move has a winning strategy. (Note that $\mathcal{P}$ stands for "previous" as the player going second is the previous player if some moves have already been made.)
- In Avoider-Enforcer games:
  - $\mathcal{A}$ if Avoider has a winning strategy.
  - $\mathcal{E}$ if Enforcer has a winning strategy.
  - $\mathcal{N}$ if the next player to move has a winning strategy.
  - $\mathcal{P}$ if the player going second has a winning strategy.
- In Waiter-Client or Client-Waiter games:
  - $\mathcal{C}$ if Client has a winning strategy.
  - $\mathcal{W}$ if Waiter has a winning strategy.

The two main goals in game theory are, given a game, to compute a winning strategy and to compute the outcome. Note that sometimes, the outcome can be computed with non-constructive arguments, and thus, it is possible to know which player has a winning strategy, but we cannot describe that strategy. Complexity studies focus mainly on the computation of the outcome, since the answer to a decision problem must be either $\top$ or $\bot$. Complexity will be discussed in Section 1.3, for now, we will define the different possible outcomes for the different conventions.

Now that the outcomes are defined, the decision problem related to them is the following one:

Problem 1.13.
Input: A hypergraph $\mathcal{H}$ together with a convention and an outcome $\mathcal{O}$.
Output: $\top$ if $\mathcal{H}$ has the outcome $\mathcal{O}$ under the considered convention.

Note that as the game is a two player game, it is often the case that the decision problem takes as input a hypergraph and a player starting, and asks which player has a winning strategy. If it happens, we often state this decision problem a "computing the winner of a positional game".

Optimal strategies and outcomes

Even though these outcomes are all the theoretical outcomes that can be obtained from a game position, some of them are not real outcomes of the game, since if the game is played optimally, they cannot appear.

Lemma 1.14 (Hales and Jewett [HJ63])**.** *Bob cannot have a winning strategy in a Maker-Maker game, i.e. the outcome $\mathcal{B}$ does not exist in Maker-Maker games.*

Proof. By contradiction, suppose that Bob has a winning strategy $\mathcal{S}$ on $\mathcal{H} = (\mathcal{X}, \mathcal{F})$. The proof is a *strategy stealing* argument for Alice:

- First Alice claims an arbitrary vertex $x_0 \in \mathcal{X}$. We denote by P_{x_0} the current position of the game and by P the position P where x_0 is removed from the vertices claimed by Alice.
- Let $\mathcal{X}_A, \mathcal{X}_B$ are already claimed by Alice and Bob respectively in P. Whenever Bob claims a vertex $x \in \mathcal{X}$ in the position P_{x_0}, Alice considers the vertex $y = \mathcal{S}(\mathcal{X}_A \cup \{x\} \setminus \{x_0\}, \mathcal{X}_B)$ that Bob would have claimed according to $\mathcal{S}$ (in P) if she had claimed x. If $y \neq x_0$, she claims y in P_{x_0}. Otherwise, she claims another available vertex x_0' and now considers the position $P_{x_0'}$.

According to this strategy, at any point in the game, the vertices played by Bob in P_{x_0} are a subset of the vertices played by Alice against $\mathcal{S}$ in P, and the vertices played by Alice in P_{x_0} contain at least the vertices played by Bob after the same number of moves following $\mathcal{S}$ in P. Therefore, Alice has a winning strategy, which contradicts that Bob has one. □

Note that this lemma does not provide a strategy for Alice that ensures a draw or a win. A direct consequence of this lemma is that if there is no draw on $\mathcal{H}$, then Alice wins the Maker-Maker game on $\mathcal{H}$. In contrast with Maker-Maker games, in Avoider-Avoider games, all three outcomes are possible. Indeed, consider the hypergraphs $\mathcal{H}_1 = (\{x\}, \emptyset), \mathcal{H}_2 = (\{x\}, \{\{x\}\})$ and $\mathcal{H}_3 = (\{x, y\}, \{\{x\}\})$, we have that $\mathcal{H}_1$ ends in a draw, Bob wins in $\mathcal{H}_2$ and Alice wins in $\mathcal{H}_3$.

Next lemma deals with zugzwang in games. A *zugzwang* is a position of a game where no player wants no move. In achievement games, it shows that there is no zugzwang, and therefore, it never benefits a player to skip his turn.

Lemma 1.15 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. For any position $P = (\mathcal{H}, \mathcal{X}_1, \mathcal{X}_2)$, if a player has a winning (resp. drawing) strategy going second on P in the Maker-Breaker or Maker-Maker convention, he has one going first on P, i.e., the outcome $\mathcal{P}$ cannot exist in the Maker-Breaker convention.*

The proof of Lemma 1.15 being very similar as the proof of Lemma 1.14, it will not be provided here. A similar result exists in Avoider-Enforcer games. However, as in general, in the Avoider-Enforcer convention, the players will try to avoid the vertices in several hyperedges, the first moves are not very significant on the outcome. Therefore, the last moves are more important. More specifically, a player who wins by playing the last move wins by playing the second to last move.

Lemma 1.16 (Galliot [Gal23, GGGP])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. For any position $P = (\mathcal{H}, \mathcal{X}_1, \mathcal{X}_2)$ in the Avoider-Enforcer convention, if a player has a winning strategy when he claims the last vertex of the graph (i.e. going first if the number of vertices is odd, or second when the number of vertices is even), he has one when he claims the second-to-last vertex.*

Proof. Even if in theory, four cases have to be proved, by similarities among them, we will only provide the proof that if Avoider wins going second when the number of vertices is even, he also wins going first. Let $\mathcal{S}$ be a strategy for Avoider going second. Consider the following strategy:

- First, Avoider claims any vertex $x_0 \in \mathcal{X}$
- Now, when Enforcer claims a vertex y, Avoider claims the vertex x he would have claimed as an answer to y according to $\mathcal{S}$. If $x = x_0$, then he claims an arbitrary vertex x_0' and considers it as the new x_0.

At the end of the game, Avoider will have claimed vertices according to $\mathcal{S}$ as if Enforcer had never claimed x_0 and had let it as the last vertex to be claimed by Avoider. Therefore, as $\mathcal{S}$ was a winning strategy, Avoider does not claim all the vertices of a hyperedge and therefore has won. □

This lemma leads to the introduction of the outcome $\mathcal{SL}$ if the second to last player has a winning strategy in the Avoider-Enforcer convention. Therefore, the only three outcomes in Avoider-Enforcer games are $\mathcal{A}, \mathcal{E}$ and $\mathcal{SL}$.

1.1.2 Link between the conventions

The various conventions presented here are not completely independent. The following statements give some implications about the outcomes of the different conventions.

Lemma 1.17 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Suppose that Breaker wins the Maker-Breaker game on $\mathcal{H}$ while going second, i.e. the outcome of $\mathcal{H}$ is $\mathcal{B}$. Then the outcome of the Maker-Maker game played on $\mathcal{H}$ is a draw, i.e. its outcome is $\mathcal{D}$.*

Proof. Suppose that Breaker has a winning strategy $\mathcal{S}$ going second in the Maker-Breaker game played on $\mathcal{H} = (\mathcal{X}, \mathcal{F})$. Recall that, by Lemma 1.14, the outcome is either $\mathcal{A}$, or $\mathcal{D}$ in the Maker-Maker convention. If Bob plays exactly the strategy $\mathcal{S}$, by construction, as it is winning for Breaker in the Maker-Breaker convention, Alice cannot fill up a hyperedge. Therefore, she cannot win and the outcome is $\mathcal{D}$. □

Note that the contrary is false in general: there exists some hypergraph $\mathcal{H}$ on which the outcome in the Maker-Maker convention is $\mathcal{D}$, but Maker can win in the Maker-Breaker convention. For instance in Tic-Tac-Toe, it is well known that the outcome in the Maker-Maker convention is $\mathcal{D}$, but the outcome in the Maker-Breaker convention is $\mathcal{N}$. For the readers that are not convinced that Maker wins going first in the Maker-Breaker convention, a proof will be provided later as illustration of Theorem 1.26.

The previous lemma can also be applied in avoidance games, with the following statement:

Lemma 1.18 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. If Alice (resp. Bob) has a winning strategy on $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ in the Avoider-Avoider game, then Enforcer has a winning strategy in the Avoider-Enforcer game played on $\mathcal{H}$ going first (resp. second).*

Proof. Suppose that Alice (resp.Bob) has a winning strategy $\mathcal{S}$ on $\mathcal{H} = (\mathcal{X}, \mathcal{F})$. Suppose that Enforcer plays $\mathcal{S}$ going first (resp. second) in the Avoider-Enforcer game played on $\mathcal{H}$. By construction of $\mathcal{S}$, Bob (resp. Alice) was supposed to claim all the vertices of a hyperedge. Therefore, this also happens in the Avoider-Enforcer convention, and Enforcer wins. □

These two implications are the main reasons explaining why *weak* and *strong* games are defined this way: a win of Maker (resp. Enforcer) in the Maker-Breaker (resp. Avoider-Enforcer) convention is called a *weak win*, as it does not imply any result for the Maker-Maker or the Avoider-Avoider convention. Reciprocally, a win of Breaker (resp. Avoider) in the Maker-Breaker (resp. Avoider-Enforcer) convention is called a *strong draw*, as it implies that Bob can draw in the Maker-Maker (resp. Avoider-Avoider) convention.

Finally, Beck [Bec02] has conjectured informally that if Maker (resp.Breaker) wins the Maker-Breaker game on some hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$, then Waiter wins the Waiter-Client (resp. Client-Waiter) game played on $\mathcal{H}$. Later, in 2009, Csernenszky, Mándity, and Pluhár [CMP09] have stated this conjecture more formally and proved that it holds in several known positional games as *k-in-a-row* or the *Shannon Switching game*. The main reason of this conjecture is that Waiter have much more control than Client, Maker or Breaker, as any move of Waiter constraints the move of Client. Therefore, the study of Waiter-Client and Client-Waiter games have partially been considered to understand better some Maker-Breaker games. However, even if this conjecture holds for several games, it has been disproved by Knox [Kno12] in 2012 for the general case. The counter-example hypergraph is provided in Figure 1.2, and we refer the reader to the paper of Knox for the proof that it disproves the conjecture.

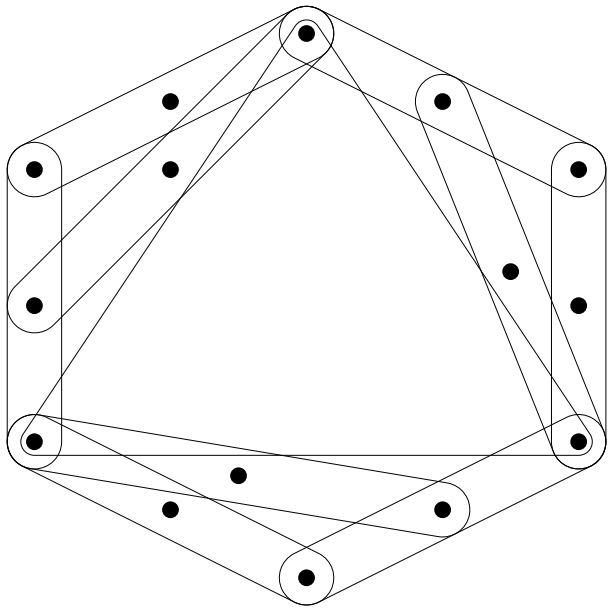


Figure 1.2: The counter example hypergraph of Beck's conjecture: Breaker wins in the Maker-Breaker convention, but Client wins in the Client-Waiter convention.

Extra sets

Similarly to the fact that, according to Lemma 1.15, the players always want to move in achievement games, it would be natural to think that adding other winning sets in Maker-Breaker games will benefit Maker. This result can be stated in a more general way that can be applied to several conventions:

Lemma 1.19 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$ be a sub-hypergraph of $\mathcal{H}$. If Maker (resp. Client, Waiter) wins the Maker-Breaker (resp. Client-Waiter, Waiter-Client) game on $\mathcal{H}'$, then Maker (resp. Client, Waiter) wins the Maker-Breaker game on $\mathcal{H}$.*

Moreover, if the number of vertices of $\mathcal{H}$ and $\mathcal{H}'$ are both odd or both even, this result also holds for Enforcer in the Avoider-Enforcer convention.

Proof. We do the proof in the Maker-Breaker convention. Let $\mathcal{S}$ be a winning strategy for Maker on $\mathcal{H}'$. Maker plays a strategy in $\mathcal{H}$ as follows:

- If Maker goes first, he plays the vertex $x \in \mathcal{X}$ that would have been played by $\mathcal{S}$ in $\mathcal{H}'$.
- When it's Maker's turn, if the last move of Breaker was not a vertex in $\mathcal{H}'$, Maker considers that he has played any unclaimed vertex instead.

Following this strategy, the vertices that Maker will claim in $\mathcal{H}$ will contain vertices corresponding to $\mathcal{S}$ in $\mathcal{H}'$. Therefore, she will fill up a hyperedge and win.

In Waiter-Client games, the same proof works, since Waiter can only choose pairs of vertices in $\mathcal{H}'$ until he wins.

In Client-Waiter games, the same proof almost works, except that Waiter can propose one vertex of $\mathcal{H}'$ and one vertex outside $\mathcal{H}'$. If this is the case, Client selects the vertex in $\mathcal{H}'$.

In Avoider-Enforcer games, we apply the same proof, using the fact that the number of vertices in $\mathcal{H} \setminus \mathcal{H}'$ is even. Therefore, is Avoider claims in $\mathcal{H} \setminus \mathcal{H}'$, Enforcer answers in $\mathcal{H} \setminus \mathcal{H}'$ and these two moves won't change the moves made on $\mathcal{H}'$. □

The parity argument in the Avoider-Enforcer case is very important, as changing the parity of the number of moves can change the winner. As shown in Lemma 1.16, what really matters in Avoider-Enforcer games is the player who plays the last move and not the first one. A counter example of Lemma 1.19 when the parity changes is provided in Figure 1.3

(a) Avoider looses going first

(b) Avoider wins going first

Figure 1.3: Example of a graph on which Enforcer wins, in Avoider-Enforcer, but not after adding a vertex and a hyperedge. Both hypergraphs have outcome $\mathcal{SL}$

Since the second player cannot win in Maker-Maker games, one might think that Lemma 1.19 can also be applied to Maker-Maker games. This is generally false, and this phenomenon is known as the *extra set paradox*, presented by Beck [Bec08]. There exists some Maker-Maker games in which the first player has a winning strategy, but she has not if a winning set is added to the game. The smallest known example of this paradox is given in Galliot's thesis [Gal23] and is provided in Figure 1.4.

1.1.3 Definitions about graphs

Even if the formal definition of positional games is made on hypergraphs, the most classical positional games are played on the edge set of graphs, and the winning sets are some structures of the graph. Most of these classical games were introduced by Beck, and we refer the reader to his book *Combinatorial game, Tic-Tac-Toe Theory* [Bec08] for a complete history of these games. For readers unfamiliar with graph theory, we present here most of the graph-theoretic notions that will be used in this manuscript, and we refer the reader to Bondy and Murty [BM76] for a more complete presentations of graphs.

Vocabulary in graphs

A *(simple) graph* G is a 2-uniform hypergraph. In general, we write it $G = (V, E)$, where V is the set of *vertices* and $E \subseteq V \times V$ is the set of *edges*. Sometimes, we will write $V(G)$ and $E(G)$ to refer to the set of vertices or edges of G. A *subgraph* of G is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. If G' is a subgraph of G, we write $G' \subseteq G$. If $V' \subseteq V$ (resp. $E' \subseteq E$), the subgraph *induced* by V' (resp. by E') is the graph $G' = (V', E \cap (V' \times V'))$ (resp. $G' = (\{v, \exists e' \in E', v \in e'\}, E')$). Let $u, v \in V$ be two vertices. If $(u, v) \in E$, we say that u is a *neighbor* of v or that u and v are *adjacent*. The *open neighborhood* of v, denoted by $N(v)$ is the set of neighbors of v. The *closed neighborhood* of v, denoted by $N[v]$ is $N[v] = N(v) \cup \{v\}$. The *degree* of v is $d(v) = |N(v)|$. A *leaf* $\ell \in V$ is a vertex of degree 1. An *isolated vertex* $v \in V$ is a vertex of degree 0.

Let $G = (V, E)$ be a graph. Let $u, v \in V$ be two vertices.

(a) Outcome: $\mathcal{A}$.

(b) Outcome: $\mathcal{D}$.

Figure 1.4: The extra-set paradox in Maker-Maker Games

A *trail* of length k between u and v is a set of vertices $u_1 = u, \ldots, u_{k+1} = v$ with $k \geq 1$ such that for any $1 \leq i \leq k$, we have $(u_i, u_{i+1}) \in E$. If in this trail, all the vertices are different, this trail is a *path*, denoted by P_{k+1}. If only u_1 and u_{k+1} are equal, and $k \geq 2$, this trail is a *cycle*, denoted by C_k. A set $C \subset V$ is said to be *connected* if for any $u, v \in C$, there exists a path between u and v. If C is maximal for inclusion, C is said to be a *connected component* of G. We denote by $CC(G)$ the set of connected components of G. If G has a single connected component, we say that G is *connected*. A graph $T = (V, E)$ is said to be a *forest* if it contains no cycle. If it is a connected forest, we say that T is a *tree*.

Structures in graphs

Now that graphs are defined, several games are constructed so that the hyperedges correspond exactly to some structure in the graph. We introduce here the structures that will be studied in this manuscript.

Let $G = (V, E)$ be a graph.

A *Hamiltonian cycle* of G is a cycle that contains all the vertices of G. A *matching* in G is a set of edges $M \subset E$, such that for any $e_1, e_2 \in M$, we have $e_1 \cap e_2 = \emptyset$. It is said to be a *perfect matching* if V is covered by M. A *spanning tree* T of G is a subgraph of G which is a tree and such that $V(T) = V$. A set $I \subset V$ is said to be *stable* or *independent*, if for any $u, v \in I$, $(u, v) \notin E$. A *dominating set* $D \subset V$ is a set such that $\bigcup_{v \in D} N[v] = V$. A *Feedback vertex set* $S \subset V$ is a set such that $G \setminus S$ is a forest. A *Feedback edge set* $S \subset E$ is a set of edges such that $G \setminus S$ is a forest. A *vertex cover* $S \subset V$ is a set of vertices such that any edge of E is adjacent to at least one vertex of S. We say that two vertices $u, v \in V$ have the same *type* if $N(v) \setminus \{u\} = N(u) \setminus \{v\}$. The graph G has *neighborhood diversity* at most w if there exists a partition of V into at most w sets such that the vertices in each set have all the same type.

The neighborhood diversity is a graph parameter introduced by Lampis [Lam10] to generalize FPT algorithms parameterized by vertex cover to larger classes of graphs. Note that each set must induce a clique or an independent set. Furthermore, if a set of graphs has bounded vertex cover, then it has bounded neighborhood diversity.

Classes of graphs

While studying some games on graphs, it will happen that a problem is too hard to be solved on general graphs. When this is the case, a classical way to handle the problem is to study it on some classes of graphs. Here we introduce most of the classes that will be studied.

We first present the two following operations:

- If G and H are two graphs, $G \otimes H$ refers to the join between G and H, i.e., the graph obtained by putting together one copy of G and one of H, and adding any edge between a vertex of G and a vertex of H.
- If G and H are two graphs, $G \cup H$ refers to the union of G and H, i.e., the graph obtained by putting together one copy of G and one of H, but no edges between G and H.

Let $G = (V, E)$ be a graph, and let n, m be two integers.

G is a *cograph* if it can be constructed from isolated vertices using only $\otimes$ and $\cup$ operations. The graph K_n called the *clique* or *complete graph* on n vertices is the graph $G = (V, E)$ with $|V| = n$ and $E = \{(u, v) \in V \times V, u \neq v\}$. A graph is said to be a *cluster* if it is a union of cliques. The *distance to cluster* of a graph G is the size of a smallest vertex subset whose deletion makes G a cluster. G is said to be *split* if its vertex set V can be partitioned into two sets K, S such that the subgraph induced by K is a clique and the one induced by S is a stable set. G is said to be *bipartite* if its vertices can be partitioned into two sets V_1 and V_2 such that $E \subset V_1 \times V_2$. The graph $K_{n,m}$ called the *complete bipartite graph* is the graph $G = (V, E)$ with $V = (V_1 \cup V_2)$ with $|V_1| = n$, $|V_2| = m$ and $E = V_1 \times V_2$. If $n = 1$, $K_{1,m}$ is said to be a *star*.

G is *planar* if it can be drawn on the plane in such a way that its edges intersect only at their endpoints.

1.1.4 Some examples of games

In this section, we will introduce the different positional games on graphs that will be studied in this manuscript.

Games on edges

Most of the studies in positional games consist of studying games played on the edge sets of graphs. Alternately, the players claim edges of a graph, and the hyperedges consist of some structure of the graph. The best-known game on edges is probably SIM, an Avoider-Avoider game played on a clique, in which the first player to claim three edges that from a triangle loses. The H-game is a natural generalization of this game, see Figure 1.5.

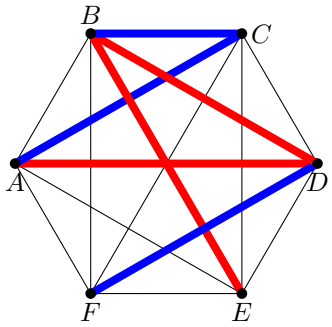


Figure 1.5: Example of position in the game SIM. If Alice (in red) claims for instance the edge (D, E), she loses by closing the triangle BDE.

Definition 1.20 (H-game)**.** *Let H be a graph. The* H-game *is defined on a graph G by the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = E(G)$ and $\mathcal{F} = \{E' \subset E(G), E'$ induces a copy of $H\}$.*

If $H = K_k$ (resp. $H = K_{1,k}$) we will call this game the clique-game (resp. star-game).

Definition 1.21 (Connectivity game)**.** *The* connectivity game *is defined on a graph G by the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = E(G)$ and $\mathcal{F} = \{E' \subset E(G), E'$ induces a spanning tree of $G\}$.*

Definition 1.22 (Perfect matching game)**.** *The* perfect matching game *is defined on a graph G by the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = E(G)$ and $\mathcal{F} = \{E' \subset E(G), E'$ is a perfect matching of $G\}$.*

Definition 1.23 (Hamiltonicity game)**.** *The* hamiltonicity game *is defined on a graph G by the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = E(G)$ and $\mathcal{F} = \{E' \subset E(G), E'$ is a Hamiltonian cycle of $G\}$.*

Note that in all of the above examples, by definition of positional games, the edges claimed by the player must contain a hyperedge in order to win (or lose in avoidance games). It is not necessary that these edges form exactly the aimed structure.

Games on vertices

The other way to play positional games on a graph is to claim vertices instead of edges. Again, it will be natural to consider some structure of the graph as the winning sets. Here we present the domination game. Several other "domination games" have been introduced in the literature under different rulesets, see [ABBS02, BKR10] for some examples. Here we focus on the positional game version of the game, introduced in the Maker-Breaker convention by Duchêne *et al.* [DGPR20] in 2020.

Definition 1.24 (Domination game)**.** *The* Maker-Breaker domination game *is defined on a graph G. Two players, Dominator and Staller, alternate claiming unclaimed vertices of the graphs. Dominator wins if she claims all the vertices of a dominating set. Otherwise, Staller wins.*

In the case of the domination game, if G is a graph, the natural hypergraph to consider would have one hyperedge for each dominating set of G. However, a graph usually has an exponential number of dominating sets. Therefore, if the underlying hypergraph is required, it is often more convenient to consider its dual, i.e., the hypergraph of the closed neighborhoods. Indeed, If Breaker fills up the close neighborhood of some vertex, Maker will not dominate this vertex, and a graph on n vertices has at most n closed neighborhoods. Because of this duality and that both points of view can be used in this game, we will refer to the players as Dominator and Staller instead of Maker and Breaker. Dominator aims to claim the vertices of a dominating set of G and Staller aims to claim the closed neighborhood of some vertex in G.

1.2 History of positional games

In this section, we will present how positional games were introduced, their connection with other areas of computer science, and the first important results on them. Since after their introduction in 1963, most of the studies on positional games were focused on complete graphs, we will present here their study on complete graphs, and we postpone the study of their complexity to Section 1.3.

1.2.1 The beginning: Maker-Maker games

Among the various conventions, the Maker-Maker convention is the most natural. Therefore, when Hales and Jewett [HJ63] introduced positional games, one motivation for the introduction of positional games was the generalization of the famous Tic-Tac-Toe game, and the initial study of positional games focused on the Maker-Maker convention. Here, both players want to fill up a hyperedge of a given hypergraph. These games are indeed the most likely to be played in real life. Hales and Jewett [HJ63] directly provided some general results, as Lemma 1.14, and a weaker version of pairing strategies, which will be presented later in Section 1.4. Then, they introduced the first generalization of Tic-Tac-Toe: the n^d game, which consists of playing Tic-Tac-Toe on the d-dimensional n-grid, the classical Tic-Tac-Toe being the 3^2 game. They proved that, if $n \geq 3^d - 1$, then this game ends in a draw. Conversely, for any n, if d is large enough, they proved that the first player has a winning strategy.

The other major generalization of Tic-Tac-Toe is the k-in-a-row game. The k-in-a-row game is played on a potentially infinite grid, and the winning sets are any set of k vertices aligned, either in a row, in a column or in a diagonal. This game includes the popular 5-in-a-row game Gomoku, which was introduced in the 1800s and is still played today. While 4-in-a-row is an easy win for the first player [Folklore], Alis *et al.* [AvdHH96] proved that the first player wins on the 15*15 grid. However, it is still unknown whether this result can be generalized to larger grids or not. Most of the other known results about the Maker-Maker k-in-a-row game come from the Maker-Breaker study of the game.

1.2.2 Introduction of Maker-Breaker games

According to Lemma 1.14, the second player cannot win in the Maker-Maker convention. Therefore, Chvátal and Erdős [CE78] have introduced the Maker-Breaker convention in 1978. Under this convention, both players have simpler goals: while in the Maker-Maker convention each player's goals are both to fill up a hyperedge and to prevent their opponent from filling one up, in the Maker-Breaker convention, two players compete for a single goal. Since there is no draw in Maker-Breaker games, one player has a winning strategy. k-in-a-row is a famous example of a studied game, that was introduced in the Maker-Maker convention, but whose study is now focused on the Maker-Breaker convention. In the Maker-Breaker version of this game, the result from Alis *et al.* [AvdHH96] proves, by Lemma 1.19, that the game is a win for Maker in any grid of dimension greater than $15 * 15$ for $k \leq 5$. In 1980, Zetters proved that Breaker can ensure a draw if $k \geq 8$ [Zet80] by providing a paving on the infinite grid that blocks any set of 8 aligned cases. This

result directly implies that k-in-a-row is a draw on any grid in the Maker-Maker convention for $k \geq 8$. The outcomes for $k = 6$ and $k = 7$ are still open.

Since Maker-Breaker games were introduced due to the fact that Bob cannot win in Maker-Maker games, Maker-Breaker games are generally studied with Maker going first. This will always be the case in this manuscript unless something is specified.

Although Maker-Breaker games seem to be artificial and were introduced only as a tool for understanding Maker-Maker games, some well-known games can be formalized as Maker-Breaker games. For example, the game of HEX is played on a hexagonal grid by two players, namely Red and Blue, who each get two opposite sides of the grid and try to connect them with a path of their color. This game is not a Maker-Maker game, since the winning sets for the two players are different, but it can be modeled in the Maker-Breaker convention, since at the end of the game, the only way to prevent a red path from connecting two opposite sides is to build a blue path between the other two.

1.2.3 Toward general results on positional games: Edős-Selfridge criterion

Soon after their introduction, some strong results were found on Maker-Breaker games. The most important is probably the Erdős-Selfridge criterion, which states that if there are not so many hyperedges and if they are large enough, Breaker will have the time to play in each of them. Formally, it is stated as follows:

Theorem 1.25 (Erdős-Selfridge [ES73]). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph.*

- *If $\sum_{f \in \mathcal{F}} 2^{-|f|} < \frac{1}{2}$, then Breaker wins the Maker-Breaker game on $\mathcal{H}$ going second.*
- *If $\sum_{f \in \mathcal{F}} 2^{-|f|} < 1$, then Breaker wins the Maker-Breaker game on $\mathcal{H}$ going first.*

This theorem can quickly solve some games. For instance, in the 4-in-a-row game played on a $4 * 4$ grid, There are ten hyperedges of order four. Therefore, $\sum_{f \in \mathcal{F}} 2^{-|f|} = \frac{10}{16}$, which proves that Breaker wins going first.

Despite the fact that Erdős-Selfridge's criterion is very general and can often be used to prove that some games are wins for Breaker, a weaker criterion, due to Beck [Bec82] in 1982, can sometimes prove that Maker can win on some games. This criterion uses something quite important for Maker, the *maximum pair degree*, usually denoted by $\Delta_2(\mathcal{H})$, and defined as the largest number of hyperedges containing a pair of vertices. Formally, we have:

$$\Delta_2(\mathcal{H}) = \max \{|\{f \in \mathcal{F} : u, v \in f\}| : u \neq v \in \mathcal{X}\}$$

.

Intuitively, the higher this parameter is, the more likely it is that, whenever Maker plays a move, Breaker can respond with a move that touches the same hyperedges.

Theorem 1.26 (Beck [Bec82]). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph, If*

$$\sum_{f \in \mathcal{F}} 2^{-|f|} > \frac{\Delta_2(\mathcal{H})|\mathcal{X}|}{8}$$

then Maker wins the Maker-Breaker game on $\mathcal{H}$ going first.

Sketch of the proof. (Theorem 1.25 and Theorem 1.26)

The main idea of the proofs of Theorem 1.25 and Theorem 1.26, is to define a potential function $p(\mathcal{H}) = \sum_{f \in \mathcal{F}} 2^{-|f|}$, and to play a strategy that optimizes the potential after each move of the considered player, where the hypergraph is updated after each move, Maker aiming to maximize the potential, and Breaker aiming to minimize it. A move of Maker removes the claimed vertex from the hyperedges containing it, while a

move of Breaker removes the hyperedges containing the claimed vertex. With these updates, Maker's moves increase the potential, while Breaker's moves decrease it. When all the vertices have been played, we have $|f| = 0$ for any hyperedge f, and therefore, the potential is an integer. If it is 0, Breaker has won as there is no hyperedge left. If it is at least one, Maker has managed to fill up a hyperedge and therefore has won. □

Using this result, we can prove that Maker wins the Maker-Breaker Tic-Tac-Toe going first. Even if Theorem 1.26 cannot be applied on the empty grid, if Maker starts claiming the center vertex, after any answer of Breaker, there will be three hyperedges of order 2 and at least two hyperedges of order 3. Therefore, we will have $\sum_{f\in\mathcal{F}} 2^{-|f|} \geq 3 * \frac{1}{4} + 2 * \frac{1}{8} = 1$. In Tic-Tac-Toe, we have $\Delta_2(\mathcal{H}) = 1$. Thus, after two moves, $|\mathcal{X}| = 7$, and $\sum_{f\in\mathcal{F}} 2^{-|f|} \geq 1 \geq \frac{7}{8}$. Therefore, Maker has a winning strategy in Maker-Breaker convention starting with playing the middle of the grid.

All games on edges presented until now are easy Maker wins when the size of the board is large enough: Hefetz *et al.* [HKSS14] showed that, on a complete graph, Maker can always win.

1.2.4 Biased Maker-Breaker games

Intuitively, the number of threats in a complete graph is large enough so that Maker can always create any structure.

Because Maker can always win in large enough graphs in most of the classical Maker-Breaker games, *bias* have been introduced to balance the game and give a chance to Breaker. The presentation of games that have been done until known is said *unbiased*, as both players claim a single vertex per turn.

Definition 1.27 (Biased Maker-Breaker game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let $p, q \leq 1$ be two integers. The* biased $(p : q)$ Maker-Breaker game *on $\mathcal{H}$ is defined similarly to the Maker-Breaker game on $\mathcal{H}$, except that when it is Maker's turn, she claims p vertices and when it is Breaker's turn, he claims q vertices. The values p and q are then called the* bias *of Maker and Breaker respectively.*

Note that the unbiased Maker-Breaker game corresponds to $p = q = 1$. Since the bias has been introduced to balance the game when the unbiased case is favorable to Maker, the value $b = \frac{q}{p}$ is quite important, and we will often consider $(1 : b)$ games instead of $(p : q)$ games. By simplicity, a game played with bias $(1 : b)$ will be said to be played with a bias b.

Lemma 1.28 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. If Breaker wins on $\mathcal{H}$ with a bias b, he wins on $\mathcal{H}$ with a bias b' for any $b' \geq b$.*

A consequence of this lemma is that, for any hypergraph $\mathcal{H}$ there exists a minimum value of b_0, called the *threshold bias* such that Maker wins the $(1 : b)$ Maker-Breaker game played on $\mathcal{H}$ if and only if $b < b_0$. The threshold bias is well-defined on any hypergraph such that $\mathcal{F} \neq \emptyset$, and such that Maker cannot win with his first move. Today, one of the main studies in positional games consists in computing the threshold bias on different games.

By studying biased games, one can aim for general results similar to those known in unbiased games. Beck [Bec82] has generalized Theorem 1.25 to biased games.

Theorem 1.29 (Beck [Bec82])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let $p, q \geq 1$ be two integers. Suppose that*

$$\sum_{f\in\mathcal{F}} (1+q)^{-|f|/p} < \frac{1}{1+q}$$

Then Breaker wins the $(p : q)$ Maker-Breaker game played on $\mathcal{H}$.

Theorem 1.26 can be stated in the biased case to give a sufficient condition for Maker to win:

Theorem 1.30 (Beck [Bec82]). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let $p, q \geq 1$ be two integers. If*

$$\sum_{f \in \mathcal{F}} \binom{p+q}{p}^{-|f|} > \frac{p^2 q^2 \Delta_2(\mathcal{H}) |\mathcal{X}|}{(p+q)^3}$$

then Maker wins the Maker-Breaker game on $\mathcal{H}$.

Another major theorem in biased games is due to Bednarska and Luczak [BL00], who give a bound on the threshold bias for the H-game for any graph H. An important part of their proof is that, if the hypergraph is large enough, Maker wins with high probability even by playing randomly.

Theorem 1.31 (Bednarska-Luczak [BL00]). *Let H be a graph which contains at least three nonisolated vertices. There exists constants c_0, C_0 and n_0 such that for every $n \geq n_0$:*

- *if $b \leq c_0 n^{1/m(G)}$, then Maker wins the H-game played on K_n with bias b,*
- *if $b \geq C_0 n^{1/m(G)}$, then Breaker wins the H-game played on K_n with bias b,*

where $m(G) = \max\limits_{\substack{H \subset G \\ |V(H)| \geq 3}} \dfrac{|E(H)| - 1}{|V(H)| - 2}$

This result is very strong, since it gives both lower and upper bounds for every nontrivial graph H. Moreover, these bounds are tight up to multiplicative constants. However, the proof of this theorem relies on the fact that, since H is fixed, if its size is small enough compared to the board of the game G, by playing randomly, the edges claimed by Maker, will behave as a random graph. Therefore, the probability that the graph induced by Maker's edges contains a copy of H depends on its density, which gives the bound for the threshold bias. This result does not hold if the aimed winning structure in G has a size that depends on G, which is the case for the connectivity game, the hamiltonicity game or the perfect matching game. For these three games, the threshold bias is $O(\frac{n}{\log(n)})$. For the first game, this result is known from Chvátal and Erdős [CE78]. For the second and third ones, only the upper bound was given by Chvátal and Erdős, who proved that, for any $\varepsilon > 0$, if $b \geq \frac{(1+\varepsilon)n}{\log(n)}$ for some constant c, Breaker can isolate a vertex. The two lower bounds are then obtained Krivelevich [Kri11] in 2011, proving that, for any ε, Maker wins if $b \leq \frac{(1-\varepsilon)n}{\log(n)}$ for the hamiltonicity game. As a consequence of this result, the threshold bias is also $\frac{n}{\log(n)}$ for the perfect matching game, since if a graph has an even number of vertices, a Hamiltonian cycle contains a perfect matching, which proves that the last bound is also tight.

Note that this idea of playing randomly had already been stated by Erdős, as the *probabilistic intuition*, which states that the outcome of an optimally played game should not differ asymptotically from the same game played randomly. This intuition is still used as a conjecture and is still studied today, see Nenadov [Nen23].

Definition 1.32 (Erdős-Selfridge [ES73]). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Consider a random partition of $\mathcal{X}$ into two sets V_M, V_B. The* probabilistic intuition *states that, if $(\mathcal{H}, V_M, V_B)$ is a winning position for Maker (resp. Breaker) with high probability, then Maker (resp. Breaker) wins the Maker-Breaker game on $\mathcal{H}$.*

This intuition explains some later results on positional games on random graphs. Let $G(n,p)$ be a random graph with n vertices on which each edge exists with probability p. Stojaković and Szabó [SS05] computed the threshold probability p_F such that Maker wins on $G(n,p)$ almost surely if and only if $p \geq p_F$. They proved that asymptotically, as n goes to infinity, for the connectivity game, the perfect matching game, the clique game and the Hamiltonian cycle game, p_F and the threshold bias differ at most by a multiplicative constant.

1.2.5 Misere version: Avoider-Enforcer games

A classical study in combinatorial games is the *misere* version of games, i.e. the transformation of winning conditions into losing conditions. This is what happens in positional games when dealing with avoidance games. The Avoider-Avoider convention is the misere version of the Maker-Maker convention. The well-known SIM game, introduced by Simmons [Sim68] in 1968 and presented in Figure 1.5, is played under this convention.

Avoider-Enforcer games were introduced by Lu [Lu91] under the name Antimaker-Antibreaker games as the misere version of Maker-Breaker. The standard name of this convention, Avoider-Enforcer (or sometimes Avoider-Forcer), has been popularized by Hefetz and his co-authors after 2007 [HKS07a, HKSS07, HKS07b, Hef07]. Note that several studies of this convention consider partial avoidance, i.e., Avoider loses if she claims a certain fraction of a hyperedge, rather than the entire hyperedge, but we will not focus on that study here. A strong connection between Avoider-Enforcer and Maker-Breaker has quickly emerged, as Lu has proved the following criterion, which is similar to the Erdős-Selfridge criterion in Maker-Breaker games.

Theorem 1.33 (Lu [Lu91])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph, if*

$$\sum_{f \in F} 2^{-|f|} < 1$$

then Avoider wins the Avoider-Enforcer game on $\mathcal{H}$.

However, most of the early studies considered that Avoider loses if he claims more than a fraction of $(1-\varepsilon)$ vertices from a hyperedge for ε positive. The biased version of Avoider-Enforcer, introduced by Hefetz *et al.* [HKS07a], consists, as in the Maker-Breaker version of the game, of letting Avoider and Enforcer claim p and q vertices per move respectively. The next point of the study should be, as for Maker-Breaker games, to introduce the threshold bias. But, this is not possible here because, with this definition, biased Avoider-Enforcer games are not monotonous. As presented by Hefetz *et al.* [HKSS10], the outcome can depend on the parity of the bias. For example, take $\mathcal{H} = \big(\{u_1, u_2, u_3\}, \{\{u_3\}\}\big)$, i.e. the hypergraph with three vertices and a losing set of size one. Avoider, going first loses the unbiased game and the biased $(3,1)$ game but wins the biased $(2,1)$ game. This phenomenon is mainly due to the fact that in Avoider-Enforcer games, players do not want to play in general, but since some vertices can be used to pass their turn, a player may want to claim some of them in order to prevent his opponent from skipping his turn. To deal with this problem, Hefetz *et al.* introduced monotone (p,q) Avoider-Enforcer games, and called the classical biased game as strict biased game.

Definition 1.34 (monotone (p,q) Avoider-Enforcer)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph, and let $p, q \geq 1$ be integers. A* monotone (p,q) Avoider-Enforcer game *on $\mathcal{H}$ is as the Avoider-Enforcer game on $\mathcal{H}$, but Avoider claims at least p vertices per move and Enforcer at least q vertices per move.*

Note that this relaxation would not be necessary in Maker-Breaker games, since if Maker (resp. Breaker) is allowed to claim up to p (resp. q) vertices per move, with a similar proof of Lemma 1.15, there will exist an optimal strategy in which they claim p (resp. q) vertices.

This definition of biased Avoider-Enforcer games is monotonous. The threshold bias is defined similarly to Maker-Breaker games as the smallest integer b such that Avoider wins the biased $(1,b)$ Avoider-Enforcer game. In [HKSS10], they determined the threshold bias for several games, such as the H-game and the connectivity game.

In contrast to Maker-Breaker games, where there are no differences between the monotone and the strict rulesets, in Avoider-Enforcer games, both rulesets have their interests. The monotone ruleset allows introducing the threshold bias and provide results similar to its Maker-Breaker analog, while the strict ruleset is more natural and can be seen as a tool to handle some Maker-Breaker games where achieving some structure for Maker is equivalent to avoiding another structure. For instance, Hefetz [HKSS08] determined the threshold bias in the planarity game, in which Maker aims to claim the edges of a non-planar graph by providing a strategy avoiding short cycles and using some graph theory results about graphs of large girth.

1.2.6 I cut, you'll choose: Client-Waiter and Waiter-Client games

The last type of convention studied in this thesis are Waiter-Client and Client-Waiter games. These games were introduced by Beck [Bec02] in 2002 under the names Picker-Chooser and Chooser-Picker respectively to represent the "I cut you'll choose" principle, and generalize it. For instance, consider the clique-game, i.e., the H-game with $H = K_q$ for some integer q. In a *I cut you'll choose* game, Waiter cuts the edges of K_n into two parts, and Client has to choose one of the two parts. Therefore, this game has a direct link with the Ramsey number: Client wins if and only if $n \geq r(q)$ where $r(q)$ is the Ramsey number of q, i.e., the minimum order of K_n such that any 2-coloring of the edges of K_n contains a monochromatic copy of K_q. Waiter-Client and Client-Waiter games are then a generalization of this principle by proposing the edges two by two, and letting Client claim exactly one of them, and letting the other one to Waiter.

The current names of the conventions, Waiter-Client and Client-Waiter were introduced by Hefetz, Krivelevich and Tan [HKT16], in 2016, as these new names are less ambiguous with respect to the names of the players. These two conventions are very similar, as they are played with the same rules. Only the winning conditions are different. In both cases, Waiter selects two vertices and offers them to Client who chooses one of them to claim, letting Waiter claims the second one. In the Waiter-Client convention, Waiter plays the role of Maker and tries to fill up a hyperedge, while Client plays the role of Breaker and tries to prevent him from succeeding. In the Client-Waiter convention, it is the opposite, i.e., Client plays the role of Maker and Waiter the role of Breaker.

This convention, was presented as a first step to understand Maker-Breaker games. In fact, Beck introduced it as an unbalanced version of the game where Waiter has more control on the game than Client. Indeed similarities between both conventions exists, such that theorems similar to Theorem 1.25 and Theorem 1.26 were provided in Waiter-Client and Client-Waiter games.

Theorem 1.35 (Beck [Bec02])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. If $\sum_{f \in \mathcal{F}} 2^{-|f|} < 1$, then Client wins the Waiter-Client game on $\mathcal{H}$.*

Theorem 1.36 (Beck [Bec02])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph, If*

$$\sum_{f \in \mathcal{F}} 2^{-|f|} > \frac{\Delta_2(\mathcal{H})|\mathcal{X}|}{8}$$

then Client wins the Client-Waiter game on $\mathcal{H}$.

The similarities between Client-Waiter games and Maker-Breaker games motivate the study of classical Maker-Breaker games in the Client-Waiter convention, when some open problems were unsolved in the first convention. For instance, even if the 7-in-a-row Maker-Breaker game is only conjectured to be a Breaker win in the Maker-Breaker convention, but Csernenszeky [Cse10] proved in 2010 that Waiter wins in the Client-Waiter convention on any subgrid of the infinite grid.

1.2.7 Connection between positional games and other areas of computer science

Positional games have strong connections to several areas of computer science. The most natural one is probably Ramsey Theory, since players, by claiming the vertices of a hypergraph, create a partition of the vertices between the two players. For example, it is known, by a Ramsey argument, that for any graph H, there exists an integer N such that every 2-coloring of the edges of K_N contains a copy of H. This result has been generalized with a hypergraph coloring result by Erdős and Selfridge [ES73], which we need to introduce before stating the result.

Definition 1.37 (Hypergraph coloring)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A k-*coloring *of $\mathcal{H}$ is a function $c : \mathcal{X} \to [1, k]$. It is said to be* proper *if for any hyperedge $f \in \mathcal{F}$, there exists $u, v \in f$ such that $c(u) \neq c(v)$.*

The minimum number k such that there exists a proper k-coloring of $\mathcal{H}$, if it exists is called the chromatic number *of $\mathcal{H}$ and is denoted $\chi(\mathcal{H})$.*

Theorem 1.38 (Erdős and Selfridge [ES73]). *Let $\mathcal{H}$ be a hypergraph. If $\chi(\mathcal{H}) \geq 3$, then Alice wins the Maker-Maker game on $\mathcal{H}$.*

Sketch of the proof. The proof relies on the fact that in Maker-Maker games, by Lemma 1.14, Bob cannot win, thus the outcome is either $\mathcal{D}$ or $\mathcal{A}$. If it is $\mathcal{D}$, at the end of the game, no player would have claimed all the vertices of a hyperedge. Therefore, the set of vertices claimed by Alice and by Bob would provide a 2-coloring of $\mathcal{H}$, which cannot exist. □

Corollary 1.39. *Let $\mathcal{H}$ be a hypergraph. If, in Maker-Maker convention, we have $o(\mathcal{H}) = \mathcal{D}$, then $\chi(\mathcal{H}) \leq 2$.*

A direct consequences of Theorem 1.38, together with Theorem 1.25 is the following:

Theorem 1.40. *Let $\mathcal{H}$ be a k-uniform hypergraph. If $|\mathcal{F}| \leq 2^{k-1}$, then $\chi(\mathcal{H}) \leq 2$.*

However, note that Theorem 1.38 does not provide a winning strategy, but only the outcome.

Another important problem connects positional game theory to hypergraph theory: the neighborhood conjecture. Let us first introduce the neighborhood of a hyperedge.

Definition 1.41 (Neighborhood of a hyperedge). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let $f \in \mathcal{F}$ be a hyperedge. The neighborhood of f, denoted by $N[f]$ is the set of hyperedges that intersect f including f itself, i.e. $N[f] = \{f' \in \mathcal{F}, f \cap f' \neq \emptyset\}$. The* maximum neighborhood size *of $\mathcal{H}$ is then the largest size of a neighborhood of a hyperedge in $\mathcal{H}$.*

Conjecture 1.42 (Beck [Bec02]). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a k-uniform hypergraph. If the maximum neighborhood size of $\mathcal{H}$ is smaller than 2^{k-1}, Breaker wins the Maker-Breaker game on $\mathcal{H}$.*

Note that this conjecture would be strictly stronger than Erdős-Selfridge criterion on regular hypergraphs, as it only requires local properties and not global ones. Indeed, sometimes, the Erdős-Selfridge criterion cannot be applied because of the number of hyperedges, even if they are not connected to each other. This phenomenon is prevented with this conjecture?

Other connections between positional games and complexity theory or logic theory exists, but this connection will be presented more in details among this manuscript as a part of the work during my thesis was focused on this connection. Most of these results will be presented in Chapter 2 and Chapter 4.

1.3 Algorithmic and complexity studies

Although positional games on complete graphs have been the focus of most studies since their introduction, algorithmic studies on general hypergraphs began soon after. To handle algorithmic studies of positional games on graphs, the graph considered will no longer be complete, and any graph will be considered as input of the problems. Recently, there has been a greater emphasis on these types of results. As it will be shown in this section, positional games are a good tool to handle some complexity classes, and determining how hard it is to know which player has a winning strategy is very important.

However, to handle correctly the complexity of positional games, we will remind the reader several definitions and results in complexity theory, then we will focus on the complexity of positional games.

1.3.1 Complexity

In this section, Σ will be a finite set called alphabet. A *word* on Σ is a finite tuple of elements of Σ. A *language* on Σ is a (possibly infinite) set of words.

The main study of complexity theory goes through *decision problems*, and classifying them into different complexity classes is an important goal of our study.

Definition 1.43 (Decision problem). *A* decision problem *is a question, that takes as input a language L over an alphabet Σ^* and a word $x \in \Sigma^*$, and asks whether $x \in L$ or not.*

Definition 1.44 (Complexity class)**.** *A* complexity class *is a set of languages L, such that for any $x \in \Sigma^*$, the decision problem taking as input (L, x) can be solved by a Turing Machine allowing a certain quantity of time or memory.*

Usually, we can identify any closed question with decision problems of determining whether a word belongs to a language. For instance, the question "Does a graph G contain a perfect matching" can be identified with the decision problem taking as input G as a word and the set of graphs admitting a perfect matching as language.

Without going into the details of Turing machines, one can simply consider that basic operations, such as comparisons, additions, or conditional branches can be computed in constant time on it. A Turing machine can then be deterministic, if only basic operations are allowed, or *nondeterministic*, if one more operation is allowed: creating a copy of itself. In this case, it is sufficient that one of the copies outputs "$\top$" to the problem to state that the word is in the language. When attempting to classify problems into different complexity classes, the concepts of reduction and hardness naturally arise.

Definition 1.45 (Reduction)**.** *A* reduction *is a computable function f that takes as input a word x of a language L and outputs a word x' of a language L' such that, $x \in L$ if and only if $x' \in L'$. If the Turing machine that computes f ends in polynomial time, then we say that the reduction is polynomial.*

Definition 1.46 (Hardness)**.** *Let $\mathcal{C}$ be a complexity class. A language L is said to be* $\mathcal{C}$-hard *if any language in the class $\mathcal{C}$ can be reduced to L in polynomial time. Furthermore, we say that L is* $\mathcal{C}$-complete *if $L \in \mathcal{C}$.*

The classes P and NP

The dichotomy between the classes P and NP is probably one of the most famous problems in computer science, and answering the question "Is P equal to NP ?" is a millennium problem. The class P is the class of languages that can be recognized in polynomial time by a deterministic Turing machine, while the class NP is the class of languages that can be recognized in polynomial time by a nondeterministic Turing machine. Thus, $\mathsf{P} \subset \mathsf{NP}$, and the question of the equality between the two classes is still open today, although most of the community agrees to say that they are probably different. Despite the fact that the formal definition of the classes requires the concept of Turing machines, the most practice definition, especially in algorithmic studies, is the one with certificates:

Definition 1.47 (NP)**.** *A language L belongs to the class* NP *if there exists a polynomial P and a function $f : \Sigma^* \times \Sigma^* \to \{\top, \bot\}$, such that for any input $x \in \Sigma^*$ we have, $x \in L$ if and only if there exists $c \in \Sigma^*$ with $|c| < P(|x|)$, $f(x, c) = \top$ and the number of operation to compute $f(x, c)$ is lower than $P(|x|)$. c, if it exists, is then called a* certificate *for x.*

The class NP is the natural class for studying problems that are easily verifiable, for example, the existence or non-existence of some structure in graphs. However, proving that some structure does no exist in a graph is, in general, not easily certifiable. By considering certificates to prove that $x \notin L$, the class under consideration becomes coNP.

Definition 1.48 (coNP)**.** *A language L belongs to the class* coNP *if there exists a polynomial P and a function $f : \Sigma^* \times \Sigma^* \to \{\top, \bot\}$ where Σ^* a set of words such that for any input $x \in \Sigma^*$ we have, $x \notin L$ if and only if there exists $c \in \Sigma^*$ with $|c| < P(|x|)$, $f(x, c) = \bot$ and the number of operation to compute $f(x, c)$ is lower than $P(|x|)$. c, if it exists, is then called a* certificate *for x.*

In the rest of the paper, we will identify a language and the decision problem of deciding whether a word is in a language or not. For instance, Clique, the language of words of the form (G, k) such that G contains a clique of size k, will be identified with the decision problem of determining, given a graph G and an integer k, whether G contains a clique of size k or not. Note that if it is not precised, we suppose that $\Sigma = \{0, 1\}$, and that the pair (G, k) is encoded in binary.

The study of NP-complete languages started with SAT, the language of satisfiable boolean formulas, which was proved NP-complete by Cook [Coo71] in 1971, and has become one of the most important research topic in complexity theory. To introduce SAT formally, we will first recall some definitions in boolean logic.

Definition 1.49. *Let X be a set called the variables.*

- *A* valuation *is a function $\nu : X \to \{\top, \bot\}$. $\top$ refers to true and $\bot$ refers to false.*
- *A* formula *is defined inductively by a variable x is a formula, and if F and F' are formulas, $\neg F$, $F \vee F'$ and $F \wedge F'$ are formulas.*
- *A* valuation ν *satisfies a formula F if:*
  - *F is a variable x and $\nu(x) = \top$.*
  - *$F = \neg F'$ and ν does not satisfy F'.*
  - *$F = F_1 \vee F_2$ and ν satisfies F_1 or ν satisfies F_2.*
  - *$F = F_1 \wedge F_2$ and ν satisfies F_1 and ν satisfies F_2.*
- *A* literal *is a formula of the for x or $\neg x$.*
- *A* clause *is a formula that contains only $\vee$ and literals.*
- *A* Conjunctive Normal Form formula (or CNF) *is a formula that can be written only using $\wedge$ and clauses. It is said to be a k-CNF formula if all its clauses contains at most k literals.*

Problem 1.50 (3-SAT)**.**
Input: a 3-CNF Formula φ of the form $\varphi = \bigwedge_{1 \leq i \leq m} C_i$ where each C_i is a clause that contains at most three literals.
Output: $\top$ if and only if φ is satisfiable

Recall that it is equivalent to consider the decision problem asking whether the word φ belongs to the language of satisfiable formulas.

Theorem 1.51 (Cook [Coo71])**.** 3-SAT*is* NP*-complete.*

The NP-completeness of SAT has been the starting point of several reductions. Karp [Kar72] in 1972 proved, by reduction from SAT, that several classical problems are NP-complete.

The class PSPACE

In game theory, however, this is usually not easy to find a certificate that a player has a winning strategy. Indeed, if $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ is a hypergraph, a strategy for a player corresponds to a function $\mathcal{S} : 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to \mathcal{X}$, and therefore, it cannot necessarily be expressed with a polynomial certificate. Therefore, in general, the strategy cannot be written in polynomial size, and one must consider a larger class to handle all the possibilities. To handle the fact that the certificates are not polynomial, we usually need to consider the class PSPACE instead of NP, which bounds the space required to compute the function instead of the time.

Definition 1.52 (PSPACE)**.** *A language L belongs to the class* PSPACE*, if there exists a polynomial P and a computable function $f : \Sigma^* \to \{\top, \bot\}$ such that $f(x) = \top$ if and only if $x \in L$, and that f uses at most $P(|x|)$ memory space.*

Lemma 1.53 (Folklore)**.** *The class* NP *is a subclass of the class* PSPACE*.*

In PSPACE, the first central problem from which most of the reduction are made is QBF. It corresponds to Quantified Boolean Formulas. Instead of just considering a conjunctive normal form formula, quantifiers are allowed here. Without loss of generality, and up to add useless variables, QBF can be written as follows:

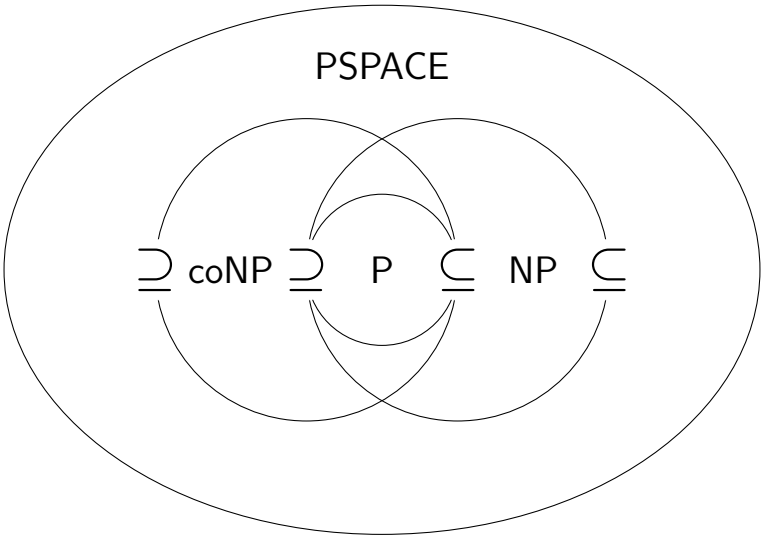


Figure 1.6: Inclusion between the different complexity classes

Problem 1.54 (QBF)**.**
Input: A quantified formula of the form $\psi = \exists x_1 \forall x_2 \dots \exists x_{2n-1} \forall x_{2n}\ \varphi(x_1, \dots, x_{2n})$ *where* φ *is a* 3*-CNF formula.*
Output: $\top$ *if the valuation provided by* $x_1, \dots, x_{2n}$ *satisfies* $\psi(x_1, \dots, x_{2n})$*.*

Theorem 1.55 (Stockmeyer and Meyer [SM73])**.** QBF *is* PSPACE*-complete.*

PSPACE is the natural complexity class to express strategies. Even if the function $\mathcal{S}$ cannot necessarily be expressed in polynomial space, it is possible, given two sets $\mathcal{X}_1$ and $\mathcal{X}_2$ to compute all the possible sequence of moves supposing that $\mathcal{X}_1$ and $\mathcal{X}_2$ are already claimed. Then, only returning as $\mathcal{S}(\mathcal{X}_1, \mathcal{X}_2)$ an optimal move is enough to compute the winner, even if the full description of $\mathcal{S}$ is not provided. Indeed, a strategy consists in choosing a move for each of the move of the opponent, which creates an alternation between $\forall$ and $\exists$ as it is in QBF. This alternation between $\exists$ and $\forall$ allows us to define QBF as a game played by two players, Satisfier and Falsifier, who alternately choose a valuation of the variables. Satisfier chooses the value of x_1, then Falsifier chooses the value of x_2, and so on until the last variable has its value chosen. At the end, a valuation ν of the variables is obtained, and Satisfier wins if and only if ν satisfies φ. This equivalent definition of the problem makes it easier to reduce QBF to games. A summary of the different complexity classes and their inclusion is provided in Figure 1.6

Parameterized complexity

Once problems have been characterized as hard (either NP or PSPACE-hard in general), two main steps are possible to still get positive results on these problems. Either simplify them, by restricting their inputs to some classes, or look for the reason why these problems are hard. We handle here the second point by considering parameterized complexity, i.e. we look at some parameters of the input, and observe what happens when this parameter is bounded.

The study of parameterized complexity was first introduced by Gurevich *et al.* [GSV84], who expressed the exponential part of the complexity of an algorithm as a function of some parameter k of the graph, instead of its size. This new study introduced different classes of complexity to better understand how hard problems are, and to find algorithms that can be efficient when some parameters are small, even if the problem is NP-hard.

Definition 1.56 (Parameterized problem)**.** *A parameterized problem is a question, that takes as input a language L such that $L \subset \Sigma^* \times \mathbb{N}$ and a word $(x, k) \in \Sigma^* \times \mathbb{N}$, and asks whether $(x, k) \in L$ or not. x is called the instance, and k the parameter.*

Definition 1.57 (FPT)**.** *A parameterized problem L is said to be* FPT*, if there exists a computable function f and a polynomial p such that for any $(x, k) \in \Sigma^* \times \mathbb{N}$, it can be decided in time $O(f(k) * p(|x|))$ whether $(x, k) \in L$.*

Definition 1.58 (XP)**.** *A parameterized problem L is said to be* XP*, if there exists a computable function f such that for any $x, k \in \Sigma^* \times \mathbb{N}$, it can be decided in time $O(|x|^{f(k)})$ whether $(x, k) \in L$.*

Note that we have FPT $\subset$ XP.

Both the classes FPT and XP have the property that if a parameterized problem is in FPT (resp. XP) and if k is bounded, then there exists a polynomial function to decide whether $(x, k) \in L$ or not. If this is the case, we say that the problem has an FPT (resp. XP) algorithm.

Usually, FPT results are provided through kernels.

Definition 1.59 (Kernelization)**.** *Let $L \subset \Sigma^* \times \mathbb{N}$ be a language. A* kernelization *is a function f that maps an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ to an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that:*

- *f can be computed in polynomial time in $|x|$.*
- *$(x, k) \in L$ if and only if $(x', k') \in L$*
- *$|x'|$ is bounded by a computable function g of k.*

If g is linear, quadratic, cubic, or polynomial, we say that L admits a linear, quadratic, cubic or polynomial kernel.

The most natural way to find kernels is through reduction rules. A *reduction rule* is a function that takes an instance (x, k) as input and returns an instance (x', k') with $|x'| < |x|$. Usually, kernels are obtained by successively applying of reduction rules, until the size of $|x'|$ is bounded by a function of k. Admitting a small kernel is a very important property in parameterized complexity.

Theorem 1.60 (Folklore)**.** *Let L be a parameterized problem. L is* FPT *if and only if L admits a kernel.*

However, not every parameterized problem has a kernel. In fact, the W-hierarchy characterizes different classes of complexity, considering the weft of the boolean circuit that recognizes the language. We refer the reader to Downey and Fellows [DF95, DF99] who introduced these classes of complexity. It is sufficient to know that unless if some complexity assumptions are added, we have FPT $= W[0] \subsetneq W[1] \subsetneq W[2] \subsetneq \cdots \subsetneq$ AW$[*]$. In particular, if a problem is $W[1]$-hard, it cannot be solved by an FPT algorithm.

When dealing with problem on graphs, a large panel of graph values have already been considered as parameters to deal with parameterized problem on graphs. An overview of graph problems parameterized by several graph parameters is provided in Jansen's dissertation [Jan13], and we will use some of these parameters in Chapter 4.

1.3.2 Complexity of positional games

Now that several complexity classes are defined, one can wonder in which complexity class are positional games. Hopefully, an argument of Schaefer [Sch78] answers this question.

Theorem 1.61 (Schaefer [Sch78])**.** *All the conventions of positional games are in* PSPACE*.*

Sketch of the proof. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Schaefer proved that if a game ends in a polynomial number of moves consisting in a polynomial number of options each, then the game is in PSPACE. In positional game, any move consists in claiming an unclaimed vertex. Therefore, the game end in at most $|\mathcal{X}|$. Moreover, a move consists in choosing one (two in Waiter-Client or Client-Waiter games) unclaimed vertex. Therefore, there is a polynomial number of options each time. To obtain an algorithm in running in polynomial space, it is enough to compute each possible games, and keeping in memory the better outcome for each. Thus, at most one game at a time is kept in memory and this method uses polynomial space. □

All the results presented here will be summarized in Table 1.1

Complexity of achievement games

Soon after the introduction of Maker-Breaker games, in 1978, Schaefer [Sch78] proved that many logic games are PSPACE-complete. Among them, he introduced POS CNF, which is defined as follows:

Definition 1.62 (POS CNF)**.** *Let φ be a CNF formula where no literal is negative. The* POS CNF *game is played by two players, namely Satisfier and Falsifier. Alternately, Satisfier (resp. Falsifier) chooses a variable whose truth value has not yet been set and sets it to $\top$ (resp. $\bot$). When all the variables have been set to a truth value, Satisfier wins if and only if the valuation ν corresponding to these moves satisfies φ.*

This game, even though it is not presented as a positional game, corresponds exactly to Maker-Breaker games. Indeed, let φ be a positive CNF formula over the variables $\{x_1, \dots, x_n\}$. Consider the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = \{x_1, \dots, x_n\}$, and every clause $C_j \in \varphi$ is a hyperedge containing the variables appearing in C_j. Consider that when Satisfier sets a variable to $\top$, Breaker claims the corresponding vertex, and when Falsifier sets a variable to $\bot$, Maker claims the corresponding vertex. If φ is satisfied, at least one variable in each clause is set to $\top$, corresponding to a vertex claimed by Breaker. Thus, all the hyperedges contains a vertex claimed by Breaker and Breaker wins. If it is not satisfied, there exists a clause C_j in which all the variables were set to $\bot$, corresponding to a hyperedge fully claimed by Maker, and Maker wins. Finally, Breaker wins if and only if Satisfier wins.

The reduction of Schaefer, from QBF to POS CNF contains only clauses of size at most 11. Therefore, he provided the following theorem:

Theorem 1.63 (Schaefer [Sch78])**.** *Deciding the outcome of a Maker-Breaker game is* PSPACE*-complete, even restricted to hypergraphs of rank* 11*.*

The proof of Theorem 1.63 was complicated, Byskov [Bys04] gave a simpler proof of the same result in 2004, but with no hypothesis on the size of the hyperedges. In the same paper, he provided a reduction from Maker-Breaker games to Maker-Maker games increasing their rank by only one, proving therefore that Maker-Maker games are PSPACE-complete, even restricted to hypergraphs of rank 12. The result of Schaefer was improved in 2021, when Rahman and Watson [RW21] proved that Maker-Breaker games are also PSPACE-complete restricted to hypergraphs of rank 6. Moreover, they proved that the hypergraph obtained in their reduction can be transformed into a 6-uniform hypergraph and the reduction remains polynomial.

Theorem 1.64 (Rahman and Watson [RW21])**.** *Deciding the outcome of a Maker-Breaker game is* PSPACE*-complete, even restricted to* 6*-uniform hypergraphs.*

Using again the argument provided by Byskov, this results improved the known bound on the complexity of Maker-Maker games.

Corollary 1.65 (Rahman and Watson [RW21], Byskov [Bys04])**.** *Deciding the outcome of a Maker-Maker game is* PSPACE*-complete, even restricted to* 7*-uniform hypergraphs.*

On the other hand, if the rank of the hypergraph is small enough, the outcome is not so difficult to compute. Considering that Maker starts, as it is often the case in Maker-Breaker games, if the hypergraph

has a hyperedge of size 1, Maker wins with her first move, and if the hypergraph is 2-uniform, Breaker wins if and only if all the hyperedges are disjoint. The case of rank 3 hypergraphs is much more complicated. In 2004, Kutz [KF04] gave a polynomial-time algorithm, for rank 3 linear hypergraphs, i.e., hypergraphs on which two hyperedges share at most one vertex. This result was improved, by Galliot *et al.* [Gal23] in 2023, who proved that the outcome can be computed in polynomial time on any hypergraph of rank 3.

Theorem 1.66 (Galliot *et al.* [Gal23])**.** *Deciding the outcome of a Maker-Breaker game played on a hypergraph of rank 3 is polynomial.*

The proof of Theorem 1.66 is a long case analysis, showing that, if Breaker is able to prevent some structure to appear in the hypergraph during his six first moves, he can prevent them to appear during all the game. Consequently, if $\mathcal{H}$ has outcome $\mathcal{M}$, after at most six moves, Maker has either win or created a structure in the hypergraph that ensures her to win.

Theorem 1.66, together with Theorem 1.64 let only the complexity of hypergraphs of rank 4 and 5 open. As most of the gadgets used by Rahman and Watson are hyperedges of order four or five, it seems likely that Maker-Breaker games restricted to hypergraphs of rank 4 are also PSPACE-complete.

Conjecture 1.67. *Deciding the outcome of Maker-Breaker games is* PSPACE*-complete, even restricted to hypergraphs of rank* 4*.*

Even if there was no result between 1978 and 2021 on general Maker-Breaker games, during this time, several particular Maker-Breaker games were proved to be PSPACE-complete. For instance Reisch [Rei81] proved in 1981 that, Generalized HEX, in which Maker aims to connect two vertices of a graph, and Breaker tries to prevent him is PSPACE-complete on planar graphs, and Duchêne *et al.* [DGPR20] proved that the domination game is PSPACE-complete on split and bipartite graphs. On the positive side, only few complexity results are known. For instance, Lehman [Leh64] proved that the winner of the Maker-Breaker connectivity game can be determined in polynomial time, and Duchêne *et al.* [DGPR20] proved that the Maker-Breaker domination game can be solved in polynomial time in forests and cographs. In Chapter 3, we prove that the winner of the Maker-Breaker H-game can be computed in polynomial time in forests when H is a star, and the winner of the Maker-Breaker perfect matching game can be computed in polynomial time in grids.

Concerning Maker-Maker games, only few complexity results are known. Even for 3-uniform Maker-Maker games, it is still unknown if computing the winner can be done in polynomial time or not. In Chapter 5, we focus on the Maker-Maker domination game and provide a polynomial time algorithm to compute the winner of this game in forests.

Complexity of avoidance games

Although avoidance games and achievement games have appeared at about the same time, the PSPACE-hardness of deciding the winner of Maker-Breaker games came much earlier than the same problem related to avoidance games. The first complexity result obtained on general Avoider-Avoider games was that it is PSPACE-complete to determine the winner of a given position, with some played vertices, by Slany [Sla02] in 2002. Next, Burke and Hearn [BH19] in 2019 have improved this results, proving that computing the winner of an Avoider-Avoider game was PSPACE-complete by providing a proof for 2-uniform hypergraphs in the game COL, which consists in claiming the vertices of a graph, and whenever a player claims the two extremity of an edge, he or she loses. Finally, no other result appeared, until 2022 when Miltzow and Stojakovic [MS22] have proved that computing the winner of Avoider-Enforcer games is NP-hard. With Valentin Gledel [GO23], we close this open question by proving that computing the winner of an Avoider-Enforcer game is PSPACE-complete, even restricted to 6-uniform hypergraphs, see Chapter 2, Theorem 2.1.

On the positive side, only rank 2 hypergraphs are known to be solvable in polynomial time. A proof of this result will be provided in an ongoing work of Galliot *et al.*.

Theorem 1.68 (Galliot *et al.* [GGGP])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a rank* 2 *hypergraph. The winner of an Avoider-Enforcer game played on $\mathcal{H}$ can be computed in polynomial time.*

The main idea of the proof, is to focus on the structures that can make Avoider loses in her last moves and removing vertices that will be used only to pass turns for the players.

Complexity of Client-Waiter games

Even though Client-Waiter and Waiter-Client games were introduced in 2002, their complexity was quickly studied, and the first hardness results appeared in 2011, when Csernenszky *et al.* [CMP11] proved the NP-hardness of both conventions. Note, however, that the construction used for Waiter-Client provides a hypergraph with an exponential number of hyperedges, so the construction is polynomial only if we allow an implicit description for the hyperedges.

The complexity of hypergraphs of small rank has not yet been studied considering these conventions, however, we present here some results obtained with Valentin Gledel and Sebastien Tavenas, proving that hypergraphs of rank 2 can be solved in polynomial time.

Theorem 1.69 (Gledel, O., Tavenas [GOT])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a rank 2 hypergraph. The winner of a Client-Waiter game played on $\mathcal{H}$ can be computed in polynomial time.*

Proof. We prove that Client wins on $\mathcal{H}$ if and only if there exists a hyperedge of order 1, or if two winning sets intersect.

- If $\mathcal{H}$ has a winning set of order 1, $\{x\}$, when Waiter proposes the vertex x, Client can claim it and win. As the last vertex goes to Client if the number of vertices if odd, if x is never proposed, Client still wins.
- If there exists two winning sets $\{a, b\}$ and $\{a, c\}$ which intersect, Client applies the following strategy:
  - Whenever Waiter proposes two of these vertices, Client claims one of them, and he takes a if it is proposed.
  - If Waiter proposes one of these vertices with any other, Client claims the one in $\{a, b, c\}$.
  - Otherwise, Client takes any vertex.

  Following this strategy, Client will claim at least two vertices from $\{a, b, c\}$, and will have claimed a. Therefore, he will win. Once again, even if the number of vertices is odd, this strategy can be applied.

Reciprocally, if all the hyperedges have size 2 and do not intersect, by always proposing a pair (a, b) such that (a, b) is a hyperedge, Waiter will get one vertex in each of the winning sets and will win. □

Theorem 1.70 (Gledel, O., Tavenas [GOT])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a rank 2 hypergraph. The winner of a Waiter-Client game played on $\mathcal{H}$ can be computed in polynomial time.*

Proof. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a rank 2 hypergraph. Up to removing some hyperedges, we can assume that there are no two hyperedges e and e' such that $e \subset e'$. If this were the case, the hyperedge e' could be removed.

We prove that if Waiter has a winning strategy in the Waiter-Client game played on $\mathcal{H}$, she has a winning strategy in three moves, which gives a polynomial algorithm by exhaustive search on the first three moves. First, note that if (x, y) is a hyperedge such that at least one of its vertex, say x, is in no other hyperedge, proposing (x, y) can be ignored from any optimal strategy for Waiter, as Client will claim y and the resulting hypergraph is a subhypergraph of $\mathcal{H}$. Thus, by Lemma 1.19, it cannot benefit Waiter.

This result is obtained by case analysis on the number of hyperedges of size 1 and the structure of the hyperedges of size 2:

1. If $\mathcal{H}$ has two hyperedges of order 1, Waiter wins in a single move by proposing the two vertices in those two hyperedges.
2. If $\mathcal{H}$ has exactly one hyperedge of order 1, and at least two other hyperedges, Waiter wins in two moves: Let x be the vertex alone in its hyperedge, and let $e_1 = (u_1, v_1)$, and $e_2 = (u_2, v_2)$ be two hyperedges that does not intersect $\{x\}$.

- If $e_1 \cap e_2 = \emptyset$, Waiter first propose $\{u_1, u_2\}$. By symmetry, assume that Client chooses u_2. Now Waiter proposes $\{v_1, x\}$ and wins.
- If $e_1 \cap e_2 \neq \emptyset$, say for instance $u_1 = u_2$, then Waiter proposes the two pairs $\{u_1, x\}$ and $\{v_1, v_2\}$ and wins.

3. If $\mathcal{H}$ has exactly one hyperedge of order 1 and at most one hyperedge of order 2, then Client wins. If (x) is a hyperedge and (y, z) is another (if it exists), Client can always claim x and one of y or z and win.

4. If $\mathcal{H}$ has no hyperedge of order 1 and at most three hyperedges of order 2, Client wins. After the first move of Waiter, Clients claims a vertex of maximum degree.

5. Otherwise, all the hyperedges have size 2. Let $\{x, y\}$ be the first move of Waiter. Suppose, without loss of generality, that both x and y are in at least one hyperedge different from (x, y). If such a move does not exist, $\mathcal{H}$ contains only one hyperedge (x, y), and Client wins by claiming any vertex of it. Suppose, without loss of generality, that Client chooses x. Since, y was in at least one hyperedge different from (x, y), say (y, z), there is now a hyperedge with only one vertex left z. Thus, we are back at one of the previous points. Therefore, if at least one pair of vertices leads Waiter to the case (1) or to the case (2), Waiter wins in at most three moves, otherwise, Client wins. □

Finally, we proved that computing the winner of a Client-Waiter game is PSPACE-complete, even restricted to hypergraphs of rank 6, see Chapter 2, Theorem 2.10.

With these results, the complexity of positional games let two directions for further studies: determining the complexity of Waiter-Client games, which are only known to be in PSPACE, or closing the different gaps for the other conventions, as except for Avoider-Avoider games, no complexity results are known for hypergraphs of rank 4 and hypergraphs of rank 5.

Rank	2	3	4, 5	6	7+
Maker-Breaker	P [Folklore]	P [Gal23]	Open	PSPACE-c [RW21]	PSPACE-c [RW21]
Maker-Maker	P [Folklore]	Open	Open	Open	PSPACE-c [RW21, Bys04]
Avoider-Avoider	PSPACE-c [BH19]	PSPACE-c [BH19]	PSPACE-c [BH19]	PSPACE-c [BH19]	PSPACE-c [BH19]
Avoider-Enforcer	P [GGGP]	Open	Open	PSPACE-c Thm 2.1	PSPACE-c Thm 2.1
Client-Waiter	P Thm 1.69	Open	Open	PSPACE-c Thm 2.10	PSPACE-c Thm 2.10
Waiter-Client	P Thm 1.70	Open	Open	Open	Open

Table 1.1: The complexity of the different conventions

As the hardness of Client-Waiter and Avoider-Enforcer games are very recent, only few results are already provided concerning the PSPACE-hardness of particular games under these conventions. In Chapter 2, we will prove that both Avoider-Enforcer and Waiter-Client domination games are PSPACE-complete, using the hardness of Avoider-Enforcer and Client-Waiter games respectively.

Parameterized complexity

As presented previously, most of the conventions of positional games are PSPACE-hard, even if the hypergraphs considered are of small rank. Therefore, it seems natural to tackle them under the paradigm

of parameterized complexity. A natural parameter for games is the number of moves. Downey and Fellows [DF99] introduced the number of moves as a parameter to handle games. In positional games, we say that a player has a winning strategy in k moves if he has satisfied his winning conditions after his k first moves. Namely, we have the following conditions:

- In Maker-Breaker games, Maker wins in k moves if she can fill up a hyperedge during her k first moves.
- In Maker-Breaker games, Breaker wins in k moves if he can fill up a transversal of the hypergraph during his k first moves.
- In Avoider-Enforcer games, Enforcer wins in k if he can force Avoider to fill up a hyperedge during her k first moves.
- In Avoider-Enforcer games, Avoider wins in k moves if she can force Enforcer to claim a transversal during his k first moves.

Downey and Fellows [DF99] conjectured that HEX, which we remind the reader can be modeled as a Maker-Breaker game, is AW[*]-hard. This problem has remained open for several years until Bonnet *et al.* [BGL$^+$17] proved that it is only W[1]-complete parameterized by the number of moves required for Maker to win, which proves, unless some complexity assumptions are made, that it is not AW[*]-hard. In this paper, they proved several meta-theorems about the parameterized complexity of positional games.

Theorem 1.71 (Bonnet *et al.* [BGL$^+$17])**.** *Determining whether Maker can fill up a hyperedge in k moves in a Maker-Breaker game is W[1]-complete, parameterized by k.*

Theorem 1.72 (Bonnet *et al.* [BGL$^+$17])**.** *Determining whether Enforcer can force Avoider to fill up a hyperedge in k moves in an Avoider-Enforcer game is co-W[1]-complete, parameterized by k.*

Theorem 1.73 (Bonnet *et al.* [BGL$^+$17])**.** *Determining whether Alice can fill up a hyperedge before Bob and during her k moves in a Maker-Maker game is AW[*]-complete, parameterized by k.*

This complexity gap between the Maker-Breaker and the Maker-Maker conventions can be explained by the fact that, in Maker-Maker games, the players have two goals, to fill up a hyperedge and to prevent their opponent from filling up one. This supports the idea that Maker-Maker games are harder to solve than Maker-Breaker games.

Despite these hardness results, Bonnet *et al.* [BJS16], together with an argument by Frick and Grohe [FG01], also provided an FPT algorithm to solve Hex parameterized by the number of moves. Indeed, Hex being played on a planar graph, it has locally bounded treewidth, and as the winning conditions can be expressed in first order formula, a result of Frick and Grohe [FG01] states that it can be verified in FPTtime parameterized by the number of variables, which corresponds here to the number of moves. More generally, they proved that any game on a graph G whose winning conditions can be expressed in first order logic is FPT parameterized by the treewidth and the number of moves Maker needs to win.

Even if the number of moves seems natural and inherent to games, as we are mostly studying games on graphs, several graph parameters could be used, as they are directly related to the game we will consider. For instance, as most games are easily solvable on trees, some distance to tree can be used.

In Chapter 4, we provide FPT algorithms for the Maker-Breaker Domination game for several graph parameters, and we prove that this game is W[2]-hard parameterized by the number of moves required by Dominator to win.

1.4 Tools in positional games

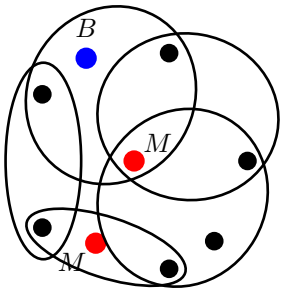
In this section, we will introduce several tools that are often used in positional games, either to provide reductions between games or to compute strategies.

1.4.1 Equivalences between hypergraphs

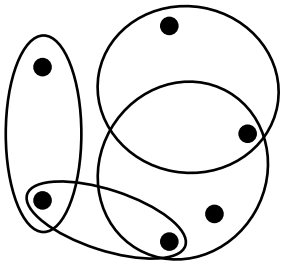
One problem with considering positional games, is that some results can only be applied when the board of the game is empty, i.e. before any move has been made. As soon as positions with already claimed vertices are considered, some results cannot be applied so easily. Hopefully, in weak games, it is possible to reduce the hypergraph after these moves to consider only hypergraphs where no vertex has been claimed.

Lemma 1.74 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Let $\mathcal{X}_1$, $\mathcal{X}_2$ be set of vertices such that $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. Let $\mathcal{X}' = \mathcal{X} \setminus (\mathcal{X}_1 \cup \mathcal{X}_2)$ and $\mathcal{F}' = \{e \setminus \mathcal{X}_1 | e \in \mathcal{F} \text{ and } e \cap \mathcal{X}_2 = \emptyset\}$. Let $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$. In Maker-Breaker, Avoider-Enforcer, Client-Waiter and Waiter-Client games, the position $\mathcal{P} = (\mathcal{H}, \mathcal{X}_1, \mathcal{X}_2)$ and $\mathcal{P}' = (\mathcal{H}', \emptyset, \emptyset)$ have the same outcome.*

The main idea using this lemma is that, as we only consider the hyperedges filled up by Maker for instance, whenever Breaker claims a vertex x, we can remove all the hyperedges that contains x, as we know that Maker cannot fill up one of them. Reciprocally, when Maker plays a vertex x, it can be removed from all the hyperedges that contains it as, to fill up this hyperedge, she no longer needs to play x, see Figure 1.7. Note that if there is an empty hyperedge, Maker has automatically won, as by definition, she has claimed all the vertices of this hyperedge.



(a) A general position on a hypergraph.



(b) The resulting hypergraph after application of Lemma 1.74.

Figure 1.7: Application of Lemma 1.74. Vertices claimed by Maker are removed. The vertex claimed by Breaker is removed with the hyperedge that contains it.

The proof of this lemma is straightforward and comes from the following observation:

Observation 1.75. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ and $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$ be two hypergraphs. Let $\mathcal{P} = (\mathcal{H}, \mathcal{X}_1, \mathcal{X}_2)$ and $\mathcal{P}' = (\mathcal{H}', \mathcal{X}_1', \mathcal{X}_2')$ be positions. Suppose that there is a bijection $f : \mathcal{X} \setminus (\mathcal{X}_1 \cup \mathcal{X}_2) \to \mathcal{X}' \setminus (\mathcal{X}_1' \cup \mathcal{X}_2')$ such that for any set $S \subset \mathcal{X} \setminus (\mathcal{X}_1 \cup \mathcal{X}_2)$ of unclaimed vertices, S contains a winning set for a player in $\mathcal{H}$ if and only if $f(S)$ contains a winning set for the same player in $\mathcal{H}'$. Then, $\mathcal{P}$ and $\mathcal{P}'$ have the same outcome.*

This reduction, together with Lemma 1.19 makes it possible to remove several vertices from the game and work on smaller hypergraphs. When this happens, it is often the case that the considered hypergraph is no longer connected. Thus, we consider tools to solve positional games that can be represented as union of games.

1.4.2 Union of hypergraphs

While playing on a hypergraph with multiple connected components, the game behaves as a union of games, in the combinatorial game sense of the union: the moves on a component will not cause any change in the other ones. Therefore, even if in most of the studies, the player who goes first is determined by the rules, usually Maker or Avoider, to handle unions, we need to study games with the other player going first. By playing first in a component, a player makes it possible for their opponent to play first in another one. Generally, the outcome is not the same in both cases. We remind the reader here that, in achievement games,

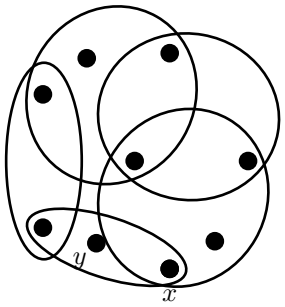


Figure 1.8: In this hypergraph x dominates y as any hyperedge containing y also contains x.

i.e., Maker-Maker and Maker-Breaker, according to Lemma 1.15, the players always want to play, and the outcome $\mathcal{P}$ does not exist. We also remind the reader that in Avoider-Enforcer games, by Lemma 1.16, what really matters is the player going second to last and not the player who starts.

We can now focus on the outcome of a union of games, in function of the outcome of its components. In Maker-Breaker games, this result is well known from the folklore, while in Avoider-Enforcer games, it is due to Galliot *et al.* [GGGP]. Both tables are provided in Table 1.2. These tables directly provides the following lemma:

Lemma 1.76 (Folklore)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph such that $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2$. Maker wins going first in $\mathcal{H}$ if and only if she wins going first in $\mathcal{H}_1$ or in $\mathcal{H}_2$.*

This lemma is very strong and makes it possible to handle unions very easily in Maker-Breaker games. However, no results are known in the Maker-Maker convention. A consequence of that is, for example, that even if cographs can easily be solved in polynomial time in the Maker-Breaker Domination game, they are really hard to handle in the Maker-Maker convention of that game.

$\cup$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{B}$
$\mathcal{M}$	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{M}$
$\mathcal{N}$	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{N}$
$\mathcal{B}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{B}$

(a) Outcome of the union of two Maker-Breaker games.

$\cup$	$\mathcal{E}$	$\mathcal{SL}$	$\mathcal{A}$
$\mathcal{E}$	$\mathcal{E}$	$\mathcal{E}$	$\mathcal{E}$
$\mathcal{SL}$	$\mathcal{E}$	$\mathcal{E}$	$\mathcal{SL}$
$\mathcal{A}$	$\mathcal{E}$	$\mathcal{SL}$	$\mathcal{A}$

(b) Outcome of the union of two Avoider-Enforcer games.

Table 1.2: Outcome of an union of two weak games

1.4.3 Dominated moves

Another phenomenon that can occur when computing a winning strategy, is that some moves are better than others. Therefore, we present here the notion of *dominated moves*, to show that under certain circumstances, we can assume without loss of generality that some moves are not played.

Definition 1.77 (Dominated move)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Let $x, y \in \mathcal{X}$ be unclaimed vertices in $\mathcal{H}$. We say that y* dominates *x in $\mathcal{H}$, or x is* dominated by *y in $\mathcal{H}$, if whenever a player has a winning strategy starting by claiming x, he has one starting by claiming y, see Figure 1.8.*

Considering this definition of dominated moves, we can always assume that, a player will never play a dominated move. In particular, we have the following result which belongs to the folklore of Maker-Breaker and Maker-Maker games, and which was proved by Miltzow and Stojaković [MS22] in Avoider-Enforcer games. Although it is not stated, the proof they gave also works in the Avoider-Avoider convention.

Lemma 1.78 (Folklore, Miltzow and Stojaković [MS22]). *Let $\mathcal{H}$ be a hypergraph. Let $x, y \in \mathcal{X}$ such that $\{e \in \mathcal{F} | x \in e\} \subseteq \{e \in \mathcal{F} | y \in e\}$. Then y dominates x in Maker-Breaker and Maker-Maker games, and x dominates y in Avoider-Enforcer and Avoider-Avoider games.*

Client-Waiter games also have a similar type of dominated moves. However, because of the different structure of Client-Waiter games, the most important part of dominated moves focuses on Waiter's choices. Indeed, when Client has to choose, its optimal moves, among the two possible vertices, are the same as the one of Maker in the Maker-Breaker convention.

Lemma 1.79 (Folklore). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Let $e \in \mathcal{F}$ such that $e = (x, y)$. If Waiter has a winning strategy in the Client-Waiter game on $\mathcal{H}$, he has one starting by proposing the pair $\{x, y\}$.*

The intuition behind this lemma is that Waiter cannot propose x and y separately, otherwise Client can take both and win. Since she has to propose them together, it is better for her to propose them first to know which vertex will be chosen by Client.

Proof. We prove this result by contraposition. Suppose that Client has a winning strategy $\mathcal{S}$ when Waiter proposes $\{x, y\}$ first. Suppose, without loss of generality that Client chooses x for his first move, and consider the following strategy for Client:

- If Waiter proposes the pair $\{x, y\}$, Client chooses x.
- If Waiter proposes a pair that contains neither x nor y, Client chooses following $\mathcal{S}$, supposing he has already claimed x and Waiter y.
- If Waiter proposes a pair containing exactly one of x or y, Client chooses it.

Following this strategy, if Waiter at some point proposes the pair $\{x, y\}$, everything happens for Client as if Waiter had proposed it first. Therefore, Client will manage to claim a winning set, since $\mathcal{S}$ is a winning strategy for Client. Otherwise, Client will claim x and y, which makes him win, since it is a winning set.

So whenever Client has a winning strategy when (x, y) is proposed first, he has one when it is not. Therefore, by contraposition, whenever Waiter has a winning strategy, she has one that proposes (x, y) first. □

1.4.4 Pairing strategies

A classic type of strategy in Maker-Breaker games is pairing strategies. A *pairing strategy* consists of pairing vertices two by two and making sure to claim at least one vertex in each pair, usually by playing in the same pair as the opponent whenever he plays in a pair. While this type of strategy is not necessarily a winning strategy, it can characterize some instances of the game where a winner can be determined in polynomial time.

Lemma 1.80 (Folklore). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph in a Maker-Breaker game. Let $(a_1, b_1), \ldots, (a_n, b_n)$ be pairwise disjoint pairs of vertices. Maker (resp. Breaker) has a strategy that claims at least one vertex in each pair (a_i, b_i). Such a strategy is called a* pairing strategy*.*

Corollary 1.81 (Folklore). *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Let $(a_1, b_1), \ldots, (a_n, b_n)$ be pairwise disjoint pairs of vertices such that any hyperedge $f \in \mathcal{F}$ contains a pair (a_i, b_i). Then Breaker wins the Maker-Breaker game on $\mathcal{H}$.*

This result can also be applied after some moves have been played. For instance, In Tic-Tac-Toe, Breaker going first can consider the pairing presented in Figure 1.9 to claim at least one vertex in each hyperedge and win.

Pairing strategies has been used several times to obtain results in Maker-Breaker games. As an application, in the Maker-Breaker domination game, Duchêne *et al.* [DGPR20] introduced *pairing dominating sets*, i.e. sets of pairs such that each vertex of the graph is dominated by the two vertices of some pair. They

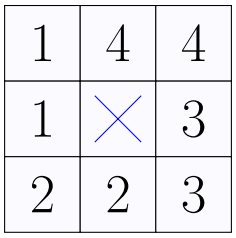


Figure 1.9: A pairing strategy for Breaker going first in Tic-Tac-Toe. The cross in the middle is his first move, vertices numbered with the same digit are paired together.

proved that the existence of a pairing dominating set ensures a win for Dominator, and that, in trees and cographs, Dominator wins going second if and only if the graph admits a pairing dominating set. More details about these results will be provided in Chapter 4.

Although pairing strategies are a classic type of strategy in achievement games, they have not been used in avoidance games. Since in avoidance games, ensuring to play some vertex cannot provide a winning strategy, pairing strategies have to be modified to be applied here. Therefore, we introduced *pairing strategies* in Avoider-Enforcer games [GO23] which force the opponent to play at least once in each pair.

Lemma 1.82 (Gledel and O. [GO23])**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Suppose that Avoider (resp. Enforcer) plays last in $\mathcal{H}$. Let $(a_1, b_1), \ldots, (a_n, b_n)$ be pairwise disjoint pairs of vertices, and let $v \notin \bigcup_{i=1}^{n} \{a_i, b_i\}$.*

Avoider (resp. Enforcer) has a strategy which forces Enforcer (resp. Avoider) to claim at least one vertex in each pair (a_i, b_i). Enforcer (resp. Avoider) has a strategy which forces Avoider (resp. Enforcer) to claim v and at least one vertex in each pair (a_i, b_i).

Proof. By symmetry of the result, we do the proof when Avoider claims the last vertex. Consider the following strategy for Avoider:

- If Enforcer claims a vertex in a pair (a_i, b_i), Avoider claims the other vertex of the pair.
- Otherwise, Avoider claims any vertex that is not in a pair.

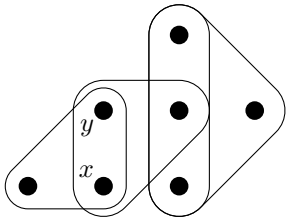
By construction, when it is Enforcer's turn, in any pair in which he has played, Avoider has also played. Therefore, when it is Avoider's turn, there is at most one pair of vertices in which he has to play. As Avoider plays the last move, whenever it is his turn to play, the number of remaining vertices is odd. Therefore, if Enforcer does not play in a pair, at least one vertex in no pair will be available for Avoider. Thus, Avoider has a strategy to force Enforcer to claim at least one vertex in each pair (a_i, b_i).

Now, consider the following strategy for Enforcer:

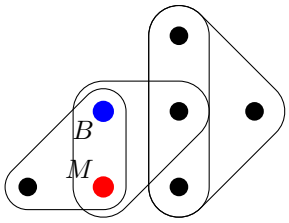
- If Avoider claims a vertex in a pair (a_i, b_i), Enforcer claims the other vertex of the pair.
- If Avoider claims v, if there exists at least one pair (a, b) in which Avoider has not played, Enforcer claims a and considers now that b is the new vertex that Avoider will be forced to claim instead of v.
- Otherwise, Enforcer claims any vertex that is not in a pair nor v.

For the same reason, with this strategy, when it is Avoider's turn, in any pair in which he has played, Enforcer has played too. When it is Enforcer's turn, note that the number of remaining moves is even, and there always exists exactly one vertex that is not in a pair, and that Enforcer wants Avoider to claim. Thus, the number of vertices on which Enforcer cannot play before Avoider is odd. Therefore, Enforcer always has an available move that fulfills this strategy. □

Corollary 1.81 can also be applied in the Avoider-Enforcer convention, as follows:



(a) Example of a hypergraph. Here x and y satisfy the hypothesis of the Super Lemma.



(b) We apply the Super Lemma, and add x to the vertices claimed by Maker and y to the vertices claimed by Breaker.

Figure 1.10: Application of the Super Lemma

Corollary 1.83. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. Let $(a_1, b_1), \dots, (a_n, b_n)$ be pairwise disjoint pairs of vertices such that any hyperedge $f \in \mathcal{F}$ contains a pair (a_i, b_i). Then Avoider wins the Avoider-Enforcer game on $\mathcal{H}$.*

Pairing strategies in Avoider-Enforcer games will be an important argument to prove that computing the winner of an Avoider-Enforcer game is PSPACE-complete in Chapter 2.

1.4.5 Symmetries and Super Lemma

In games on graphs, it is often the case that symmetries appear. For example, in Arc Kayles [BDO24], Influence [DOP24] or the Largest connected subgraph game [BFMIN22], it was shown that some symmetric strategies can be used to compute the outcome of games with some symmetric properties. However, these properties are very restrictive, and sometimes, the board has no global symmetries but only local ones. The next lemma, that we first introduced in [BDD$^+$23], answers the question of how to handle local symmetries. This lemma can be used in several games, see [DGI$^+$23, BHOP], and therefore, it has been given the name of *Super Lemma*.

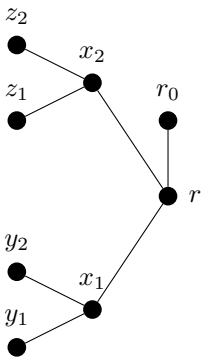
Lemma 1.84 (Super Lemma, Maker-Breaker version)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let V_M (resp. V_B) be the vertices already claimed by Maker (resp. Breaker). Let $x, y \in \mathcal{X} \setminus (V_M \cup V_B)$ be two unclaimed vertices such that for any $S \subset 2^{\mathcal{X} \setminus \{x,y\}}$, we have $S \cup \{x\} \in \mathcal{F}$ if and only if $S \cup \{y\} \in \mathcal{F}$. Then $(\mathcal{H}, V_M, V_B)$ and $(\mathcal{H}, V_M \cup \{x\}, V_B \cup \{y\})$ have the same outcome.*

The main idea behind the Super Lemma is that, if two vertices play the same role, when a player will want to claim one of them, his opponent will want to claim the second one, and therefore, at the end, each player will have claimed one, see Figure 1.10.

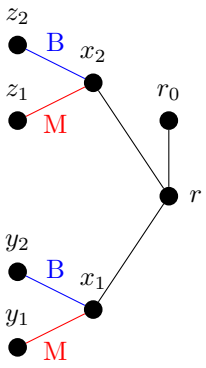
Proof. Suppose that Maker has a winning strategy $\mathcal{S}$ in $(\mathcal{H}, V_M \cup \{x\}, V_B \cup \{y\})$. We provide a winning strategy for her in $(\mathcal{H}, V_M, V_B)$ as follows:

- While Breaker does not claim a vertex in $\{x, y\}$, Maker claims the same vertex as she would have claim according to $\mathcal{S}$.
- If Breaker claims a vertex in $\{x, y\}$, Maker answers by claiming the other vertex in $\{x, y\}$.
- If only $\{x, y\}$ are remaining, Maker claims x.

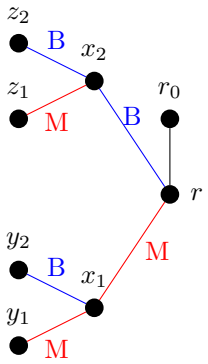
Following this strategy, as $\mathcal{S}$ was a winning strategy in $(\mathcal{H}, V_M \cup \{x\}, V_B \cup \{y\})$, there exists a hyperedge $e \in \mathcal{F}$ in which all the vertices are claimed by Maker. If $x \notin e$, Maker has also claimed all the vertices of e in $(\mathcal{H}, V_M, V_B)$. If $x \in e$, Maker has claimed all the vertices of $S = e \setminus \{x\}$, and one vertex among $\{x, y\}$. By hypothesis, as $e = S \cup \{x\} \in \mathcal{F}$, we have $e' = S \cup \{y\} \in \mathcal{F}$. Therefore, Maker has claimed either all the vertices of e or all the vertices of e' and has won.



(a) Apply the Super Lemma to $\{(z_2, x_2), (z_1, x_2)\}$ and $\{(y_2, x_1), (y_1, x_1)\}$



(b) We can now apply the Super Lemma to the pair of edges $\{(r, x_2), (r, x_2)\}$.



(c) The resulting graph after application of the Super Lemma.

Figure 1.11: Application of the Super Lemma to solve the P_4-game in a complete binary tree. Labelled M edges (in red) are claimed by Maker and labelled B edges (in blue) are claimed by Breaker. It is now easy to see that the outcome of the game is $\mathcal{N}$

Suppose now that Breaker has a winning strategy $\mathcal{S}$ in $P' = (\mathcal{H}, V_M \cup \{x\}, V_B \cup \{y\})$. Breaker follows the following strategy in $P = (\mathcal{H}, V_M, V_B)$:

- While Maker does not claim a vertex in $\{x, y\}$, Breaker claims the same vertex as he would have claim according to $\mathcal{S}$.
- If Maker claims a vertex in $\{x, y\}$, Breaker answers by claiming the other vertex in $\{x, y\}$.
- If only $\{x, y\}$ are remaining, Breaker claims y.

Following this strategy until the end, by hypothesis in P', for any hyperedge $e \in \mathcal{F}$, there is a vertex claimed by Breaker. Let e be a hyperedge after the game played in P. If $\{x, y\} \subset e$, then Breaker has played in e, as he has played one vertex among x and y. If $y \notin e$, as there is a vertex $z \in e$ that Breaker has claimed following $\mathcal{S}$ in P', he also has claimed z in P as $z \notin \{x, y\}$. If $y \in e$, consider $S = e \setminus \{y\}$. By hypothesis, $S \cup \{x\} \in \mathcal{F}$. Therefore, in P', there exists a vertex $z \in S$ that Breaker has claimed (as $z \in S \cup \{x\}$ and Maker has claimed x). Therefore, $z \in e$, and once again, Breaker has claimed a vertex in e. Finally, for any $e \in \mathcal{F}$, there exists $z \in e$ claimed by Breaker, so Breaker has won. □

The Super Lemma is very general. Moreover, using also Lemma 1.74, it can be applied even with vertices that do not satisfy the hypothesis of the lemma at the beginning of the game, but which will satisfy them after other vertices have been played. By applying twice the Super Lemma in the tree provided in Figure 1.11, we see that the P_4-game on this tree has outcome $\mathcal{N}$, as on the resulting graph, only the edge (r, r_0) is unclaimed.

This lemma will be used in almost all the chapters of this thesis to simplify some graphs, and will be the main tool in Chapter 4 to prove that the Maker-Breaker domination game is FPT parameterized by the modular-width of the input graph.

Finally, the proof provided for the Maker-Breaker version of the Super Lemma consists mostly in pairing x and y and applying the same strategy in the rest of the graph. Therefore, the same proof can be applied in the Avoider-Enforcer convention, which leads to the following lemma.

Lemma 1.85 (Super Lemma, Avoider-Enforcer version)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and let V_A (resp. V_E)) be the vertices already claimed by Avoider (resp. Enforcer). Let $x, y \in \mathcal{X} \setminus (V_A \cup V_E)$ be two unclaimed vertices such that for any $S \subset 2^{\mathcal{X} \setminus \{x,y\}}$, we have $S \cup \{x\} \in \mathcal{F}$ if and only if $S \cup \{y\} \in \mathcal{F}$. Then $(\mathcal{H}, V_A, V_E)$ and $(\mathcal{H}, V_A \cup \{x\}, V_E \cup \{y\})$ have the same outcome.*

Proof. Suppose that Avoider has a winning strategy $\mathcal{S}$ in $(\mathcal{H}, V_A \cup \{x\}, V_E \cup \{y\})$. We provide a winning strategy for her in $(\mathcal{H}, V_A, V_E)$ as follows:

- While Enforcer does not claim a vertex in $\{x, y\}$, Avoider claims the same vertex as she would have claim according to $\mathcal{S}$.
- If Enforcer claims a vertex in $\{x, y\}$, Avoider answers by claiming the other vertex in $\{x, y\}$.
- If only $\{x, y\}$ are remaining, Avoider claims x.

Following this strategy until the end, by hypothesis in P', for any hyperedge $e \in \mathcal{F}$, there is a vertex claimed by Enforcer. Let e be a hyperedge after the game played in P. If $\{x, y\} \subset e$, then Enforcer has played in e, as he has played one vertex among x and y. If $y \notin e$, as there is a vertex $z \in e$ that Enforcer has claimed following $\mathcal{S}$ in P', he also has claimed z in P as $z \notin \{x, y\}$. If $y \in e$, consider $S = e \setminus \{y\}$. By hypothesis, $S \cup \{x\} \in \mathcal{F}$. Therefore, in P', there exists a vertex $z \in S$ that Enforcer has claimed (as $z \in S \cup \{x\}$ and Avoider has claimed x). Therefore, $z \in e$, and once again, Enforcer has claimed a vertex in e. Finally, for any $e \in \mathcal{F}$, there exists $z \in e$ claimed by Enforcer, so Avoider has won.

Suppose now that Enforcer has a winning strategy $\mathcal{S}$ in $P' = (\mathcal{H}, V_A \cup \{x\}, V_E \cup \{y\})$. Enforcer follows the following strategy in $P = (\mathcal{H}, V_A, V_E)$:

- While Avoider does not claim a vertex in $\{x, y\}$, Enforcer claims the same vertex as he would have claim according to $\mathcal{S}$.
- If Avoider claims a vertex in $\{x, y\}$, Enforcer answers by claiming the other vertex in $\{x, y\}$.
- If only $\{x, y\}$ are remaining, Enforcer claims y.

Following this strategy, as $\mathcal{S}$ was a winning strategy in $(\mathcal{H}, V_A \cup \{x\}, V_E \cup \{y\})$, there exists a hyperedge $e \in \mathcal{F}$ in which all the vertices are claimed by Avoider. If $x \notin e$, Avoider has also claimed all the vertices of e in $(\mathcal{H}, V_A, V_E)$. If $x \in e$, Avoider has claimed all the vertices of $S = e \setminus \{x\}$, and one vertex among $\{x, y\}$. By hypothesis, as $e = S \cup \{x\} \in \mathcal{F}$, we have $e' = S \cup \{y\} \in \mathcal{F}$. Therefore, Avoider has claimed either all the vertices of e or all the vertices of e' and Enforcer has won. □

However, the Super Lemma cannot in general be applied in the Maker-Maker or the Avoider-Avoider convention. For example, in the graph of Figure 1.12, Alice wins in four moves, but if we apply the Super Lemma to (b_1, b_2), which satisfy the hypothesis of the lemma, Bob can ensure at least a draw.

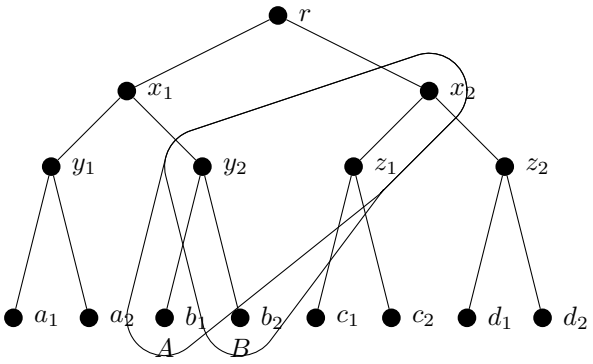


Figure 1.12: A counterexample of the Super Lemma in Maker-Maker. The winning sets are all the paths going from r to a leaf together with the two hyperedges shown in the figure. This hypergraph is a win for Alice, since it is 4-uniform and she can win in four moves. But if we apply the Super Lemma to (b_1, b_2), Bob can ensure a draw: If Alice does not play r with her first move, Bob plays it and then ensure a draw by claiming one of x_2, y_2 or z_1. If she does, Bob claims x_2. Alice is forced to play x_1 otherwise Bob plays it and draws. The he plays y_2, forcing for the same reason Alice to claim y_1. Then Bob wins by claiming z_1.

Chapter 2

Complexity of the different conventions

Gonna catch 'em all.

This chapter focuses on the general complexity of two conventions of positional games. Despite the fact that determining the winner of a Maker-Breaker game is PSPACE-complete since Schaefer [Sch78] in 1978, most of the other conventions have remained open during a long time. An argument from Byskov [Bys04] solved Maker-Maker games in 2004, and the Avoider-Avoider convention has remained open until 2019, when it was proved to be PSPACE-complete by Burke and Hearn [BH19]. We present here two new PSPACE-hardness reductions which prove that Avoider-Enforcer and Client-Waiter games are PSPACE-complete.

In Section 2.1 we will prove that Avoider-Enforcer games are PSPACE-complete. In Section 2.2, we will prove that this is also the case for Client-Waiter games. Finally, in Section 2.3, we reduce these games to the domination game under Avoider-Enforcer and Waiter-Client conventions.

This chapter contains the results of two collaborations. The first one was with Valentin Gledel, and was published in STACS 2023 [GO23]. The second one is a collaboration with Valentin Gledel and Sebastien Tavenas [GOT].

2.1 Avoider-Enforcer games are PSPACE-complete

This section is dedicated to proving the PSPACE-completeness of the Avoider-Enforcer convention, by reduction from QBF. The proof is divided into four parts. First, we present the reduction and introduce the *legitimate order* on the vertices in Subsection 2.1.1, which consists is the order that mimics the valuation of QBF. Secondly, we prove that if this order is respected, it simulates QBF in Subsection 2.1.2. Then we prove that if Enforcer respects the order, Avoider is forced to respect it too, otherwise he loses in Subsection 2.1.3. Next, we prove that if Avoider respects the order, it cannot benefit Enforcer not to respect it in Subsection 2.1.4. Finally, in Subsection 2.1.5, we conclude the proof and present a tool to make the hypergraph 6-uniform.

Theorem 2.1. *Computing the outcome of an Avoider-Enforcer game is* PSPACE*-complete, even restricted to hypergraphs of rank* 6*.*

First, we recall that, according to Theorem 1.61, Avoider-Enforcer games are in PSPACE.

2.1.1 Construction of the hypergraph and of the order

Construction of the hypergraph

Given a QBF-formula of the form $\psi = \forall X_1 \exists X_2, \ldots, \forall X_{2n-1} \exists X_{2n} \varphi$, with φ a 3-CNF formula, we construct a hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ whose vertex set is $\mathcal{X} = \{x_1, \overline{x_1}, \ldots, x_{2n}, \overline{x_{2n}}, u_1, u_2, \ldots, u_{6n}\}$.

A round in ψ corresponds to a step i during which Falsifier gives a valuation to X_{2i-1} and then Satisfier gives a valuation to X_{2i}. In this reduction, for any round in ψ, we consider ten vertices and eight hyperedges. Four of the ten vertices are $\{x_{2i-1}, \overline{x_{2i-1}}, x_{2i}, \overline{x_{2i}}\}$, and the six others vertices are $u_{6i-5}, u_{6i-4}, u_{6i-3}, u_{6i-2}, u_{6i-1}, u_{6i}$. The eight hyperedges are constructed as follows:

$$A_{2i} = (x_{2i}, \overline{x_{2i}}, u_{6i+1}, u_{6i+3}) \qquad B_{2i-1} = (x_{2i-1}, \overline{x_{2i-1}}, u_{6i-1})$$
$$C^+_{6i} = (u_{6i}, u_{6i+1}, u_{6i+3}, x_{2i}) \qquad C^-_{6i} = (u_{6i}, u_{6i+1}, u_{6i+3}, \overline{x_{2i}})$$
$$C^+_{6i-2} = (u_{6i-2}, u_{6i-1}, u_{6i+1}, x_{2i}) \qquad C^-_{6i-2} = (u_{6i-2}, u_{6i-1}, u_{6i+1}, \overline{x_{2i}})$$
$$C^+_{6i-4} = (u_{6i-4}, u_{6i-3}, u_{6i-1}, x_{2i-1}) \qquad C^-_{6i-4} = (u_{6i-4}, u_{6i-3}, u_{6i-1}, \overline{x_{2i-1}})$$

If some of these vertices do not exist (if i is too large), we still add the hyperedges, but with fewer vertices in them. For instance, $A_{2n} = \{x_{2n}, \overline{x_{2n}}\}$. Moreover, for each clause $F_j = l^j_1 \vee l^j_2 \vee l^j_3$ of ψ, we add a hyperedge D_j. For $k \in \{1, 2, 3\}$, if l^j_k is a positive variable X_p, then x_p is in D_j, if l^j_k is a negative one $\neg X_p$, then $\overline{x_p}$ is in D_j. Moreover, If $p = 2i - 1$ is odd, then u_{6i-1} is in D_j, if $p = 2i$ is even, then u_{6i+1} is in D_j.

For instance if $F_1 = X_1 \vee X_2 \vee \neg X_4$, we have $D_1 = (x_1, x_2, \overline{x_4}, u_5, u_7, u_1 3)$.

Finally, the CNF game φ is reduced to the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ with

$$\mathcal{X} = \{\{x_i\}_{1 \le i \le 2n} \cup \{\overline{x_i}\}_{1 \le i \le 2n} \cup \{u_j\}_{1 \le j \le 6n}\}$$
$$\mathcal{F} = \left\{\{A_{2i}\}_{1 \le i \le n} \cup \{C^+_{2i}\}_{1 \le i \le 3n} \cup \{C^-_{2i}\}_{1 \le i \le 3n} \cup \{B_{2i-1}\}_{1 \le i \le n} \cup \{D_j\}_{1 \le j \le m}\right\}$$

Legitimate order

With this construction, we say that Avoider and Enforcer follow a *legitimate order* if they claim board elements in the following order for increasing i:

1. Avoider starts and claims u_{6i-5}.
2. Enforcer claims u_{6i-4}.
3. Avoider claims u_{6i-3}.
4. Enforcer claims one of x_{2i-1} or $\overline{x_{2i-1}}$.
5. Avoider claims the remaining vertex in $\{x_{2i-1}, \overline{x_{2i-1}}\}$
6. Enforcer claims u_{6i-2}.
7. Avoider claims u_{6i-1}.
8. Enforcer claims u_{6i}.
9. Avoider claims one of x_{2i} or $\overline{x_{2i}}$.
10. Enforcer claims the remaining vertex in $\{x_{2i}, \overline{x_{2i}}\}$.

We say that the game is restricted to the legitimate order if Avoider and Enforcer claims the vertices according to this order, i.e. their only choices are among vertices x_i and $\overline{x_i}$.

2.1.2 Winner in the legitimate order

Lemma 2.2. *When the game is restricted to the legitimate order, Avoider has a winning strategy in the Avoider-Enforcer game on H if and only if Satisfier has a winning strategy for the* 3-QBF *game on φ.*

Proof. If the order of moves is legitimate, the only choices available for Avoider and Enforcer are on the vertices x_i and $\overline{x_i}$. For each $1 \leq i \leq 2n$, Avoider claims one of $x_i, \overline{x_i}$ and Enforcer the other. Therefore, if both Avoider and Enforcer play one vertex in $\{x_i, \overline{x_i}\}$, we define the *underlying valuation* given to ψ as the following one:

$$X_i = \begin{cases} \top & \text{if Avoider has claimed } \overline{x_i} \text{ and Enforcer has claimed } x_i \\ \bot & \text{if Avoider has claimed } x_i \text{ and Enforcer has claimed } \overline{x_i} \end{cases}$$

Consider a game played on $\mathcal{H}$ for which both Avoider and Enforcer have respected the legitimate order through the whole game.

Claim 2.3. *Avoider won the game on $\mathcal{H}$ if and only if the formula ψ is satisfied by the underlying valuation of the X_is.*

Proof of the claim. Since the legitimate order is respected, Enforcer claimed all the vertices u_{2i} and thus played at least once in all the hyperedges C^+_{2i} and C^-_{2i}. Moreover, for each pair of variables $(x_i, \overline{x_i})$, Enforcer claimed one of the vertices of the pair, and so he has claimed at least one vertex in all the hyperedges A_i and B_i. Thus, the only hyperedges that could possibly be fully played by Avoider are the hyperedges D_j.

Since, in the legitimate order, Avoider claimed all the vertices u_{2i+1}, a hyperedge D_j corresponding to a clause F_j is fully played by Avoider if and only if she played on all the vertices $x(l^j_k)$ for $l_k \in F_j$, where $x(l^j_k) = x_p$ if $l^j_k = X_p$ and $x(l^j_k) = \overline{x_p}$ if $l^j_k = \neg X_p$. If this is the case, then this means that the formula ψ is not satisfied by the underlying valuation because the clause F_j has all its literals assigned to $\bot$. On the contrary, if the formula ψ is satisfied by the underlying valuation, then, for all clause F_j, at least one of the literals in it is assigned to $\top$ and so Enforcer played at least once in each hyperedge D_j.

Therefore, Avoider won the game on $\mathcal{H}$ if and only if ψ is satisfied. ◇

Suppose Satisfier has a winning strategy $\mathcal{S}$ on ψ. We define a strategy for Avoider as follows: Whenever Avoider has to play a vertex x_{2k} or $\overline{x_{2k}}$, Avoider considers the underlying valuation given to the X_is with $i < 2k$. Then, if Satisfier had put X_{2k} to $\top$, she claims $\overline{x_{2k}}$. Otherwise, she claims x_{2k}. With this strategy, at the end of the game, the underlying valuation of the variables of H will be the same as the valuation given by the game that Satisfier played on ψ. Since Satisfier has a winning strategy on ψ, the underlying valuation satisfies ψ and so Avoider wins the game.

Similarly, if Falsifier has a winning strategy, Enforcer can follow the strategy in such a way that at the end, the valuation of variables in the game played by Falsifier corresponds to the underlying valuation in $\mathcal{H}$. Since Falsifier wins on ψ, Enforcer wins the game on $\mathcal{H}$. □

2.1.3 Enforcer's winning strategy

We prove that, if Enforcer has a winning strategy when the legitimate order is respected by both players, then he has a winning strategy when it is not. We introduce here several sets of variables, defined in such a way that the global strategy can be decomposed into local ones according to these sets.

For $i = 1$ to $4n$, we define the set of vertices S_i as $S_{4n} = \{u_{6n}, x_{2n}, \overline{x_{2n}}\}$ and for $i < 4n$:

- if $i = 4k$, $S_i = \{u_{6k}, x_{2k}, \overline{x_{2k}}, u_{6k+1}\} \cup S_{i+1}$
- if $i = 4k-1$, $S_i = \{u_{6k-2}, u_{6k-1}\} \cup S_{i+1}$
- if $i = 4k-2$, $S_i = \{x_{2k-1}, \overline{x_{2k-1}}\} \cup S_{i+1}$
- if $i = 4k-3$, $S_i = \{u_{6k-4}, u_{6k-3}\} \cup S_{i+1}$

Lemma 2.4. *If Enforcer has a winning strategy in $\mathcal{H}$ when the legitimate order is respected by the two players, then he has a winning strategy in $\mathcal{H}$.*

Proof. Suppose Enforcer has a winning strategy when the legitimate order is respected. Consider a strategy for Enforcer in which he plays according to the legitimate order until Avoider does not. If Avoider respects the order until all the vertices are played, by assumption, Enforcer wins. Otherwise, the proof of the following claim provides a winning strategy for Enforcer.

Claim 2.5. *If, during the game, Avoider plays in a set S_i in which Enforcer has not played yet, then, after this move, Enforcer has a strategy to win the game.*

Proof of the claim. The proof is by induction on i.

First, notice that each S_i has an odd number of vertices and, as the total number of vertices in H is $10n$, there is also an odd number of vertices outside S_i. Therefore, if Avoider plays first in a set S_i for some i, Enforcer answers by playing an arbitrary vertex that is not in S_i and considers an arbitrary pairing outside S_i, which exists as there is an even number of vertices outside S_i after his move. This way, Avoider will be the next player to play in S_i.

Base cases:

- Case $i = 4n$: If Avoider plays first is S_{4n}, by pairing the two other vertices in S_{4n}, and by using Lemma 1.82, Enforcer can force Avoider to play another vertex in S_{4n}. Hence, as $C^+_{6n} = (u_{6n}, x_{2n})$, $C^-_{6n}(u_{6n}, \overline{x_{2n}})$ and $A_{2n} = (x_{2n}, \overline{x_{2n}})$ are three hyperedges, Avoider will claim the two vertices of one of them and will lose.

- Case $i = 4n - 1$: As shown previously, Enforcer has a strategy such that Avoider is the next player to play in S_{4n-1}. If Avoider has played at least one of her two first moves in S_{4n}, she has lost by the case $i = 4n$. Otherwise, she has claimed exactly u_{6n-2} and u_{6n-1}. In this case, Enforcer claims u_{6n} and pairs x_{2n} and $\overline{x_{2n}}$ and by Lemma 1.82 he forces Avoider to claim all the vertices of either $C^+_{6n-2} = \{u_{6n-2}, u_{6n-1}, x_{2n}\}$ or $C^-_{6n-2} = \{u_{6n-2}, u_{6n-1}, \overline{x_{2n}}\}$.

Induction steps:

Suppose that the first time Avoider does not respect the order of the move, she plays in a set S_i for $i \leq 4n - 2$. Enforcer plays a vertex outside S_i. If the second move of Avoider in S_i is in S_{i+1}, Enforcer wins by induction hypothesis. Thus, we can suppose that Avoider has claimed two vertices in $S_i \setminus S_{i+1}$. Moreover, as Enforcer has arbitrarily paired the vertices outside S_i, we describe here the strategy in S_i, and Enforcer plays according to the pairing outside S_i. This strategy ensures that the moves in S_i alternate between both players.

- Case $i = 4k$: Avoider has played twice in $\{u_{6k}, x_{2k}, \overline{x_{2k}}, u_{6k+1}\}$. At least one of $\{u_{6k}, x_{2k}, \overline{x_{2k}}\}$ is available. Enforcer claims it. Avoider has to claim the third vertex in this quadruple, otherwise, she plays first in S_{i+1} and loses by induction, and necessarily one of the three vertices she has claimed is u_{6k+1}. Enforcer claims u_{6k+2}. Avoider either plays first in S_{i+2} and loses by induction hypothesis, or claims u_{6k+3}. At this moment, Avoider has claimed the vertices u_{6k+1} and u_{6k+3}, and two of the vertices of $\{u_{6k}, x_{2k}, \overline{x_{2k}}\}$. So she has filled up one of the hyperedges $C^+_{6k} = (u_{6k}, u_{6k+1}, u_{6k+3}, x_{2k})$, $C^-_{6k} = (u_{6k}, u_{6k+1}, u_{6k+3}, \overline{x_{2k}})$ or $A_{2k} = (x_{2k}, \overline{x_{2k}}, u_{6k+1}, u_{6k+3})$.

- Case $i = 4k - 1$: Avoider has claimed u_{6k-2} and u_{6k-1}. Enforcer claims u_{6k}. Avoider has to play on vertex in $\{x_{2k}, \overline{x_{2k}}, u_{6k+1}\}$, otherwise she plays first in S_{i+2} and loses by induction. Enforcer claims either x_{2k} or $\overline{x_{2k}}$, as at least one of them is available. If Avoider plays a vertex in S_{i+2} she loses by induction. So she has to play the last vertex available in $S_i \setminus S_{i+1}$. With this strategy, Avoider has necessarily claimed u_{6k+1} and one of x_{2k} and $\overline{x_{2k}}$. Thus, she has claimed all the vertices of either $C^+_{6k-2} = (u_{6k-2}, u_{6k-1}, u_{6k+1}, x_{2k})$ or $C^-_{6k-2} = (u_{6k-2}, u_{6k-1}, u_{6k+1}, \overline{x_{2k}})$.

- Case $i = 4k - 2$: Avoider has claimed x_{2k-1} and $\overline{x_{2k-1}}$. Enforcer claims u_{6k-2}. Either Avoider plays first in S_{i+2} and loses by induction, or she claims u_{6k-1}, the last available vertex in S_{i+1} and loses by having filled up $B_{2k-1} = (x_{2k-1}, \overline{x_{2k-1}}, u_{6k-1})$.

- Case $i = 4k-3$: Avoider has claimed u_{6k-4} and u_{6k-3}. Enforcer claims x_{2k-1}. Avoider has to claim $\overline{x_{2k-1}}$, otherwise she plays first in S_{i+2} and loses by induction. Then Enforcer claims u_{6k-2}. If Avoider plays in S_{i+3} she loses by induction. The last vertex available in $S_i \setminus S_{i+3}$ is u_{6k-1}, and if Avoider claims it, she loses by having claimed all the vertices $C^-_{6k-4} = (u_{6k-4}, u_{6k-3}, u_{6k-1}, \overline{x_{2k-1}})$.

By applying this induction, at any moment of the game, if Avoider plays first in a set S_i, she looses. $\diamond$

Finally, if Enforcer has played according to the legitimate order, at any moment of the game, Avoider has to play in a set S_i in which Enforcer has already played. Therefore, she has to respect the order of the moves. The only moment when she can change this order is by claiming u_{6k+1} instead of one of the vertices $x_{2k}, \overline{x_{2k}}$. But if she does so, Enforcer can claim one of them, for instance x_{2k}, and Avoider will be forced to claim $\overline{x_{2k}}$. If this happens, everything happens as if Avoider has claimed $\overline{x_{2k}}$ first and u_{6k+1} after. Since these moves could have occurred in the legitimate order, the strategy can then continue as if the order has been respected.

To conclude, if Enforcer has a winning strategy when the legitimate order is respected, he has one in $\mathcal{H}$ even without this restriction. $\square$

2.1.4 Avoider's winning strategy

We now prove that, if Avoider has a winning strategy when the legitimate order is respected by both players, then she has a winning strategy when it is not. The main idea of the strategy is to respect the order, while Enforcer respects it. If at some point, Enforcer does not respect the order, Avoider, can choose a valuation that satisfies ψ, and has a pairing strategy to force Enforcer to claim for each pair $(x_i, \overline{x_i})$, either the vertex corresponding to any valuation she has chosen or the odd u_j that follows it. By construction, any hyperedge containing a vertex x_i or $\overline{x_i}$ also contains that vertex u_j, and this will prove that whenever Enforcer does not respect the order, it benefits Avoider.

Lemma 2.6. *If Avoider has a winning strategy in $\mathcal{H}$ when the legitimate order is respected by the two players, then she has a winning strategy in $\mathcal{H}$.*

Proof. Suppose Avoider has a winning strategy when the legitimate order is respected. We now describe a winning strategy for Avoider in the general case.

While Enforcer respects the legitimate order, Avoider also respects it. Suppose that at some moment of the game, Enforcer does not respect the legitimate order. Denote by y_A the vertex he would have played according to the legitimate order, and by y_E the vertex he has claimed instead. If y_A is a vertex x_{2i} or $\overline{x_{2i}}$, Avoider pairs it with u_{6i+1} and continues as if Enforcer had to play u_{6i+2}. If this is the case, consider $y_A = u_{6i+2}$. Note that according to Lemma 1.78, we can suppose that y_E is not a vertex u_j with j odd. Indeed, for each vertex u_{2i+1}, the hyperedges that contain the previous vertex in the legitimate order (if this vertex is a vertex x_j or $\overline{x_j}$ this is true for either of them) also contain u_{2i+1}.

Denote by k the smallest integer such that $y_E \notin S_k$, and by k' the largest integer such that $y_A \in S_{k'}$. We consider $k = 4n+1$ if $y_E \in S_{4n}$, with $S_{4n+1} = \emptyset$. Note that all the vertices outside $S_{k'}$ have already been played or are paired, and that $S_{k'} \setminus S_k$ is then the set of vertices perturbed by the move of Enforcer. As all the sets S_i have an odd number of vertices, we know that the number of remaining vertices outside S_k is odd, as an even number of moves have been played. By assumption, Avoider was following a winning strategy for the legitimate order. She can then consider an arbitrary sequence of moves for Enforcer following the legitimate order and her answers according to her strategy until all the vertices in $S_{k'} \setminus S_k$ are all claimed. According to these moves, we will denote by x_j^E the vertex among $(x_j, \overline{x_j})$ claimed by Enforcer and by x_j^A the vertex played by Avoider.

Avoider claims y_A, the vertex that Enforcer should have claimed according to the legitimate order, and will consider one strategy in S_k and another one outside S_k:

- In $\mathcal{H} \setminus S_k$, she plays according to a pairing strategy, that will be presented in the next paragraph.

- In S_k, Avoider considers the strategy she would have played if all the vertices outside S_k were played according to the legitimate order, with the vertices x_j^E claimed by Enforcer and the vertices x_j^A claimed by Avoider.

The pairing we define is the following one: (u_{6i-4}, x_{2i-1}^A), (u_{6i-3}, u_{6i-6}), (x_{2i-1}^E, u_{6i-1}), (u_{6i-2}, x_{2i}^A), (u_{6i+1}, x_{2i}^E). This pairing concerns all the vertices that have to be played after y_A in the legitimate order that are not in S_k, and we consider only pairs containing at least one vertex outside S_k. Note that by construction, exactly one vertex of this pairing is already played, and exactly one paired with a vertex in S_k. Therefore, to make the pairing contain only vertices not played and outside S_k, some modifications are done. These modifications are presented in Table 2.1. By applying Lemma 1.82, Avoider can ensure that Enforcer plays at least one in each of these pairs.

y_A	changes
u_{6i-4}	$u_{6i-3} \leftrightarrow x_{2i-1}^A$
x_{2i-1}, $\overline{x_{2i-1}}$	$x_{2i-1}^* \leftrightarrow u_{6i-1}$
u_{6i-2}	$u_{6i-1} \leftrightarrow x_{2i}^A$
u_{6i}	$u_{6i+3} \leftrightarrow x_{2i}^A$

y_E	changes
u_{6i-4}	no changes
x_{2i-1}, $\overline{x_{2i-1}}$	$x_{2i-1}^* \leftrightarrow u_{6i-4}$
u_{6i-2}	no changes
u_{6i}	no changes
x_{2i}, $\overline{x_{2i}}$	$x_{2i}^* \leftrightarrow u_{6i+1}$, $u_{6i-2} \leftrightarrow u_{6i}$

Table 2.1: Changes of the matching. x_j^* refers to the variable in $\{x_j, \overline{x}_j\}$ that has not been played, arrows represent new pairs of vertices in the matching.

Claim 2.7. *The pairing strategy ensures that Enforcer plays at least once in each hyperedge A_i, B_i or C_i containing all their vertices in $S_{k'}$ and at least one outside S_k.*

Proof of the claim.

First, if the hyperedge contains no vertex whose pairing has been modified because of their belonging to y_A or y_E, it contains two paired vertices. We show in bold text the paired vertices:

$$A_{2i} = (x_{2i}^A, \boldsymbol{x_{2i}^E}, \boldsymbol{u_{6i+1}}, u_{6i+3}) \qquad C_{6i-2}^E = (u_{6i-2}, u_{6i-1}, \boldsymbol{u_{6i+1}}, \boldsymbol{x_{2i}^E})$$
$$C_{6i}^A = (\boldsymbol{u_{6i}}, u_{6i+1}, \boldsymbol{u_{6i+3}}, x_{2i}^A) \qquad B_{2i-1} = (x_{2i-1}^A, \boldsymbol{x_{2i-1}^E}, \boldsymbol{u_{6i-1}})$$
$$C_{6i}^E = (u_{6i}, \boldsymbol{u_{6i+1}}, u_{6i+3}, \boldsymbol{x_{2i}^E}) \qquad C_{6i-4}^A = (\boldsymbol{u_{6i-4}}, u_{6i-3}, u_{6i-1}, \boldsymbol{x_{2i-1}^A})$$
$$C_{6i-2}^A = (\boldsymbol{u_{6i-2}}, u_{6i-1}, u_{6i+1}, \boldsymbol{x_{2i}^A}) \qquad C_{6i-4}^E = (u_{6i-4}, u_{6i-3}, \boldsymbol{u_{6i-1}}, \boldsymbol{x_{2i-1}^E})$$

We prove now that the first hyperedges of the matching also contains two paired vertices, according to the changes presented in Table 2.1. Recall first that if Enforcer was supposed to play in $\{x_{2i}, \overline{x_{2i}}\}$, Avoider pairs this vertex with u_{6i+1} and considers $y_A = u_{6i+2}$. The only one hyperedge among the A_i, B_i and C_i that was concerned with this change is A_i, in which two vertices are now paired. In any other case, the following vertices are paired:

- If $y_A = u_{6i-4}$, only the hyperedges C_{6i-4}^E and C_{6i-4}^A are concerned by the changes. In the former x_{2i-1}^E is paired with u_{6i-1}, in the latter x_{2i-1}^A is paired with u_{6i-3}.
- If $y_A \in \{x_{2i-1}, \overline{x_{2i-1}}\}$, the only hyperedges concerned by the change is B_{2i-1}. In it, the other vertex in $\{x_{2i-1}, \overline{x_{2i-1}}\}$ is paired with u_{6i-1}
- If $y_A = u_{6i-2}$, only the hyperedges C_{6i-2}^E and C_{6i-2}^A are concerned by the changes. In the former x_{2i}^E is paired with u_{6i+1}, in the latter x_{2i}^A is paired with u_{6i-1}.
- If $y_A = u_{6i}$, only the hyperedges C_{6i}^E and C_{6i}^A are concerned by the changes. In the former x_{2i}^E is paired with u_{6i+1}, in the latter x_{2i}^A is paired with u_{6i+3}.

For the last hyperedges that contain vertices of the matching, the following happens:

- If $y_E = u_{6i-4}$, the pairing stops at u_{6i-3}. The only two hyperedges that contain at least one vertex in S_k and one vertex outside S_k are C^+_{6i-4} and C^-_{6i-4}, in which Enforcer has claimed y_E.
- If $y_E = x_{2i-1}$ or $\overline{x_{2i-1}}$, the pairing stops after the second vertex in $\{x_{2i-1}, \overline{x_{2i-1}}\}$. The only one hyperedge containing at least one vertex in S_k and one outside S_k is B_{2i-1}, in which Enforcer has already claimed y_E.
- If $y_E = u_{6i-2}$, the pairing stops at u_{6i-1}. The only two hyperedges that contain at least one vertex in S_k and one vertex outside S_k are C^+_{6i-2} and C^-_{6i-2}, in which Enforcer has claimed y_E.
- If $y_E = u_{6i}$, the pairing stops at u_{6i+1}. The three hyperedges that contain both vertices in S_k and vertices outside S_k are C^+_{6i}, C^-_{6i} and A_{2i}. In C^+_{6i}, C^-_{6i}, Enforcer has claimed y_E, and in A_{2i}, Enforcer will play one of x^E_{2i} or u_{6i+1} as these two vertices are paired together.
- If $y_E = x_{2i}$ or $\overline{x_{2i}}$, the pairing stops at u_{6i+1}. The three hyperedges that contain vertices inside S_k and outside S_k are C^+_{6i}, C^-_{6i} and A_{2i}. As the second vertex in $\{x_{2i}, \overline{x_{2i}}\}$ is paired with u_{6i+1}, either Enforcer has claimed both x_{2i} and $\overline{x_{2i}}$, and any of these three hyperedges contains at least one of them; or Enforcer has claimed u_{6i+1} which is in these three hyperedges.

If the pairing stops because it goes until the end (i.e. $k = 4n + 1$), one vertex is not paired. According to Lemma 1.82, as Enforcer plays the last move in $\mathcal{H}$, Avoider can force him to play it and still play once in each pair of the pairing.

Finally, in any hyperedge A_i, B_i or C_i containing at least one vertex of the matching, Enforcer has played at least one vertex. ◇

Now, we can prove that the strategy we defined for Avoider is a winning strategy. In all the hyperedges A_i, B_i or C_i, Enforcer played at least once. Indeed, if Enforcer has respected the order until he plays in one of these hyperedges, there is nothing to do. Otherwise, by Claim 2.7, Avoider can force Enforcer to play in it as this hyperedge is considered in a set of hyperedges in which Enforcer has not respected the order, and thus contains two paired vertices.

In the hyperedges D_j, as the strategy in the legitimate order is a winning strategy, it forces at least one vertex x_i or $\overline{x_i}$ to be claimed by Enforcer in D_j (since all the vertices u_k of odd indices are played by Avoider in the legitimate order). Then, by construction, if when this vertex has to be claimed the order was respected, Enforcer has played it. If the order was not respected, then Avoider has paired this vertex with the next vertex u_k of odd index, forcing Enforcer to play either that vertex u_k or the vertex he would have played among x_i and $\overline{x_i}$ according to the legitimate order. In both cases, Enforcer has played in D_j. □

2.1.5 Conclusion

Using the previous lemmas, we can prove Theorem 2.1.

Proof. First, we remind that, according to Theorem 1.61, we know that Avoider-Enforcer games are in PSPACE. We now prove the PSPACE-hardness of the problem by reduction from 3-QBF.

Let ψ be a QBF quantified boolean formula. Consider the hypergraph H obtained from ψ by following the construction of Subsection 2.1.1. This construction has polynomial size. According to Lemma 2.2, when the order is respected, if Satisfier (resp. Falsifier) has a winning strategy in ψ, Avoider (resp. Enforcer) has a winning strategy in H. Thus, according to Lemma 2.6 (resp. Lemma 2.4), if Avoider (resp. Enforcer) has a winning strategy on H when the legitimate order is respected, she (resp. he) also has one in general in H. Thus, Satisfier wins on ψ if and only if, Avoider wins on H. Therefore, determining the winner of an Avoider-Enforcer game is PSPACE-complete.

The construction provides a hypergraph H in which all the hyperedges have of size at most six, determining the winner of an Avoider-Enforcer game is PSPACE-complete even restricted to hypergraphs or rank 6. □

This proof builds a hypergraph of rank 6, which is not uniform. A similar argument to the one used by Rahman and Watson [RW21] proves that the hypergraphs can be made 6-uniform without loss of generality.

Lemma 2.8. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph of rank k. Let $m = \min_{e \in \mathcal{F}} |e|$. If $m < k$, there exists a hypergraph $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$ of rank k where $\min_{e \in \mathcal{F}} |e| = m + 1$, having $|\mathcal{F}'| \leq 2|\mathcal{F}|$ and $|\mathcal{X}'| \leq |\mathcal{X}| + 2$ such that Avoider has a winning strategy in the Avoider-Enforcer game on $\mathcal{H}$ if and only if she has one in $\mathcal{H}'$.*

Proof. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph of rank k. Let $m = \min_{e \in \mathcal{F}} |e|$. We define $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$ as follows. We start from $\mathcal{X}' = \mathcal{X}$. We add two vertices $\{a_1, a_2\}$ in $\mathcal{X}'$. For each hyperedge $e \in \mathcal{F}$, we add hyperedges in $\mathcal{F}'$ as follows:

- If $|e| > m$, we add a copy of e in $\mathcal{F}'$.
- If $|e| = m$, we add two hyperedges $e_1 = e \cup \{a_1\}$ and $e_2 = e \cup \{a_2\}$ in $\mathcal{F}'$.

We have $|\mathcal{X}'| = |\mathcal{X}| + 2$, $|\mathcal{F}'| \leq 2|\mathcal{F}|$ and $\min_{e \in \mathcal{F}'} |e| = m + 1$. The construction of $\mathcal{H}'$ is provided in Figure 2.1

Now, by the Super Lemma, applied to a_1 and a_2 in $\mathcal{H}'$, we know that $(\mathcal{H}', \emptyset, \emptyset)$ has the same outcome as $(\mathcal{H}', \{a_1\}, \{a_2\})$. They, by Lemma 1.74, $(\mathcal{H}', \{a_1\}, \{a_2\})$ and $\mathcal{H}$ has the same outcome, as $\mathcal{H}$ is exactly the hypergraph obtained with this reduction. □

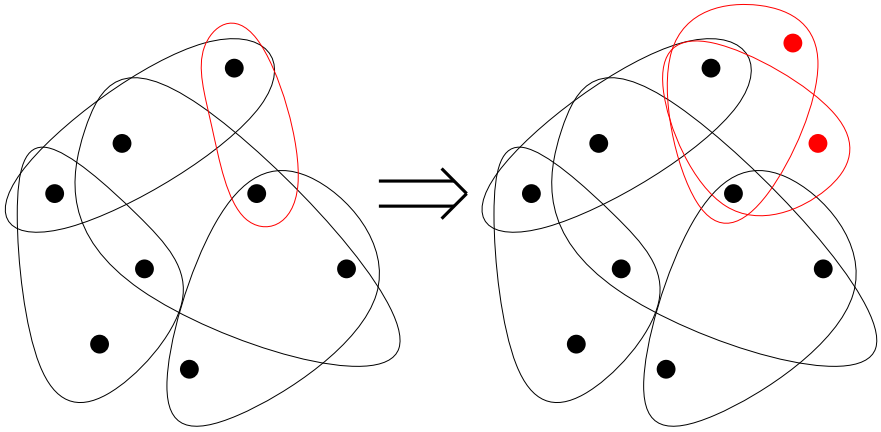


Figure 2.1: Application of Lemma 2.8. The hyperedge of size 2 has been transformed into two hyperedges of size 3.

Corollary 2.9. *Avoider-Enforcer is* PSPACE*-complete even restricted to* 6*-uniform hypergraphs*

Proof. The hypergraph obtained in the proof of Theorem 2.1 has rank k and all its hyperedges have size at least two. Therefore, by applying Lemma 2.8 four times, with $m = 2, 3, 4, 5$, we obtain a hypergraph having at most eight more vertices and eight times more hyperedges. Thus, this construction is still polynomial and the hypergraph obtained is 6-uniform. □

2.2 Client-Waiter games are PSPACE-complete

In this section, we will introduce the boolean problem Paired SAT, and we prove its PSPACE-completeness. From Paired SAT, we prove that determining the winner of a Client-Waiter game is PSPACE-complete, even restricted to 6-uniform hypergraphs:

Theorem 2.10. *Computing the winner of a Client-Waiter game is* PSPACE*-complete, even restricted to hypergraphs of rank* 6*.*

Subsection 2.2.1 is dedicated to the problem Paired SAT and the proof of its PSPACE-completeness. Then, in Subsection 2.2.2, we introduce blocks in hypergraphs, which will be the main tool of the proof, as they allow us to create a pairing strategy even for Client in Client-Waiter games. In Subsection 2.2.3, we present the construction of the hypergraph $\mathcal{H}$. Then in Subsection 2.2.4, we prove that if Satisfier has a winning strategy in the instance of Paired SAT then Waiter has a winning strategy in $\mathcal{H}$. Next, in Subsection 2.2.5, we prove the opposite, i.e., that if Falsifier has a winning strategy in the instance of Paired SAT, then, Client has one in $\mathcal{H}$. Finally, in Subsection 2.2.6 we introduce a tool to transform the hypergraph into a uniform one.

2.2.1 Paired SAT is PSPACE-complete

We introduce here the game Paired SAT. The main idea of this game is to introduce a CNF-game mimicking the fact that one player chooses which variable the second player will have to play on.

Definition 2.11 (Paired SAT)**.** *Let φ be a* 3*-CNF Formula over a set of pair of variables $X = \{(x_1, y_1), \dots, (x_n, y_n)\}$. The* Paired SAT*-game is played by two players, Satisfier and Falsifier as follows: while there is a variable that has not been assigned a valuation, Satisfier chooses a pair of variables (x_i, y_i) that she has not chosen yet and gives a valuation, $\top$ or $\bot$, to x_i. Then Falsifier gives a valuation to y_i. When all the variables have their valuation chosen, Satisfier wins if and only if the valuation they have provided to the x_is and y_is satisfies φ.*

Theorem 2.12. *Determining the winner of the* Paired SAT*-game is* PSPACE*-complete.*

Proof. We provide a reduction from QBF. Let $\psi = \exists x_1 \forall y_1 \cdots \exists x_n \forall y_n, \varphi$ be a QBF formula. We construct an instance of the PAIRED SAT-game (φ', X) as follows:

- $X = \{(z_0, y_0), (x_1, t_1), (z_1, y_1), \dots, (x_n, t_n), (z_n, y_n)\}$, where y_0, the z_is and the t_js are new variables.
- $\varphi' = \varphi \bigwedge\limits_{1 \leq i \leq n} (y_{i-1} \oplus t_i \oplus z_i)$.

  where $\oplus$ refers to the XOR operator and we have:

$$a \oplus b \oplus c = (a \vee b \vee c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c).$$

We prove that Satisfier wins on ψ if and only if she wins on (φ', X).

First note that whenever two values of $a \oplus b \oplus c$ are known, the player who chooses the last value can always decide to satisfy or not $(a \oplus b \oplus c)$. Suppose first that Satisfier has a winning strategy $\mathcal{S}$ on ψ, and consider the following strategy for him on (φ', X):

- Satisfier chooses to instantiate the pairs $(z_0, y_0), (x_1, t_1), \dots (x_n, t_n), (z_n, y_n)$ in that order.
- Whenever Satisfier has to choose a value for a variable x_i, he follows $\mathcal{S}$ with the corresponding values of the x_j and y_j for $1 \leq j < i$. This is always possible as he chooses the order
- Whenever Satisfier has to give a value to a variable z_i, she gives the value so that $y_{i-1} \oplus t_i \oplus z_i$ is satisfied (if $i = 0$ she can instantiate the variable z_0 by either $\top$ or $\bot$).

Following this strategy, all the clauses $(y_{i-1} \oplus t_i \oplus z_i)$ are satisfied as Satisfier always chooses the last vertex of these clauses (which appears in exactly one of them), and as $\mathcal{S}$ was a winning strategy in ψ, it will satisfy φ, as the variables will be chosen in the same order.

Now Suppose that Falsifier has a winning strategy $\mathcal{S}$ on ψ, and consider the following strategy on (φ', X, Y):

- While Satisfier plays the pairs according to the order $(z_0, y_0), (x_1, t_1), \dots (x_n, t_n), (z_n, y_n)$, Falsifier gives to the corresponding t_i the value $\perp$, and to y_i the value given by $\mathcal{S}$.
- If Satisfier plays a pair (x_j, t_j) before she should, Falsifier still gives the valuation $\perp$ to t_j and ignores this move while choosing the valuations of the y_i for $i \leq j$.
- If Satisfier plays a vertex z_j before he should, the first time it happens, all the unplayed variables in the clause $(y_{j-1} \oplus t_j \oplus z_j)$ will be played by Falsifier. Therefore, Falsifier can just win by choosing a good valuation for y_{j-1} and t_j.

Following this strategy, if Satisfier instantiates a variable z_i whereas there is $j \leq i$ such that the variable x_j has not yet been instantiated then Falsifier wins. Indeed, the first time it happens either z_{i-1} or x_i has not been instantiated (otherwise it already happened when Satisfier instantiated z_{i-1}). Consequently, Falsifier wins through the clause $(y_{i-1} \oplus t_i \oplus z_i)$. Otherwise, each time Falsifier has to choose a valuation for a variable y_i, all the vertices x_j with $j \leq i$ have already been played and so, he can play according to $\mathcal{S}$. As $\mathcal{S}$ was a winning strategy, in both case, Falsifier can make a clause unsatisfied and therefore wins the game. □

2.2.2 Blocks in Client-Waiter games

The main tool of several reductions of positional games is pairing strategies. However, this cannot be applied to Client-Waiter games, since only Waiter has choices about how to make the pairs. We present *block-hypergraphs* and *block-strategies* that will be used similarly to pairing strategies to ensure that client can claim some vertices. A block-hypergraph is depicted in Figure 2.2.

Definition 2.13 (Blocks)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* block *$B \subset \mathcal{X}$ is a set of vertices such that $|B| = 2k$ for some $k \geq 1$, and any set of $k+1$ vertices of B is a hyperedge.*

If $\mathcal{H}$ can be partitioned into blocks, we say that $\mathcal{H}$ is a block-hypergraph.

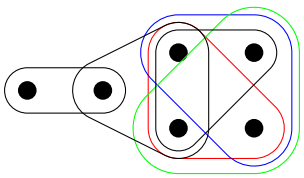


Figure 2.2: A block a hypergraph. The two vertices on the left form a block. The four on the right a second one. The hyperedge between them is in no block

Lemma 2.14. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph, and let B be a block of $\mathcal{H}$. If Waiter has a winning strategy in $\mathcal{H}$, she has to propose the vertices of B two by two.*

Proof. Suppose that Waiter has a winning strategy in which she does not propose all the vertices of B two by two. Let $k = \frac{|B|}{2}$. The first time she proposes a vertex $x \in B$ with a vertex $y \notin B$, Client can choose x. Then, each time Waiter proposes at least one vertex in B, Client claims it. In the end, Client will claim at least $k+1$ vertices of B and therefore will win. Thus, if Waiter has a winning strategy, she has to propose the vertices of B two by two. □

Corollary 2.15. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a block-hypergraph. If Waiter has a winning strategy in $\mathcal{H}$, any pair of vertices she proposes belongs to a same block of $\mathcal{H}$.*

2.2.3 Construction of the hypergraph

Now that block-hypergraphs are defined, we can present the reduction. The main idea of the reduction is that we construct a block-hypergraph, such that each block corresponds to the valuation that will be given to a variable.

Let (φ, X) be an instance of Paired SAT where $X = \{(x_1, y_1), \dots, (x_n, y_n)\}$, and $\varphi = \bigwedge_{1 \leq j \leq m} C_j$ is a 3-CNF on the variables of X. We build a hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ as follows.

Let us define the set $\mathcal{X}$ of $8n$ vertices. Let $\mathcal{X} = \bigcup_{1 \leq i \leq n} S_i \cup F_i$, with for $1 \leq i \leq n$, $S_i = \{s_i^0, s_i^T, s_i^F, s_i^1\}$ (gadget which encodes Satisfier's choice for the variable x_i) and $F_i = \{f_i^0, f_i^T, f_i^{T'}, f_i^F\}$ (gadget which encodes Falsifier's choice for the variable y_i).

Now we focus on the construction of the hyperedges.

- The block-hyperedges $B = \bigcup_{1 \leq i \leq n} B_i$, which make each S_i and each F_i a block:

$$B_i = \bigcup_{i=1}^{n} \left(\{H \subseteq S_i \mid |H| = 3\} \cup \{H \subseteq F_i \mid |H| = 3\} \right).$$

- The pair-hyperedges $P = \bigcup_{1 \leq i \leq n} P_i$:

$$P_i = \left\{ \{s_i^0, s_i^T, f_i^0, f_i^T\}, \{s_i^0, f_i^F, f_i^T, s_i^F\}, \{s_i^0, f_i^F, s_i^T, f_i^{T'}\}, \{s_i^0, s_i^F, f_i^0, f_i^{T'}\} \right\}$$

- The clauses-hyperedges. Each clause $C_j \in \varphi$ is a set of three literals $\{\ell_j^1, \ell_j^2, \ell_j^3\}$. We define first, for $1 \leq j \leq m$ and $k \in \{1, 2, 3\}$, the set H_j^k which encodes the property that the literal ℓ_j^k is instantiated to $\bot$.

$$H_j^k = \begin{cases} \{\{s_i^0, s_i^T\}\} & \text{if } \ell_j^k = x_i \\ \{\{s_i^0, s_i^F\}\} & \text{if } \ell_j^k = \neg x_i \\ \{\{f_i^0, f_i^T\}, \{f_i^0, f_i^{T'}\}\} & \text{if } \ell_j^k = y_i \\ \{\{f_i^F\}\} & \text{if } \ell_j^k = \neg y_i. \end{cases}$$

  We define now the set of hyperedges:

$$C = \bigcup_{C_j \in \varphi} H_j.$$

  with $H_j = \left\{ h_1 \cup h_2 \cup h_3 \mid \forall k \in \{1, 2, 3\},\ h_k \in H_j^k \right\}$

  For example, if $C_j = x_1 \vee y_1 \vee \neg y_2$, we have $H_j^1 = \{\{s_1^0, s_1^T\}\}$, $H_j^2 = \{\{f_1^0, f_1^T\}, \{f_1^0, f_1^{T'}\}\}$ and $H_j^3 = \{\{f_2^F\}\}$. Finally, we have two hyperedges to encode C_j: $H_j = \{(s_1^0, s_1^T, f_1^0, f_1^T, f_2^F), (s_1^0, s_1^T, f_1^0, f_1^{T'}, f_2^F)\}$

The gadget for the pair (x_i, y_i) is depicted in Figure 2.3.
Intuitively, these hyperedges are constructed in such a way that:

- The block hyperedges B force Waiter to always propose two vertices in the same set. This way, we can compute a valuation from the moves of the players
- The pair hyperedges P force Waiter to give to Client the choice of the value of y_i after she has made her choice for x_i
- The clause hyperedges C which represents the clauses of φ and make the equivalence between a win of Waiter and a valuation that satisfies φ

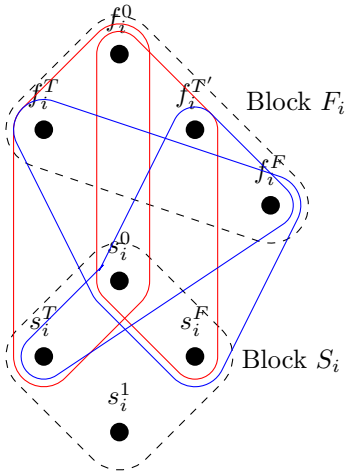


Figure 2.3: Gadget for the vertices in S_i and F_i. A dashed set represents a block, i.e. all the hyperedges of size 3 are present in it.

Finally, the reduction associates to the instance (φ, X) of Paired SAT the instance $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ with $\mathcal{F} = B \cup P \cup C$ of Client-Waiter. This reduction is polynomial as B contains $8n$ hyperedges, P contains $4n$ hyperedges, and C contains at most $8m$ hyperedges (where m is the number of clause of φ).

We define an underlying valuation of the variables, corresponding to the moves on the hypergraph as follows:

- If Client has claimed s_i^T, $x_i = \bot$
- If Client has claimed s_i^F, $x_i = \top$
- If Client has claimed f_i^0 and one of f_i^T, $f_i^{T'}$, $y_i = \bot$
- If Client has claimed f_i^F, $y_i = \top$

This valuation simulates that the literals corresponding to Client's claims equal $\bot$.

2.2.4 Waiter's winning strategy

In this section, we prove that if Satisfier has a winning strategy in φ, then Waiter has a winning strategy in $\mathcal{H}$.

Lemma 2.16. *If Satisfier has a winning strategy in φ, then Waiter has a winning strategy in $\mathcal{H}$.*

Proof. Let $\mathcal{S}$ be a winning strategy for Satisfier, consider a strategy for Waiter as follows. If $\mathcal{S}$ selects an integer $1 \leq i \leq n$, and puts x_i to $\top$ (resp. $\bot$), Waiter plays in Block i and selects the pair (s_i^0, s_i^T) (resp. (s_i^0, s_i^F)). Then, she plays the pair corresponding to the two other vertices in the block S_i. To compute the value of y_i, she plays $(f_i^F, f_i^{T'})$ (resp. (f_i^F, f_i^T)) and finally, the remaining pair of the block F_i. If Client has chosen f_i^F, she considers that $y_i = \top$, otherwise, she considers that $y_i = \bot$ in $\mathcal{S}$.

As this strategy always propose vertices in blocks, Client cannot win with the hyperedges in B:

- If Waiter has proposed (s_i^0, s_i^F), if Client has not chosen s_i^0, as it is in all the hyperedges of P_i, Waiter cannot lose on P_i. Otherwise, Waiter has claimed s_i^F, and has proposed the pairs (f_i^T, f_i^0) and $(f_i^F, f_i^{T'})$ which covers all the remaining hyperedges of P_i.

- If Waiter has proposed (s_i^0, s_i^T), once again, if Client has not chosen s_i^0, Waiter cannot lose on P_i. Otherwise, Waiter has claimed s_i^T, and has proposed the pairs $(f_i^{T'}, f_i^0)$ and (f_i^F, f_i^T) which covers all the remaining hyperedges of P_i.

We now consider the clauses hyperedges:

Let C_j be a clause of φ. It is sufficient to prove that there exists a $1 \leq k \leq 3$ such that Waiter has claimed a vertex in each set of H_j^k. As $\mathcal{S}$ was a winning strategy for Satisfier in φ, there exists a literal ℓ_j^k for $k \in \{1, 2, 3\}$ satisfying C_j.

- If $\ell_j^k = x_i$, by construction of the strategy, Waiter has proposed the pair (s_i^0, s_i^T), therefore she has claimed a vertex in H_j^k.
- If $\ell_j^k = \neg x_i$, by construction of the strategy, Waiter has proposed the pair (s_i^0, s_i^F), therefore she has claimed a vertex in H_j^k.
- If $\ell_j^k = y_i$, by construction of the strategy, Waiter has considered that y_i was put to $\top$ according to the choices of Client, which corresponds to the case where Client has chosen f_i^F. Therefore, she has claimed in H_j^k two of the three vertices $\{f_i^0, f_i^T, f_i^{T'}\}$, and thus in all the sets of H_j^k, either by having claimed f_i^0 or by having claimed f_i^T and $f_i^{T'}$.
- If $\ell_j^k = \neg y_i$, by construction of the strategy, Waiter has considered that y_i was put to $\bot$ according to the choice of Client, which corresponds to the case where Client has not chosen f_i^F, therefore Waiter has claimed it, and thus has claimed in H_j^k.

Finally, Client has not filled up the hyperedges in H_j for any $1 \leq j \leq m$, and Waiter has won. □

2.2.5 Client's winning strategy

We prove now that if Falsifier has a winning strategy in φ, then Client has a winning strategy in $\mathcal{H}$.

Lemma 2.17. *If Falsifier has a winning strategy in φ, then Client has a winning strategy in $\mathcal{H}$.*

Proof. Suppose now that Falsifier has a winning strategy $\mathcal{S}$ in φ. We provide a strategy for Client. Note that as the hypergraph can be decomposed into blocks or order 4, once the first pair of a block is proposed, Client knows that the second pair will be proposed at some point, and therefore can consider a strategy knowing the two pairs.

First, notice that, as $\mathcal{H}$ is a block-hypergraph, we can suppose that Waiter always propose vertices in blocks, according to Corollary 2.15.

Let $1 \leq i \leq n$ be an integer, and suppose that Waiter proposes pairs in a block S_i or F_i. Client considers that Satisfier has chosen the pair i in $\mathcal{S}$ and answers as follows:

- If Waiter proposes two vertices in S_i, as Client knows that Waiter will have to propose the second pair of S_i at some point (as $\mathcal{H}$ is a block-hypergraph), he can always consider the two pairs of S_i. Among them, he can ensure to claim s_i^0 and one of s_i^T and s_i^F. If he has claimed s_i^F, he considers that Satisfier has put x_i to $\top$ and if he has claimed s_i^T, he considers that it has been put to $\bot$.
  - Now, if $\mathcal{S}$ puts y_i to $\bot$, as Client knows that he will have s_i^0 and s_i^F (resp. s_i^T), Waiter is forced to propose (f_i^0, f_i^T) (resp. $(f_i^0, f_i^{T'})$) as they are the only remaining vertices of a hyperedge of P. Thus, Client can take f_i^0 and by the block strategy of Waiter, he will be able to take $f_i^{T'}$ (resp. f_i^T) after the next move if F_i.
  - If $\mathcal{S}$ puts y_i to $\top$, the same happens, and he can claim f_i^F and any other vertex.
- if Waiter proposes two vertices of F_i, as F_i is a block of four vertices, Waiter has only three possible ways to pair them.
  - If the pairs are (f_i^0, f_i^T) and $(f_i^F, f_i^{T'})$, Client assume that x_i was put to $\bot$ in $\mathcal{S}$. If he has to put y_i to $\bot$, he will claim f_i^0 and $f_i^{T'}$, forcing Waiter to propose the pair (s_i^0, s_i^F) in order not to lose because of the hyperedge $(f_i^0, s_i^0, f_i^{T'}, s_i^F)$. If he has to put y_i to $\top$, he will claim f_i^T and f_i^F, forcing Waiter to propose (s_i^0, s_i^F) in order not to lose because of the hyperedge $(s_i^0, f_i^F, f_i^T, s_i^F)$ and everything continues as if (s_i^0, s_i^F) has been proposed before.

– If the pairs are $(f_i^0, f_i^{T'})$ and (f_i^F, f_i^T), Client assume that x_i was put to $\top$ in $\mathcal{S}$. If he has to put y_i to $\bot$, he will claim f_i^0 and f_i^T, forcing Waiter to propose the pair (s_i^0, s_i^T) in order not to lose because of the hyperedge $(f_i^0, s_i^0, f_i^T, s_i^T)$. If he has to put y_i to $\top$, he will claim $f_i^{T'}$ and f_i^F, forcing Waiter to propose (s_i^0, s_i^T) in order not to lose because of the hyperedge $(s_i^0, f_i^F, s_i^T, f_i^{T'})$ and everything continues as if (s_i^0, s_i^T) has been proposed before.

– If the pairs are (f_i^0, f_i^F) and $(f_i^T, f_i^{T'})$, if for any valuation of x_i, $\mathcal{S}$ puts y_i to $\bot$, Client takes f_i^0 and f_i^T and will take s_i^0 and one of s_i^T or s_i^F. Otherwise, if for $x_i = \top$ (resp. $x_i = \bot$), Falsifier would put y_i to $\top$, Client takes f_i^F and $f_i^{T'}$ (resp. f_i^T) forcing Waiter to propose the pair (s_i^0, s_i^T) (resp. (s_i^0, s_i^F)) because of the hyperedge $(s_i^0, f_i^F, s_i^T, f_i^{T'})$ (resp. $(s_i^0, f_i^F, f_i^T, s_i^F)$) in both cases, the variables are assigned as if Satisfier has chosen before Falsifier in $\mathcal{S}$.

Following this strategy until the end, the underlying valuation of the vertices is the one obtained by $\mathcal{S}$ in φ. By hypothesis, there exists a clause $C_j \in \varphi$ in which all the literals are set to $\bot$. Let ℓ_j^k for $1 \leq k \leq 3$ be one of these literals.

- If $\ell_j^k = x_i$, i.e. $x_i = \bot$, by hypothesis, Client has claimed s_i^0 and s_i^T.
- If $\ell_j^k = \neg x_i$, i.e. $x_i = \top$, by hypothesis, Client has claimed s_i^0 and s_i^F.
- If $\ell_j^k = y_i$, i.e. $y_i = \bot$, by hypothesis, Client has claimed f_i^0 and one of f_i^T or $f_i^{T'}$.
- If $\ell_j^k = \neg y_i$, i.e. $y_i = \top$, by hypothesis, Client has claimed f_i^F.

Thus, in any case, Client has claimed all the vertices of a set in H_j^k, and therefore wins by filling up a hyperedge in H_j. □

2.2.6 Conclusion

We can now conclude the proof of Theorem 2.10.

Proof of Theorem 2.10. First, we recall that, according to Theorem 1.61, Client-Waiter games are in PSPACE. We prove the hardness by reduction from Paired SAT.

Let (φ, X) be an instance of Paired SAT. Consider the hypergraph $\mathcal{H}$ obtained from the reduction provided in Subsection 2.2.3. It has $O(|X|)$ vertices of $O(|X| + |\varphi|)$ hyperedges, so it is polynomial. According to Lemma 2.16 and Lemma 2.17, we have that Satisfier wins in (φ, X) if and only if Waiter wins in $\mathcal{H}$. Therefore, determining the winner of a Client-Waiter game is PSPACE-complete.

Moreover, as any hyperedge of $\mathcal{H}$ has size at most 6, the problem is still PSPACE-complete restricted to hypergraphs of rank 6. □

As it was done by Rahman and Watson for Maker-Breaker games [RW21], or by Gledel and myself for Avoider-Enforcer games [GO23], a next step of the study would be to transform the hypergraph into a uniform one, to get a stronger result. We achieved this with the following lemma:

Lemma 2.18. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph of rank k. Let $m = \min_{e \in E} |e|$. If $m < k$, there exists a hypergraph $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$ of rank k where $\min_{e \in E} |e| = m+1$, having $|\mathcal{F}'| \leq |\mathcal{F}| + \binom{2(k-1)}{k}$ and $|\mathcal{X}'| \leq |\mathcal{X}| + 2(k-1)$ such that Client has a winning strategy in the Client-Waiter game on $\mathcal{H}$ if and only if he has one in $\mathcal{H}'$.*

Proof. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph of rank k. We construct the hypergraph $\mathcal{H}' = (\mathcal{X}', \mathcal{F}')$. Let $A = \{a_1, \ldots, a_{2(k-1)}\}$ be $2(k-1)$ new vertices, and set $\mathcal{X}' = \mathcal{X} \cup A$. We make A a block, i.e. we define the set $U = \{B \subset A \mid |B| = k\}$ of hyperedges. Finally, we introduce $L = \{e \cup \{a_1\} \mid e \in \mathcal{F} \text{ and } |e| = m\}$. Finally, we define our hyperedges as follows:

$$\mathcal{F}' = \{e \in \mathcal{F} \mid |e| \geq m+1\} \cup L \cup U$$

The construction is depicted in Figure 2.4.

We have that Client wins in $\mathcal{H}$ if and only if he wins in $\mathcal{H}'$. Indeed, suppose that Client wins in $\mathcal{H}$, as A is a block, Waiter will have to propose the vertices of A two by two. Therefore, Client can claim a_1 when it will be proposed and apply his strategy on $\mathcal{H}$. Then, if Client had filled up a hyperedge $e \in \mathcal{F}$, if $|e| = m$, he has filled up $e \cup \{a_1\}$ in $\mathcal{H}'$, and if $|e| \geq m+1$, he has filled up e in $\mathcal{H}'$. Reciprocally, if Waiter has a winning strategy in $\mathcal{H}$, by proposing the same pairs in $\mathcal{H}'$ and the vertices of A two by two, for any hyperedge $e \in \mathcal{F}'$, either $e \in WS$ or $e \setminus \{a_1\} \in \mathcal{F}$, in both cases, Waiter has claimed a vertex of it by her winning strategy in $\mathcal{H}$. Otherwise, $e \in U$, in this case, she has not lost of e as Client has claimed $k-1$ vertices of A and $|e| = k$. □

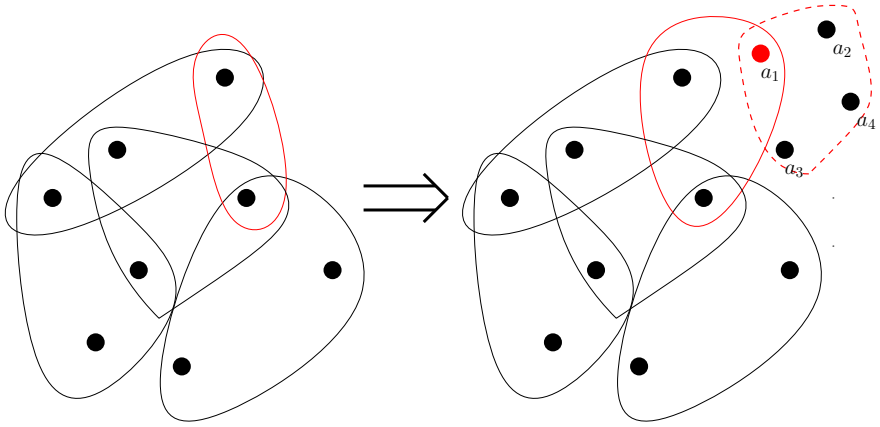


Figure 2.4: The construction of Lemma 2.18 with $k = 3$ and $m = 2$. The dashed set is a block and contains the four hyperedges of size 3. The resulting hypergraph is 3-uniform.

Corollary 2.19. *Client-Waiter games are* PSPACE*-complete even restricted to* 6*-uniform hypergraphs.*

Proof. The hypergraph obtained in the proof of Theorem 2.10 has rank 6 and all its hyperedges have size at least two. Therefore, by applying four times Lemma 2.18, with $m = 2, 3, 4, 5$, we obtain a hypergraph having at most $4 * 10$ more vertices and $4 * \binom{10}{6} = 840$ more edges. Thus this construction is still polynomial, and the hypergraph obtained is 6-uniform. □

2.3 Applications

Most of the proofs that Maker-Breaker games on graphs are PSPACE-complete are reductions from 6-uniform POS CNF, i.e. the general Maker-Breaker game. Therefore, the proofs that Avoider-Enforcer and Client-Waiter games are PSPACE-complete makes it possible to reduce these games to other Avoider-Enforcer or Client-Waiter games on graphs. In the next section, we illustrate the power of these theorems by proving that two other conventions of the domination game are PSPACE-complete.

The Maker-Breaker domination game has been introduced in 2020 by Duchêne *et al.* [DGPR20] and was proved PSPACE-complete by reduction from the general 6-uniform Maker-Breaker game. We prove here that, thanks to the PSPACE-completeness of Avoider-Enforcer and Client-Waiter games, the domination game is also PSPACE-complete under Avoider-Enforcer and Waiter-Client conventions. We provide a construction for the two conventions similar to the one provided by Duchêne *et al.*, and prove that the domination game is still PSPACE-complete restricted to split graphs under Avoider-Enforcer and Waiter-Client conventions.

2.3.1 The construction of the graph

Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. The construction provided is similar to the one provided by Duchêne *et al.* [DGPR20] in the Maker-Breaker convention. We construct the following graph $G = (V, E)$ as follows:

- For each vertex u_i in $\mathcal{X}$, we add a vertex v_i in V. Denote by K the set of vertices created during this step.
- For each hyperedge C in $\mathcal{F}$, we add p new vertices $v_C^1, \ldots, v_C^p$ in V. Denote by S the set of vertices created during this step. The value of p will be provided depending on the convention.
- For each pair of vertices $u_i, u_j \in \mathcal{X}$, we add an edge between v_i and v_j in E.
- If a vertex u_i of $\mathcal{X}$ belongs to a hyperedge C of $\mathcal{F}$, we add the edges $(v_i, v_C^1), \ldots, (v_i, v_C^p)$ to E.

Note that K is a clique and S a stable set. Therefore, G is a split graph.

The graph obtained is depicted in Figure 2.5 for $p = 2$. The next two section will consist in the proofs that the reduction provided attests that the Avoider-Enforcer domination game and the Waiter-Client domination games are PSPACE-complete.

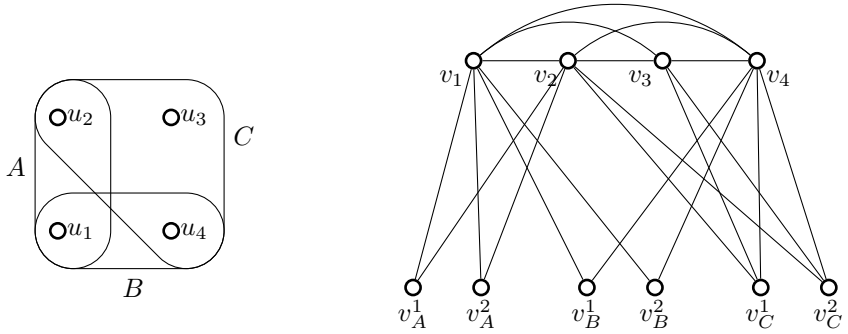


Figure 2.5: Reduction from a hypergraph to a graph in the Domination game.

2.3.2 Avoider-Enforcer domination game

We recall that, in the Avoider-Enforcer domination game, Avoider loses if the vertices she claims a dominating set of the graph. Otherwise, Enforcer loses. In order to differentiate the players of the Avoider-Enforcer game on $\mathcal{H}$ and the players of the Avoider-Enforcer domination game, we will call Eric, the player who wants not to dominate the graph and Aline his opponent.

Theorem 2.20. *The Avoider-Enforcer domination game is* PSPACE*-complete, even restricted to split graphs.*

Proof. First, the Avoider-Enforcer domination game is in PSPACE, as the number of moves in the game is the number of vertices, and as determining if a set is a dominating set or not can be done in polynomial time, the game is in PSPACE.

Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and $G = (V, E)$ be the graph constructed in Section 2.3.1 with $p = 2$.

Suppose Avoider (resp. Enforcer) has a winning strategy $\mathcal{S}$ in $\mathcal{H}$. We define a strategy $\mathcal{S}'$ for Aline (resp. Eric) in G as follows.

- If Avoider (resp. Enforcer) plays the first move in H, Aline (resp. Eric) claims first a vertex v_i such that u_i is the first vertex claimed in $\mathcal{S}$.
- If the opponent claims a vertex v_i, she (resp. he) claims a vertex v_j such that u_j is the answer to the vertex u_i in $\mathcal{S}$.
- If a player plays a vertex v_C^k for $k \in \{1, 2\}$, she (resp. he) claims the vertex $v_C^{k'}$ for $k' \neq k \in \{1, 2\}$.

Now if Avoider had a winning strategy in $\mathcal{H}$, by applying the strategy $\mathcal{S}'$, for any vertex v_C^i, Aline has not played all the v_js adjacent to it. Therefore, Eric has claimed one of them and all the v_C^is are dominated. Moreover, all the vertex u_js are forming a clique, and therefore dominated by Eric, as he has played at least once in K. Finally, Eric has dominated the graph and Aline has won.

Reciprocally, if Enforcer had a winning strategy in $\mathcal{H}$, by applying the strategy $\mathcal{S}'$, Eric knows that there exists a pair of vertices (v_C^1, v_C^2), such that Aline has played all the v_js adjacent to them. As Aline has played one of them, and all its neighbors, this vertex is not dominated. Therefore, Eric has won. □

2.3.3 Waiter-Client domination game

The graph G considered for the Waiter-Client domination game is the one constructed in Subsection 2.3.1 with $p = 2(n + m)$, where $n = |V_H|$ and $m = |E_H|$. We recall that, in the Waiter-Client Domination game, Waiter aims to claim a Dominating set, while Client aims to claim a vertex and all its neighbors, preventing Waiter to dominate. In order to differentiate the players of the Client-Waiter game on H and the players of the Waiter-Client domination game, we will call Dominator, the player who wants to fill up a hyperedge (Waiter), and Staller her opponent (Client). Note that we consider here the Waiter-Client convention and not the Client-Waiter one as we consider the neighborhood of the vertices as hyperedges rather than the dominating sets.

Theorem 2.21. *The Waiter-Client domination game is* PSPACE*-complete, even restricted to split graphs, when Waiter aims to dominate the graph.*

Proof. Suppose that Waiter has a winning strategy $\mathcal{S}$ in the Client-Waiter game on H. We provide the following strategy for Dominator on G:

- For each winning set $e \in E_H$, she proposes the vertices $v_e^1, \ldots, v_e^{2(n+m)}$ two by two.
- Now if $\mathcal{S}$ wants to propose a pair of vertices (u_i, u_j) in H, she proposes the pair (v_i, v_j) in G.

Following this strategy, as $\mathcal{S}$ was a winning strategy in H, for any hyperedge $e \in E_H$, there exists a vertex u_i claimed by Waiter. Therefore, Waiter has also claimed v_i in G. Thus, Dominator dominates all the vertices $v_e^1, \ldots, v_e^{2(n+m)}$. As the v_is make a clique, she also dominates the v_is as she has claimed at least one of them. Therefore, Dominator wins.

Conversely, suppose that Client has a winning strategy $\mathcal{S}$ in the Client-Waiter game on H. We provide the following strategy for Staller on G:

- If Dominator proposes a pair (v_i, v_j) corresponding to a pair (u_i, u_j) in H, Staller chooses the vertex corresponding to the one taken by $\mathcal{S}$ in H, ignoring the possible other vertices v_i he has claimed.
- If Dominator proposes a pair containing exactly one vertex v_i corresponding to a vertex u_i in H, Staller takes u_i.
- If Dominator proposes a pair of two vertices (v_e^i, v_f^j), if Staller has currently claimed no vertex v_e^k for $1 \leq k \leq 2(n + m)$, he claims v_e^i, otherwise, he claims v_f^j.

Following this strategy until all the vertices are claimed, if a vertex u_i would be claimed according to $\mathcal{S}$ by Client in H, v_i will also be claimed by Staller in G. Moreover, as there are $2(n + m)$ copies of each clause vertex, Client with the third point of the strategy ensure to take at least one copy of each. Therefore, if $e \in E_H$ is a hyperedge that was filled up by Client, consider a vertex v_e^k claimed by Staller in G, all its neighbor will also be claimed by Staller, and therefore, Staller wins. □

Note that even if we proved that the Waiter-Client domination game is PSPACE-complete, this does not prove that general Waiter-Client games are PSPACE-complete as a graph generally has an exponential number of dominating sets. Therefore, the reduction to the hypergraph corresponding to the domination game is not a polynomial reduction.

2.4 Further work

In this chapter, we proved that determining the winner in a 6-uniform Avoider-Enforcer game or Client-Waiter game is a PSPACE-complete problem. The two natural questions concern the optimality of the 6 and the other conventions.

In the reduction provided in the Avoider-Enforcer convention, most hyperedges were of size at most 4. In fact, only the clause hyperedges could be of order 6. Therefore, we can expect a better bound by reducing the size of this gadget.

On the other side, in contrast to Maker-Maker games, where rank 2 hypergraphs are in P, in Avoider-Avoider games, it is already PSPACE-complete to determine the winner in a 2-uniform hypergraph [BH19]. Moreover, the main argument in the proof that 3-uniform Maker-Breaker games are in P is the fact that Maker can build a simple structure on which she will win with her first moves. This phenomenon cannot occur in Avoider-Enforcer, isolated vertices are played first and not last, without giving any information about the outcome of the game. More generally, in Avoider-Enforcer games, it is the last moves that matter, and not the first moves. This makes the first moves approach difficult to apply. Therefore, the computational complexity of rank 3 Avoider-Enforcer games can be a good problem to look at.

However, in Client-Waiter games, 3-uniform hypergraphs seem to be easier to handle. After the first claim of Client, some hyperedges have only two vertices unclaimed. Thus, according to Lemma 1.79, Waiter can propose these two vertices in an optimal strategy. It follows that many moves can be forced which makes the problem unlikely to be PSPACE-hard. Therefore, we make the following conjecture:

Conjecture 2.22. *Determining the winner of a Client-Waiter game on a hypergraph of rank 3 is in* P*.*

The last convention for which no hardness result is known is the Waiter-Client convention. However, the proof of Theorem 1.70 seems to be adaptable to larger ranks, up to considering more moves for Waiter. Indeed, if $\mathcal{H}$ is k-uniform and has 2^k disjoint hyperedges, Waiter can easily win by a dichotomy strategy. Therefore, the hypergraphs to be considered for a reduction must have a minimum transversal of bounded size, which makes the game unlikely to be PSPACE-hard. Therefore, this convention seems to be very different from the others, and we make the following conjecture:

Conjecture 2.23. *Let k be an integer. Determining the winner of a Waiter-Client game restricted to hypergraphs of rank k is in* P*.*

Chapter 3

Maker-Breaker games on edges

A classic.

In Chapter 1, we saw that the study of the complexity of Maker-Breaker games on graphs is a trendy ongoing work. However, even if most of the games studied on complete graphs are played on the edges of graphs, most of the recent work on the complexity of games considers games played on the vertices, and only few of them are known to be PSPACE-complete. We initiate here the study of the complexity of the most classical Maker-Breaker games played on the edge set of graphs. Since the connectivity game is already known to be in P by Lehman [Leh64], we present here results for two of the other most studied games: the H-game and the perfect matching game. In the case of the H-game, Galliot *et al.* [Gal23] proved that 3-uniform Maker-Breaker games can be solved in polynomial time. Therefore, if H has three edges, the winner of the H-game can be computed thanks to their algorithm. However, for general graphs, there are no known result, and we provide here the first proofs of PSPACE-completeness of positional games played on the edges of graphs.

Instead of stating the results game by game, we will organize them according to the nature of the complexity of the study, i.e. PSPACE-hard, polynomial or FPT, as the proof of the PSPACE-hardness between the two games have similarities. In Section 3.1, we prove that both the H-game and the perfect matching game are PSPACE-complete. Then in Section 3.2, we provide some positive results on some particular instances of the H-game, in particular we prove that if H is a star and G a tree, the winner can be computed in polynomial time. Finally, in Section 3.3, we generalize the latest result by considering either that G can be a general graph, or that H can be a tree.

This work was a collaboration with Eric Duchêne, Valentin Gledel, Fionn Mc Inerney, Nicolas Nisse, Aline Parreau and Miloš Stojaković [DGI+23].

3.1 PSPACE-completeness results

In this section, we prove that it is PSPACE-complete to decide who wins the perfect matching game and the H-game via reductions from POS CNF. Note that in [RW21], it is also proven that POS CNF remains PSPACE-complete if Falsifier goes first and/or the number of variables n is odd. This property will be used in our reductions.

3.1.1 PSPACE-completeness of the perfect-matching game

We first prove that it is PSPACE-complete to determine the outcome of the perfect matching game. To simplify the reduction, the proof is done allowing parallel edges, that the following lemma enables to remove, see Figure 3.1.

Lemma 3.1. *Let G be any graph containing two parallel edges e_1 and e_2 connecting two of its vertices u and v. Then, there exists a complete bipartite graph H, of bounded order, with bipartition (A, B) such that, if e_1 and e_2 are removed from G, and a copy of H is added instead, where u (v, respectively) is made adjacent to two vertices of A (B, respectively), then the outcome of the perfect matching game in the resulting graph G' is the same as that in G.*

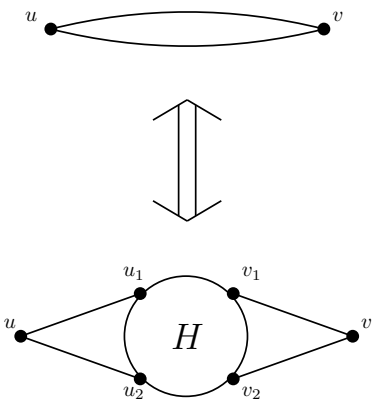


Figure 3.1: How parallel edges are removed in the perfect matching game in Lemma 3.1.

Proof. With the parallel edges e_1 and e_2 in G, according to the Super Lemma, we can suppose that Maker claims e_1 and Breaker claims e_2 in the perfect matching game. That means that, playing in G, she has two possibilities to claim a perfect matching:

(i) claim a perfect matching in $G - \{e_1, e_2\}$, or

(ii) claim a perfect matching in $G \setminus \{u, v\}$.

Let H be a balanced complete bipartite graph with parts A and B, and let $u_1, u_2 \in A$ and $v_1, v_2 \in B$. We will show that when $|A| = |B| = k$, for a suitable constant k to be chosen later, Maker, playing second, has a strategy in H that satisfies the following five conditions:

- Maker has a perfect matching in H, and
- for every $i, j \in \{1, 2\}$, Maker has a perfect matching in $H \setminus \{u_i, v_j\}$.

Once this is shown, forming G' by replacing the parallel edges e_1 and e_2 in G by H, and adding the two pairs of edges connecting u to u_1 and u_2, and v to v_1 and v_2, as depicted in Figure 3.1, finishes the proof. Indeed, assume Maker, playing second, has a strategy in H to satisfy the above-mentioned five conditions, and, on top of that, in G', she pairs uu_1 with uu_2, and vv_1 with vv_2. Then, she can claim a perfect matching in G' if and only if she can satisfy (i) or (ii) in G, which corresponds to her being able to claim a perfect matching in G.

To show that Maker can satisfy the five conditions playing second in H, we define an auxiliary positional game in H. Let

$$\mathcal{F}_H := \{E(X, Y) \mid X \subseteq A,\, Y \subseteq B,\, |X| + |Y| = k + 1\},$$

and, for every $i, j \in \{1, 2\}$, let

$$\mathcal{F}_{i,j} := \{E(X, Y) \mid X \subseteq A \setminus \{u_i\},\, Y \subseteq B \setminus \{v_j\},\, |X| + |Y| = k\}.$$

In the auxiliary game, Maker will assume the role of Breaker (to avoid confusion, we will call this player Auxiliary Breaker), trying to claim one edge in every winning set in $\mathcal{F} := \mathcal{F}_H \cup \big(\cup_{i,j \in \{1,2\}} \mathcal{F}_{i,j}\big)$. If she achieves

that, then Hall's condition [Hal35] for the existence of a perfect matching in a bipartite graph implies that Maker will have a claimed perfect matching in H, as well as in $H \setminus \{u_i, v_j\}$, for every $i, j \in \{1, 2\}$.

To show that Auxiliary Breaker can win the auxiliary game in H, we apply the Erdős-Selfridge Criterion (see Theorem 1.25):

$$\begin{aligned}
\sum_{E' \in \mathcal{F}} 2^{-|X|} &\leq \sum_{\ell=1}^{k} \binom{k}{\ell}\binom{k}{k-\ell+1} 2^{-\ell(k-\ell+1)} + 4\sum_{\ell=1}^{k-1} \binom{k-1}{\ell}\binom{k-1}{k-\ell} 2^{-\ell(k-\ell)} \\
&\leq \sum_{\ell=1}^{k} \binom{k}{\ell}\binom{k}{k-\ell+1} 2^{-\ell(k-\ell)} + 4\sum_{\ell=1}^{k} \binom{k}{\ell}\binom{k}{k-\ell} 2^{-\ell(k-\ell)} \\
&= \sum_{\ell=1}^{k} \binom{k}{\ell}\binom{k}{\ell-1} 2^{-\ell(k-\ell)} + 4\sum_{\ell=1}^{k} \binom{k}{\ell}^2 2^{-\ell(k-\ell)} \\
&\leq 2\sum_{\ell=1}^{\lceil k/2 \rceil} \binom{k}{\ell}^2 2^{-\ell(k-\ell)} + 8\sum_{\ell=1}^{\lceil k/2 \rceil} \binom{k}{\ell}^2 2^{-\ell(k-\ell)} \\
&\leq 10\sum_{\ell=1}^{\lceil k/2 \rceil} k^{2\ell} 2^{-\ell(k-\ell)} \\
&\leq 10\sum_{\ell=1}^{\lceil k/2 \rceil} k^{2\ell} 2^{-\ell(k-\lceil k/2 \rceil)} \\
&= 10\sum_{\ell=1}^{\lceil k/2 \rceil} \left(2^{2\log_2 k - \lfloor k/2 \rfloor}\right)^{\ell}.
\end{aligned}$$

This expression tends to zero when k grows, and hence, for k large enough, it is less than $1/2$, and so, the Erdős-Selfridge criterion implies Auxiliary Breaker's win. □

Theorem 3.2. *Deciding whether Maker wins the perfect matching game in a given graph G is* PSPACE*-complete.*

Proof. The problem is clearly in PSPACE since both the number of turns and the number of possible moves at each turn are bounded from above by n^2, and determining whether a set of edges contains a perfect matching or not can be done in polynomial space. To prove it is PSPACE-hard, we give a reduction from POS CNF where there are an odd number of variables, which, as mentioned before, is PSPACE-hard [RW21].

Let ϕ be an instance of POS CNF where there are an odd number of variables. Denote the variables in ϕ by $x_1, \ldots, x_{2n+1}$, and the clauses in ϕ by $C_1, \ldots, C_m$. From ϕ, we construct the graph G as follows, and recall that, by Lemma 3.1, we can allow pairs of parallel edges.

- For all $1 \leq i \leq 2n+1$, introduce two vertices $v^i_{i_0}$ and $\overline{v}^i_{i_0}$, and the edge $e_i = v^i_{i_0}\overline{v}^i_{i_0}$.
- Then, add n new vertices $a_1, \ldots, a_n$, and, for all $1 \leq i \leq 2n+1$ and $1 \leq \ell \leq n$, add two parallel edges between $v^i_{i_0}$ and a_ℓ. See Figure 3.2 for an illustration.
- For each clause C_j in ϕ, add a vertex C^j in G.
- For each variable x_i in ϕ, let $C_{i_1}, \ldots, C_{i_{k_i}}$ be the clauses containing x_i in ϕ. For all $1 \leq i \leq 2n+1$ and for all $1 \leq j \leq k_i$, add the vertices $u^i_{i_j}, \overline{u}^i_{i_j}, v^i_{i_j}, \overline{v}^i_{i_j}$, $x^i_{i_j}$, and $y^i_{i_j}$. Also, for all $1 \leq i \leq 2n+1$, add the vertices $y^i_{i_{k_i}+1}$. Then, connect them as follows for all $1 \leq i \leq 2n+1$ and $1 \leq j \leq k_i$ (see Figures 3.3 and 3.4):
  - Add the two edges $\overline{v}^i_{i_{j-1}}u^i_{i_j}$ and $\overline{v}^i_{i_{j-1}}\overline{u}^i_{i_j}$.

- Add two parallel edges between $u^i_{i_j}$ and $x^i_{i_j}$.
- Add two parallel edges between $x^i_{i_j}$ and C^{i_j}.
- Add two parallel edges between $x^i_{i_j}$ and $y^i_{i_j}$.
- Add two parallel edges between $u^i_{i_j}$ and $\overline{u}^i_{i_j}$.
- Add two parallel edges between $\overline{u}^i_{i_j}$ and $v^i_{i_j}$.
- Add two parallel edges between $v^i_{i_j}$ and $\overline{v}^i_{i_j}$.

Also, for all $1 \leq i \leq 2n+1$, add two parallel edges between $\overline{v}^i_{i_{k_i}}$ and $y^i_{i_{k_i}+1}$.

- If $|V(G)|$ is currently an odd number, then add the vertex $y^0_{0_0}$ in G.
- For each pair of vertices among the $y^i_{i_j}$'s (including $y^0_{0_0}$ if it exists), add two parallel edges between them (see Figure 3.5).

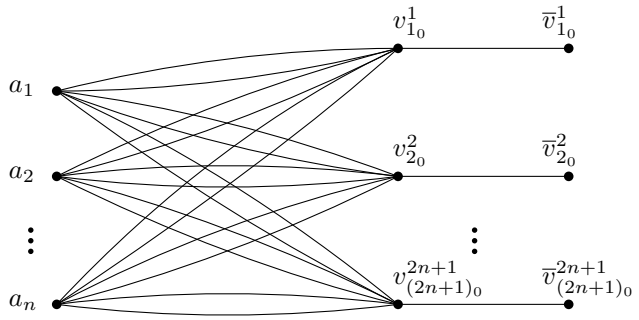


Figure 3.2: The variable gadget in the graph G constructed in the proof of Theorem 3.2.

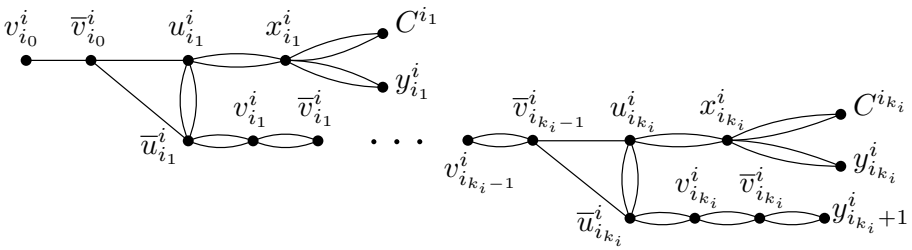


Figure 3.3: Construction for a variable x_i in clauses $C^{i_1}, \ldots, C^{i_{k_i}}$ in ϕ in the graph G constructed in the proof of Theorem 3.2.

Note that G is clearly constructed in polynomial time. We prove that Satisfier wins in ϕ if and only if Maker wins the perfect matching game in G. By the Super Lemma, the outcome of the perfect matching game in G is the same as the outcome in G, where, for each pair of parallel edges, both Maker and Breaker have claimed one of the two edges, and so, we can assume they have done so in what follows.

We first give the idea above this construction:

- The vertices a_i will be matched to n of the $v^i_{i_0}$, so Maker will have to play the $n+1$ other edges adjacent to the $v^i_{i_0}$s in her first $n+1$ moves, otherwise she loses by not matching one of them.

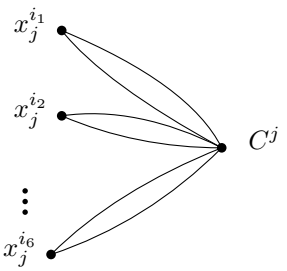


Figure 3.4: Construction for a clause $C_j = (x_{i_1} \vee \cdots \vee x_{i_6})$ in ϕ in the graph G constructed in the proof of Theorem 3.2.

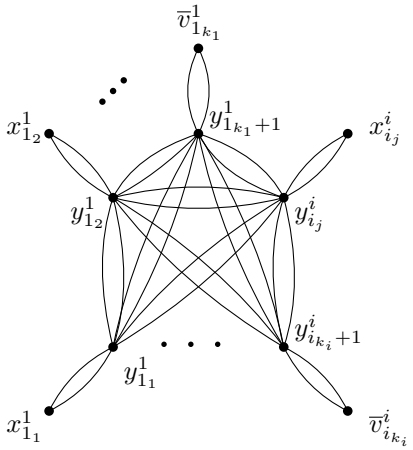


Figure 3.5: The $y^i_{i_j}$ vertices in the graph G constructed in the proof of Theorem 3.2.

- The gadget going from $v^i_{i_0}$ to $v^i_{i_{k_i}}$ is made so that if Maker has played $(v^i_{i_0}, \overline{v^i_{i_0}})$, she can take in her matching the edges $(u^i_{i_1}, \overline{u^i_{i_1}})$, $\ldots$, $(u^i_{i_{k_i}}, \overline{u^i_{i_{k_i}}})$ and the edges $(v^i_{i_0}, \overline{v^i_{i_0}}), \ldots, (v^i_{i_{k_i}}, \overline{v^i_{i_{k_i}}})$, letting the vertices $x^i_{i_1}, \ldots, x^i_{i_{k_i}}$ available to match the clauses. If she has not claimed $(v^i_{i_0}, \overline{v^i_{i_0}})$, Breaker can force her moves to have the vertex $x^i_{i_k}$ of his choice matched with the vertex $u^i_{i_k}$, and therefore not matching some clause vertex.
- All the vertices y^i_j are used to match the vertices remaining of the previous gadget, so that Maker does not lose outside of a clause. In particular, they will be matched with the vertices x^i_j that are in clauses which is already matched with an other variable in it.

First, we prove the simpler of the two directions, that is, if Falsifier wins in ϕ, then Breaker wins the perfect matching game in G. Assume that Falsifier has a winning strategy $\mathcal{S}$ in ϕ. Consider the following strategy for Breaker in G:

- If Maker claims an edge $e_i = v^i_{i_0}\overline{v}^i_{i_0}$, then Breaker answers by claiming an edge $e_j = v^j_{j_0}\overline{v}^j_{j_0}$, where x_j is the variable that would have been claimed by Falsifier in ϕ according to $\mathcal{S}$ if Satisfier claimed x_i in ϕ.
- If Maker claims any edge other than an e_i before all the e_i's have been claimed, then Breaker claims an arbitrary e_i. Then, as there is an odd number of e_i's, by pairing them, Breaker can ensure claiming at least $n+1$ of them. Thus, by construction, Maker will not be able to have a perfect matching containing all the $v^i_{i_0}$'s. Indeed, after this step, at least $n+1$ of the e_i's are claimed by Breaker, and so, their respective $n+1$ $v^i_{i_0}$'s must be matched with their only remaining neighbors, the a_ℓ's, of which there are only n, and thus, this is not possible.

Hence, we can assume that all the edges e_i have been claimed during the first $2n+1$ moves. Consider the valuation obtained in ϕ if all the x_i variables associated to the e_i edges claimed by Maker are the variables set to true, and all the x_i variables associated to the e_i edges claimed by Breaker are the variables set to false. By the hypothesis that $\mathcal{S}$ is a winning strategy for Falsifier in ϕ, there exists a clause C_z that is not satisfied by this valuation in ϕ. Let $x_{f_1}, \ldots, x_{f_6}$ be the variables in C_z in ϕ. Since Breaker claimed all the edges e_i corresponding to the variables x_i in ϕ that Falsifier would have claimed according to $\mathcal{S}$, the edges $e_{f_1}, \ldots, e_{f_6}$ are claimed by Breaker in G. Recall that, as the number of variables is odd, it is Breaker's turn. Breaker plays as follows for all $\ell \in \{f_1, \ldots, f_6\}$ and for $j = 1$ to k_ℓ while $\ell_j \leq z$:

- If $\ell_j < z$, then Breaker claims $\overline{v}^\ell_{\ell_{j-1}} \overline{u}^\ell_{\ell_j}$. As the only remaining edge available to match $\overline{v}^\ell_{\ell_{j-1}}$ is $\overline{v}^\ell_{\ell_{j-1}} u^\ell_{\ell_j}$, Maker has to claim it. Now, all the edges adjacent to $\overline{u}^\ell_{\ell_j}$ have been claimed, and Maker has claimed only two of them, of which only one does not interfere with the edges already forced in the matching: $\overline{u}^\ell_{\ell_j} v^\ell_{\ell_j}$. Thus, $\overline{u}^\ell_{\ell_j} v^\ell_{\ell_j}$ has to be in any perfect matching claimed by Maker, which forces the edge $v^\ell_{\ell_j} \overline{v}^\ell_{\ell_j}$ claimed by Maker to not be in the matching.
- If $\ell_j = z$, then Breaker claims $\overline{v}^\ell_{\ell_j - 1} u^\ell_z$, forcing Maker to claim $\overline{v}^\ell_{\ell_j - 1} \overline{u}^\ell_z$ to match $\overline{v}^\ell_{\ell_j - 1}$. Now, the only edge that Maker can use to match u^ℓ_z is the edge $u^\ell_z x^\ell_z$.

By the above strategy for Breaker, any perfect matching contained in the edges claimed by Maker has to contain the edges $u^\ell_z x^\ell_z$ for all $\ell \in \{f_1, \ldots, f_6\}$. Therefore, all the vertices adjacent to C^z are already matched, and thus, it cannot be matched, and Breaker wins.

Now, we prove that if Satisfier wins in ϕ, then Maker wins the perfect matching game in G. Assume that Satisfier has a winning strategy $\mathcal{S}$ in ϕ. We construct a strategy for Maker in G as follows:

First, Maker claims the edge $e_i = v^i_{i_0} \overline{v}^i_{i_0}$ corresponding to the variable x_i that Satisfier would have claimed first in ϕ according to $\mathcal{S}$. Then,

- If Breaker claims an edge $e_j = v^j_{j_0} \overline{v}^j_{j_0}$, then Maker claims the edge $e_i = v^i_{i_0} \overline{v}^i_{i_0}$ corresponding to the variable x_i that Satisfier would have claimed according to $\mathcal{S}$ if Falsifier had claimed x_j in ϕ.
- For any variable x_i in a clause C_j in ϕ, Maker pairs the edges $\overline{v}^i_{i_{j-1}} u^i_{i_j}$ and $\overline{v}^i_{i_{j-1}} \overline{u}^i_{i_j}$ in G, and so, if Breaker claims one of them, she claims the other one.

Note that by construction, one of these moves is always available, as after the first move of Maker, any set where moves are considered has an even number of unclaimed edges remaining.

Suppose that Maker employs this strategy until the end of the game. Then, she will have claimed the e_i edges corresponding to the x_i variables claimed by Satisfier in ϕ according to $\mathcal{S}$. We extract a perfect matching from the edges she claimed as follows:

- Add in the matching, the $n+1$ e_i's claimed by Maker. The n $v^i_{i_0}$'s that are not in these edges are paired with the n a_ℓ's.
- Let $i_1, \ldots, i_{n+1}$ be the indices of the e_i edges claimed by Maker. For $\ell = i_1$ to i_{n+1}, add in the matching all the edges $u^\ell_j \overline{u}^\ell_j$ and $v^\ell_j \overline{v}^\ell_j$ such that x_ℓ is in C_j in ϕ.
- For each clause C_j, as at least one variable x_i in C_j in ϕ corresponding to an edge x_i claimed by Maker is set to true, consider such an i, and add the edge $x^i_j C^j$ in the matching. For each other variable x_ℓ in C_j in ϕ corresponding to an edge x_ℓ claimed by Maker ($\ell \neq i$), add the edge $x^\ell_j y^\ell_j$ in the matching.
- For each variable x_i with $i \notin \{i_1, \ldots, i_{n+1}\}$ being in clauses $C_{i_1}, \ldots, C_{i_{k_i}}$ in ϕ:
  - For $j = 1$ to k_i, while Maker has claimed $\overline{v}^i_{i_{j-1}} u^i_{i_j}$, add in the matching the edges, $\overline{v}^i_{i_{j-1}} u^i_{i_j}$, $x^i_{i_j} y^i_{i_j}$, and $\overline{u}^i_{i_j} v^i_{i_j}$. If and once Maker has claimed an edge $\overline{v}^i_{i_{z-1}} \overline{u}^i_{i_z}$, add in the matching $\overline{v}^i_{i_{z-1}} \overline{u}^i_{i_z}$, $u^i_{i_z} x^i_{i_z}$, and $v^i_{i_z} \overline{v}^i_{i_z}$, and exit the for loop. Then, for $j = z+1$ to k_i, add in the matching the edges $u^i_{i_j} \overline{u}^i_{i_j}$, $x^i_{i_j} y^i_{i_j}$, and $v^i_{i_j} \overline{v}^i_{i_j}$.

– If Maker has claimed no edge $\overline{v}^i_{i_{z-1}}\overline{u}^i_{i_z}$ at the end, add in the matching $\overline{v}^i_{i_{k_i}}y^i_{i_{k_i}+1}$.

- Note that this strategy matches all the vertices except some of the $y^i_{i_j}$'s (including $y^0_{0_0}$ if it exists). As G contains an even number of vertices, and every pair of remaining vertices is connected by two parallel edges, by considering any matching among the remaining $y^i_{i_j}$'s, Maker has claimed a perfect matching and wins. □

Usually, after a game has been proved to be PSPACE-hard, we study the complexity of that game on simpler classes of graphs. However, in the case of the perfect matching game, it is quite difficult to focus on standard classes of graphs on which there are no direct results. In fact, in order to win the perfect matching game, the board must contain several perfect matchings, otherwise simple arguments can prove that Breaker wins. For instance, if G has two leaves, Breaker can always play an edge that disconnects one leaf from the rest of the graph and prevents Maker from claiming a perfect matching, which proves that trees with at least three vertices have outcome $\mathcal{B}$. More precisely, the following result holds:

Lemma 3.3. *Let $G = (V, E)$ be a graph with $|V| = 2n$. If the number of perfect matchings of G is less than 2^{n-1}, Breaker wins the perfect matching game on G.*

Proof. This result is straightforward using the Erdős-Selfridge criterion of Theorem 1.25. Indeed, as any perfect matching contains n edges, the criterion states that if:

$$\sum_{\substack{M\subset E\\ M \text{ perfect matching}}} 2^{-|M|} = \frac{|\{M \subset E \mid M \text{ perfect matching}\}|}{2^n} < \frac{1}{2}$$

Breaker wins. □

As a direct application of this Lemma, is G is a large enough grid, according to the bound on the number of perfect matchings provided by Benhmaram and Friedland [BF12], Breaker wins:

Theorem 3.4. *Let $G_{2n,m}$ be the grid with $2n$ rows and m columns. If $2n + m \geq 4$, Breaker wins the perfect matching game on $G_{2n,m}$.*

Note that the result is stated with $2n$ rows, as we need at least n or m even to have at least one perfect matching.

Proof. Denote by $pm(G)$ the number of perfect matchings of a graph G.

According to Behmaram and Friedland [BF12], a grid of size $2n * m$ satisfies:

$$pm(G_{2n,m}) = \prod_{v\in G_{2n,m}} d(v)^{\frac{1}{4}} = 4^{\frac{(2n-2)(m-2)}{4}} 3^{\frac{2(2n-2+m-2)}{4}} 2^{\frac{4}{4}} = 2^{nm-2n-m+3}\sqrt{3}^{2n+m-4}$$

This value has to be compared to the bound of Lemma 3.3, i.e. 2^{nm-1} as $G_{2n,m}$ has $2nm$ vertices. Therefore, the Erdős-Selfridge criterion can be applied if and only if

$$\begin{aligned}
& \frac{2^{nm-1}}{2^{nm-2n-m+3}\sqrt{3}^{2n+m-4}} > 1 \\
\iff \quad & \frac{2^{2n+m-4}}{\sqrt{3}^{2n+m-4}} > 1 \\
\iff \quad & \left(\frac{2}{\sqrt{3}}\right)^{2n+m-4} > 1 \\
\iff \quad & 2n + m > 4
\end{aligned}$$

To conclude the proof it remains to see that Maker wins the perfect matching game on the grid $G_{2,1}$ and that Breaker wins on the grid $G_{2,2}$. Any other grid of even order satisfies $2n + m \geq 5$ vertices. □

3.1.2 PSPACE-completeness of the H-game

We now prove that the H-game is PSPACE-complete in graphs of small diameter when H is a tree. Hence, even when H is a relatively basic graph, determining the outcome of the H-game is hard. In Section 3.3, we will prove that by being a bit more restrictive on the instance, we will be able to obtain FPT-results.

Theorem 3.5. *There exists a tree H such that deciding whether Maker wins the H-game in a given graph G is* PSPACE*-complete, even if G has diameter at most* 6.

Proof. We show that the statement holds for the tree H in Figure 3.6.

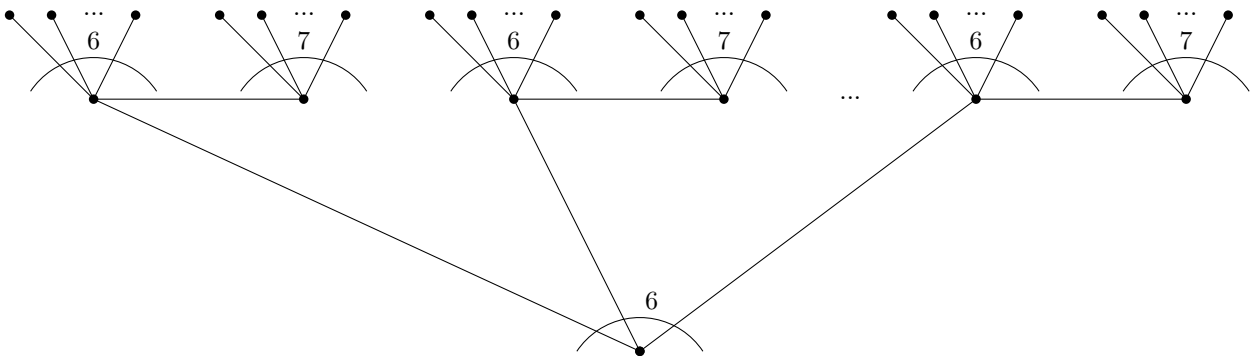


Figure 3.6: The tree H in the proof of Theorem 3.5.

The problem is clearly in PSPACE since both the number of turns and the number of possible moves at each turn are bounded from above by n^2 [Sch78, Lemma 2.2]. To prove it is PSPACE-hard, we give a reduction from POS CNF where Falsifier plays first, which as mentioned before, is known to be PSPACE-hard [RW21].

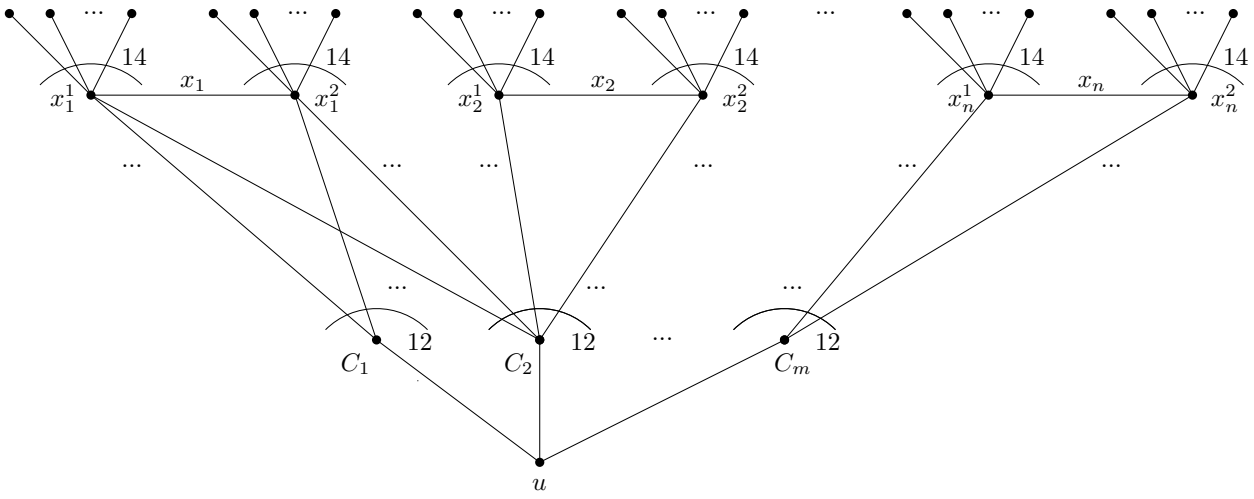


Figure 3.7: The graph G constructed in the proof of Theorem 3.5. In this example, the variable x_1 appears in the clauses C_1 and C_2, the variable x_2 appears in the clause C_2, and the variable x_n appears in the clause C_m.

Let ϕ be an instance of POS CNF in which Falsifier plays first. From ϕ, we construct the graph G as follows. For each clause C_j in ϕ, introduce a new clause vertex C_j in G, and, for each variable x_i in ϕ, introduce a new variable edge $x_i = x_i^1 x_i^2$ in G, all pairwise vertex-disjoint from each other. For all $1 \leq i \leq n$ and $1 \leq j \leq m$, if the variable x_i is contained in the clause C_j in ϕ, then add the edges $x_i^1 C_j$ and $x_i^2 C_j$ in

G. For each $1 \leq i \leq n$, add 28 vertices, and make 14 of them adjacent to x_i^1, and the other 14 adjacent to x_i^2. Lastly, to ensure G has diameter at most 6, add a vertex u and, for all $1 \leq j \leq m$, make it adjacent to C_j. See Figure 3.7 for an illustration of G, and note that G is clearly constructed in polynomial time.

We prove that Satisfier wins in ϕ (recall that Falsifier plays first) if and only if Breaker wins the H-game in G. First, we prove the following useful claims.

Claim 3.6. *Suppose that, for all $1 \leq i \leq n$ and $1 \leq j \leq m$ such that the edges $x_i^1C_j$ and $x_i^2C_j$ exist, Breaker claims at least one of $x_i^1C_j$ and $x_i^2C_j$. In that case, if Maker is to claim a copy of H in G, then the unique vertex of degree 6 in H must be a clause vertex in G.*

Proof of the claim. Every other vertex in G either has degree 1 or cannot be adjacent to 6 vertices that have at least 8 edges incident to each of them that Maker can claim, due to Breaker's strategy. ◇

Claim 3.7. *Suppose that, for all $1 \leq i \leq n$ and $1 \leq j \leq m$ such that the edges $x_i^1C_j$ and $x_i^2C_j$ exist, Breaker claims at least one of $x_i^1C_j$ and $x_i^2C_j$. In that case, if Maker is to claim a copy of H in G, then each of the pairs of adjacent degree-8 vertices in H must be the two vertices of a variable edge in G.*

Proof of the claim. By Claim 3.6, the unique vertex of degree 6 in H is a clause vertex. Thus, the 6 vertices of degree 8 adjacent to this vertex of degree 6 in H must each be a vertex of 6 different variable edges in G. Indeed, u cannot be one of these vertices since all of u's neighbors can have at most 7 incident edges claimed by Maker due to Breaker's strategy. Due to Breaker's strategy, for each of these vertices of degree 8, the only vertex adjacent to them that has at least 8 edges incident to it that Maker can claim, is the other vertex in each of the same variable edges. ◇

First, by the Super Lemma, we can suppose that the edges between any vertex x_i^j and a leaf for $1 \leq j \leq 2$ and $1 \leq i \leq n$ are equitably distributed among Maker and Breaker.

We prove that if Satisfier wins in ϕ, then Breaker wins the H-game in G. Assume that Satisfier wins in ϕ. Breaker employs the following pairing strategy. If Maker claims a variable edge x_i, then Breaker follows his winning strategy as Satisfier in ϕ by claiming the variable edge in G corresponding to the variable he wants to set to true in ϕ assuming that Maker just set the variable x_i to false in ϕ. If Maker claims an edge $x_i^1C_j$ (resp. $x_i^2C_j$), then Breaker claims $x_j^2C_j$ (resp. $x_j^1C_j$). Lastly, if Maker claims an edge incident to u, then Breaker claims another edge incident to u. Whenever Breaker cannot employ his strategy, he claims an arbitrary edge, and then goes back to following his strategy. For a contradiction, assume that, at the end of the game, Maker claimed a copy of H. Then, by Claims 3.6 and 3.7, there exists a clause such that all of the variable edges corresponding to the variables it contains in ϕ have been claimed by Maker. This contradicts the fact that Satisfier wins in ϕ since Breaker followed Satisfier's winning strategy in ϕ on the variable edges of G.

Now, we prove that if Falsifier wins in ϕ, then Maker wins the H-game in G. Assume that Falsifier wins in ϕ. Maker first claims a variable edge in G that corresponds to the variable she wants to set to false in ϕ according to her winning strategy as Falsifier in ϕ. Then, Maker employs the following pairing strategy. If Breaker claims a variable edge x_i, then Maker follows her winning strategy as Falsifier in ϕ by claiming the variable edge in G corresponding to the variable she wants to set to false in ϕ assuming that Breaker just set the variable x_i to true in ϕ. If Breaker claims an edge $x_i^1C_j$ (resp. $x_i^2C_j$), then Maker claims $x_j^2C_j$ (resp. $x_j^1C_j$). Lastly, if Breaker claims an edge incident to u, then Maker claims another edge incident to u. Whenever Maker cannot employ her strategy, she claims an arbitrary edge, and then goes back to following her strategy. Since Maker followed Falsifier's winning strategy in ϕ on the variable edges of G, for at least one clause, she will have claimed all of the variable edges corresponding to the variables contained in that clause in ϕ. We can easily locate a Maker's copy of H containing that clause's vertex as the unique vertex of degree 6 in H. □

As can be seen in Figure 3.6, the tree H from the proof of Theorem 3.5 has order 91. It would be interesting to know the order and/or size of the smallest graph H for which the H-game remains PSPACE-complete. As a step in this direction, in a more involved proof, we showed in [DGI+23] that the H-game is PSPACE-complete for a graph H of order 51 and size 57. It is worth noting that the orders and sizes of our

graphs H in our reductions are dependent on the fact that POS CNF is PSPACE-hard, but it is not known whether the analogously defined UNIFORM POS CNF 5 or UNIFORM POS CNF 4 are PSPACE-hard, which would allow for smaller H's to be constructed.

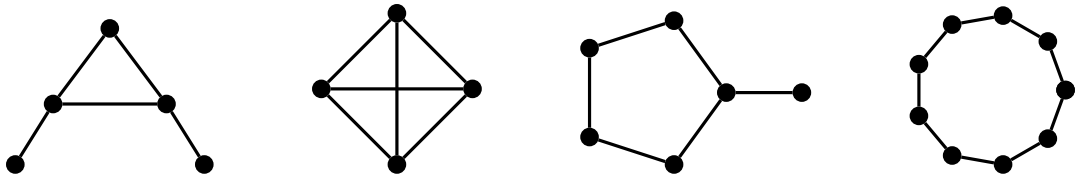
Theorem 3.8 (Duchêne, *et al.* [DGI+23])**.** *There exists a graph H of order 51 and size 57, such that deciding whether Maker wins the H-game in a given graph G is* PSPACE*-complete.*

3.2 Polynomial time algorithms

Knowing that the H-game is PSPACE-complete, we try to restrict the input of the problem to obtain positive results. Using the result of Galliot [Gal23], if H has three edges, the winner of the H-game can be computed in polynomial time. However, the algorithm provided by Galliot has complexity $O(|V(\mathcal{H})|^6)$ where $\mathcal{H}$ is the hypergraph of the game. In the case of the P_4-game, we improve this result by providing a linear time algorithm in any graph. Next, we consider that H is a star, i.e., that Maker aims to claim several edges adjacent to the same vertex. This game will be referred to as the $K_{1,\ell}$-game, and we provide here a polynomial time algorithm for solving the $K_{1,\ell}$-game in trees for any fixed integer $\ell \geq 1$.

3.2.1 Linear-time algorithm for the P_4game

We first give a necessary and sufficient structural condition for Breaker to win the P_4-game in any graph G. This leads to a linear-time algorithm to decide the outcome of the P_4-game.



(a) The bull (b) K_4 (c) C_5 with a leaf (d) Odd cycle

Figure 3.8: Maximal non-bipartite graphs for which Breaker wins the P_4-game.

Theorem 3.9. *For any connected graph G, Breaker wins the P_4-game in G if and only if*

1. *G is bipartite and all the vertices of degree at least 3 are in the same part; or*
2. *G is an odd cycle; or*
3. *G is a subgraph of the bull, K_4, or a C_5 with a leaf attached to one vertex (see Figure 3.8).*

Proof. We first prove the "if" part. One can check with a small case analysis that Breaker wins in the first three graphs of Figure 3.8, and thus, in all their subgraphs by Lemma 1.19. For the odd cycle, Breaker wins with the following strategy. Let $(e_1, \ldots, e_{2\ell+1})$ be the edges of the cycle in this order. Without loss of generality, let e_1 be the first edge claimed by Maker. Then, Breaker claims $e_{2\ell+1}$, and then follows a pairing strategy by pairing edges e_{2i} with e_{2i+1}, for $1 \leq i < \ell$ ($e_{2\ell}$ is not paired).

Thus, we can assume that G is bipartite with all the vertices of degree at least 3 in the same part. Let $V(G) = A \cup B$ be a bipartition of the vertices of G with the part A containing all the vertices of degree at least 3, and so, the vertices of B have degree at most 2. Note that any path on four vertices must contain an inner vertex in B, and thus, two edges incident to the same vertex in B. Thus, Breaker can win by following a pairing strategy where, for each vertex v of B, the (potentially) two edges incident to v are paired together.

We now prove the "only if" part. Let G be a graph that does not satisfy the three conditions of Theorem 3.9. The proof is divided into four cases:

1. G contains a diamond (K_4 minus one edge).
2. G contains a triangle, but not a diamond.
3. G contains an odd path between two vertices of degree at least 3, but not a triangle.
4. G contains an odd cycle with a unique vertex of degree at least 3.

These cases will cover all the possible graphs where Maker wins. Indeed, if G is bipartite, then it would be treated in Case 3. If G is not bipartite and not an odd cycle, then it has a vertex of degree at least 3 in an odd cycle C. If C contains a unique vertex of degree at least 3, then we are in Case 4. Otherwise, C contains two vertices of degree at least 3, and thus, there is an odd path between vertices of degree at least 3, and we are in Case 3 if G is triangle-free. Otherwise, G contains a triangle, but not a diamond (Case 2) or G contains a diamond (Case 1).

Case 1. G contains a diamond. Since G is not restricted to a subgraph of K_4, G contains a subgraph that is a diamond with a leaf connected to it (either to a vertex of degree 3 or to a vertex of degree 2 in the diamond). One can easily check that Maker wins the P_4-game in these subgraphs, and thus, by Lemma 1.19, she wins in all the graphs containing them.

Case 2. G contains a triangle, but not a diamond. Let uvw be this triangle. Since G is not a triangle (it would be a subgraph of the bull), at least one vertex, say u, must be connected to another vertex z. Note that z is not connected to any other vertex of the triangle, as otherwise G would contain a diamond. Since G is not restricted to the graph on the vertices $\{u, v, w, z\}$ (because otherwise it would be a subgraph of K_4), there is another vertex x in the graph.

Assume first that x is connected to z. Then, Maker claims zu. Then, she has a pairing strategy with pairs (xz, vw) and (uv, uw). Assume now that x is connected to u. Then, Maker claims vw, and then follows a pairing strategy with pairs (xu, zu) and (uv, uw). Finally, assume that x is connected to the triangle, without loss of generality, by v. Since G is not a bull and does not contain a diamond, there must either be an additional edge, and it can only be xz (otherwise G would contain a diamond) or there is another vertex connected to the graph. The first case returns to the case where x was connected to z. For the second case, the only possibility not covered yet is that there is a vertex t connected to w. Then, Maker claims uw. If Breaker then claims vx or vu, then Maker claims uz, and pairs tw with wv. Otherwise, w.l.o.g., Breaker then claims uz. Then, Maker claims uv and pairs tw with vx.

From now on, we can assume that G is triangle-free. Thus, Maker just needs to claim any three consecutively incident edges and will be sure to obtain a P_4.

Case 3. G contains an odd path between two vertices of degree at least 3, but not a triangle. Let u and v be two vertices of degree at least 3 connected by an odd path P. We choose u and v such that P has minimum length. Let $e_1, \dots, e_{2\ell+1}$ be the edges of P with e_1 incident to u, and $e_{2\ell+1}$ incident to v. Let e_u and e'_u be the two other edges incident to u, and e_v and e'_v the two other edges incident to v.

If $\ell = 0$, then Maker claims e_1 and follows a pairing strategy with pairs (e_u, e'_u) and (e_v, e'_v) (e_u and e'_u are vertex-disjoint from e_v and e'_v since G is triangle-free). Assume now that $\ell > 0$. Maker starts by claiming e_2. Assume first Breaker does not answer by claiming e_1. Then, Maker can claim e_1 as her second move. Then, either she pairs e_u with e'_u if Breaker did not claim any of these edges on his first move, or she pairs e_u or e'_u (the one that is unclaimed) with e_3. Therefore, we can assume that Breaker answers by claiming e_1.

Assume now, by induction on $1 < i < \ell$, that before their i^{th} moves, Maker has claimed all the even edges e_{2j}, and Breaker has claimed all the odd edges e_{2j-1} for $j = 1, \dots, i-1$ (we have shown this is true for $i = 2$ above). Then, on her i^{th} move, Maker claims e_{2i}. Breaker has to answer by claiming e_{2i-1}, since otherwise Maker can claim e_{2i-1}, creating a P_4 with e_{2i} and e_{2i-2}. Then, the inductive hypothesis holds for $i+1$. Repeating this argument, for her ℓ^{th} move, Maker claims $e_{2\ell}$, and Breaker has to answer by claiming $e_{2\ell-1}$. Then, Maker claims $e_{2\ell+1}$ and pairs e_v with e'_v, which will create a P_4.

Case 4. G contains an odd cycle with a unique vertex of degree at least 3. Let u be the unique vertex of degree at least 3 in an odd cycle in G. Let $e_1, \dots, e_{2\ell+1}$ be the edges of the cycle, with u incident to e_1 and $e_{2\ell+1}$. Let v be a vertex adjacent to u, but not in the cycle (it exists since u has degree at least 3). Since G is triangle-free, $\ell > 1$. Assume first that $\ell = 2$. Since G is not restricted to a C_5 with a leaf, and since any additional edge will create another vertex of degree at least 3 in the cycle, there must be another vertex w in the graph. If w is adjacent to u, then Maker claims e_2. Then, as before for the odd path, Breaker should answer by claiming e_1. Then, Maker claims e_4, and Breaker should answer by claiming e_3. Finally, Maker claims e_5, and then pairs uv with uw. If w is adjacent to v, then Maker claims uv. Breaker should answer by claiming vw, as otherwise Maker can make a P_3 with two free extremities. Then, Maker can force moves by claiming e_2 (Breaker then claims e_1), then e_4 (Breaker then claims e_3), and then win by claiming e_5.

Assume now that $\ell > 2$, *i.e.*, the cycle has length at least 7. Maker claims e_4. Breaker should claim either e_3 or e_5 to avoid a P_3 with two free extremities. If, w.l.o.g., Breaker claims e_3, Maker can again force moves by claiming the even edges $e_6, \dots, e_{2\ell}$. Breaker always has to answer by claiming the preceding odd edges $e_5, \dots, e_{2\ell-1}$. Then, Maker claims $e_{2\ell+1}$ and pairs e_1 with uv, making a P_4. □

The conditions given in Theorem 3.9 are checkable in linear time, which implies the following:

Corollary 3.10. *It can be decided in linear time whether Maker wins the P_4-game in a given connected graph G.*

Solving the P_ℓ-game for $\ell > 4$ seems difficult, even for trees, since Maker's winning strategy can be highly non-trivial. Indeed, already for $\ell = 4$, there are trees for which Maker needs an unbounded number of moves to win and needs to play disconnected. An example is given in Figure 3.9, where there can be an arbitrary even number of vertices of degree 2 in the middle path. Moreover, this can be generalized for the P_ℓ-game for any $\ell \geq 4$.

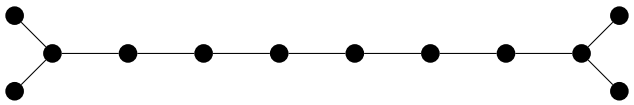


Figure 3.9: A tree where Maker wins the P_4-game, but must play disconnected. By adding an even number of vertices of degree 2 in the middle path, this gives a family of trees where the number of moves to win for Maker is unbounded.

3.2.2 Star-game in trees

In this section, we consider the $K_{1,\ell}$-game in trees. In other words, Maker needs to claim ℓ edges adjacent to the same vertex. We prove that this game is solvable in linear time in trees.

Theorem 3.11. *For any tree T and any fixed integer $\ell \geq 1$, it can be decided in linear time whether Maker wins the $K_{1,\ell}$-game in T.*

To prove this theorem, we need three structural lemmas. The first one is true for the $K_{1,\ell}$-game in any graph.

Lemma 3.12. *For any graph G and any fixed integer $\ell \geq 1$, if G contains a vertex of degree at least $2\ell - 1$, then Maker wins the $K_{1,\ell}$-game in G.*

Proof. Let u be a vertex of degree at least $2\ell - 1$. Maker claims any ℓ edges incident to u in her first ℓ moves. □

Lemma 3.13. *For any fixed integer $\ell \geq 1$, if T is a tree with maximum degree at most $2\ell - 2$ and at most one vertex of degree $2\ell - 2$, then Breaker wins the $K_{1,\ell}$-game in T.*

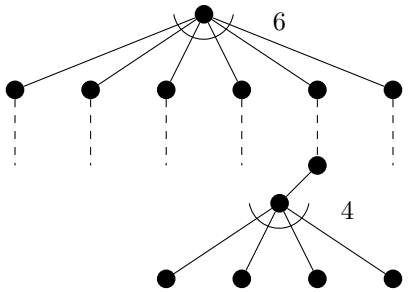


Figure 3.10: The structure of a tree satisfying the hypothesis of Lemma 3.13 with $\ell = 4$

Proof. Let r be a vertex of maximum degree (possibly $2\ell-2$) and root T in r. The structure of T is presented in Figure 3.10 with $\ell = 4$. We define a pairing strategy for Breaker as follows. For any vertex $u \in V(T)$, let $v_1, \ldots, v_{t_u}$ be its children, and pair together the edges $\{uv_{2i-1}, uv_{2i}\}$ for $i = 1$ to $i = \lfloor t_u/2 \rfloor$. Breaker, playing this pairing strategy, will claim at least half of the edges incident to r (so Maker will claim at most $\ell - 1$ edges incident to r) and, for any other vertex $w \in V(T)$, Breaker will claim at least $\lfloor \frac{t_w-1}{2} \rfloor$ edges incident to w. Since $t_w < 2\ell - 2$ as r is the only vertex that could have degree $2\ell - 2$, Maker will claim at most $\ell - 1$ edges incident to u. □

The next lemma will enable us to cut a tree into several components when there is a vertex of degree $2\ell - 2$. We first describe the cut operation we are using. Let T be a tree and let $uw \in E(T)$. Let T_u and T_w be the two trees composing the forest $T \setminus \{uw\}$, with $u \in V(T_u)$ and $w \in V(T_w)$. Let T_1 be the tree obtained from T_u by adding one pendent edge incident to u, and let T_2 be the tree obtained from T_w by adding two pendent edges incident to w. We call the forest $T_1 \cup T_2$ a (T, u, uw)-*cut*. See Figure 3.11 for an illustration.

Lemma 3.14. *For any fixed integer $\ell \geq 1$, let T be a tree with a vertex u of degree $2\ell - 2$, w any neighbor of u, and $T_1 \cup T_2$ the (T, u, uw)-cut. Maker wins the $K_{1,\ell}$-game in T if and only if she wins the $K_{1,\ell}$-game in $T_1 \cup T_2$.*

Proof. Let e be the edge added to u in T_1, and e_1, e_2 the two edges added to w in T_2. By the Super Lemma, we can without loss of generality suppose that e_1 is played by Maker and e_2 by Breaker. Assume that Maker wins in T. In $T_1 \cup T_2$, Maker plays as follows. She associates all the edges of $T_1 \cup T_2$ with their corresponding edges in T (e is associated with uw). Then, she plays as in T. At the end of the game in T, there is a vertex x in T that has ℓ incident edges claimed by Maker. If $x \neq w$, it has the same ℓ incident edges at the end of the game in $T_1 \cup T_2$ (with possibly the edge uw replaced by e if $x = u$). If $x = w$, and Maker claimed the edge uw that is not present in T_2, in the game in T, then Maker claimed the same ℓ edges in the game in $T_1 \cup T_2$ except that uw has been replaced by one of the edges e_1 or e_2.

For the other direction, assume that Maker has a winning strategy in $T_1 \cup T_2$. By Lemma 1.76, she wins either in T_1 or in T_2. Assume she wins in T_1. Then, she can follow the same strategy in T without taking care of Breaker's moves in the rest of T (and considering that $e = uw$). Assume now she wins in T_2. To win in T, Maker first claims uw. Then, Breaker should answer by claiming an edge incident to u since u has degree $2\ell - 2$ (otherwise, Maker can claim ℓ edges incident to u). Then, Maker follows her winning strategy in $(T_2, \{e_1\}, \{e_2\})$ in T. The unclaimed edges are in a one-to-one correspondence, and the vertices have the same number of edges claimed by Maker (that is, one for w, and 0 for the other vertices). Thus, Maker will win in T. □

The next theorem gives a necessary and sufficient structural condition for Maker to win the $K_{1,\ell}$-game in trees. This will imply Theorem 3.11 since it is easy to check if a tree has this structure (see the proof of Theorem 3.11 below).

Theorem 3.15. *Let T be a tree and $\ell \geq 1$ a fixed integer. Maker wins the $K_{1,\ell}$-game in T if and only if there is a subtree T' of T such that every vertex x of T' has degree at least $2\ell - 1 - d_{T'}(x)$ in T, where $d_{T'}(x)$ denotes the degree of x in T'.*

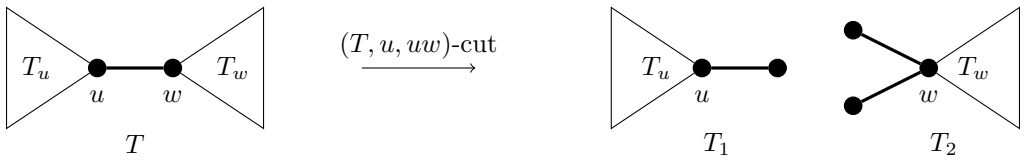


Figure 3.11: Illustration of a (T, u, uw)-cut. If u has degree $2\ell - 2$, then the $K_{1,\ell}$-game in T or in the cut has the same outcome.

Note that if T' is reduced to a single vertex, it must have degree $2\ell-1$ in T, which corresponds to Lemma 3.12. Otherwise, all the leaves of T' must be of degree $2\ell-2$ in T. The theorem says that these leaves must be connected by vertices of sufficiently large degree.

Proof. We prove the equivalence by induction on the number of inner vertices of T (*i.e.*, vertices of degree at least 2). Assume that T has only one inner vertex u. The only possibility for T' is to be the single vertex u, and Maker wins if and only if u has degree $2\ell - 1$. Thus, the equivalence is true.

Assume now, by induction, that the result is true for any tree with at most $i - 1 \geq 1$ inner vertices. Let T be a tree with i inner vertices. If T has at least one vertex u of degree at least $2\ell - 1$, then take T' to be the single vertex u, and Maker wins by Lemma 3.12. If T has maximum degree at most $2\ell - 2$, and T contains at most one vertex of degree $2\ell - 2$, it is not possible to have a subtree T' satisfying the property, since T' cannot be a single vertex, it should have at least two leaves of degree $2\ell - 2$ in T. By Lemma 3.13, Breaker wins, and so, the equivalence is true.

Thus, we can assume that T has at least two vertices of degree exactly $2\ell-2$. Let u and v be two vertices of degree $2\ell - 2$ at maximum distance from each other. Let w be the neighbor of u on the path between u and v. Let $T_1 \cup T_2$ be the (T, u, uw)-cut. Note that by the maximality of the distance between u and v, the vertex u is the only vertex of T_1 of degree $2\ell - 2$. In particular, Breaker wins in T_1 by Lemma 3.13. By Lemma 3.14, Maker wins in T if and only if she wins in $T_1 \cup T_2$. Then, since Breaker wins in T_1, by Lemma 1.76, Maker wins in T if and only if she wins in T_2. The tree T_2 has strictly less than i inner vertices, and thus, by induction, Maker wins in T_2 if and only if there is a tree T_2' such that every vertex of degree t in T_2' has degree $2\ell - 1 - t$ in T_2.

Assume Maker wins in T. This means that such a T_2' exists. If $w \notin V(T_2')$, then it is a valid tree T' for T. If $w \in V(T_2')$, then T_2' does not contain the two pendent edges incident to w (since no leaf of T_2 can belong to T_2'). Then, let T' be the subtree of T obtained from T_2' by adding u. Note that T_2' cannot have leaves of T as vertices, and thus, all its vertices are inner vertices of T. Since u is a leaf of T' and has degree $2\ell-2$ in T, it satisfies the degree condition. This is also true for all the other vertices of T' except w. Let t_2 be the degree of w in T_2'. By hypothesis, its degree in T_2, $d_{T_2}(w)$, is at least $2\ell - 1 - t_2$. The vertex w has degree $t_2 + 1$ in T' and $d_{T_2}(w) - 1$ in T. Thus, $d_T(w) = d_{T_2}(w) - 1 \geq 2\ell - 1 - t_2 - 1 = 2\ell - 1 - d_{T'}(w)$, and T' satisfies the property.

For the other direction, assume that there is a subtree T' valid in T. Let uw' be the pendent edge in T_1 (*i.e.*, $w' \notin V(T)$). If T' is a subtree of $T_1 \setminus \{w'\}$, then, by induction, Maker wins in T_1, and, by Lemmas 1.76 and 3.14, she wins in T. Otherwise, let $V(T_2') = V(T') \cap V(T_2)$. As before, one can prove that the degree condition is correct for each vertex in T_2' in T_2 (indeed, if u was in T', then w has one less neighbor in T_2' than it did in T', but this is compensated by the fact that w has one more neighbor in T_2). Thus, by induction, Maker wins in T_2, and so, by Lemmas 1.76 and 3.14, Maker wins in T.

Therefore, the equivalence is true for T, and, by induction, is true for all T. □

This characterization implies a linear-time algorithm for the $K_{1,\ell}$-game in trees.

Proof of Theorem 3.11. To find such a T', one can do a breadth-first search starting from any vertex of T. Label each vertex with its degree in T. Then, consider each vertex x, starting from the deepest level. If x has label $2\ell-2$, then increment the label of the parent of x by 1, and otherwise, do nothing. If at some point a vertex is labeled with $2\ell-1$, then Maker wins, and otherwise, Breaker wins. Note that when we increment

the label of a vertex, it corresponds to the cut operation. Precisely, if a vertex v receives a label $2\ell-1$, let T_v be the subtree rooted in v. Then, a subtree T' satisfying the requirement is the inclusion-minimal subtree of T_v containing v and whose leaves ($\neq v$) have degree $2\ell-2$ in T. □

This result is quite restrictive, since it requires conditions on both H and G. In the next section, we extend this result by removing either the condition on H or the condition on G, leading to an FPT algorithm, parameterized by the number of moves, but not a polynomial time algorithm.

3.3 Parameterized results

In this section, we consider the H-game parameterized by the number of moves k to build the graph H. Note that it would not be pertinent to consider the perfect matching game parameterized by the number of moves, since every winning set in the perfect matching game is of order $\frac{n}{2}$, and thus Maker needs at least $\frac{n}{2}$ moves to win.

We prove that determining the outcome for the $K_{1,\ell}$-game in any graph and the H-game in any tree are both FPT parameterized by the number of moves k that Maker has to build a copy of H. The next theorem is crucial for these results, and is interesting on its own since it allows bounding the diameter of the graph after Maker's first move, which could lead to other positive results for the H-game, and the proof technique could be generalized to other games on graphs.

For any graph G, $e=(u,v)\in E(G)$, and $r\in\mathbb{N}$, let $B_G(e,r)$ be the ball of center e and radius r, i.e. the set of edges at distance at most r from the two extremities of e. Formally, we have:

$$B_G(e,r)=\{(w,w')\in E(G)\mid \max\{dist_G(u,w),dist_G(v,w),dist_G(u,w'),dist_G(v,w')\}\ \leq r\}.$$

Given a graph G and $X,Y\subseteq E(G)$ (with $|X|=|Y|+1$), Breaker wins the H-game in position (G,X,Y) in $i\geq 1$ moves if it is Breaker's turn and Maker cannot create H in at most $i-1$ moves after Breaker's next move.

The next theorem will be the key of our parameterized complexity results about the H-game. It relies on the idea that, whenever the number of moves of Maker is bounded, the moves that she makes have to be close enough to each other in order to create threats of creating a copy of H. More precisely, we prove that if two moves of Maker are at a distance larger than 3^k, then Maker cannot use both of them to create a threat on a hyperedge in less than k moves.

Theorem 3.16. *Let H be a connected graph, G any graph, and k a positive integer. Maker wins the H-game in G in at most k moves if and only if there exists $e\in E(G)$ such that Maker wins the H-game in $G[B_G(e,3^k)]$ in at most k moves.*

Proof. First, if there exists $e\in E(G)$ such that Maker wins the H-game in $G[B_G(e,3^k)]$ in at most k moves, then Maker wins the H-game in G in at most k moves by Lemma 1.19. Now, assume that Breaker wins the H-game in $G[B_G(e,3^k)]$ in at most k moves for each $e\in E(G)$. We describe a winning strategy for Breaker in the H-game in G that takes at most k moves.

Let e_1 be the first edge claimed by Maker, and let $G_1=G[B_G(e_1,3^k)]$. Note that Breaker wins the H game in $(G_1,\{e_1\},\emptyset)$ in at most k moves by the initial assumption. In particular, if Maker always claims an edge in G_1, then Breaker wins.

First, Breaker answers to Maker claiming e_1 by following his winning strategy in G_1. Let $2\leq i\leq k$ be the i^{th} round of the game, before the i^{th} move of Maker, and let $M_i=\{e_1,\dots,e_{i-1}\}$ be the edges claimed by Maker, and $B_i=\{f_1,\dots,f_{i-1}\}$ the edges claimed by Breaker. Assume, by induction on i, that there exist edge-disjoint subgraphs $G_1,\dots,G_{s_i}$ such that:

- for every $1\leq j\leq s_i$, there exists $e_j\in E(G)$ such that $E(G_j)\subseteq B_G(e_j,3^k)$;
- for every $e\in M_i$, there exists a unique $1\leq j_e^i\leq s_i$ such that $e\in E(G_{j_e^i})$ and, moreover, $B_G(e,3^{k-i+1})\subseteq E(G_{j_e^i})$;

- for every $1 \leq j \leq s_i$, Breaker wins the H-game in $(G_j, E(G_j) \cap M_i, E(G_j) \cap B_i)$ in at most $k - i + 1$ moves.

The inductive hypothesis holds if $i = 2$ by remarks above (in particular, $s_2 = 1$). Assume that the inductive hypothesis holds for $i \geq 2$. Let e_i be the i^{th} edge claimed by Maker.

The main idea of the induction is to see how the ball $B_i = B_G(e_i, 3^i)$ interacts with the already considered balls. If B_i is included in one of them, the winning strategy of Breaker in that ball will ensure that he prevents any creation of a copy of H. If it is disjoint to all the other balls, Breaker considers this new ball and has a winning strategy in it by hypothesis. If it intersects some of the previous balls, by hypothesis, since Maker now has one less move, we can reduce the radius of the balls that intersect B_i and he considers B_i as a new ball.

- If there exists $1 \leq j \leq s_i$ such that $B_G(e_i, 3^{k-i}) \subseteq E(G_j)$, then Breaker answers by following his winning strategy in G_j. Note that in this case, j is unique since $G_1, \ldots, G_{s_i}$ are edge-disjoint subgraphs by the inductive hypothesis for i. Then, the inductive hypothesis holds for $i+1$ with the same subgraphs $G_1, \ldots, G_{s_i}$ (in particular, $s_i = s_{i+1}$).

- If $B_G(e_i, 3^{k-i}) \cap E(G_j) = \emptyset$ for all $1 \leq j \leq s_i$, then let $s_{i+1} = s_i + 1$ and $G_{s_{i+1}} = G[B_G(e_i, 3^{k-i})]$. Then, Breaker answers by following his winning strategy in $G_{s_{i+1}}$, which exists by the assumption that Breaker wins the H-game in $G[B_G(e, 3^k)]$ in at most k moves for each $e \in E(G)$, and since $G_1, \ldots, G_{s_{i+1}}$ are edge-disjoint subgraphs by the inductive hypothesis for i and the case we are in. Indeed, by Lemma 1.19, Breaker has a winning strategy in $G[B_G(e_i, 3^{k-i})]$ in $k - i$ moves since he has one in $G[B_G(e_i, 3^k)]$ in k moves (if Maker cannot create H in k moves, then she clearly cannot do it in $k - i$ moves in a subgraph). Then, the inductive hypothesis holds for $i + 1$.

- Lastly, if there exists $\emptyset \neq J \subseteq \{1, \ldots, s_i\}$ such that, for all $j \in J$, $B_G(e_i, 3^{k-i}) \cap E(G_j) \neq \emptyset$ and $B_G(e_i, 3^{k-i}) \setminus E(G_j) \neq \emptyset$, then let $s_{i+1} = s_i + 1$ and $G_{s_{i+1}} = G[B_G(e_i, 3^{k-i})]$. Now, for every $j \in J$, let $E_j = \{f \in E(G_j) \mid B(f, 2 \cdot 3^{k-i}) \nsubseteq E(G_j)\}$. Note that for every $f' \in M_i$ and $j \in J$, $E_j \cap B(f', 3^{k-i}) = \emptyset$ by the second assumption of the inductive hypothesis for i. For all $j \in J$, let $G_j = G[E(G_j) \setminus E_j]$ (intuitively, the edges of G_j that are "too close" to $G_{s_{i+1}}$ are removed from G_j), and note that $G_1, \ldots, G_{s_{i+1}}$ are now edge-disjoint subgraphs since $G_1, \ldots, G_{s_i}$ were edge-disjoint subgraphs by the inductive hypothesis for i. Now, Breaker plays his next move according to his winning strategy in $G_{s_{i+1}}$, which, as in the previous case, exists by the assumption that Breaker wins the H-game in $G[B_G(e, 3^k)]$ in at most k moves for each $e \in E(G)$, and since $G_1, \ldots, G_{s_{i+1}}$ are edge-disjoint subgraphs. Then, the inductive hypothesis holds for $i + 1$.

The inductive hypothesis and the strategy described above guarantee that Breaker wins, *i.e.*, Maker cannot win in G in at most k moves. □

With Theorem 3.16 in hand, we now have one of the main tools to prove our FPT results, which rely on the fact that we only need to consider the ball (of edges) of bounded diameter in the length of the game centered at the first edge claimed by Maker. The next corollary focuses on the $K_{1,\ell}$-game in graphs. As we know that, whenever a graph G has a vertex of high degree, Maker can win the $K_{1,\ell}$ game in ℓ moves, we can bound the number of vertices in each ball considered in Theorem 3.16, and therefore, the size of the graph.

Corollary 3.17. *For any graph G and any fixed integer $\ell \geq 1$, deciding whether Maker wins the $K_{1,\ell}$-game in G is* FPT *parameterized by the length of the game.*

Proof. Consider the $K_{1,\ell}$-game, for a positive constant ℓ (recall that H is a fixed graph in the H-game), in any graph G. Let k be the length of the game. If there is a vertex of degree at least $2\ell - 1$, then Maker wins in ℓ moves by Lemma 3.12. Hence, we can assume that the maximum degree is at most $2\ell - 2$. By Theorem 3.16, Maker wins in G in k moves if and only if she wins in k moves in one of the balls $B(e, 3^k)$ for some edge $e \in E(G)$. Since $\Delta(G) \leq 2\ell - 2$, for any edge $f \in E(G)$, the ball $B(f, 3^k)$ has size at most

$(2(\Delta(G)-1))^{3^k} = (4\ell-6)^{3^k}$, *i.e.*, a function of k since ℓ is a constant. Therefore, one can check if Maker wins by first checking the maximum degree, and then checking the outcomes of all possible games in the $|E(G)|$ balls (of edges) of diameter 3^k which have size bounded by a function $f(k)$. Indeed, this leads to an FPT algorithm since, in any graph of size bounded by a function $f(k)$, the output of the H-game in at most k moves can be determined by an exhaustive search in time $f'(k)$ for some computable function f' (the length of the game is k, and the number of possible moves at each step is at most the number of edges which is at most $f(k)$). □

Theorem 3.16 combined with the particular structure of trees also leads to an FPT algorithm on trees. Here, the main idea is, when considering a ball of diameter 3^k, its farthest vertices from the center are leaves. Therefore, as Maker plays only k moves, she cannot claim more than k of them, and so only $2k$ moves need to be considered. Then, inductively, as there is a function of k vertices at most to be considered at distance d from the center of the ball, there is a function of d and k possible subtrees for the vertices at distance $d-1$. Therefore, we only need at most $2k$ of each different neighborhood to compute the winner. Following this induction until we reach the center of the ball, we obtain a function of k vertices to be considered in the ball.

Theorem 3.18. *For any connected graph H and any tree T, deciding whether Maker wins the H-game in T is* FPT *parameterized by the length of the game.*

Proof. First, note that if H is not a tree, then Breaker trivially wins the H-game in T, and so, we can assume that H is a tree. We prove that the H-game parameterized by the length of the game k admits a kernel in T. That is, from T, we build, in polynomial time, a forest F of size at most a function of k (precised below) such that Maker wins in T in at most k moves if and only if there exists a connected component of F in which Maker wins in at most k moves.

First, for every edge $e \in E(T)$, let T_e be the subtree of T induced by the edges at distance at most 3^k from e, *i.e.*, T_e is the subtree induced by $B(e, 3^k)$. Let F be the forest that consists of the disjoint union of the T_e's, $e \in E(T)$. By Theorem 3.16, Maker wins in T in at most k moves if and only if there exists $e \in E(T)$ such that Maker wins in T_e in at most k moves.

The *depth* of a rooted tree is the maximum distance from its root to a leaf. For every $e \in E(T)$, let us root T_e in such a way that it has depth $d_e \leq 3^k$ (this is possible by the definition of T_e). A vertex $v \in V(T_e)$ has *level* $i \geq 0$ if the subtree of T_e rooted in v has depth i. Let us iteratively, for $i = 1$ to d_e, replace T_e^{i-1} ($T_e = T_e^0$) by a tree T_e^i such that: Maker wins in T_e^{i-1} in at most k moves if and only if Maker wins in at most k moves in T_e^i; and, for every vertex $v \in V(T_e^i)$ at level i, the subtree of T_e^i rooted in v has size at most $n_i(k)$ (a function of k whose recursive definition is given below).

First, for $i = 1$, for every vertex v at level 1 in T_e, (*i.e.*, all children of v are leaves), if v has more than $2k$ children, then remove all but $2k$ of its children. Let T_e^1 be the obtained tree. By construction, every vertex $v \in V(T_e^1)$ at level 1 is the root of a subtree of size at most $n_1(k) = 2k+1$. Moreover, since, for every vertex v at level 1 in T_e, at most $2k$ edges between v and leaves can be claimed (as the length of the game is k), then the output of the H-game is the same in T_e and T_e^1.

Now, by induction on $i \geq 1$, let us assume that we have built a tree T_e^i such that Maker wins in T_e^i in at most k moves if and only if Maker wins in at most k moves in T_e; and, for every vertex $v \in V(T_e^i)$ at level i, the subtree of T_e^i rooted in v has size at most $n_i(k)$. Let $g_i(k)$ be the number of rooted trees of depth at most i and of size at most $n_i(k)$. For every $v \in V(T_e^i)$ at level $i+1$, let $S_1, \ldots, S_r$ be the subtrees rooted in the children of v (note that each of these subtrees has depth at most i and size at most $n_i(k)$). For every possible rooted subtree S of depth i and size at most $n_i(k)$, if there are more than $2k$ copies of S in the multiset of trees $\{S_1, \ldots, S_r\}$, then remove all but $2k$ copies of S. Let T_e^{i+1} be the resulting tree (after having done the above process for every vertex at level $i+1$ of T_e^i). In T_e^{i+1}, every vertex at level $i+1$ has at most $2kg_i(k)$ children and those children are the roots of subtrees of size at most $n_i(k)$, and hence, every vertex at level $i+1$ is the root of a subtree of size at most $n_{i+1}(k) = 2kg_i(k)n_i(k)$. Moreover, for every vertex v at level $i+1$ in T_e^i, at most $2k$ edges in the subtree rooted at v can be claimed. Therefore, the output of the H-game is the same in T_e^i and T_e^{i+1}.

After the above process has been done for $i = d_e \leq 3k$ for each subtree T_e, $e \in E(T)$, F consists of the disjoint union of the trees T_e^{3k}, $e \in E(T)$, each of size at most $n_{3k}(k)$, and Maker wins in T in at most k moves if and only if she wins in at most k moves in some connected component of F. While two of these subtrees are isomorphic, let us remove one of the two isomorphic subtrees. This clearly preserves the fact that Maker wins the H-game in T in at most k moves if and only if she wins in at most k moves in some connected component of F. Moreover, eventually, F has size at most $g_{3k}(k)n_{3k}(k)$, *i.e.*, this is the desired kernel.

To conclude, this leads to an FPT algorithm since, in any graph of size bounded by a function $f(k)$, the output of the H-game in at most k moves can be determined by an exhaustive search in time $f'(k)$ for some computable function f' (the length of the game is k, and the number of possible moves at each step is at most the number of edges which is at most $f(k)$). □

3.4 Further work

We proved in this chapter that for both the H-game and the perfect matching game, it is PSPACE-complete to compute the winner. Since it was already known from Lehman [Leh64] that the connectivity game can be solved in polynomial time, among the most famous game on edges in the literature, the last one whose complexity is unknown is the Hamiltonicity game. The similarities between Hamiltonian cycles and perfect matchings invite us to make the following conjecture.

Conjecture 3.19. *Deciding whether Maker wins the Hamiltonicity game in a given graph G is* PSPACE*-complete.*

Considering the H-game, the graph provided in our reduction has 57 edges. Recall that by Theorem 1.66, if H has three edges, the winner can be computed in polynomial time. The size of the smallest graph H for which the H-game is PSPACE-complete can still be reduced. In particular, note that if Conjecture 1.67 holds, we can remove two branches of the graph given in Theorem 3.8, removing 18 edges. The resulting graph H would then have 39 edges. On the other hand, we can ask what is the largest integer k such that the H-game is polynomial for any graph H of size at most k.

In terms of positive results, we have managed to provide a linear algorithm for the P_4-game. Two possible extensions would be either to look at another graph with three edges, or to the P_k-game for larger values of k. Among the graphs with three edges, one should look at the $K_{1,3}$-game. The algorithm provided to solve this game in trees is linear, and one can wonder if we can adapt it to solve it in general graphs.

In terms of parameterized complexity, we have only considered here the number of moves, which is the most natural parameter in combinatorial games. However, since we are dealing with games on graphs, a study of the H-game parameterized by some graph parameters could be interesting, as we will do in Chapter 4 for the domination game.

Finally, using the results of Theorem 2.1 and Theorem 2.10, it would be natural to prove that the H-game and the perfect matching game are also PSPACE-complete in the Avoider-Enforcer or the Client-Waiter convention. If so, it would be interesting to know if there exists some graphs H such that the H-game is PSPACE-complete in one convention but not in another.

Chapter 4

Parameterized complexity of the Maker-Breaker domination game

Modern problems require modern solutions.

In Chapter 3, we focused on games on edges of graphs. The other way to play positional game on graphs is to claim vertices instead of edges. Among the game played on vertices of graphs, we focus here on the Maker-Breaker domination game, introduced by Duchêne *et al.* [DGPR20] in 2020. In their study, they proved that the problem of determining the winner is PSPACE-complete, even when restricted to split or bipartite graphs, but polynomial when restricted to trees and cographs. Therefore, we aim to fill the gap between these two complexity classes by considering the game under the parameterized complexity paradigm. This study completes the study of Bonnet *et al.* [BGL$^+$17], as in particular we prove that determining whether Breaker can claim a transversal in a hypergraph in k moves is W[2]-hard (Corollary 4.14), opposite to their results concerning Maker filling up a hyperedge which is W[1]-complete. On the positive side, we provide FPT algorithms for several graph parameters that generalize classes of graphs on which the problem can be solved in polynomial time. In practice, we provide kernels for the parameters modular-width, size of a minimum feedback edge set and distance to cluster, which are the generalization of cographs, trees, and clusters respectively.

In Section 4.1, we will present some known results about the Maker-Breaker domination game. Next, in Section 4.2 we introduce some lemmas that will be useful in the proofs of the rest of this chapter. In Section 4.3, we prove that determining the winner of the Maker-Breaker domination game is $W[1]$-complete when parameterized by the number of moves required for Staller to win, but that it is $W[2]$-hard when parameterized by the number of moves required for Dominator to win. We then focus on the Maker-Breaker domination game parameterized by different graph parameters. First, in Section 4.4 we prove that the game remains PSPACE-complete, even restricted to game having a domination set of size 2. Then, we consider the modular-width as parameter in Section 4.5, and we obtain an FPT for this parameter. In Section 4.6, we focus on the size of a minimum feedback edge set, providing a quadratic kernel. Finally, in Section 4.7, we propose the distance to cluster as a parameter, but the kernel provided is not polynomial here.

This work was a collaboration with Guillaume Bagan, Mathieu Hilaire and Aline Parreau [BHOP].

4.1 Maker-Breaker domination game

We present here the state of the art on the Maker-Breaker domination game, introduced by Duchêne *et al.* [DGPR20]. Recall first that the Maker-Breaker domination game is played on a graph G. Two players, Dominator and Staller take turns claiming an unclaimed vertex of the graph. If Dominator claims a winning set, she wins. Otherwise, if Staller prevent her for claiming a winning set before all the vertices are claimed, Staller wins. Note that given a graph $G = (V, E)$, the natural hypergraph for this game should

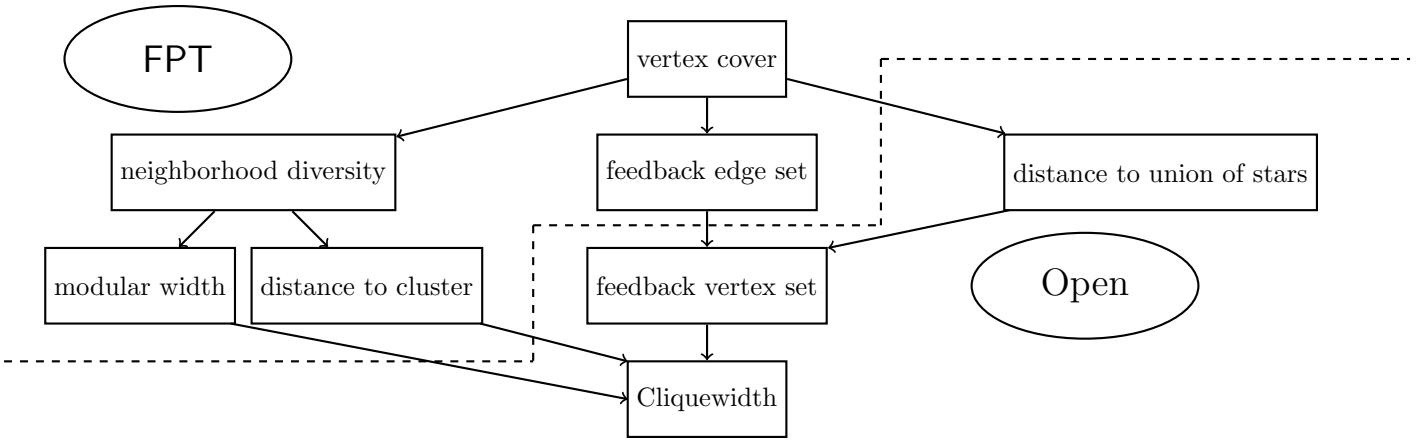


Figure 4.1: Hasse diagram of the parameters presented here. If a node a parameter is bounded for a class of graph, it is bounded for its descendant in the tree. Consequently, being FPT for the modular width, or the distance to cluster shows that computing the outcome of the Maker-Breaker Domination game is also FPT for the neighborhood diversity and the size of a minimum vertex cover.

be the hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = V$ and $\mathcal{F}$ is the set of dominating sets of G. However, this hypergraph can have an exponential number of hyperedges. Therefore, the hypergraph considered will often be defined by $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ where $\mathcal{X} = V$ and $\mathcal{F} = \{N[v] | v \in V\}$, so that it has $o(|V|)$ hyperedges. Indeed, filling up a hyperedge for Maker will then be equivalent to isolating a vertex for Staller, and thus corresponds to claiming a transversal in the hypergraph of the dominating sets.

The players in the Maker-Breaker domination game will then be named Dominator and Staller, to avoid confusion between the two representations. We adapt the notations of the outcomes and positions. If G is a graph its outcome is defined by:

- $o(G) = \mathcal{D}om$ if Dominator has a winning strategy on G going either first or second.
- $o(G) = \mathcal{N}$ if player going first has a winning strategy on G.
- $o(G) = \mathcal{S}tall$ if Staller has a winning strategy on G going either first or second.

Positions will be denoted (G, D, S), where G is a graph, D is the set of vertices claimed by Dominator and S is the set of vertices claimed by Staller.

When they introduced the game, Duchêne *et al.* studied the complexity of this game, and they obtained different results for several classes of graphs. The result that motivates the study of the parameterized complexity of the Maker-Breaker domination game is its PSPACE-completeness.

Theorem 4.1 (Duchêne *et al.* [DGPR20])**.** *Determining the outcome of the Maker-Breaker domination game is* PSPACE*-complete, even if the input graph is supposed to be split or bipartite.*

The proof of this theorem is a reduction from POS CNF, and the graph provided is similar to the one provided in Figure 2.5 for split graphs, and the same without edges between the v_is for bipartite graphs.

Next, they introduced *pairing dominating sets*, as an application of pairing strategies in the Maker-Breaker domination game. Later, they proved that is several classes of graphs, the outcome is $\mathcal{M}$ if and only if the graph admits a pairing dominating set.

Definition 4.2 (Pairing Dominating Set [DGPR20])**.** *Let $G = (V, E)$ be a graph. A subset of pair of vertices $\mathcal{P} = \{(u_1, v_1), ..., (u_k, v_k)\}$ of V is a* pairing dominating set *if all the vertices are distinct and if the intersections of the closed neighborhoods of each pair cover all the vertices of the graph:*

$$V = \cup_{i=1}^{k} N[u_i] \cap N[v_i].$$

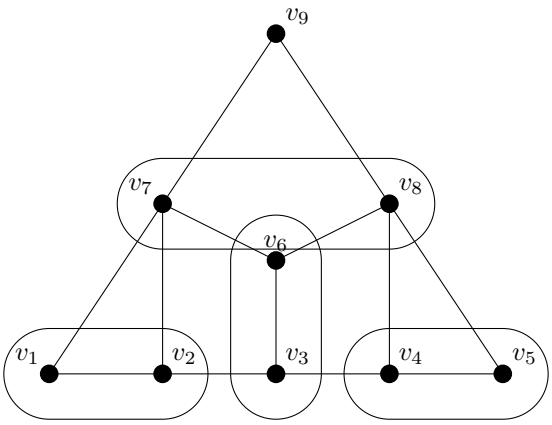


Figure 4.2: An example of pairing dominating set. The set $P = \{(v_7, v_8), (v_1, v_2), (v_3, v_6), (v_4, v_5)\}$ is a pairing dominating set of the graph.

An example of pairing dominating set is represented in Figure 4.2.

Lemma 4.3 (Duchêne *et al.* [DGPR20])**.** *Let G be a graph. If G contains a pairing dominating set, Dominator wins the Maker-Breaker domination game on G going first or second, i.e., $o(G) = \mathcal{D}om$.*

A simple example of application of this lemma is when the graph can be covered by cliques of order at least two. If this is the case, Dominator has a winning strategy which consists in playing at least once in each clique. However, note that this lemma is only an implication. The difficult part is then to find conditions in which there is an equivalence. They proved in [DGPR20] that the equivalence is true for cographs and forests. Moreover, in cographs and forests, pairing dominating sets can be computed, in polynomial time.

Theorem 4.4 (Duchêne *et al.* [DGPR20])**.** *Let G be a cograph or a forest. G admits a pairing dominating set if and only if $o(G) = \mathcal{D}om$.*

The proof for cographs relies mostly on the fact that union in Maker-Breaker games can be handled by using Table 4.1. Moreover, the join of two graphs is also easy to handle, as they proved with the following theorem.

outcome	$\mathcal{S}tall$	$\mathcal{N}$	$\mathcal{D}om$
$\mathcal{S}tall$	$\mathcal{S}tall$	$\mathcal{S}tall$	$\mathcal{S}tall$
$\mathcal{N}$	$\mathcal{S}tall$	$\mathcal{S}tall$	$\mathcal{N}$
$\mathcal{D}om$	$\mathcal{S}tall$	$\mathcal{N}$	$\mathcal{D}om$

Table 4.1: Outcome of the union of two Maker-Breaker domination games.

Theorem 4.5 (Duchêne *et al.* [DGPR20])**.** *Let G and H be two graphs. If $G = K_1$ and $o(H) = \mathcal{S}tall$ (or $o(G) = \mathcal{S}tall$ and $H = K_1$), then $o(G \otimes H) = \mathcal{N}$. Otherwise $o(G \otimes H) = \mathcal{D}om$.*

For forests, it relies mostly on the next important lemma, which provides optimal moves for Staller and forcing moves for Dominator.

Lemma 4.6 (Duchêne *et al.* [DGPR20])**.** *Let (G, D, S) be a position. Let v_0 be an unclaimed leaf of G and v_1 its private neighbor. If it is Staller's turn and v_1 is unclaimed, then playing v_1 is an optimal move and (G, D, S) and $(G \setminus \{v_0, v_1\}, D, S)$ have the same outcome*

By applying this Lemma until it cannot be applied, there is only three possibility, depicted in Figure 4.3:

- The resulting graph is empty. In this case, what was removed is a perfect matching, which is a pairing dominating set and the outcome is $\mathcal{D}om$.
- The resulting graph is a star with either zero or at least two branches. In this case by playing the center vertex of the star, the first player wins and the outcome is $\mathcal{N}$.
- Any other tree contains two cherries, i.e., an internal vertex adjacent to at least two leaves. In this case Staller wins, and the outcome is $\mathcal{S}tall$.

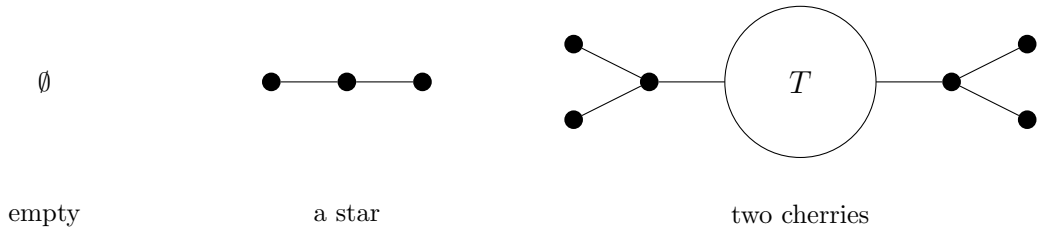


Figure 4.3: Trees that cannot be reduced

4.2 Preliminary results

We first introduce reductions lemmas that will be used to remove the vertices already played by some players. The first one consider vertices claimed by Staller but already dominated by Dominator. The intuition behind Lemma 4.7 is that, a vertex claimed by Staller but already dominated cannot be used to dominate or create traps.

Lemma 4.7. *Let (G, D, S) be a position. Let $v \in S$, that has already a neighbor in D. Then (G, D, S) and $(G \setminus \{v\}, D, S \setminus \{v\})$ have the same outcome.*

Proof. Let (G, D, S) be a position. Let $X \subset V(G) \setminus (D \cup S)$ be a set of vertices. We have that $X \cup D$ is a dominating set of G if and only if $X \cup D$ is a dominating set of $G \setminus \{v\}$. Therefore, the winning sets of (G, D, S) and $(G \setminus \{v\}, D, S \setminus \{v\})$ are the same, so, by Observation 1.75 they have the same outcome. □

Next lemma consider vertices played by Dominator. These vertices cannot be removed as we have to get track of which vertices are already dominated or not, but as they can be transformed into leaves to better understand the structure of the graph, see Figure 4.4.

Lemma 4.8. *Let (G, D, S) be a position. Let $v \in D$ of degree d, and let $v_1, \ldots, v_d$ be the neighbors of v. Let G' be the graph obtained from $G \setminus \{v\}$ and by adding d new vertices $u_1, \ldots, u_d$, private neighbors of $v_1, \ldots, v_d$. Then (G, D, S) and $(G', D \cup \{u_1, \ldots, u_d\} \setminus \{v\}, S)$ have the same outcome. If this happens, we say that we* split *the vertex v.*

Proof. Let (G, D, S) be a position. Let $X \subset G \setminus (D \cup S)$ be a set of vertices. We have that $D \cup X$ is a dominating set of G if and only if $X \cup D \cup \{u_1, \ldots, u_d\} \setminus \{v\}$ is a dominating set of G'. Therefore, the winning sets of (G, D, S) and $(G', D \cup \{u_1, \ldots, u_d\} \setminus \{v\}, S)$ are the same. Thus, by Observation 1.75 they have the same outcome. □

Finally, we introduce the following lemma, to deal with pending paths, as they may occur when we apply Lemma 4.8 in larger paths.

Lemma 4.9. *Let P_n be a path of order n. We have $o(P_n, \{v_1\}, \emptyset) = o(P_n, \{v_1, v_n\}, \emptyset) = \mathcal{D}om$ where v_1 and v_n are the leaves of P_n.*

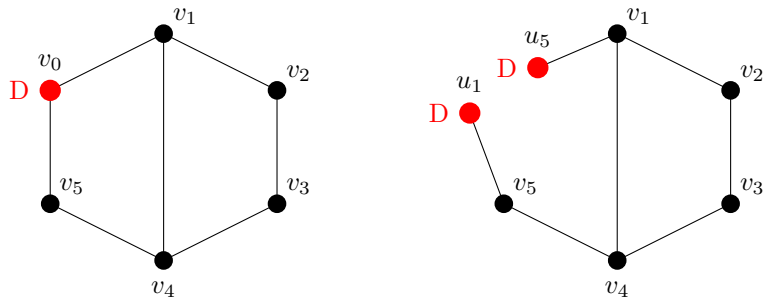


Figure 4.4: Split on the vertex v_0 using Lemma 4.8

Proof. It is sufficient to prove this lemma with only v_1 claimed by Dominator by Lemma 1.19, since when Dominator has also claimed v_n, the hypergraph obtained is a subhypergraph to the one where she has not.

Let P_n be a path of order n. Denote its vertices $v_1, \ldots, v_n$ such that $(v_i, v_{i+1}) \in E(P_n)$. Suppose that Dominator has already claimed v_1. If n is odd, $P = \{(v_{2i}, v_{2i+1})\}_{1 \leq i \leq \frac{n-1}{2}}$ is a pairing dominating set. If n is even, $\mathcal{P} = \{(v_{2i-1}, v_{2i})\}_{2 \leq i \leq \frac{n}{2}}$ is a pairing dominating set. Thus by Lemma 4.3, P_n has outcome $\mathcal{D}om$ □

4.3 Number of moves

In this section, we consider as parameter the number of moves a player needs to win. In the Maker-Breaker domination game. Given an integer k, the two related decision problems become: "Can Staller claim the neighborhood of a vertex in k moves?", or "Can Dominator claim a dominating set in k moves ?".

In Subsection 4.3.1, we prove that determining whether Staller can isolate a vertex in k moves is W[1]-complete, parameterized by k. In Subsection 4.3.2, we prove that this problem is W[2]-hard when it is Dominator that has to dominate in few moves. As an application of the later result, we obtain that Maker-Breaker games are W[2]-hard parameterized by the number of moves required for Breaker to win. A consequence of this result is that computing the winner of a Maker-Breaker positional game, parameterized by the number of moves of Breaker needs to win, is W[2]-hard. This result is surprising in contrast to its pendant parameterized by the number of moves of Maker which is W[1]-complete according to Bonnet *et al.* [BGL$^+$17].

4.3.1 When Staller must win in few moves

The next theorem proves that the Maker-Breaker domination game is W[1]-hard parameterized by the number of moves required for Staller to win. The reduction mostly consists in adapting the reduction provided by Duchêne *et al.* [DGPR20] from POS CNF so that it can handle the number of moves.

Theorem 4.10. *Determining the winner of the Maker-Breaker domination game parameterized by the number of moves required for Staller to win is W[1]-complete.*

Proof. For the appurtenance to W[1], we recall that the hypergraph considered when Staller is Maker is the hypergraph of the neighborhood of the vertices of G. Thus, it has a polynomial number of vertices and of edges. Therefore, by a result from Bonnet *et al.* [BGL$^+$17], the game is in W[1].

We provide a proof from the general Maker-Breaker game parameterized by the number of moves required for Maker to win, which is known to be W[1]-complete from Bonnet *et al.* [BGL$^+$17]. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph and k be an integer. Without loss of generality, suppose that every vertex in $\mathcal{H}$ is in at least one hyperedge. The graph $G = (V, E)$ we construct is similar to the one provided in Chapter 2, Section 2.3.1 with $p = k + 2$:

- For each vertex u_i in $\mathcal{X}$, we add a vertex v_i in V. Denote by K the set of vertices created during this step.

- For each hyperedge f in $\mathcal{F}$, we add $k+2$ new vertices $v_f^1, \ldots, v_f^{k+2}$ in V. Denote by S the set of vertices created during this step.
- For each pair of vertices $u_i, u_j \in \mathcal{X}$, we add an edge between v_i and v_j in E.
- If a vertex u_i of $\mathcal{X}$ belongs to a hyperedge f of $\mathcal{F}$, we add the edges $(v_i, v_f^1), \ldots, (v_i, v_f^{k+2})$ to E.

We prove that Staller can isolate a vertex of G in $k+1$ moves if and only if Maker wins on $\mathcal{H}$ in k moves.

Suppose that Maker has a winning strategy $\mathcal{S}$ on $\mathcal{H}$ in k moves. We provide a winning strategy for Staller in $k+1$ moves on G as follows:

- If Maker goes first in H, Staller claims first the vertex v_i such that u_i is the first vertex claimed in $\mathcal{S}$ by Maker.
- If Dominator claims a vertex v_i, Staller answers by claiming the vertex v_j such that u_j is the answer to the vertex u_i in $\mathcal{S}$.
- If Dominator plays a vertex v_f^i for $i \in \{1, k+2\}$, Staller considers that she has played an arbitrary unclaimed vertex u_j instead, and plays a vertex v_j according to this move in $\mathcal{S}$.
- When Staller will have claimed k vertices, as $\mathcal{S}$ was a winning strategy in $\mathcal{H}$, there exists a hyperedge $f \in \mathcal{F}$ fully claimed by Maker. Staller claims with his $(k+1)^{th}$ move a vertex v_f^i with $1 \leq i \leq k+2$. Note that Dominator has claimed at most $k+1$ of them (k if Staller was first to play on $\mathcal{H}$), thus one of them is available.

By construction, Staller with this strategy has claimed one vertex v_f^i and all its neighbors, thus has won in $k+1$ moves.

Suppose now that Breaker has a strategy $\mathcal{S}$ in $\mathcal{H}$ forcing Maker to play more than k moves. We provide a strategy for Dominator in G as follows:

- If Breaker goes first in H, Dominator claims first the vertex v_i such that u_i is the first vertex claimed in $\mathcal{S}$.
- If Staller claims a vertex v_i, Breaker answers by claiming the vertex v_j such that u_j is the answer to the vertex u_i in $\mathcal{S}$.
- If Staller plays a vertex v_f^i for $i \in \{1, k+2\}$, Dominator plays an arbitrary vertex of the graph, and considers that Maker has not played this move yet.
- When Staller claims his k^{th} vertex, if he has played no vertex v_f^i, Dominator claim any unclaimed vertex of the graph. If he has claimed at least one vertex v_f^i, she claims the vertex v_j corresponding to the vertex u_j that would have been claimed according to $\mathcal{S}$, which exists as the move on v_f^i was ignored in $\mathcal{S}$.

Following this strategy, if Staller did not play a vertex v_f^i during his k first moves, he cannot have claimed all the vertices corresponding to a hyperedge of $\mathcal{H}$. Therefore, he cannot isolate a vertex with his $k+1^{th}$ move. Since the vertices v_i have at least $k+2$ neighbors each (we assumed that each vertex is in at least one hyperedge), and even if his last move would fill up a hyperedge, he has not played any vertex v_f^i, and therefore cannot have isolated any of them. If Staller has played at least one vertex v_f^i, he has played at most k vertices v_j. Since Dominator's answers followed $\mathcal{S}$, Staller cannot have played all the neighbors of a vertex v_f^i. In both cases, no vertex was isolated in the first $k+1$ moves of Staller, and Dominator has won. This reduction is clearly polynomial.

Finally, the game is both in W[1] and W[1]-hard, thus, it is W[1]-complete. □

Note also that, as the hypergraph of the Maker-Breaker domination game, where Staller needs to isolate a vertex in few moves has linear size, the results from Bonnet *et al.* [BJS16] can be applied. Therefore, this problem, can be solved in FPT time on graphs of locally bounded treewidth.

4.3.2 When Dominator must win in few moves

This subsection focuses on the proof that, when parameterized by the number of moves required for Dominator to win, the Maker-Breaker domination game is W[2]-hard. Note that contrary to the previous subsection, the hypergraph considered for Dominator does not have polynomial size in general, so we cannot apply the result from Bonnet *et al.* proving that this game is in W[1]. To prove its hardness, we will provide a reduction from the k-Dominating set problem, which is defined as follows:

Problem 4.11 (k-Dominating set)**.**
Input: a graph G and an integer k
Output: $\top$ if G has a dominating set of size k, $\bot$ otherwise.

Theorem 4.12 (Downey and Fellows [DF95])**.** *The k-Dominating set problem is W[2]-complete parameterized by k.*

Theorem 4.13. *Determining the winner of the Maker-Breaker domination game is W[2]-hard, parameterized by the number of moves required for Dominator to win.*

Proof. The proof is a reduction from the k-dominating set problem. Let $G = (V, E)$ be a graph, and k be an integer. We build a graph $G' = (V', E')$ as follows:

- For any vertex $x_i \in V$, we add two vertices y_i and y'_i in V' and the edge (y_i, y'_i) in E'.
- For any edge $(x_i, x_j) \in V$, we add the edges $(y_i, y_j), (y_i, y'_j), (y'_i, y_j)$ and (y'_i, y'_j) in E'.

We prove that Dominator wins in k moves in G' if and only if G admits a dominating set of size k. First, suppose that G admits no dominating set of size k, and consider any game on G'. Let $D = \{x_{i_1}, \dots, x_{i_k}\}$ be the vertices of G such that for each $j \in \{i_1, \dots, i_k\}$, Dominator has claimed either y_j or y'_j in G' at the end of the game. By hypothesis, D is not a dominating set. Thus, there exists a vertex x_{i_0} such that $N[x_{i_0}] \cap D = \emptyset$. Thus, Dominator has not claimed x_{i_0} nor any of its neighbor, thus has not won in k moves.

Suppose now that G has a dominating $D = \{x_{i_1}, \dots, x_{i_k}\}$ of size k. Dominator can play a pairing strategy in G' by pairing for each $j \in \{i_1, \dots, i_k\}$ the vertices y_j and y'_j. By hypothesis, she will win in k moves by claiming once in each of these pairs. □

As, the hypergraph of the Maker-Breaker domination game where Dominator plays the role of Breaker has linear size, we directly obtain the following corollary by considering the Maker-Breaker game on that hypergraph instead of playing on a graph:

Corollary 4.14. *Determining whether Breaker can win in k moves in a Maker-Breaker positional game is W[2]-hard.*

Note that the main point of this result is that, it proves that, when parameterized by the number of moves required for Breaker ton win, Maker-Breaker games are not in W[1] (unless W[1] = W[2] which is believed to be false). Whether it is W[2]-complete or whether it can be W[k] hard for some $k \geq 3$ is still open, but we do believe that it is W[2]-complete.

Conjecture 4.15. *Determining whether Breaker can win in k moves in a Maker-Breaker positional game is W[2]-complete.*

A consequence of this conjecture is that the Maker-Breaker domination game would also be W[2]-complete.

On the positive side, using the same method as Bonnet *et al.* [BJS16], together with an argument by Frick and Grohe [FG01], we obtain the following theorem:

Theorem 4.16. *Determining whether Dominator can build a dominating set in k moves in a graph G is* FPT *parameterized by k on locally bounded treewidth graph.*

Proof. The winning condition for Dominator can be expressed as follows:

$$\exists x_1 \forall y_1 \dots \exists x_k \forall u \left(\bigwedge_{1 \leq i < j \leq k} x_j \neq y_i \right) \wedge \left(\bigvee_{1 \leq i \leq k} E(u, x_i) \right)$$

Therefore, according to Frick and Grohe [FG01], this property can be verified in FPT-time on graphs of locally bounded treewidth, and consequently on bounded treewidth graphs or planar graphs. □

Since planar graphs and bounded treewidth graphs have locally bounded treewidth, this directly implies that determining whether Dominator can win in k moves in a planar graph or a bounded treewidth graph is FPT.

4.4 Size of a minimum dominating set

We will now focus on the parameterized complexity of the Maker-Breaker domination game considering graph parameters. One parameter that seems natural is γ, the size of a minimum dominating set of G. However, this parameter does not provide results in general, as we prove that even for $\gamma = 2$, the problem remains PSPACE-complete.

Lemma 4.17. *Determining the winner of the Maker-Breaker domination game is* PSPACE*-complete, even restricted to graphs G satisfying $\gamma(G) = 2$ (Dominator starts), or $\gamma(G) = 1$ (Staller starts).*

Proof. The reduction is direct from the Maker-Breaker domination game, which is known to be PSPACE-complete, when Dominator starts by Theorem 4.1. Let G be a graph.

If Staller starts, we consider the graph G' obtained from G by adding a universal vertex v_0 in G, thus $\gamma(G') = 1$. Staller has to play v_0 first, otherwise Dominator plays it and wins. Therefore, after this move, the game is played on G' as it was played on G as it is Dominator's turn and any move of Dominator will dominate v_0.

If Dominator starts, we consider the graph G' obtained from G by adding a universal vertex v_0 and an isolated vertex v_1 (not connected to v_0). Thus $\gamma(G') = 2$, as $\{v_0, v_1\}$ is a dominating set. When Dominator starts, she has to claim v_1 first, otherwise, Staller plays it and isolates it. Now, Staller has to play v_0 as otherwise Dominator plays it in her second move and wins. Therefore, the outcome on G' is the same as the outcome on G. □

Note that if $\gamma(G) = 1$, G has a universal vertex. Thus, if Dominator starts, she can claim the universal vertex of G and win with her first move.

4.5 The modular-width

As presented in Chapter 4.1, the union and the join of two graphs can easily be handled in the Maker-Breaker domination game. Together with the Super Lemma, it makes it possible to handle modules. All together, modules, joins, and unions makes it possible to define the modular-width of graphs. We provide here an FPT algorithm for the Maker-Breaker domination game parameterized by the modular-width. We recall first some definitions about modules.

Let us first define what is a module in a graph. Intuitively, a module is a set of vertices behaving the same regarding the rest of the graph.

Definition 4.18 (Module of a graph)**.** *Let $G = (V, E)$ be a graph. A* module *of G is a set of vertices M such that, for any $x, y \in M$, we have $N[x] \setminus M = N[y] \setminus M$.*

Since any vertex of M has the same neighborhood in G, we say that a vertex $x \in G \setminus M$ is adjacent to *M, if it is adjacent to the vertices of M, see Figure 4.5.*

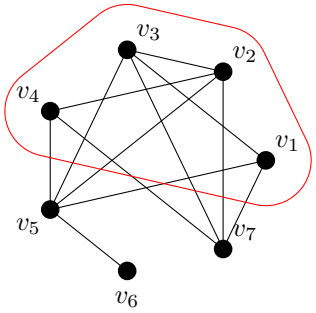


Figure 4.5: $M = \{v_1, v_2, v_3, v_4\}$ is module in the graph. v_5 and v_7 are adjacent to M. v_6 is not.

The *modular-width* is a graph parameter, introduced by Gajarský *et al.* [GLO13], that consists in decomposing the graph into modules. A, example of modular decomposition is provided in Figure 4.6

Definition 4.19 (Modular-width [GLO13])**.** *We consider a graph $G = (V, E)$ built from using the following operations:*

1. *Create an isolated vertex.*
2. *The disjoint union of two graphs G_1 and G_2 denoted by $G_1 \cup G_2$.*
3. *The complete join of two graphs G_1 and G_2 denoted by $G_1 \otimes G_2$.*
4. *The substitution operation with respect to some graph G' with vertices $v_1, \ldots, v_n$ for graphs $G_1, \ldots, G_n$ denoted by $G'(G_1, \ldots, G_n)$. The graph $H = G'(G_1, \ldots G_n)$ is the graph with $V(H) = \bigcup_{1 \leq i \leq n} V(G_i)$, and $E(G') = \bigcup_{1 \leq i \leq n} E(G_i) \cup \{(u, v) \mid u \in G_i, v \in G_j, (v_i, v_j) \in E(G')\}$. Intuitively, this operation consists in replacing each vertex v_i by a G_i which behaves as a module.*

A modular-decomposition *of G is an expression that constructs G using the above operations only.*
The width *of a modular-decomposition is the largest number of operands in any occurrence of the operation* (4)*. The* modular width *of G, denoted by $mw(G)$ is the minimum integer m such that G can be obtained from a modular-decomposition with width m.*

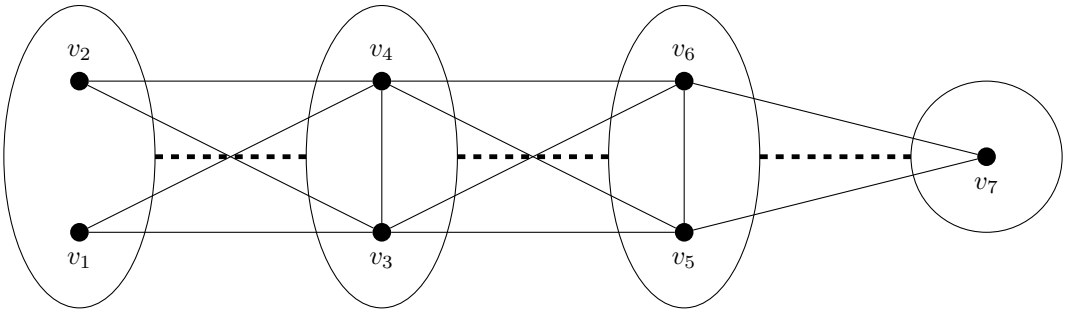
Note that cographs are exactly the graphs obtained using only (1), (2) and (3) and therefore have modular-width 0.

Tedder *et al.* [TCHP08] have provided a linear time algorithm which, given a graph G, compute a modular decomposition of G of width $mw(G)$.

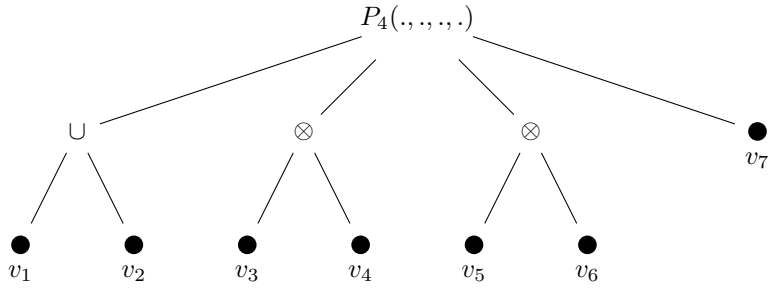
The notion of module is very strong in the Maker-Breaker domination game. Indeed, when a module is identified in a graph, depending on the outcome of the game played on the module, next lemma says that it can be replaced by a module of order at most 3 without changing the outcome of the whole graph. An example of application of this lemma in provided in Figure 4.7

Lemma 4.20. *Let G be a graph, and M be a module of G such that $|M| \geq 3$.*

- *If $o(M) = \mathcal{D}om$, let H be the graph obtained from G by replacing M by a copy of P_2 and adding all the edges between a vertex of P_2 and a vertex $v \in V(G) \setminus M$ such that v is adjacent to M in G.*
- *If $o(M) = \mathcal{S}tall$, let H be the graph obtained from G by replacing M by two independent vertices and by adding all the edges between any of these two vertices and any vertex $v \in V(G) \setminus M$ such that v is adjacent to M in G.*
- *If $o(M) = \mathcal{N}$, let H be the graph obtained from G by replacing M by a copy of P_3 and adding all the edges between a vertex of P_3 and a vertex $v \in V(G) \setminus M$ such that v is adjacent to M in G.*



(a) A graph G, the big circles are modules, dashed edges between them shows the connection between the modules.



(b) A modular decomposition of width 4.

Figure 4.6: Example of a modular decomposition

Then G and H have the same outcome.

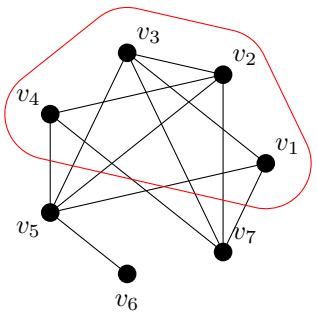
Proof. Let $G = (V, E)$ be a graph and let M be a module of G. Let H be the graph obtained from G by replacing M by a P_2, a P_3 or two isolated vertices, depending on $o(M)$ and according to the statement of the lemma.

First, we apply the Super Lemma to P_2, the two isolated vertices or the two leaves of P_3, which is always possible as they have the same neighborhood in G.

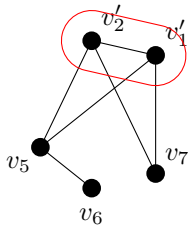
Suppose that Dominator has a winning strategy $\mathcal{S}$ in G'. We build a strategy for Dominator, considering that G is decomposed into two components, M and $G \setminus M$ as follows:

- If $o(M) = \mathcal{D}om$, then, Dominator plays in $G \setminus M$ following $\mathcal{S}$, and in M following her winning strategy as a second player, answering always in the same component as Staller, or claiming an arbitrary vertex if it is not possible.
- If $o(M) = \mathcal{N}$, Dominator follows $\mathcal{S}$ in $G \setminus M$, supposing that if Staller claims in M, it corresponds to claiming the middle vertex of P_3, and if $\mathcal{S}$ would make Dominator claim the middle vertex of P_3, she claims the first vertex she would have claimed in M according to her winning strategy in M going first. Once the first move in M has been made, she always answers in the same component as Staller, following $\mathcal{S}$ in $G \setminus M$, and her winning strategy in M if she played first in M, or any strategy in M otherwise.
- If $o(M) = \mathcal{S}tall$, Dominator follows $\mathcal{S}$ in $G \setminus M$, and claims any arbitrary vertex whenever Staller plays in M.

Following this strategy, Dominator will always dominate the graph:



(a) $M = \{v_1, v_2, v_3, v_4\}$ is module. Since $\{(v_1, v_3), (v_2, v_4)\}$ is a pairing dominating set, it has outcome $\mathcal{D}om$.



(b) The reduced graph, where M has been replaced by an edge.

Figure 4.7: Reduction of module using Lemma 4.20

- In $(G \setminus M) \cap N[M]$. Since $|M| \geq 2$, she will always claim at least one vertex x_0 of M, and thus any vertex dominated in G' by the vertices replacing M will be dominated by x_0.
- In $G \setminus N[M]$, by hypothesis, as her strategy was a winning strategy, all the vertices are dominated, as she followed $\mathcal{S}$.
- In M, if $o(M) = \mathcal{D}om$ or $o(M) = \mathcal{N}$ and she has played first in M, she dominates M by following her winning strategy in it.
- In M, if $o(M) = \mathcal{S}tall$ or $o(M) = \mathcal{N}$ and she played second in M, according to $\mathcal{S}$, either one leaf of P_3 or one of the two independent vertex is not dominated by M. Therefore, it must be dominated by a vertex of $G \setminus M$, which dominates all the vertices of M in G as M is a module. Thus, Dominator wins.

Suppose now that Staller has a winning strategy $\mathcal{S}$ in G'. We build a strategy for Staller, considering that G is decomposed into two components, M and $G \setminus M$ as follows:

- If $o(M) = \mathcal{D}om$, then Staller plays according to $\mathcal{S}$ in $G \setminus M$, always answering in the same component as Dominator, and plays arbitrary vertices in M.
- If $o(M) = \mathcal{N}$, Staller follows $\mathcal{S}$ in $G \setminus M$, supposing that if Dominator claims in M, it corresponds to claiming the middle vertex of P_3, and if $\mathcal{S}$ would make Staller claim the middle vertex of P_3, he claims the first vertex he would have claimed in M according to his winning strategy in M going first. Once the first move in M has been made, he always answers in the same component as Dominator, following $\mathcal{S}$ in $G \setminus M$, and his winning strategy in M if he played first in M, or any strategy in M otherwise.
- If $o(M) = \mathcal{S}tall$, Staller follows $\mathcal{S}$ in $G \setminus M$, and his winning strategy in M.

For the same reason as previously, if Staller manages to isolate a vertex of $G \setminus M$ in G' according to $\mathcal{S}$, he still isolates it in G since he has played according to $\mathcal{S}$, and necessarily, this vertex has no neighbor in M, as otherwise, it would be adjacent to a vertex claimed by Dominator in G'. If $o(M) = \mathcal{N}$ or $\mathcal{S}tall$ he might isolate a vertex of P_3 or one of the two vertices added instead of M. In this case, he has played following his winning strategy in M, and all the vertices adjacent to M. Thus, he isolated a vertex of M that is isolated in G.

Finally, we have proved that G and G' have the same outcome. □

This lemma enables us to threat modules quite easily, since if we can compute the outcome of a module, we can reduce it. This will provide us an FPT algorithm parameterized by the modular-width.

Theorem 4.21. *The Maker-Breaker domination game is* FPT *parameterized by the modular-width of the graph taken as input.*

Proof. Let $G = (V, E)$ be a graph, and let $k = mw(G)$. First, we compute a decomposition of G in linear time. We embed it in a tree such that, the nodes are the operations $(2), (3)$ and (4), and the leaves are isolated vertices. Now we compute the outcome of G, starting from the leaves. Let H the graph obtained at a node of the decomposition as follows:

1. If $|V(H)| \leq 3k$, we compute the winner by an exhaustive computation of all the possible games.
2. If $H = G_1 \cup G_2$, we compute inductively the outcome of G_1 and G_2 which must have modular-width at most k and we return the outcome according to Table 4.1.
3. If $H = G_1 \otimes G_2$ suppose $|G_1| \leq |G_2|$, If $|G_1| = 1$, if $o(G_2) = \mathcal{N}$ or $o(G_2) = \mathcal{D}om$, we have $o(H) = \mathcal{D}om$. Otherwise, we have $o(H) = \mathcal{N}$.
4. If $H = G'(G_1, \ldots, G_p)$, necessarily, we have $p \leq k$. We replace, according to Lemma 4.20, each G_i having more than three vertices by a P_2 a P_3 or two isolated vertices depending on its outcome. Then, we compute $o(H)$ as it has at most $3k$ vertices.

This algorithm runs in time $O((3k)!n)$. as any of these four steps can be done in $O((3k)!)$ operations, and the modular decomposition contains at most n nodes. □

Note that this proof does not provide a kernel, and the size of an optimal kernel for the modular width is still open. However, this lemma has consequences on other parameters. Recall that this results proves that determining the winner of the Maker-Breaker domination game is also FPT parameterized by the neighborhood diversity or the vertex cover.

4.6 Size of a minimum feedback edge set

In this section, we provide an FPT algorithm to solve the Maker-Breaker domination game parameterized by the size of a minimum feedback edge set. The study of this parameter is motivated by the fact that the Maker-Breaker domination game can be easily solved in forests, and that a feedback edge set is a set of edges to be removed to obtain a forest. The rest of this section is devoted to proving the following theorem.

Theorem 4.22. *The Maker-Breaker domination game has a quadratic kernel, parameterized by the size of a minimum feedback edge set.*

Structure of the proof of Theorem 4.22

The proof is composed of several lemmas, we first present here the intuition of the proof by explaining it step by step. We denote by $fes(G)$ the size of a smallest feedback edge set of $G = (V, E)$. Note that if G is connected, it satisfies $fes(G) = |E| - |V| + 1$.

- After the first move of Dominator, by Lemma 4.6, Staller can play all the neighbors of the leaves in the graph. Once this lemma cannot be applied, one of the following happens:
  - Staller can win with his next move.
  - G has bounded size in function of k.
  - G has at most one leaf, Dominator has claimed its neighbor.
- If $fes(G) \leq k$, and G has at most one leaf, G has at most $2k$ vertices of degree larger than 2 (see Lemma 4.23). Thus, in G most of the vertices of G are of degree 2, or the size of G is bounded by some function of $fes(G)$.

- If there is a long path containing only vertices of degree 2 in G, it can be reduced to path containing at most seven vertices, using Lemma 4.25.
- Once the previous point cannot be applied, the graph has a number of vertices bounded by some function of $fes(G)$. We can test all the strategies on it.

Structure of graphs of bounded feedback edge set

We first prove that if G has a large order, but a small feedback edge set, once all the leaves of G but one, and their neighbors have been removed, there are a lot of paths of degree 2.

Lemma 4.23. *Let G be a graph, with at most one leaf. Let L be the set of vertices of G of degree 3 or more. We have $|L| \leq 2fes(G) - 1$.*

Proof. Up to apply this result to all the connected components of G, suppose G connected. Let H be the graph obtained by contracting any vertex of degree 2 in G and by removing its leaf (if any). Note that $fes(H) = fes(G)$ and that the vertices of H are exactly L or $L \setminus \{x\}$ if x is the neighbor of the removed leaf. All the vertices of H have degree at least 3, so we have $|E(H)| \geq \frac{3}{2}|V(H)|$. Since $|E(H)| - |V(H)| + 1 = fes(H)$, we have by subtracting this equality to the previous one $|V(H)| - 1 \geq \frac{3}{2}|V(H)| - fes(H)$. Finally, we obtain $2fes(H) - 2 = 2fes(G) - 2 \geq |V(H)|$. As putting back the leaf can at most create one vertex of degree 3, we have $2fes(G) - 1 \geq |L|$ □

Shorten long pending paths

A *pending path* is an induced path $v_1, \dots v_k$ such that v_k is a leaf and for all $1 \leq i \leq k-1$, $d(v_i) = 2$. The next step of the algorithm is to prove that if G has a long pending path, we can reduce it to a smaller one of same parity. Let G be a graph and u a vertex of G. Let $k \geq 1$ be an integer. We denote by $G(k, u)$ the graph G with a pending path of order k attach to u. More formally, we add k vertices $v_1, ..., v_k$ to G and, the edges (v_i, v_{i+1}) for $i \in \{1, ..., k\}$ and the edge (u, v_1). v_k is then the leaf of the pending path, see Figure 4.8.

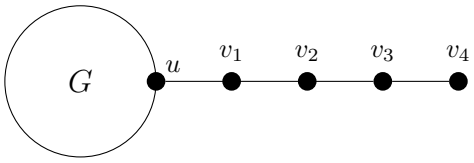


Figure 4.8: A pending path of order 4 attached to graph G. The resulting graph is $G(4, u)$.

To deal with pending paths, we already have Lemma 4.6 to handle them when the extremity and its neighbor are unclaimed. We now introduce a lemma to shorten paths when Dominator has claimed one extremity. Note that when Staller claim the neighbor of a leaf in a pending path, Dominator must answer by playing the leaf and the two vertices can be removed.

Lemma 4.24. *Let $k \geq 3$, $G_k = (G(k, u), D, S)$ be a position on $G(k, u)$ such that the only vertex claimed on the vertices $\{v_1, ..., v_k\}$ is v_k and is claimed by Dominator. If Dominator wins in the position G_k, then she wins in all the positions $G_{k'} = (G(k', u), D \setminus \{v_k\} \cup \{v_{k'}\}, S)$, for any $k' \geq 1$.*

Proof. When $k' < k$ it is clear : the winning sets for Dominator in P_k are winning sets of $G_{k'}$ (when restricted to the unclaimed vertices). Thus, Dominator can just follow her strategy in the smaller graph. If at some point she has to claim a vertex not in $G_{k'}$, she just claims any unclaimed vertex.

We now prove the result for $k' = k + 1$. By induction, it will prove the result for all $k' \geq k$.

Dominator follows her strategy in G_k until an element of $X = \{v_1, ..., v_k\}$ is claimed. Note that up to change the sets D and S, we can assume that it is the first move.

- Assume first that it is Dominator that claims a vertex v_j in X. Then, using Lemma 4.8 the game G_k (resp. G_{k+1}) is split into two games: G_j and a path with $k-j+1$ (resp. $k-j+2$) vertices with only the two end vertices claimed by Dominator. Since playing v_j was an optimal move for Dominator in G_k, she is winning playing second in the game G_k with v_j claimed by Dominator. Therefore, she is winning second in G_j (by Table 4.1. Since she is also winning second in any path with extremities claimed by Dominator by Lemma 4.9, she is winning in G_{k+1} by claiming v_j.

- Assume now that it is Staller that claims a vertex v_j in X. If $j > 1$, then Dominator claims v_{j-1} (that is unclaimed by hypothesis). Then as before the game G_{k+1} is split into a path with one extremity claimed by Dominator (where Dominator wins second, by Lemma 4.9, and the game G_{j-1} that is winning for Dominator playing second (since $1 \leq j-1 \leq k$ and the first remark of the proof). Thus, the union is winning by Dominator playing second by Table 4.1.

  Assume now $j = 1$: Staller claims v_1. If u is unclaimed, Dominator claims u and wins since $(G, D \cup \{u\}, S)$ is winning for Dominator playing second, as a subgraph of G_k which had outcome $\mathcal{D}om$ and a path with one extremity claimed by Dominator, which has outcome $\mathcal{D}$ by Lemma 4.9. If u was already claimed by Dominator, Dominator can answer in G, and we have the same result. Otherwise, if it is claimed by Staller, then in both games, Dominator has to answer in v_2 since $k \geq 3$ and v_1 need to be dominated. Then both games are split into two components that are Dominator wins as second player by hypothesis for G and Lemma 4.9.

□

Reduction of long induced (non-pending) paths

Next lemma is the one which provide the main reduction rule of our kernelization algorithm. Its application is depicted in Figure 4.9

Lemma 4.25. *Let G be a graph, and u, v be two vertices of G. We denote by $G_k^{u,v}$ the graph obtained from G by adding a path on k vertices to G and connecting u and v to the two extremities of the path. If $k \geq 7$, then $G_k^{u,v}$ and $G_{k+2}^{u,v}$ have the same outcome, even if some vertices of G have already been played by Dominator.*

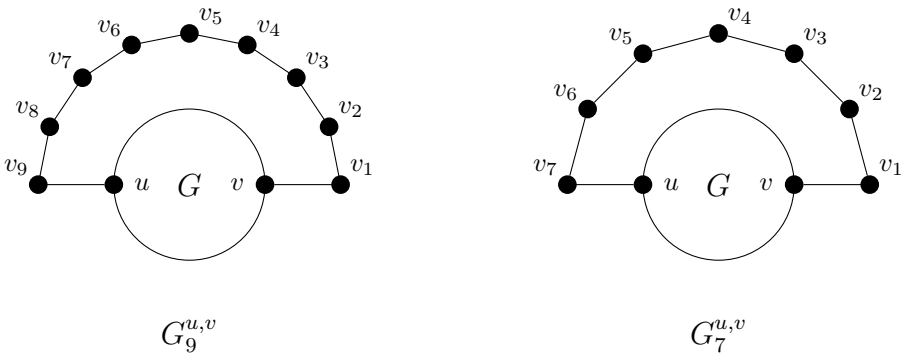


Figure 4.9: Two graphs with the same outcome, using Lemma 4.25.

Proof. Denote by P_k the path attached to u and v, and D the set of vertices already claimed by Dominator in G.

Suppose first, that Dominator has a winning strategy $\mathcal{S}$ in $G_k^{u,v}$ where she has already claimed D. We prove that Dominator wins in $G_{k+2}^{u,v}$. For her first move, if it is her turn, and while Staller plays in G, Dominator answers following $\mathcal{S}$ in G. If before any vertex is played on P_{k+2}, Dominator played u or v, by Lemma 4.8, we can split this vertex. Then, by applying Lemma 4.24, we can remove two vertices on this

path, and we obtain the result directly. Thus, we can consider that u and v are either unclaimed or claimed by Staller. We consider the first time a vertex has to be played on P_k or is played on P_{k+2}.

- If $\mathcal{S}$ wants Dominator to claim in P_k, we denote the vertices of P_k by $v_1, \ldots, v_k$, in such a way that $\mathcal{S}$ makes Dominator plays v_i with i minimal. Dominator plays v_i in P_{k+2}. Now, by Lemma 4.8, we can split the path into two, one of length i, and one of length $k+3-i$, by removing v_i and replacing it with two leaves claimed by Dominator and connected to its neighbors. Then, by Lemma 4.24, as either i or $k+3-i$ is greater than 3, (since $k \geq 7$), we can shorten one of these two paths by two vertices (or both by one). Then, using again Lemma 4.8, the resulting graph has the same outcome as if v_i was played in P_k, which concludes the proof.

- If Staller claims first in P_{k+2}, we denote the vertices of P_{k+2} by $v_1, \ldots, v_{k+2}$, in such a way that Staller played v_i with i minimal. Suppose v_1 is connected to u. The following happens depending on the value of i:

  - If $i \geq 2$, Dominator, in $\mathcal{S}$ has to play a neighbor of v_i, otherwise Staller claims either v_{i-1} or v_{i+1} and threaten to isolate a vertex with his next move.

    Thus, Dominator can claim the same vertex in P_k (either v_{i-1} or v_{i+1}). Now, by Lemma 4.7, v_i can be removed. Since $k \geq 7$, On the path $v_{i+1}, \ldots, v_{k+2}$, we can apply either Lemma 4.24, if Dominator has claimed v_{i+1}, or Lemma 4.6, if she has played v_{i-1}, which gives the result by induction.

  - If $i = 1$, Dominator has to claim a vertex in $\{u, v_2, v_3\}$, otherwise Staller can claim v_2 and threaten to isolate v_1 by claiming u or v_2 by claiming v_3 (or wins if he already has claimed u). Thus, after this move, we can split this vertex using Lemma 4.8. Once again, if Dominator has claimed u, we conclude by Lemma 4.6. If Dominator has claimed v_2 or v_3, we conclude by Lemma 4.24.

Suppose now that Dominator has a winning strategy in $G_{k+2}^{u,v}$ where she has already claimed the vertices of D. We prove that Dominator wins in $G_k^{u,v}$. For her first move, if it is her turn, and while Staller plays in G, Dominator answers following $\mathcal{S}$ in G. If before any vertex is played on P_k, Dominator played u or v, by Lemma 4.8, we can split this vertex. Then, by applying Lemma 4.24, applied on $G_k^{u,v}$, we can add two vertices on this path, and we obtain the result directly. Thus, we can consider that u and v are either unclaimed or claimed by Staller. We consider the first time a vertex has to be played on P_k or is played on P_k.

- If $\mathcal{S}$ wants Dominator to claim in P_{k+2}, we denote the vertices of P_{k+2} by $v_1, \ldots, v_{k+2}$, in such a way that $\mathcal{S}$ makes Dominator plays v_i with i minimal. Dominator plays v_i in P_k. Now, by Lemma 4.8, we can split the path into two, one of length i, and one of length $k+1-i$, by removing v_i and replacing it with two leaves claimed by Dominator and connected to its neighbors. Then, by Lemma 4.24, as either i or $k+1-i$ is greater than 3, (since $k \geq 7$), we can add two vertices to one of these two paths. Then, using again Lemma 4.8, the resulting graph has the same outcome as if v_i was played in P_{k+2}, which concludes the proof.

- If Staller claims first in P_k, we denote the vertices of P_k by $v_1, \ldots, v_k$, in such a way that Staller played v_i with i minimal. Suppose v_1 is connected to u. The following happens depending on the value of i:

  - If $i \geq 2$, Dominator, in $\mathcal{S}$ has to play a neighbor of v_i, otherwise Staller claims either v_{i-1} or v_{i+1} and threaten to isolate a vertex with his next move.

    Thus, Dominator can claim the same vertex in P_k (either v_{i-1} or v_{i+1}). Now, by Lemma 4.7, v_i can be removed. Since $k \geq 7$, On the path $v_{i+1}, \ldots, v_k$, we can apply either Lemma 4.24, if Dominator has claimed v_{i+1}, or Lemma 4.6, if she has played v_{i-1}, which gives the result by induction.

  - If $i = 1$, Dominator has to claim a vertex in $\{u, v_2, v_3\}$, otherwise Staller can claim v_2 and threaten to isolate v_1 by claiming u or v_2 by claiming v_3 (or wins if he already has claimed u). Thus, after

this move, we can split this vertex using Lemma 4.8. Once again, if Dominator has claimed u, we conclude by Lemma 4.6. If Dominator has claimed v_2 or v_3, we conclude by Lemma 4.24.

Finally, we proved that if Dominator has a winning strategy on one of $G_k^{u,v}$ or $G_{k+2}^{u,v}$, she has one on both, which proves that they have the same outcome. □

This lemma enables to shorten any path in graphs. We can now prove the main theorem of this section.

Proof of Theorem 4.22. Let G be a graph and let $k = fes(G)$. Let $n = |V(G)|$. Up to consider the $O(n)$ positions reached after the first move of Dominator, suppose that it is Staller's turn. By Lemma 4.6, Staller claims all the unclaimed neighbors of leaves forcing answers to Dominator after each move on the attached leaf. By applying Lemma 4.7, and by removing the leaves connected to the first move of Dominator, which are now in no winning sets, we can assume that the resulting graph has at most one leaf, and if so, it has been played by Dominator.

The resulting graph, by Lemma 4.23, has at most $2k-1$ vertices of degree 3 or more. Now, by applying Lemma 4.8, we can split the first vertex of Dominator to transform it into a set of leaves, without changing the degree of any other vertex, nor the number of unclaimed vertices. Now, the only vertices that are already claimed by Dominator are leaves of pending paths. Thus, by Lemma 4.25 and Lemma 4.24, we can shorten any path containing only vertices of degree 2 in this graph to a path of order at most 7.

Thus, the resulting graph has at most $7\binom{2k}{2}$ vertices of degree 2, no unclaimed leaves, and at most $2k-1$ vertices of degree at least 3. Thus, it has a kernel of size $O(k^2)$ vertices, which provides an FPT algorithm, using Theorem 1.60. □

4.7 Distance to cluster

In this section, we study the complexity of the Maker-Breaker domination game parameterized by the distance to cluster, which is defined as the number of vertices to remove to transform the graph into a union of cliques. This parameter is quite natural to study as the outcome on cliques is easy to compute in the Maker-Breaker domination game.

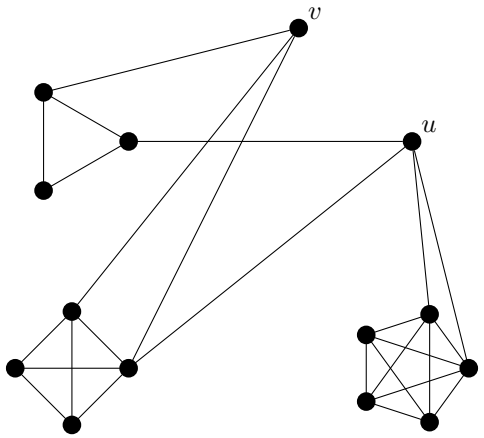


Figure 4.10: A graph G whose distance to cluster is 2: by removing u and v, G becomes a union of cliques. The signature of the triangle is $\{\emptyset, \{u\}, \{v\}\}$. The signature of the K_4 is $\{\emptyset, \{u\}, \{u, v\}\}$.

Lemma 4.26. *Let G be a cluster.*

- *If G has no isolated vertices, we have $o(G) = \mathcal{D}om$*
- *If G has a single isolated vertex, we have $o(G) = \mathcal{N}$*

- *If G has at least two isolated vertices, we have $o(G) = \mathcal{S}tall$*

Proof. This result is straightforward since, if K is a clique, it has outcome $\mathcal{N}$ if it has order 1, otherwise it has outcome $\mathcal{D}om$. The result is then given by Table 4.1. □

Theorem 4.27. *Deciding whether Dominator wins the Maker-Breaker domination game is* FPT *for the parameter distance to Cluster.*

Proof. Let X be a set of vertices such that $G \setminus X$ is a cluster. Let $k = |X|$. Let $G_1, ..., G_s$ be the connected components of $G \setminus X$. All the G_i are cliques.

We first bound the size of the cliques.

Claim 4.28. *Let $v_1, \dots, v_k$ be vertices of G_i such that $N[v_1] = \cdots = N[v_k]$. Then $o(G) = o(G \setminus \{v_3, \dots, v_k\})$.*

Proof of the claim. Consider one clique G_i. Let $Y \subseteq X$ and let $v_1, \dots, v_k$ be the vertices of G_i that have neighborhood exactly Y in X. Then $\{v_1, \dots, v_k\}$ is a module of G. By Lemma 4.20, if $k \geq 3$ has more than two vertices, it can be replaced by an edge (since the v_is induces a clique, its outcome is $\mathcal{D}om$), without changing the final outcome. Thus, all the v_is but two can be removed. ◇

The consequence of this claim is that for $1 \leq i \leq s$, we can suppose $|V(G_i)| \leq 2^{k+1}$, as there is at most 2^k different neighborhood in X, and each neighborhood has at most two neighbors in each clique.

The rest of the proof consists in bounding the number of cliques. We define the *signature* of a clique G_i as the multiset of the neighborhoods of the vertices of G_i in X, see Figure 4.10. Since at most two vertices have the same neighborhood, there are at most 3^{2^k} possible signatures (for each of the 2^k possible neighborhoods, either zero, one or two vertices share this neighborhood). Moreover, if G_i and G_j have the same signature, since we consider multisets, they must have the same number of vertices. For cliques of size not equal to 2, there are at most two other cliques with the same signature. More precisely:

Claim 4.29. *Let G_i be a clique of size ℓ.*

1. *If $\ell = 1$, and if at least two other cliques have the same signature, then all of them but two can be removed without changing the outcome of G.*
2. *If $\ell \geq 3$, and if at least three other cliques have the same signature, then all of them but three can be removed without changing the outcome of G.*

Proof of the claim.

1. Assume G_i has size 1, let u_i be the unique vertex of G_i and Y its neighborhood in X. Let H be all the other single vertices having signature $\{Y\}$. Then H is a stable set that is a module of G. If $|H| > 2$, by Lemma 4.20, it can be replaced by a stable set of size 2.

2. Assume now that G_i has size at least 3. Let $\mathcal{C}$ be all the cliques with the same signature as G_i (with $G_i \in \mathcal{C}$). Assume that $\mathcal{C}$ contains at least four cliques. Let Y be all the vertices of X connected to at least one vertex of G_i. We claim that $\mathcal{C}$ can be replaced by an edge connected by a join to the set Y. Let G' be this new graph and x, y be the vertices of the new edge. We prove that G and G' have the same outcome.

   First note that using the Super Lemma, since x and y have exactly the same neighborhood, we can assume that Dominator will claim x and Staller will claim y in G'. In particular, all the set Y is dominated by Dominator.

   Assume first that Staller is winning in G' (going first or second). Then he applies the same strategy in G ignoring the moves of Dominator in $\mathcal{C}$. Since one can assume in G' that Dominator will claim x, the vertex, says z, isolated by Staller in G' cannot be a vertex among $Y \cup \{x, y\}$. In particular, z is not connected to any vertex of $\mathcal{C}$ in G and thus will also be isolated by Staller in G.

   Assume now that Dominator is winning in G' (first or second). As said before, we can already assume that x is claimed by Dominator and y by Staller in G'. Let $\mathcal{S}'$ be the strategy of Dominator in G'

when assuming that x is claimed by Dominator and y by Staller. Let $C_1, ..., C_m$ be the cliques of $\mathcal{C}$ ($m \geq 4$). For $j \in \{1, ..., m\}$, let $\{u_j^t\}$ with $t \in \{1, ..., \ell\}$ be the vertices of C_j such that for a fixed t, the vertices u_j^t have the same neighborhood in X. While Staller is not playing in $\mathcal{C}$, Dominator follows her strategy $\mathcal{S}'$. Without loss of generality, one can assume that u_1^1 is the first vertex claimed by Staller in $\mathcal{C}$. Then Dominator claims u_2^1 and thus dominates the whole clique C_2. After that, Dominator goes on following $\mathcal{S}'$ outside $\mathcal{C}$ and follows a pairing strategy $\mathcal{P}$ in $\mathcal{C}$ with the following pairs:

(a) For $t \in \{2, ..., \ell - 2\}$, (u_1^t, u_2^t).
(b) $(u_2^{\ell-1}, u_3^{\ell-1})$ and (u_2^ℓ, u_4^ℓ)
(c) $(u_1^{\ell-1}, u_1^\ell)$
(d) For $j \in \{3, ..., m\}$, (u_j^1, u_j^2).

The pairs from items (a) and (b) ensure that at least one u_i^t is claimed by Dominator for each t while items (c) and (d) ensure that at least one vertex in each clique C_j is claimed by Dominator. Note that the vertices of $\mathcal{C}$ that are not paired can be ignored as they will be dominated by (c) and (d), and cannot dominate new vertices.

This way, Dominator will dominate all the vertices of $\mathcal{C}$ and will dominate all the vertices of Y. All the other vertices of G are present in G' and dominated by another vertex than x. Thus, Dominator will dominate the whole graph G. $\diamond$

We now consider the cliques of size 2. We prove that if there are enough cliques of size 2 with the same signature, then we can remove one of them without changing the outcome. By repeating this argument, it proves that we can assume that the number of cliques of size 2 with the same signature is bounded by a function of k.

Claim 4.30. *Let $f(k) = (2^k + 3)3^{2^k} + 2$. Let $X_1, X_2 \subseteq X$. Assume that G has more than $f(k)$ cliques of size 2 with the same signature (X_1, X_2). Let G' be the graph G with one of them removed. Then G and G' have the same outcome.*

Proof of the claim.

Let $\mathcal{C}$ be the set of cliques of size 2 with signature (X_1, X_2). Let $C_1, ...C_{|\mathcal{C}|}$ be the cliques of $\mathcal{C}$. We assume that $|\mathcal{C}| > f(k)$. For $j \in \{1, ..., \mathcal{C}\}$, let u_j and v_j be the two vertices of C_j such that all the u_j (respectively all the v_j) have neighborhood X_1 (resp. X_2). Let G' be the graph where the clique $C_{|\mathcal{C}|}$ is removed from G. We prove that G and G' have the same outcome. If Dominator wins in G' (first or second), then she wins in G using a pairing strategy on $C_{|\mathcal{C}|}$ and the same strategy she was using before on the rest of G.

Thus assume that Staller wins in G'. Let $\mathcal{S}'$ be a strategy for Staller in G'. We want to prove that Staller wins in G.

We simulate a game in G until $f(k) - 1$ cliques of $\mathcal{C}$ have at least one vertex claimed by a player with Staller that follows $\mathcal{S}'$ in G. Without loss of generality, we can assume the $f(k) - 1$ cliques played are C_1 to $C_{f(k)-1}$.

We observe the following facts:

1. If Dominator has claimed one vertex u_i and one vertex v_j, then by the Super Lemma, one can assume that for all the empty cliques of $\mathcal{C}$ there will be one vertex claimed by Staller and one by Dominator, since for all $k \notin \{i, j\}$ and all set D of vertices containing u_i and v_j, we have that $D \cup \{u_k\}$ is a dominating set if and only if $D \cup \{v_k\}$ is. Then the clique $C_\mathcal{C}$ can be removed, and we obtain the graph G' where Staller will win by hypothesis.

2. If Dominator is claiming two vertices in $\mathcal{C}$ with no vertices claimed by Staller in $\mathcal{C}$ in between, then Dominator has interest to claim two vertices u_i and v_j by Lemma 1.78, and then we are back to Case 1.

3. Assume that Dominator has already claimed a vertex v_i. After this move, if Staller claims a vertex u_j and immediately after Dominator claims v_j then the clique C_j can be removed: Dominator does not dominate any new vertex except u_j and v_j and thus the resulting game is a configuration of G' on which Staller wins. If Dominator claim at some point v_j, then it means that v_j had no unclaimed neighbor in X_2 nor neighbor claimed by Dominator otherwise it is better for Dominator to claim any neighbor of v_j by Lemma 1.78. In particular, when Staller claims a vertex u_t after that, Dominator has to answer v_t immediately, otherwise Staller isolates a vertex by playing it, and then we can remove the two vertices.

4. Thus one can consider that Dominator has only claimed the vertex v_1 in $\mathcal{C}$ and that Staller claims all the vertices u_1 to $u_{f(k)-2}$. When Staller was claiming a vertex u_j, Dominator answered outside $\mathcal{C}$. We will now count the number of optimal moves outside $\mathcal{C}$. Dominator could claim one of the k vertices of X. For the vertices outside X, there are at most 3^{2^k} different signature. For each signature not corresponding to a clique of size 2, there are at most $2^k + 3$ vertices that are vertices with distinct neighborhood in X or last vertex of a clique that need to be dominated (at most one for each class). For cliques of size 2, once Dominator has claimed the two vertices with distinct neighborhood, from what precedes, she will never claim a vertex in a clique except if Staller is claiming a vertex and threats to isolate a vertex. Thus, when Staller is claiming a vertex of $\mathcal{C}$, Dominator will play at most twice in each class of cliques of size 2. Therefore, in total there are at most $(2^k + 3)3^{2^k} = f(k) - 2$ interesting vertices. Once all these vertices have been claimed by Dominator, she should answer in $\mathcal{C}$ directly after the next move of Staller. Since there is still a vertex $u_{f(k)-1}$ unclaimed she can claim it, and we are back to Case 1. ◇

To conclude the proof, We proved that, for each signature, we can reduce the graph G to a graph in which at most $f(k)$ cliques share a signature. As there are at most 3^{2^k} signatures in total, and as each clique has at most 2^{k+1} vertices, we can reduce the graph to a kernel having $f(k)3^{2^k}2^{k+1} = O(3^{2^{k+2}})$ vertices.

We recall again, that, by Lemma 1.60 having a kernelization algorithm is equivalent to be FPT. Therefore, the Maker-Breaker domination game is FPT parameterized by the distance to cluster. □

4.8 Further work

We obtained several results for the Maker-Breaker domination game, parameterized by different graph parameters. We studied the distance to cluster as clusters are graphs in which Dominator wins easily, we can wonder what happens to other classes of graphs where the winner can be computed in polynomial time. For instance, considering the feedback vertex set instead of the feedback edge set would give a distance to forests in terms of number of vertices and not edges. This parameter seems a bit difficult to handle, since we cannot use it to bound the number of vertices of large degree, as we did for the feedback edge set. A first step could be the distance to union of stars, as stars are graphs on which Staller wins easily.

Concerning the modular-width, the proof provided here is quite general but does not provide us a kernel. Thus, on the one hand, one could try to obtain a kernel for this parameter. On the other hand, since this method mostly relies on an application of the Super Lemma, it would be interesting to try to find lemmas similar to Lemma 4.20 for other positional games, as it would imply that several games are FPTparameterized by the modular-width, which would be stronger than the neighborhood diversity.

Moreover, among the parameterized results obtained for the H-game in the previous chapter, we never considered graph parameters. It would be interesting to consider some of them, such as the feedback edge set, which would be pertinent since we managed to obtain several results on trees.

The W[2]-hardness of Maker-Breaker games parameterized by the number of moves required for a win of Breaker, obtained is Section 4.3 almost completes the result of Bonnet *et al.* [BGL$^+$17] when parameterized by the number of moves required for Maker. However, we only proved the hardness here, and proving that this parameterized game belongs to W[2] would be a good result to complete this chapter.

Conjecture 4.31. *Determining the winner of a Maker-Breaker game parameterized by the number of moves required for Breaker to win is* W*[2]-hard.*

Finally, in this study, we focused on Maker-Breaker games, and we did not provide results for other conventions. Since Bonnet *et al.* proved that the general Avoider-Enforcer is co-W[1]-complete parameterized by the number of moves required for Avoider to win, and since the Avoider-Enforcer and Maker-Breaker games are quite similar, we make the following conjecture:

Conjecture 4.32. *Determining the winner of an Avoider-Enforcer game parameterized by the number of moves required for Enforcer to win is co-*W*[2]-hard.*

Chapter 5

Maker-Maker domination game

Despite the fact that Maker-Maker games are more natural than Maker-Breaker games, they are harder to handle. Therefore, more games have been introduced in Maker-Breaker convention, and only few results are known about the Maker-Maker convention on them. This is especially the case for the domination game, introduced by Duchêne *et al.* [DGPR20]. Our main result is that, similar to the Maker-Breaker convention, the Maker-Maker domination game can be solved in polynomial time in forests. However, unlike the result provided in Maker-Breaker, which proves that if Maker wins, she can win with a pairing strategy, here we have a much harder algorithm and most of this chapter will consist of proving it. We refer the reader to the previous chapter for a summary of the results about the Maker-Breaker domination game.

In Section 5.1, we present some results from Maker-Breaker that can be applied to the Maker-Maker convention. Then, in Section 5.2 we present some useful results, that will help us to reduce some instances, and we introduce the notion of traps on which most of the proof relies. In Section 5.3, we compute the outcome on paths and cycles. Next, in Section 5.4, we present the main theorem and introduce *favorable components*, which are the main tool in the proof of the algorithm. In Section 5.5, we give the proof of the correctness of the algorithm.

This work was a collaboration with Eric Duchêne, Arthur Dumas, Aline Parreau and Eric Remila. It was published in Discrete Applied Mathematics [DDO+24]

Notations

Since several proofs in this chapter require considering positions with different players starting, we introduce the following notations:

- If $P = (G, V_A, V_B)$ is a position, we denote by (P, A) (resp. (P, B)) the *pointed position*, consisting in considering the game on P when it is Alice's turn (resp. Bob's turn).
- If P is a position and t a player, we denote by $o(P, t)$ the outcome on the pointed position (P, t). We recall that, in Maker-Maker games, by Lemma 1.14, we have $o(P, t) \in \{\mathcal{A}, \mathcal{D}\}$ if the moves are made optimally. Therefore, it may happen that moves of Alice are forced if she tries to dominate but not she tries to draw. By considering these moves, Bob can dominate before Alice. If it happens, it shows that Alice has no optimal strategy, and therefore we state that the outcome is $\mathcal{D}$ instead of $\mathcal{B}$.
- If $P = (G, V_A, V_B)$ is a position, and u, v are two unclaimed vertices of P, we denote by $P_{u,v}$ the position $(G, V_A \cup \{u\}, V_B \cup \{v\})$, i.e. the position obtained after Maker has claimed u and Breaker has claimed v.
- Two positions P and P' are said to be *equivalent* if for any $t \in \{A, B\}$, $o(P, t) = o(P', t)$. Two pointed positions (P, t) and (P', t) with the same first player are said to be *equivalent* if $o(P, t) = o(P', t)$.
- We order outcomes, stating that $\mathcal{A} > \mathcal{D}$. With this convention, a pointed position (P, t) is *better* for Alice (respectively Bob) than a pointed position (P', t') if $o(P, t) \geq o(P', t')$ (respectively $o(P, t) \leq o(P', t')$).

5.1 Comparison between Maker-Breaker and Maker-Maker conventions

In this section, we first present the results from the Maker-Breaker domination game that have applications in the Make-Maker convention.

5.1.1 General complexity

Among the results presented by Duchêne *et al.* [DGPR20], there is the PSPACE-completeness of the game, presented in Theorem 4.1. We use this result to prove that it is also PSPACE-complete in Maker-Maker convention.

Theorem 5.1. *Computing the winner of the Maker-Maker domination game is* PSPACE*-complete even if G is bipartite or split.*

Proof. First note this problem is in PSPACE, by application of Lemma 2.2 from Schaefer [Sch78], since there are most $|V|$ turns and during each turn, at most $|V|$ moves are available.

We prove that it is PSPACE-hard by a reduction from Maker-Breaker domination game, which is proved to be PSPACE-hard from Theorem 4.1.

We do the reduction as follows. Let $G = (V, E)$ be a graph with Staller starting. Consider $G' = (V \cup \{v_0\}, E)$ with v_0 a new isolated vertex. Note that the property of being split or bipartite is maintained by this operation. We prove that Dominator wins the Maker-Breaker domination game on G going second, if and only if Alice wins the Maker-Maker domination game on G'.

Suppose first that Dominator has a winning strategy $\mathcal{S}$ on G going second. We define the following strategy for Alice on G': first claim v_0, then apply $\mathcal{S}$. By hypothesis, this strategy is a winning strategy for Dominator, thus, the set of vertices claimed by Alice at the end of the game will dominate the graph. As Bob cannot dominate v_0, he cannot dominate before her, thus Alice wins.

Reciprocally, suppose that Staller has a winning strategy $\mathcal{S}$ on G going first. We define the following strategy for Bob on G'. If Alice does not claim first v_0, Bob claims it. Alice cannot dominate v_0 any longer, so the outcome is $\mathcal{D}$. Otherwise, apply $\mathcal{S}$. By hypothesis, this strategy is a winning strategy for Staller, thus, the set of vertices claimed by Alice at the end of the game will not dominate G, and the outcome is $\mathcal{D}$. □

5.1.2 Pairing strategies

Pairing dominating sets have also been introduced, see Definition 4.2, as they are a sufficient condition for a win of Dominator. Here, in Maker-Maker, this is no longer true, and a counterexample is given in Section 5.3. However, Lemma 4.3 can be adapted to graphs with a pairing dominating set of the same size as their smallest dominating set of G.

Lemma 5.2. *Let G be a graph. If G has a pairing dominating set of size $\gamma(G)$, then Alice has a winning strategy in G.*

Proof. Let G be a graph. Suppose G has a pairing dominating set of size $\gamma(G)$. By claiming her γ first moves in it, and claiming in the same pair as Bob if Bob claims a first vertex in a pair, Alice can dominate in γ moves. Bob cannot dominate before by definition of $\gamma(G)$. Therefore, Alice has a winning strategy in G. □

Corollary 5.3. *Let G be a connected cograph. We have $o(G) = \mathcal{A}$.*

Proof. This result is straightforward as any connected cograph either has a universal vertex, and Alice wins by claiming it, or it has a pairing dominating set of size two and has $\gamma(G) = 2$. □

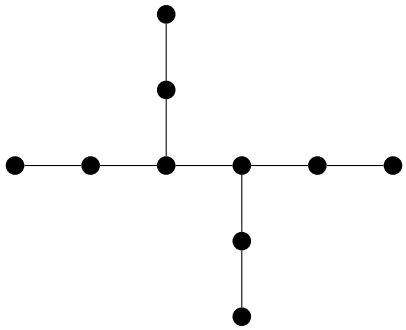


Figure 5.1: Example of a graph where Bob ensures a Draw in Maker-Maker, but if we remove any leaf with its neighbor, Alice wins.

Even if connected cographs are solved easily in Maker-Maker, dealing with disconnected graphs is not easy in the Maker-Maker convention, as Table 4.1 cannot be applied in Maker-Maker games. Actually, we did not manage to determine the outcome of a general cograph, and we let it as an open problem. Note that determining the outcome for cographs in Maker-Breaker convention is polynomial (see [DGPR20]), but finding the minimum number of moves needed by Dominator to win is surprisingly open [GIK19].

We now generalize the notion of pairing dominating sets to the notion of A-pairing, which consists in pairing strategies on some positions of the game, i.e. with some vertices already claimed. We do not introduce B-pairing, since Bob is not supposed to win in a Maker-Maker game.

Definition 5.4. *Let $G = (V, E)$ be a graph and $P = (G, V_A, V_B)$ a position. A set of disjoint pairs of unclaimed vertices $\mathcal{P} = \{(u_1, v_1), ..., (u_k, v_k)\}$ of V is a* A-pairing *of position P if for each transversal T of $\mathcal{P}$, the set $V_A \cup T$ dominates V.*

5.1.3 Removing leaves

The key ingredient to solve trees in the Maker-Breaker convention was to remove leaves using Lemma 4.6. The situation is a quite different in the Maker-Maker convention. Indeed, Lemma 4.6 is not true anymore and one cannot reduce trees as in Maker-Breaker convention. Indeed, leaves are still playing an important role as shown by Figure 5.1. Actually, one can prove that it is always optimal for Bob to claim the unique neighbor of a leaf:

Lemma 5.5. *Let $P = (G, V_A, V_B)$ be a position with an unclaimed leaf ℓ for which its unique neighbor u is also unclaimed. Then $o(P, B) = o(P_{\ell,u}, B)$*

Proof. Suppose first that $(P_{\ell,u}, B)$ is $\mathcal{D}$. Bob can, in (P, B), claims u first. Then ℓ has no unclaimed neighbor. Thus, Alice must claim it otherwise Bob claims it and isolates it. Thus, the game is now $(P_{\ell,u}, B)$ that is $\mathcal{D}$ by hypothesis. Thus, (P, B) is $\mathcal{D}$.

We prove the other implication by contraposition. Suppose $(P_{\ell,u}, B)$ has outcome $\mathcal{A}$. Let $\mathcal{S}$ be a winning strategy for Alice in $(P_{\ell,u}, B)$. We consider a strategy $\mathcal{S}'$ for Alice in (P, B) defined as follows:

- If Bob claims a vertex in $\{\ell, u\}$, Alice claims the other one.
- Otherwise, Alice claims according to $\mathcal{S}$ (ignoring the moves on ℓ and u) until she dominates all the vertices of $V \backslash \{\ell, u\}$. If u, ℓ have been claimed at this moment, either Bob has claimed u and Alice ℓ, and she wins since the strategy played is the same as $\mathcal{S}'$, or Bob has claimed ℓ and Alice u, which dominates more vertices than ℓ. Therefore, in both cases, her winning strategy in $\mathcal{S}'$ ensures her that she dominates first. Otherwise, when she dominates all the vertices of $V \backslash \{\ell, u\}$, Bob does not dominate in the game played on $(P_{\ell,u}, B)$. Thus, claiming in $\{u, \ell\}$ does not make him dominate G and if he

does not claim in $\{u, \ell\}$, he does not dominate ℓ. Therefore, by claiming at her next move a vertex in $\{\ell, u\}$, Alice wins.

Thus, Alice has a winning strategy in (P, B), finishing the proof. □

In other words, after the first move of Alice, one can always assume that Bob has claimed all the vertices that are adjacent to leaves of G, and that Alice has answered by claiming all the leaves. This will be particularly important when dealing with trees.

5.2 Preliminaries

We can now focus on new results in Maker-Maker convention. In this section, we introduce several lemmas and structure on which the proof for forest relies.

5.2.1 Union of components

First, we consider union of components, even if the Maker-Breaker results from Table 4.1 cannot be applied here. The first results which is quite natural states that a component already dominated by both players can be removed without changing the outcome.

Observation 5.6. *Let $P = (G, V_A, V_B)$ and $P' = (G', V'_A, V'_B)$ be two positions on disjoint graphs. Assume that V'_A and V'_B are dominating sets of G'. Then $P \cup P'$ and P are equivalent.*

One cannot in general determine the outcome of a position $P \cup P'$ knowing the outcome of P and P'. However, when both positions have outcome $\mathcal{A}$ when Bob starts, their union still have the outcome $\mathcal{A}$:

Observation 5.7. *Let P and P' be two positions such that $o(P, B) = o(P', B) = \mathcal{A}$. Then we have $o(P \cup P', B) = \mathcal{A}$.*

Proof. Alice follows her strategies responding as the second player in both P and P' until she dominates one of the component, say P. Then she just plays in P' following her strategy. She might claim elements of the board in P' several times in a row (if Bob goes on playing in P), but by Lemma 1.15, it is always better for Alice to play first than second. Thus, she will be able to dominate P', before Bob does. □

5.2.2 Splitting the game

When considering a position, it could be useful to decompose it into several disjoint games, as it was done in the previous chapter with Lemma 4.8. To do such a decomposition, the winning sets for both players should be the same. This is the case when a cut set is completely claimed and dominated by both players.

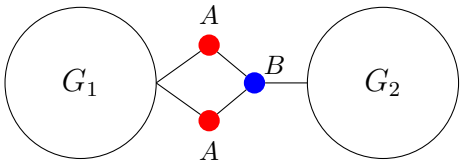
Lemma 5.8. *Let $P = (G, V_A, V_B)$ be a position. Assume $V(G)$ can be partitioned into three sets V_1, V_2, X such that:*

- *There are no edges between V_1 and V_2.*
- *All the vertices of X have been claimed: $X \subseteq V_A \cup V_B$.*
- *The vertices in X are already dominated by $V_A \cap X$ and $V_B \cap X$.*

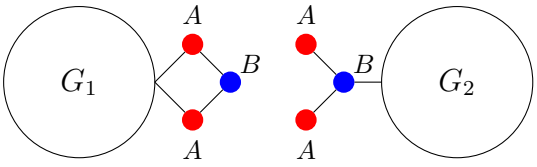
Let P_1 and P_2 be the subpositions of P induced by $V_1 \cup X$ and $V_2 \cup X$ respectively (vertices of P_1 and P_2 are disjoint). Then the position P and the position $P_1 \cup P_2$ are equivalent.

Proof. We consider the trivial bijective map f between the unclaimed vertices of P and $P_1 \cup P_2$. Let S be a set of unclaimed vertices of P and $t \in \{A, B\}$. Assume first that S is a winning set of P for Player t Let $S_1 = f(S) \cap V_1$. We prove that S_1 is a winning set of P_1 for Player t. Let u be a vertex of P_1 and $u' = f^{-1}(u)$. Either $u' \in X$ and thus is dominated by a vertex of $V_t \cap X$ in P and thus u is still dominated

in P_1. Or $u' \notin X$, and then u' is dominated by some vertex s of $S \cup V_t$ in P. Since $u' \notin X$ and X is a cutset, s should be in $V_1 \cup X$ and $f(s)$ is still in $S_1 \cup V_t$. Similarly, we can prove that $S_2 = f(S) \cap V_2$ is a winning set of P_2 for Player t, and thus $f(S)$ is a winning set of $P_1 \cup P_2$ for Player t. The reverse is easier: the union of two winning sets in P_1 and P_2 is clearly a winning set of P. Therefore, Observation 1.75 applies and the two positions are equivalent. □



(a) A graph G. The three vertices depicted are a cut set and is fully played



(b) The same graph after a cut on these three vertices

Figure 5.2: A cut in a graph, using Lemma 5.8

This result will be particularly useful in forests, since any vertex of a tree that is not a leaf is a cut vertex. Therefore, each time Alice and Bob claim two adjacent vertices that are not leaves, we can apply this lemma and cut the tree in which they were playing in into smaller ones.

5.2.3 Traps

We will frequently use the notion of *trap* that is defined in this section. Roughly speaking, an A-trap (respectively B-trap) is a vertex of a game position such that, if it is not claimed by Alice (resp. Bob) by the end of the game, it means that Alice (resp. Bob) will never build a dominating set. Formally, traps can be defined as follows:

Definition 5.9. *Let $P = (G, V_A, V_B)$ be a position of the game. An A-trap (respectively B-trap) is an unclaimed vertex v such that there exists a vertex w with $N[w] \cap V \setminus V_B = \{v\}$ (resp. $N[w] \setminus V_A = \{v\}$)*

In this definition, v corresponds to the vertex that must be claimed, and w to the vertex that will not be dominated if v is not claimed. Note that we may have $v = w$. Figure 5.3 illustrates the notion of traps.

The next lemma shows that if there is an A-trap in a position, one can consider that the next player will claim it immediately.

Lemma 5.10. *Let P be a position of the game and v be an A-trap of P. Claiming v is an optimal move for both players. Moreover, $o(P, B) = \mathcal{D}$.*

Proof. Let w such that $N[w] \cap V \setminus V_B = \{v\}$. Suppose it is Bob's turn. By claiming v, Bob isolates the vertex w as now $N[w] \subseteq V_B$. Therefore, Alice cannot dominate w any longer and the outcome is $\mathcal{D}$.

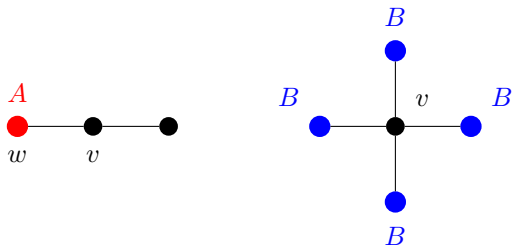


Figure 5.3: On the left, v is a B-trap and w may be isolated; on the right v is an A-trap

If it is Alice's turn, if she does not claim v, Bob claims it and once again isolates w. Therefore, v is forced for Alice. Thus, v is an optimal move. □

Corollary 5.11. *Let P be a position of the game. If there exist two A-traps v_1 and v_2 of P such that $v_1 \neq v_2$, then $o(P, A) = o(P, B) = \mathcal{D}$.*

Proof. Since there are two distinct A-traps, the two vertices w_1 and w_2 that might be isolated are necessarily distinct (otherwise v_1 and v_2 would both be in their neighborhood, contradicting the definition of a trap). Hence, even if Bob is not the first player, he will be able to claim in one of the two traps and hence isolate either w_1 or w_2. □

The next lemma proposes another example of a forced move for Alice when there exists an unclaimed P_5 in a position as a subgraph.

Lemma 5.12. *Let (P, B) be a pointed position of the game with $P = (G, V_A, V_B)$. If there exists a path $G' = (v_1, v_2, v_3, v_4, v_5)$ such that G' is a subgraph of G with $V(G') \cap (V_A \cup V_B) = \emptyset$ and v_2, v_3, v_4 of degree 2, then if Bob claims v_3, Alice is then forced to answer on v_2 or v_4.*

Proof. If Bob claims v_3, then if Alice answers elsewhere than on v_1, v_2 or v_4, Bob claims his second move on v_2 and creates two A-traps in v_1 and v_4. Indeed, since the vertices v_2 and v_3 are of degree 2, there remains only one way for Alice to dominate v_2 (i.e. by claiming v_1) and v_3 (i.e. by claiming v_4). By corollary 5.11, the resulting position has outcome $\mathcal{D}$. If Alice claims v_1, then Bob claims v_4 and by symmetry creates two A-traps equivalent to the previous case. □

Traps play a strong role in the computation of opitmal moves, as they can quite often provide simple strategies. For instance, if Alice can prevent Bob to dominate the graph, A-pairings are then enough for Alice to win. This can be used in particular where there are some B-traps in the game.

Lemma 5.13. *Let P be a position with a B-trap and an A-pairing. Then, we have $o(P, A) = \mathcal{A}$.*

Proof. Let P be a position containing a B-trap v and an A-pairing $\mathcal{S}$. Alice playing first can claim v. Now, Bob cannot dominate G anymore. Therefore, by following a pairing strategy using $\mathcal{S}$, Alice will claim a dominating set, by definition of A-pairings. Thus, she will dominate the whole graph and win. □

Lemma 5.14. *Let P be a position with two B-traps v and v' and an A-pairing that contains nor v nor v' in P. Then, we have $o(P, B) = \mathcal{A}$.*

Proof. Alice follows a pairing strategy using $\mathcal{S} \cup \{v, v'\}$. Bob cannot dominate since she will claim either v or v' and thus, there will be a vertex not dominated by Bob. She will dominate the graph since she will claim a transversal of $\mathcal{S}$. □

5.3 Path and cycles

In this section, we consider paths. In a tree, the structure of the positions obtained after some moves will be basically union of paths where the extremities are claimed. Thus, we first deal with these paths and derive some general results on them that will help us to solve paths, and also cycles and forest later.

5.3.1 Bounded baths

A *bounded path* is a path on at least four vertices where the four vertices at its extremities (the two leftmost and the two rightmost) are already claimed by Alice and Bob in such a way that the four vertices are already dominated by both players.

Definition 5.15. *A bounded path of length n is a position (G, V_A, V_B) such that:*

- *G is a path $(v_{-1}, v_0, v_1, ..., v_n, v_{n+1}, v_{n+2})$;*

- *the unclaimed vertices are exactly vertices v_1 to v_n;*
- *exactly one vertex among $\{v_{-1}, v_0\}$ (respectively $\{v_{n+1}, v_{n+2}\}$) is in V_A, the other being in V_B.*

According to this definition, the knowledge of the label of v_0 and v_{n+1} is sufficient to deduce the label of v_{-1} and v_{n+2}. Therefore, for t, t' in $\{A, B\}$, we will denote by $[to^n t']$ the bounded path of size n such that $v_0 \in V_t$ and $v_{n+1} \in V_{t'}$. See Figure 5.4 for an illustration of $[Ao^5A]$ and $[Bo^3A]$.

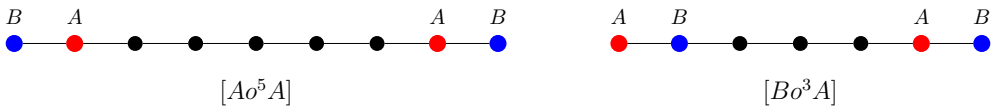


Figure 5.4: The bounded paths $[Ao^5A]$ and $[Bo^3A]$.

In some situations, bounded paths can be considered as a neutral structure that preserves the outcome when adjoined to another position. The next lemma illustrates a first case in which this may occur. It will also lead to a natural resolution of paths.

Lemma 5.16. *For any position P and any integer n, $o(P, B) = o(P \cup [Ao^n B], B)$.*

Proof. Let $P = (G, V_A, V_B)$ be a position. We do the proof by induction on the number of unclaimed vertices in $P \cup [Ao^n B]$. First note that if $n \leq 1$, the result is true since $[Ao^n B]$ is already dominated by both players. Hence, by Observation 5.6, the result is true when the number of unclaimed vertices is at most 1.

Now assume there are at least two unclaimed vertices and $n \geq 2$. Suppose first that $(P, B) = \mathcal{D}$. In this case, Bob starts by claiming v_{n-1} in $P \cup [Ao^n B]$ creating an A-trap in v_n. By Lemma 5.10, Alice has to answer v_n. By induction hypothesis $o(P \cup [Ao^{n-2} B], B) = \mathcal{D}$, which ensures that $o(P \cup [Ao^n B], B) = \mathcal{D}$.

Now assume that (P, B) has outcome $\mathcal{A}$, and let $\mathcal{S}$ be a winning strategy for Alice in this pointed position. We give a strategy for Alice in $(P \cup [Ao^n B], B)$. Let v_B be the vertex claimed by Bob in $(P \cup [Ao^n B], B)$. Alice answers as follows:

- If $v_B \in V(G)$ and if Alice does not dominate P, she claims the same vertex v_A she would have answered following $\mathcal{S}$ if Bob had claimed v_B in (P, B). The resulting position has two vertices less, and has outcome $\mathcal{A}$ by induction hypothesis.
- If $v_B \in V(G)$ and Alice already dominates P, we necessarily have $n \geq 2$, otherwise, she would already have won. She claims v_2. The resulting position is better than $(P \cup [AoA] \cup [Ao^{n-2} B], B)$, which itself is better than $([AoA] \cup [Ao^{n-2} B], B)$, as Alice already dominates P. Thus, by induction hypothesis applied with $P' = [AoA]$, which is not dominated by Bob, the position has outcome $\mathcal{A}$.
- If v_B is a vertex of $[Ao^n B]$, then v_B corresponds to some vertex v_k, with $k \in \{1, \ldots, n\}$. If $k = n$, Alice answers v_{n-1}. Otherwise, she claims v_{k+1}. Note that if $k = n$, the position obtained is better for Alice than the one obtained by $k = n - 1$ (since she dominates a superset of vertices). Therefore, and without loss of generality, we will assume that $k < n$. By Lemma 5.8, the resulting position is then equivalent to $(P \cup [Ao^{k-1} B] \cup [Ao^{n-1-k} B], B)$ which has less unclaimed vertices than the original one. By induction hypothesis applied twice, it has the same outcome as $(P \cup [Ao^{k-1} B], B)$, which also has the same outcome as (P, B), which has outcome $\mathcal{A}$ by hypothesis.

This analysis ensures that $o(P \cup [Ao^n B], B) = \mathcal{A}$, since for each claim of Bob, there exists an answer of Alice leading to a position on which Alice wins. □

The next lemma presents another situation where the adjunction of some bounded paths with particular constraints does not change a winning outcome for Alice.

Lemma 5.17. *Let $P = (G, V_A, V_B)$ be a position, let n be an integer such that $n \not\equiv 0 \mod 3$ and let n' be an integer such that $n' \equiv 0 \mod 2$. Then if $o(P, B) = \mathcal{A}$, then $o(P \cup [Ao^n A] \cup [Bo^{n'} B], B) = \mathcal{A}$.*

Proof. We prove this result by induction on the number m of unclaimed vertices of $P \cup [Ao^nA] \cup [Bo^{n'}B]$. For initialization, if $m = 1$, then P has no unclaimed vertices, $n = 1$ and $n' = 0$. Thus, Alice dominates $P \cup [Ao^nA] \cup [Bo^{n'}B]$ and the result is true, by definition.

Assume now that $m \geq 2$, If Alice dominates $(P \cup [Ao^nA] \cup [Bo^{n'}B])$ (which implies that $n \leq 2$ and $n' = 0$) then the result is true, by definition. Otherwise, we have to prove that, for each vertex x claimed by Bob, there exists an answer y for Alice such that $o(P \cup [Ao^nA] \cup [Bo^{n'}B], B) = \mathcal{A}$, the pointed position obtained after the two claim has outcome $\mathcal{A}$.

Denote by $(u_1, \ldots, u_n)$ the unclaimed vertices of $[Ao^nA]$ and by $(v_1, \ldots, v_{n'})$ the unclaimed vertices of $[Bo^{n'}B]$. We consider all the possible moves for Bob.

- If Bob claims u_1 or u_2 in $[Ao^nA]$ and $n \leq 2$, we consider two subcases. If $n' = 0$, then Alice dominates $[Ao^nA] \cup [Bo^{n'}B]$, which obviously gives the result. If $n' \geq 2$, Alice claims v_1. By Observation 5.6, the resulting game is equivalent to $P \cup [Ao^{n'-1}B]$, which is equivalent to P by Lemma 5.16.

- If Bob claims u_1 or u_2 in $[Ao^nA]$ and $n \geq 4$, by Lemma 1.78, we can suppose he claims u_2. then Alice claims u_3. Thus, the resulting position is equivalent to $P \cup [Ao^{n-3}A] \cup [Bo^{n'}B]$, which gives the result using the induction hypothesis. The case where Bob claims u_n or u_{n-1} is symmetric.

- If Bob claims a vertex u_k in $[Ao^nA]$ with $k \notin \{1, 2, n-1, n\}$, Alice answers by claiming a vertex u_i with $i \in \{k-1, k+1\}$ such that the resulting component is $[Ao^jA] \cup [Ao^{j'}B]$ with $j \not\equiv 0[3]$. Note that since $n \not\equiv 0[3]$, this vertex always exists. By Lemmas 5.8 and 5.16, the resulting position is equivalent to $P \cup [Ao^jA] \cup [Bo^{n'}B]$, which gives the result using the induction hypothesis.

- If Bob claims some vertex v_k in $[Bo^{n'}B]$, Alice answers by claiming a vertex v_i with $i \in \{k-1, k+1\}$ such that the resulting component is $[Bo^jB] \cup [Ao^{j'}B]$ with $j \equiv 0[2]$. Note that as $n' \equiv 0[2]$, this vertex always exists. By Lemma 5.16 and Lemma 5.8, the resulting position is equivalent to $P \cup [Ao^nA] \cup [Bo^jB]$, which gives the result using the induction hypothesis.

- If Bob claims a vertex x in P, we consider two subcases. If Alice does not dominate G yet, she claims the vertex she would have claimed as an answer to x in her winning strategy in P. The resulting position has two vertices less, which gives the result using the induction hypothesis.

  If Alice already dominates G, and $n' \geq 2$, then Alice claims v_1 in $[Bo^{n'}B]$, turning it into $[Ao^{n'-1}B]$. By Lemma 5.16, the game is now equivalent to $P \cup [Ao^nA]$, and Alice already dominates G, so by induction hypothesis, it is a winning position for her. For $n' = 0$ and $n \geq 4$ (cases where $n \leq 2$ are trivial), then Alice claims u_2. The resulting position is better for Alice than the position $P \cup [AoA] \cup [Ao^{n-2}B]$, which, by Lemma 5.16, is equivalent to $P \cup [AoA]$ and therefore is a winning position for Alice. □

We finish this subsection by proving that bounded paths $[Bo^nB]$, where n is odd, and $[Ao^nA]$, when $n \equiv 0 \mod 3$, are not good for Alice when Bob starts. The first one is natural and give a condition for a draw whatever the rest of the position is. The second one is more surprising. Here Bob obtains a draw mostly by threatening Alice to dominate before her. Thus, one cannot add any position and maintain a draw.

Lemma 5.18. *For any position P and any odd integer n, $o(P \cup [Bo^nB], B) = \mathcal{D}$.*

Proof. We prove the result by induction on n. If $n = 1$, there is an A-trap: by Lemma 5.10, $o(P \cup [BoB], B) = \mathcal{D}$. Now, let $n \geq 3$. Bob claims v_2 which forces Alice to claim v_1. The position is then equivalent to $(P \cup [Bo^{n-2}B], B)$ which has outcome $\mathcal{D}$ by induction. □

Lemma 5.19. *For any positive integer k, $o([Ao^{3k}A], B) = \mathsf{D}$.*

Proof. We prove the result by induction on k.

If $k = 1$, Bob claims v_2 and directly dominates $[Ao^3A]$. Thus, $o([Ao^3A], B) = \mathsf{D}$.

If $k = 2$, Bob claims v_2. If Alice does not answer in $\{v_1, v_3, v_4\}$, Bob claims v_3 at his second turn and create two A-traps in v_1 and v_4 which ensures a draw. Thus, Alice should claim a vertex among $\{v_1, v_3, v_4\}$

and does not dominate the graph at her first claim. Then Bob can claim v_5 and dominates the graph. Hence $o([Ao^6A], B) = \mathsf{D}$.

Assume now that $k \geq 3$ and that the result is true for any positive $k' < k$. Consider the position $([Ao^{3k}A], B)$. Bob claims v_5 at his first turn. By Lemma 5.12 applied on $(v_3, v_4, v_5, v_6, v_7)$, Alice should answer on v_4 or v_6. If she claims v_4, then by Lemma 5.8, the position is equivalent to the position $([Ao^3A] \cup [Bo^{3k-5}A], B)$, which is equivalent, by Lemma 5.16 to $([Ao^3A], B)$ which has outcome $\mathcal{D}$.

If she claims v_6, in the same way, the position is equivalent to $([Ao^4B] \cup [Ao^{3(k-2)}A], B)$, which is equivalent to $([Ao^{3(k-2)}A], B)$ which has outcome $\mathcal{D}$ by induction. □

5.3.2 Paths

Lemma 5.16 can be used to prove that Alice always wins on paths.

Theorem 5.20. *Let P_n be the path of length n. Then $o(P_n) = \mathcal{A}$.*

Proof. Let $v_1, ..., v_n$ be the vertices of the path. In $n \leq 3$, then Alice wins at her first turn, so we can assume that $n \geq 4$. Alice starts by claiming v_2. By Lemma 5.5, we can assume that Bob claims v_{n-1} and Alice answers by claiming v_n. Let (P, B) be the actual pointed position of the game. Using Observation 1.75, and since v_1 should be in any winning set of Bob, this position is equivalent to the position $([AoA] \cup [Ao^{n-4}B], B)$. By Lemma 5.16, $o([AoA] \cup [Ao^{n-4}B], B) = o([AoA], B) = \mathcal{A}$, which ensures that $o(G) = \mathcal{A}$. □

When playing in the Maker-Breaker convention, note that this result implies that Dominator also has a winning strategy on any path playing first. This result was already known since [DGPR20], but the strategy developed here is different from the other one.

5.3.3 Cycles

The case of cycles in the Maker-Maker convention is more subtle than for the Maker-Breaker convention, where Dominator always wins. More precisely, we will show that there are infinitely many $\mathcal{A}$ and $\mathcal{D}$ outcomes that depend on the size of the cycle modulo 3.

From now, we will denote by C_n the cycle of order n. The vertices of C_n will be denoted by v_0 to v_{n-1}.

Theorem 5.21. *Let n be an integer. We have $o(C_n) = \mathcal{D}$ if and only if $n \geq 10$ and $n \equiv 1 \mod 3$.*

Proof. We first prove the "if" part. Let $n = 3k + 1$, with $k \geq 3$, and let C_n be a cycle of order n.

By symmetry, we can assume that Alice first claims v_0 and thus $o(C_n) = o(C_n, \{v_0\}, \emptyset, B)$. We give a strategy for Bob to obtain at least a draw. The strategy on G_{10} is provided in Figure 5.5. Bob first claims v_5. By Lemma 5.12 with $G' = (v_3, v_4, v_5, v_6, v_7)$, Alice has to answer v_4 or v_6.

- If Alice claims v_6. Then Bob can claim v_3 which forces Alice to claim the A-trap in v_4 and then Bob can claim v_1 which forces Alice to claim the A-trap in v_2. At this point, the position is equivalent to $([Ao^{3(k-2)}A], B)$ which has outcome $\mathcal{D}$ by Lemma 5.19 (remember that $k - 2 \geq 1$).

- If Alice claims v_4. Bob can claim successively all the v_{2i+1} starting from v_7, creating an A-trap in v_{2i} that Alice is forced to claim. If n is even, Bob follows this strategy until claiming v_{n-1}. Then Alice has to claim v_{n-2} and does not dominate the cycle (she does not dominate v_2). Then Bob can dominate the cycle by claiming v_2.

  If n is odd, Bob follows this strategy until claiming v_{n-4}. After Alice has claimed v_{n-5}, Bob claims v_2. Alice is forced to claim v_{n-1}, otherwise Bob wins by claiming it. Then Bob claims v_{n-2}, creating a trap in v_{n-3}, forcing Alice to claim it. At this point, Alice still does not dominate v_2. Then Bob can dominate the cycle by claiming v_1.

In all cases, the game either ends in a draw or Bob dominates the graph, thus $o(C_n) = \mathcal{D}$.

We now prove the "only if" part. First consider that $n \not\equiv 1 \mod 3$. Without loss of generality, one can assume that Alice will claim v_0 at her first turn, and thus we consider the pointed position $(P, B) = ((C_n, \{v_0\}, \emptyset), B)$. Consider now the position P' obtained from P by adding a vertex claimed by Bob adjacent only to v_0. This position is better for Bob since he dominates more vertices than in P. Thus, it is enough to prove that (P', B) is a win for Alice to ensure that $o(P, B) = \mathcal{A}$. Using Observation 1.75, (P', B) is equivalent to $([Ao^{n-1}A], B)$. By Lemma 5.17, since $n - 1 \not\equiv 0 \mod 3$, $o([Ao^{n-1}A], B) = \mathcal{A}$, and thus $o(C_n) = \mathcal{A}$

It remains to prove that Alice wins on C_4 and C_7. On C_4, Alice wins by claiming any two vertices and as it is not possible to dominate in one move, she will dominate first. On C_7, Alice can claim v_0, and then use a pairing strategy with pairs (v_2, v_3) and (v_4, v_5). This way, she dominates the cycle in three moves, and Bob cannot dominate before since at least three vertices are required to dominate C_7. □

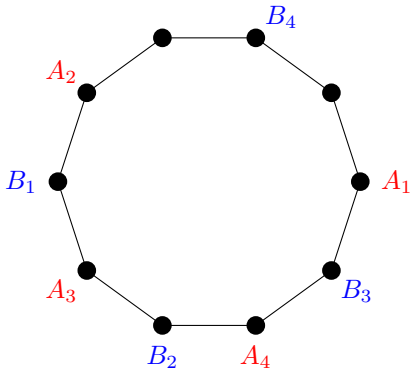


Figure 5.5: Bob's strategy on C_{10}. Moves are labelled by the turn they have been made. As most of the moves for Alice are forced, and considering symmetries, there is only one possibility. After these moves, Bob dominates the graph, but Alice does not.

5.4 A polynomial algorithm on forests

In this section, we give the essential elements to solve the case of forests, given by Theorem 5.22 below. In particular, we will first reduce the problem to any *standard* forest, meaning that non-standard forests correspond to cases that can be solved easily. Then we will give necessary conditions about the first move of Alice, yielding to the introduction of the skeleton of a forest. From that definition, we will be able to present the general algorithm that computes the outcome of any forest, as depicted by the decision tree of Figure 5.10. In the next section, we will give the full proof of the validity of the algorithm.

Theorem 5.22. *Deciding the outcome of a forest can be done in linear time.*

5.4.1 Removing small components

If there is an isolated vertex in the forest, this is like playing in the Maker-Breaker convention.

Lemma 5.23. *Let F be a forest with an isolated vertex v_0. Then $o(F) = \mathcal{A}$ if and only if $F \setminus \{v_0\}$ contains a perfect matching.*

Proof. Alice has to claim first v_0. Then Bob is playing first in $F \setminus \{v_0\}$. Since he cannot dominate anymore, he has the same role as Staller in the Maker-Breaker domination game. In [DGPR20], it is proved that Dominator playing second in a forest in the Maker-Breaker domination game wins if and only if there is a perfect matching, which gives the result. □

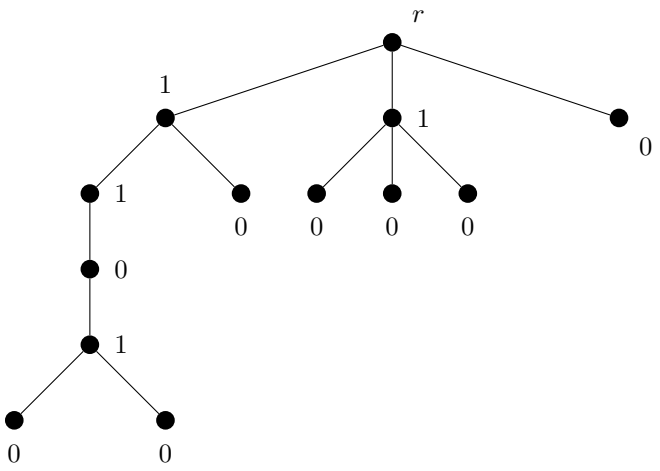


Figure 5.6: Example of labelling

Isolated edges can be removed without changing the outcome.

Lemma 5.24. *Let F be a forest with an isolated edge $e = (u, v)$. Then $o(F) = o(F \setminus \{u, v\})$.*

Proof. If Alice (respectively Bob) has a winning (resp. draw) strategy in $F \setminus \{u, v\}$, then she can apply her strategy in F by pairing u with v. The resulting strategy will still be winning (resp. leading to a draw) in F. □

By applying Lemma 5.23 and Lemma 5.24, one can consider in what follows that all the connected components have at least three vertices.

5.4.2 Bottom-to-top strategies for Bob

In this subsection, we describe a strategy for Bob that will often be considered to obtain draws or to reduce trees. Let T be a tree rooted on a vertex r, vertices of T except r are labeled inductively with values 0 and 1, starting from the leaves as follows:

- If all children of v are labeled 1, then v is labeled 0 (hence, all the leaves are labeled 0);
- If at least a child of v is labeled by 0, then v is labeled by 1.

Figure 5.6 gives an example of such a labeling.

Let T be a tree rooted in r and consider the pointed position (P, B) with $P = (T, V_A, \emptyset)$ and $V_A \subseteq \{r\}$. A *bottom-to-top strategy* for Bob on (P, B) consists, at each step, in claiming a vertex v labeled by 1 such that all the successors of v labeled by 1 are already claimed by Bob. The following property is maintained during this process: Alice is forced to claim only vertices labeled by 0, and any vertex claimed by Alice (except r) has his parent claimed by Bob. Indeed, it is true before the first claim of Bob. Assume it is true before Bob's turn. Let v be a vertex labeled by 1 such that all the successors of v labeled by 1 are already claimed by Bob. By definition of the labeling and the assumption, v has a child u labeled by 0 that is unclaimed. All the children of u are by definition labeled by 1 and thus already claimed by Bob. Thus, u is an A-trap and Alice is forced to claim it, maintaining the property true. Such a strategy can also be applied when all the leaves of T are adjacent to vertices already claimed by Bob. In this strategy, Bob can claim all the vertices labeled by 1. A particular interesting case for Bob is when there exists a vertex v labeled by 1 with two children labeled by 0:

Lemma 5.25. *Let T be a tree and consider the labeling of T rooted in v_0. If there exists a vertex labeled by 1 with two children labeled by 0, then the position (P, B) with $P = (T, \{v_0\}, \emptyset)$ has outcome $\mathcal{D}$.*

Proof. Bob uses a bottom-to-top strategy on T. When claiming v, he will create two A-traps on the two children of v labeled by 0, which concludes the proof by Corollary 5.11. □

Bob can also use bottom-to-top strategies to reduce the forest to a smaller one where he has a draw strategy.

Lemma 5.26. *Let F be a forest and v_0 a vertex. Consider the labeling of F rooted in v_0 (for the components not containing v_0, root on any vertex). Let $v \neq v_0$ be a vertex labeled by 1 and S_v be the set of successors of v in the rooted tree. Let $F_{\overline{v}}$ be the tree obtained by removing all vertices of S_v and adding a leaf v' connected to v. Then, if $o(F_{\overline{v}}, \{v_0, v'\}, \{v\}, B) = \mathcal{D}$, then $o(F, \{v_0\}, \emptyset, B) = \mathcal{D}$.*

Proof. Bob follows by a bottom-to-top strategy on S_v and can claim all the vertices of S_v labeled by 1 until claiming v. Alice is always forced to claim a child of the claimed vertex. Afterward, Bob plays his strategy to obtain a draw in $(F_{\overline{v}}, \{v_0, v'\}, \{v\}, B)$, leading to a draw for $(T, \{v_0\}, \emptyset, B)$. □

5.4.3 Cherries

A particular case where the bottom-to-top strategy will be useful is when there are some cherries in the forest. Recall that a cherry is a vertex c connected to two leaves ℓ_1 and ℓ_2. It will be denoted by the triple $C = (c, l_1, l_2)$. If F contains some cherries, the outcome of F can be easily computed.

Lemma 5.27. *Let F be a forest. If F has two cherries $C = (c, \ell_1, \ell_2)$ and $C' = (c', \ell'_1, \ell'_2)$, with $c \neq c'$, then $o(F) = \mathcal{D}$.*

Proof. Let F be such a forest. After Alice has claimed her first vertex, she cannot have claimed both c and c'. Suppose without loss of generality that Alice has claimed c'. Then Bob claims c. The resulting pointed position contains two A-traps and thus is draw by Corollary 5.11. □

Lemma 5.28. *Let F be a forest with exactly one cherry $C = (c, \ell_1, \ell_2)$. Then $o(F) = \mathcal{A}$ if and only if there is a matching in $F \setminus \{c\}$ that covers $V(F) \setminus N[c]$.*

Proof. Suppose first that F has a matching M in $F \setminus \{c\}$ that covers $V(F) \setminus N[c]$. Then Alice claims c, which creates a double B-trap in ℓ_1 and ℓ_2. The matching M is actually an A-pairing, and, we can suppose it contains neither ℓ_1 nor ℓ_2 since these two vertices have their neighborhood included in $N[c]$. Then by Lemma 5.14, Thus $o(F) = \mathcal{A}$.

Suppose now that F has no such matching M. We define a strategy for Bob as follows. If Alice's first claim is not element of the cherry, then Bob claims c and creates two A-traps, leading to a draw position. Thus, we can assume that Alice's first claim r is and element of $\{c, \ell_1, \ell_2\}$. Let T be the connected component of F containing r. If there exists another connected component T' of F that has no perfect matching, then Bob can apply the strategy of Staller playing first in T' that prevents Dominator to dominates T' (see [DGPR20]). Thus, one can assume that there is a perfect matching in all the components of F distinct from T. Now label the vertices of T rooted in r as defined in Section 5.4.2. We want to prove that there exists a vertex labeled 1 with two children labeled 0, which will ensure a draw strategy for Bob by applying the bottom to top strategy. If it is not the case, then consider the matching M' where all the vertices labeled 1 (except c) are paired with their unique child labeled 0. We claim that M' covers $V(T) \setminus N[c]$. Indeed, assume $x \in V(T) \setminus N[c]$ is not covered by M. Then x must be labeled 0 and its parent should be r since it is the only vertex not labeled. But then $x \in N[c]$. Thus, there exists a matching in $F \setminus \{c\}$ that cover $V(F) \setminus N[c]$: take the union of M' and the perfect matchings of all the other components.

□

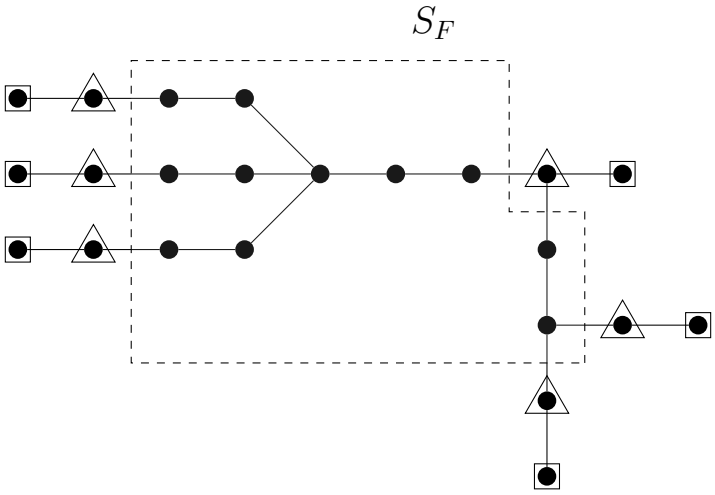


Figure 5.7: A forest F and its skeleton S_F. Note that F is connected but S_F is not. M_F is the set of vertices in triangles, and L_F is the set of vertices in squares.

5.4.4 Definition of the skeleton and easy cases

Considering Lemmas 5.27 and 5.28, we will assume now that there is no cherry in F, no isolated vertex and no isolated P_2. From Lemma 5.5, one can assume that after the first turn of Alice, Bob will claim all the unclaimed vertices of F with a leaf as a neighbor. Alice is then forced to answer on each leaf. This motivates us to define the *skeleton* of F, denoted by S_F, as the vertices that are not a leaf nor a parent of a leaf.

More formally, we denote by L_F the leaves of F and by M_F their parents. Remark that, with the hypothesis that there is neither cherry nor isolated vertex, we have $L_F \cap M_F = \emptyset$ and the mapping : $L_F \to M_F$, which associates to each vertex v of L_F its parent, is bijective. Then, let S_F be defined by $S_F = V(F) \setminus (L_F \cup M_F)$. Figure 5.7 is an illustration of a forest with its skeleton.

In some simple cases, we can directly give the outcome of F.

Lemma 5.29. *If S_F is empty, then $o(F) = \mathcal{A}$.*

Proof. Set $L_F = \{\ell_1, \ldots, \ell_k\}$. By hypothesis, F has no cherry, therefore, no vertex can be adjacent to two of these leaves. Thus, we can denote $M_F = \{m_1, \ldots, m_k\}$ with m_i adjacent to ℓ_i for $1 \leq i \leq k$. Note that k vertices (one on each set $\{\ell_k, m_k\}$) are necessary and sufficient to dominate F. Thus $\gamma(F) = k$ and there is a pairing dominating set of size k (the set of pairs $\{(\ell_i, m_i), i \in \{1, ..., k\}\}$). By Lemma 5.2, Alice has a winning strategy in F.

□

Lemma 5.30. *Let T be a connected component of F such that S_T is empty. Then $o(F) = o(F \setminus T)$.*

Proof. Assume first that $o(F \setminus T) = \mathcal{A}$ and let x be the first claim of Alice in a winning strategy. Then Alice claims x in F. By Lemma 5.5, Bob will claim all the vertices of M_T and Alice will answer all the vertices of L_T. After these moves, T is completely claimed and dominated by both players. By Observation 5.6, we can remove this component. The game is then equivalent to $(F \setminus T, \{x\}, \emptyset)$ which is a win for Alice when Bob starts, leading to $o(F) = \mathcal{A}$.

Assume now that $o(F \setminus T) = \mathcal{D}$. Consider the game played in F and a first claim x of Alice. If $x \in V(F \setminus T)$, then as before, the position $(F, \{x\}, \emptyset)$ is equivalent to the position $(F \setminus T, \{x\}, \emptyset)$ which ends in a draw when Bob starts. If $x \in V(T)$, let x' be the unique neighbor of x if x is a leaf or the leaf connected to x if $x \in M_T$. Then we can assume by Observation 5.6 that Bob claims all the vertices in M_T (except x or x') and that Alice answers by claiming all the vertices in L_T (except x or x'). At this point, all the vertices of T have been claimed except x'. Then Bob claims x'. Then both players dominate T and the position is

equivalent to $(F \setminus T, A)$ which is a draw position. In conclusion, whatever Alice claims, Bob can ensure a draw in F. Thus $o(F) = \mathcal{D}$ □

Lemma 5.31. *If S_F induces a star of center c such that c has no neighbor in M_F, then $o(F) = \mathcal{A}$.*

Proof. Alice claims the center c as a first move. Then as explained above, Bob claims all the vertices of M_F and Alice answers all the leaves of L_F. After that, Alice dominates the whole graph and Bob does not, since he does not dominate c.

□

We say that that F is *standard* if all its connected components have at least four vertices and a non-empty skeleton, if F has no cherries, and if S_F does not induce a star of center c where c has no neighbor in M_F. We now focus on the standard forests.

5.4.5 First move of Alice

If F is standard, next lemmas say that it can be assumed that Alice must claim outside S_F and must connect it. The main idea behind this result is that, if Alice claims in S_F, she claims too far from the leaves and Bob can win with a bottom-to-top strategy.

Lemma 5.32. *Let F be a standard forest and $v_0 \in S_F$. We have $o((F, \{v_0\}, \emptyset), B) = \mathcal{D}$.*

Proof. Assume first that the graph induced by S_F is either a unique vertex, a unique edge, or a star (whose center has necessarily a neighbor in M_F). Let $L_F = \{\ell_1, \ldots, \ell_k\}$ and $M_F = \{m_1, \ldots, m_k\}$ such that for $1 \leq i \leq k$, ℓ_i and m_i are neighbors. Starting from $(F, \{v_0\}, \emptyset)$, Bob successively claims $m_1, \ldots, m_k$, which forces Alice to reply $\ell_1, \ldots, \ell_k$. When Bob has just claimed m_k, he dominates the whole graph while Alice does not yet dominate ℓ_k. This ensures that $o((F, \{v_0\}, \emptyset), B) = \mathcal{D}$.

It can now be assumed that the graph induced by S_F is neither a unique vertex, a unique edge, nor a star. Thus, we can consider the position $P' = (F, \{v_0\} \cup L_F, M_F)$, and from successive applications of Lemma 5.5, we just need to prove that $o(P', B) = \mathcal{D}$.

First focus on components of S_F which do not contain v_0. Let C such a component, if each vertex of C is dominated by a vertex of M_F, then Bob dominates C while Alice does not. Otherwise, let v_1 be a leaf of the subtree T_C of F induced by C. Bob plays the bottom-to-top strategy in T_C rooted in v_1. It can be done since the leaves of T_C are only connected to vertices of M_F already claimed by Bob. If two vertices labeled by 0 have the same parent, then the strategy creates a double trap, which ensures a draw. If v_1 has all its children labeled by 1, when Bob claims the last vertex labeled by 1, a double trap is created, since v_1 will be a trap. This also ensures a draw. The only non directly conclusive case is when v_1 has one child labeled by 0, the reached position after Bob has followed the bottom-to-top strategy is such that Bob dominates C while Alice does not. Indeed, Bob dominates v_1 since it is a leaf of C and thus should be connected to M_F but Alice does not dominate v_1.

Bob follows this strategy on each component of S_F not containing v_0. All the answers from Alice are forced. If a double trap appears, we are done. Otherwise, each component is dominated by Bob and not by Alice. Moreover, it is Bob's turn. In such a case , we now need to focus on the component C_0 of S_F which contains v_0.

- If the diameter of C_0 is at most 1, (which implies that the graph induced by S_F is not connected), Bob already dominates C_0, and therefore F, but Alice does not.
- If the diameter of C_0 is 2, the graph induced by C_0 is a star centered in a vertex c. Since it is assumed that the graph induced by S_F is not a star, C_0 is not the only component of the graph induced by S_F. Thus, if Bob does not dominate C_0 yet, he claims c or any of its neighbor and dominates the whole graph while Alice does not.

- If the diameter of C_0 is at least 3, there exists a vertex v_2 at distance exactly 2 from v_0 in the subgraph induced by C_0. Let (v_0, v_1, v_2) be the path from v_0 to v_2. Bob can then use a bottom-to-top strategy in the tree induced by C_0 rooted in v_0.
  - If v_1 is labeled by 0, then v_2 is labeled by 1. Bob can play a bottom-to-top strategy claiming all the vertices labeled by 1 except v_2. Then he claims v_1. At this moment, Bob dominates F but Alice does not dominate v_2.
  - If v_1 is labeled by 1, Bob plays a bottom-to-top strategy in all branches, finishing by claiming v_1. At this moment, he dominates F but, by construction, Alice does not dominate v_2.

Thus, in any case, Bob has a strategy which leads to a draw position, and thus $o((F, \{v_0\}, \emptyset), B) = \mathcal{D}$. □

Lemma 5.33. *If F is standard and $o(F) = \mathcal{A}$, then there exists a vertex $v_0 \in M_F$ such that $o((F, \{v_0\}, \emptyset), B) = \mathcal{A}$. Moreover, v_0 satisfies:*

1. *the subgraph of F induced by $S_F \cup \{v_0\}$ is a tree ;*
2. *in the labeling of F rooted in v_0, each vertex v labeled by 1 has a unique child.*

Proof. As it is supposed that $o(F, A) = \mathcal{A}$, there exists a vertex $v_0 \in V(F)$ such that $o((F, \{v_0\}, \emptyset), B) = \mathcal{A}$. From Lemma 5.32, we have $v_0 \in M_F \cup L_F$. From Lemma 1.78, it can be assumed that $v_0 \notin L_F$. Indeed, if $l \in L_F$ and $m \in M_F$ is its private neighbor, $N[l] \subset N[m]$.

Thus, we can assume that $v_0 \in M_F$. We now prove the two other properties.

1. Since F has no cherry, there exists a unique vertex $v_{-1} \in L_F$ which is a neighbor of v_0. Assume that the graph induced by $S_F \cup \{v_0\}$ is not connected. We prove that, under this hypothesis, $o((F, \{v_0\}, \emptyset), B) = \mathcal{D}$, which gives the result by contraposition. First, using Lemma 5.5, we have $o((F, \{v_0\}, \emptyset), B) = o((F, \{v_0\} \cup L_F \setminus \{v_{-1}\}, M_F \setminus \{v_0\}), B)$.

   With the same arguments as in the previous lemma, the implementation of a bottom-to-top strategy on each component of the graph induced by $S_F \cup \{v_0\}$ not containing v_0 leads to either a double trap (which gives the result), or a position where each of these components is dominated by Bob but not by Alice. Now, focus on the component C containing v_0. Bob then plays a bottom-to-top strategy on the tree induced by C rooted in v_0. Then Bob claims v_{-1} after all the vertices labeled by 1 have been claimed. By this way, if no trap has appeared before, Bob dominates the whole forest F, while Alice does not totally dominate F. This gives the result.

2. For the second item, consider the labeling of F (that is actually a tree) rooted in v_0. First note that all the leaves of S_F are labeled by 0. Thus, each vertex labeled by 1 has at least one child. Assume that there exists a vertex v labeled by 1 with has at least two children.
   - If two children v' and v" of v are labeled by 0, Bob can then use a bottom-to-top strategy until v is claimed. When he claims v, he creates a double A-trap in v' and v" and thus obtains a draw.
   - Otherwise, there exists one child v' of v labeled by 0 and the other one, v'' labeled by 1. Then Bob uses a bottom-to-top strategy but without claiming v", until all the vertices labeled by 1 (except v" are claimed). All replies of Alice remain forced. After this is done, Bob claims v_{-1}, the neighbor of v_0 which belongs to L_F, and then dominates the whole graph while Alice does not dominate v", which ensures that $o((T, \{v_0\}, \emptyset), B) = \mathcal{D}$. □

As a consequence of this result, if F is standard and winning for Alice, then F is necessarily a tree.

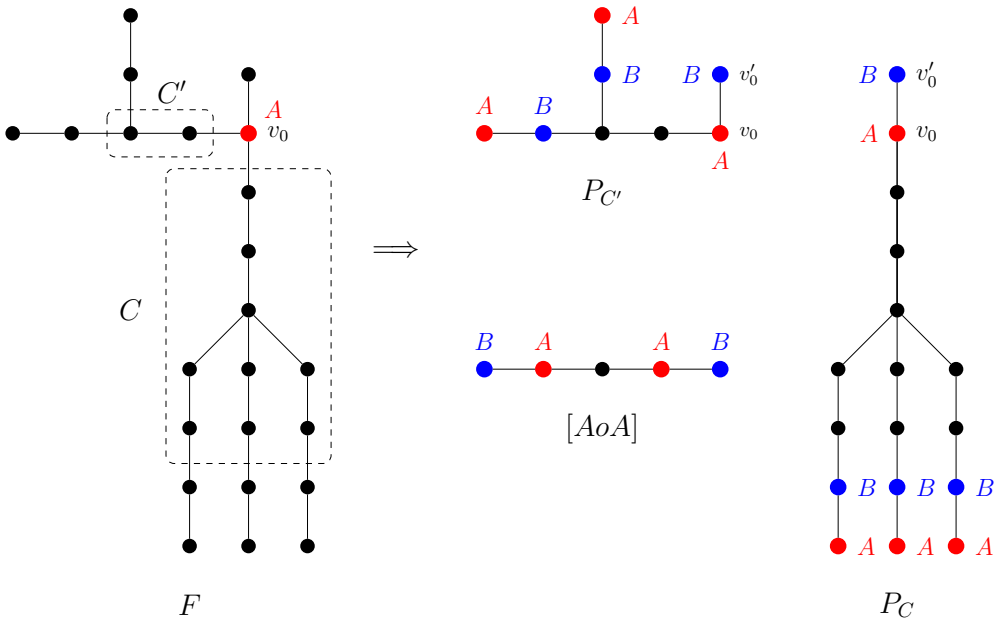


Figure 5.8: On the left a forest F. On the right, the equivalent position obtained by splitting F in Lemma 5.35. The component C is a fork whereas C' is a P_2.

5.4.6 Splitting the graph

By Lemma 5.33, it can be assumed that Alice first claims a vertex $v_0 \in M_F$ that is connected to all the components of S_F (note that if S_F is not connected, there is at most one such vertex). It can also be assumed that each vertex labeled by 1 in F rooted in v_0 has degree exactly 2. In all the remaining, v_0 will denote this first claim of Alice. Let v_{-1} be the leaf connected to v_0. After this first move, it can be assumed, using Lemma 5.5, that Bob will claim all the other vertices of M_F one by one. At each time, Alice must answer to the corresponding leaf in L_F. After this step, the free vertices are the vertices in S_F with the vertex v_{-1}. Formally, the obtained position is $P = (F, L_F \setminus \{v_{-1}\} \cup \{v_0\}, M_F \setminus \{v_0\})$ and we have $o(F) = o(P, B)$. In that follows, we will split the graph into several components defined from the connected components of the skeleton.

Definition 5.34. *For a connected component C of S_F, let T be the connected component of $F \setminus \{v_0\}$ that contains C. The position P_C is defined as the position induced by $T \cup \{v_0, v'_0\}$ in the position $P = (F, L_F \setminus \{v_{-1}\} \cup \{v_0\}, M_F \cup \{v'_0\} \setminus \{v_0\})$, where v'_0 is an additional leaf connected to v_0 and claimed by Bob.*

Figure 5.8 illustrates the two positions P_C and $P_{C'}$ derived from the forest F of Figure 5.7 when played on v_0. Lemma 5.35 shows that this splitting (with an additional $[AoA]$ position) yields to an equivalent position.

Lemma 5.35. *The position $P = (F, L_F \setminus \{v_{-1}\} \cup \{v_0\}, M_F \setminus \{v_0\})$ is equivalent to the position*

$$\left(\bigcup_{C \in CC(S_F)} P_C \right) \cup [AoA]$$

where $CC(S_F)$ denotes the set of connected components of S_F.

Proof. Let P' be the position $(\bigcup_{C \in CC(S_F)} P_C) \cup [AoA]$. Note that the unclaimed vertices are in a one-to-one correspondence in the two positions (v_{-1} is corresponding to the unclaimed vertex of $[AoA]$). By Observation 1.75, we just need to prove that P and P' have the same winning sets for both players.

A set S of unclaimed vertices is winning for Alice in P and in P' if and only if it dominates all the vertices of S_F except the ones connected to (a copy of) v_0, which corresponds to the same condition in both positions. A set S of unclaimed vertices is winning for Bob in P if and only if $v_{-1} \in S$ and S dominates all the vertices of S_F that are not connected to a vertex of $M_F \setminus \{v_0\}$. In P', S is winning for Bob if it contains the unclaimed vertex of $[AoA]$ and if it is dominating all the unclaimed vertices not already dominated by Bob, that are exactly all the vertices of S_F not connected to $M_F \setminus \{v_0\}$. Thus, the winning sets are in bijection and by Observation 1.75, the positions are equivalent. □

Using this decomposition, we now prove that Bob can just focus on a subset of components where he has a strategy that ensures a draw.

Lemma 5.36. *Assume there exists a set $\mathcal{S}$ of connected components of S_F such that $o((\bigcup_{C\in\mathcal{S}} P_C) \cup [AoA], B) = \mathcal{D}$. Then $o(F) = \mathcal{D}$.*

Proof. Using Lemma 5.35, it suffices to prove that $o((\bigcup_{C\in CC(S_F)} P_C) \cup [AoA], B) = \mathcal{D}$. For each $C \notin \mathcal{S}$, Bob plays a bottom-to-top strategy on P_C rooted on v_0. At each time Alice is forced to answer in the same component C, and, at the end, Bob dominates the component C. This is successively done for all such components.

Afterward, the remaining position is equivalent to $((\bigcup_{C\in\mathcal{S}} P_C) \cup [AoA], B)$ which has outocome $\mathcal{D}$ by hypothesis. □

5.4.7 Favorable skeletons for Alice

In this subsection, we give some necessary and sufficient conditions for a component C to be winning for Alice.

Definition 5.37. *A fork is a star with at least three branches where each is subdivided exactly once.*

On Figure 5.8, the unclaimed vertices of C is an example of a fork with four branches.

Lemma 5.38. *If $o((F, \{v_0\}, \emptyset), B) = \mathcal{A}$, then all the connected components of S_F must induce a path or a fork that is connected to v_0 by a leaf.*

Proof. Proceeding by contraposition, assume that there exists a component C of S_F, which is neither a fork nor a path connected to v_0 by a leaf . We will prove $o(P_C \cup [AoA], B) = \mathcal{D}$, which gives the result using Lemma 5.36.

Since C is not a path connected by a leaf to v_0, there exists a vertex $c \in C$ of degree at least 3 in the tree induced by $C \cup \{v_0\}$. Let $P_a = (c, a_1, a_2, ...a_p)$ be the path linking c to v_0 (a_p is adjacent to v_0), $P_b = (c, b_1, b_2, ..., b_q)$ be a path of C of maximal length such that $b_1 \neq a_1$ and $P_c = (c, c_1, c_2, ..., c_r)$ be another maximal path of S_F starting in c. Note that possibly $p = 0$, but that $q \geq r \geq 1$. In the labeling of F rooted in v_0, the vertex c is necessarily labeled by 0 since it has degree 3. This implies that both q and r are even since b_q and c_r, as leaves of C, are labeled by 0. Moreover, all the vertices labeled by 1 have degree 2. This implies that all the vertices of the three paths P_a, P_b and P_c that are connected to other vertices of F must be labeled by 0. In particular, using Lemma 5.26, it is enough to prove that there is a draw strategy when C is reduced to these three paths.

Assume first that $q \geq 4$. By Lemma 5.26, it is enough to give a draw strategy for Bob for $q = 4$ and $r = 2$ with C reduced to the union of the three paths. The first claim of Bob is b_2. By Lemma 5.12, Alice should claim either b_1 or b_3.

- If Alice replies by claiming b_3, then Bob claims c, which forces Alice to claim b_1. Then, he successively claims a_2, a_4, and so on until a_{p-1} is dominated by B. Successive replies of Alice are forced: when Bob claims a_{2i}, Alice necessarily replies in a_{2i-1}. Finally, Bob claims in $[AoA]$ and gets a draw by dominating before Alice (Alice does not dominate c_1 and c_2).

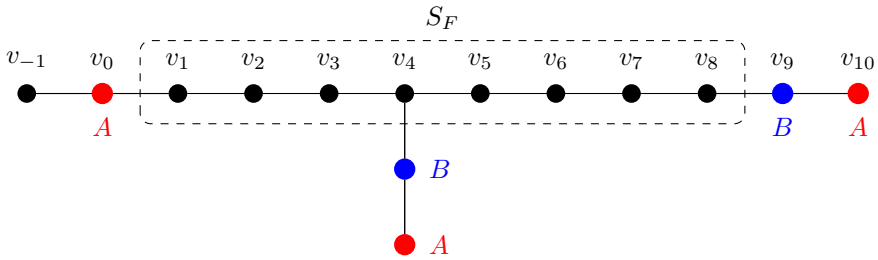


Figure 5.9: Example of a tree F such that $o(F) = \mathcal{D}$ but S_F is a path. Alice should claim first v_0 or v_9. If Alice claims v_0, Bob claims the other vertices of M_F. Then Bob can win by claiming v_2 (Alice should answer v_1, v_3 or v_4), v_6(Alice should answer v_5,v_7 or v_8) and v_{-1}. The case where Alice claims first v_9 is similar.

- If Alice replies by claiming b_1, then Bob claims c_1, which forces Alice to claim c_2. Then Bob claims the unclaimed vertex of $[AoA]$. Then the position is equivalent to the position $([Ao^{p-1}B] \cup [Bo^2B], A)$. By Lemma 5.16, this position is equivalent to $([Bo^2B], A)$ which is a draw since Bob already dominates.

Assume now that $r = q = 2$. As before, it is enough to give a strategy for C restricted to the three paths.

- If $p \geq 5$, then Bob claims a_3, which enforces Alice to reply either a_2 or a_4 by Lemma 5.12.
  - If Alice claims a_4, then Bob successively claims b_1, c_1 (with forced Alice to claim b_2 and c_2), and then a_1, which creates two A-traps in a_2 and c.
  - If Alice replies by a_2, then, first, Bob claims, $a_5, a_7, \ldots$, and so on until a_p is dominated by Bob, replies of Alice being forced on $a_4, a_6, \ldots$. Second, Bob claims c. If Alice does not answer in the set $\{a_1, b_1, b_2, c_1, c_2\}$, Bob successively claim b_1 creating an A-trap in b_2, c_1, creating an A-trap in c_2 and a_1 isolating c. Thus, Alice must claim a vertex in the set $\{a_1, b_1, b_2, c_1, c_2\}$. Then Bob claims the unclaimed vertex of $[AoA]$ and dominates the whole position before Alice.
- If $p = 4$, the position can be treated as for $p = 5$ if Alice replies in a_2 or a_4. But she can also claim a_1. In this case, Bob claims the unclaimed vertex of $[AoA]$ with the threat to claim c and dominate before Alice. Even if Alice replies in c, Bob succeeds in dominating before Alice by successively claiming b_1, c_1 and a_2 (Alice will not dominate a_3 during this time).
- If $p = 1$ or $p = 3$, then Bob claims c. If Alice claims the unclaimed vertex of $[AoA]$, a_2 or a_3, then Bob can claim b_1 and c_1, forcing Alice to reply by claiming b_2 and c_2. Then Bob can isolate c by claiming a_1. Thus, Alice should answer by claiming a vertex in $\{a_1, b_1, b_2, c_1, c_2\}$. Then Bob can claim the unclaimed vertex of $[AoA]$. If $p = 1$, he wins. If $p = 3$, he can dominate in one move by claiming either a_2 or a_3 whereas Alice need at least two moves to dominate.
- If $p = 2$, then we have a fork (since all the other branches of C starting from c must have length 2 and the vertices adjacent to C are labeled by 1, and thus have degree 2), which is not possible, by hypothesis.
- If $p = 0$, then Bob claims the unclaimed vertex of $[AoA]$. Alice needs at least two moves to dominate. If Alice does not claim c, Bob wins at his second turn by claiming it. If Alice claims c she still need two moves to dominate. Then Bob can dominate before by claiming b_1 and c_1. □

Lemma 5.38 gives us the possible structures of the connected components of S_F to have a position on which Alice can win. But actually, this condition is not sufficient: there are for example trees where S_F is a path, but Bob can obtain a draw (see for example Figure 5.9). We need to consider which vertices of S_F are already dominated by Bob.

Definition 5.39. *Let $X, Y, \in \{A, B\}$, n be a positive integer, and a subset $U \subseteq \{1, 2, \ldots, n\}$. We denote by $[Xo^nY]^U$ the position obtained from the bounded path $[Xo^nY]$ where for each $i \in U$ a pendant edge x_iy_i is added to the vertex v_i, with x_i is linked to v_i and claimed by Bob, and y_i claimed by Alice.*

Informally, an $[Xo^nY]^U$ is a bounded path, where some vertices are already dominated by Bob. As an illustration, if we set that v_{-1} is claimed by Bob on Figure 5.9, then we obtain the position $[Ao^8B]^{\{4\}}$. When U is empty, $[Xo^nY]^U = [Xo^nY]$. If $Y = B$ (respectively $X = B$), one can assume that $n \notin U$ (resp. $1 \notin U$). Indeed, v_n (resp. v_1) is already dominated by Bob.

Finally, if C is a connected component of S_F that is a path connected to v_0 by a leaf, then P_C is equivalent to a position $[Ao^nB]^U$, for a fixed n and a fixed U. Indeed, set for U all the integers i such that v_i is connected to a vertex of $M_F \setminus \{v_0\}$.

The following three observations give natural properties about bounded paths. The first one is about the existence of a pairing.

Observation 5.40. *A position $[Xo^nY]^U$ contains a A-pairing, except if $X = Y = B$ and n is odd.*

Roughly speaking, the next two observations say that it is always better for Alice to play on a bounded path with the extremities claimed by her, and with fewer vertices dominated by Bob in U.

Observation 5.41. *For any integer n and any set $U \in \{1, ..., n\}$, and any $X, \in \{A, B\}$, we have*

$$o([Ao^nA]^U, X) \geq o([Ao^nB]^U, X) \geq o([Bo^nB]^U, X).$$

Observation 5.42. *For any integer n and any sets $U \subseteq U' \subseteq \{1, ..., n\}$, and each $X, Y, Z \in \{A, B\}$, we have*

$$o([Xo^nY]^U, Z) \geq o([Xo^nY]^{U'}, Z)$$

.

In the next definition, we define the components C that are *favorable* for Alice, and among them, the ones that are *strongly* and *weakly* favorable. Theorem 5.44 will justify this terminology.

Definition 5.43. *Let v_0 be the first move of Alice, satisfying Lemma 5.33. Let C be a connected component of S_F. We say that C is* favorable for Alice *if it satisfies one of the following case:*

1. *C is a path connected to v_0 by a leaf, i.e, $P_C = [Ao^nB]^U$ with $U \subseteq \{1, 2, \ldots, n-1\}$, and at least one of the following cases holds:*
   - *(a) $n \in \{1, 2\}$ and $U = \emptyset$;*
   - *(b) $n = 3$ and $U \subseteq \{1\}$ or $U \subseteq \{2\}$;*
   - *(c) $n \geq 4$ and $U \subseteq \{2, 3, n-2\}$;*
   - *(d) $n \geq 9$, n is odd and $U \subseteq \{2, 5, n-2\}$;*
   - *(e) $n \in \{9, 11\}$ and $\{3, 5\} \subseteq U \subseteq \{2, 3, 5, n-2\}$;*
2. *C induces a fork and the only vertices of C that can be connected to M_F are the center c, the leaves except the one connected to v_0, and eventually the neighbor of c between c and v_0.*

Moreover, if C belongs to the cases (1.a), (1.b) or (1.c), we say that it is strongly *favorable. On the opposite, if C belongs to the cases (1.e) or (2), we say that it is* weakly *favorable. If C corresponds to the case (1.d), it is neither strongly nor weakly favorable.*

We can now state the final theorem that ends the characterization of trees.

Theorem 5.44. *Let F be a tree and $v_0 \in M_F$ be a first move of Alice that satisfies the condition of Lemma 5.33. The position $((F, \{v_0\}, \emptyset), B)$ has outcome $\mathcal{A}$ if and only if all the components of S_F are favorable to Alice and at most one of them is weakly favorable.*

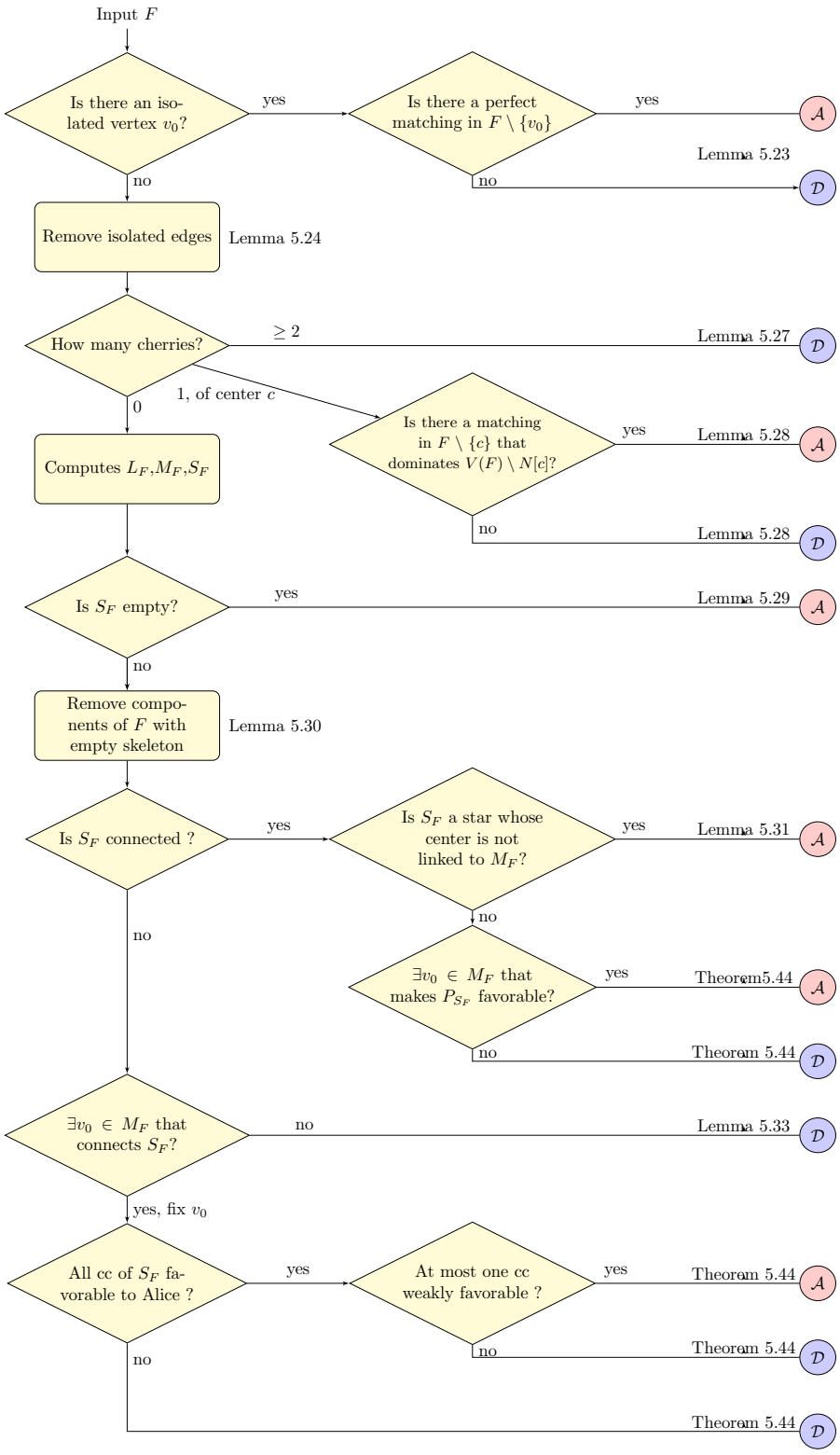


Figure 5.10: The decision tree to compute the outcome of any forest.

The proof of Theorem 5.44 is a long case analysis, and is postponed to Section 5.5. For now, we will prove Theorem 5.22 assuming Theorem 5.44. We summarize the algorithm into the diagram of Figure 5.10.

Proof of Theorem 5.22. Let F be a forest. If F has an isolated vertex v_0, then by Lemma 5.23, $o(F) = \mathcal{A}$ if and only if $F \setminus \{v_0\}$ has a perfect matching. If F has an isolated edge $e = uv$, then by Lemma 5.24, F has the same outcome as $F \setminus \{u, v\}$. Thus, we can assume that all the connected components of F have at least three vertices. If F has two cherries, then $o(F) = \mathcal{D}$ by Lemma 5.27. If it has one cherry (c, l, l'), then by Lemma 5.28, $o(F) = \mathcal{A}$ if and only if there is a matching in $F \setminus \{c\}$ that covers $V(F) \setminus N[c]$. Thus, we can assume that F has no cherry. If $S_F = \emptyset$, then $o(F) = \mathcal{A}$ (Lemma 5.29). Otherwise, one can remove components of F that have an empty skeleton (Lemma 5.30). If S_F induces a star of center c such that c is not adjacent to M_F, then $o(F) = \mathcal{A}$ (Lemma 5.31). Otherwise, F is standard. If F is not connected, then $o(F) = \mathcal{D}$. If S_F is not connected, Alice should claim the vertex v_0 of M_F that connects S_F (Lemma 5.33) if it exists (otherwise $o(F) = \mathcal{D}$). Then Alice wins if and only if all the components of S_F are favorable to her and at most one is weakly favorable (Theorem 5.44). Assume now that S_F is connected. Let C be the tree induced by S_F. Alice should claim in M_F and connected to a leaf of C (Lemma 5.33). If there exists $v_0 \in M_F$ that is adjacent to a leaf of C such that P_C is favorable to Alice, then $o(F) = \mathcal{A}$, otherwise $o(F) = \mathcal{D}$ (Theorem 5.44).

In terms of complexity, almost all the operations described in the algorithm are elementary (finding a matching in a forest, identifying the isolated vertices, edges and cherries, computing L_F, M_F and S_F, deciding if S_F is connected or a star) and can be done in linear time by examining the tree from the leaves. When S_F is not connected, there exists at most one v_0 that can connect them and this is easy to check. If S_F is connected and thus reduce to a single component, one have to check all the possible v_0 that could make P_{S_F} favorable. If S_F induces a path, there are at most two possibilities for v_0 since it should be connected to an extremity of the path. If S_F induces a fork (easy to check) of center c, to be favorable, there must be at most one neighbor of c that is adjacent to M_F. If there is exactly one neighbor a_1 of c adjacent to M_F, then the leaf of S_F adjacent to a_1 must have only one neighbor in M_F and this neighbor will be the vertex v_0. If all the neighbors at distance 1 of c in S_F are not adjacent to vertices of M_F, then there must be at least one leaf of S_F adjacent to exactly one vertex of M_F. This vertex of M_F would be v_0. In all the other cases, P_{S_F} cannot be favorable. Thus, finding v_0 can be done in linear time. Finally, once v_0 is fixed, deciding whether a component is favorable or not can also be done in linear time (because it is a fork or a path with pending edges).

□

5.5 Proof of the main theorem

We here prove Theorem 5.44 by considering successively both directions of the equivalence. Subsection 5.5.1 proves that the favorable components are necessary conditions. The sufficient condition is then proved in Subsection 5.5.2.

5.5.1 The direct part

In this subsection, we assume that $o((F, \{v_0\}, \emptyset), B) = \mathcal{A}$ and we prove that all the components are favorable and at most one is weakly favorable. We already know by Lemma 5.38 that all the connected components of S_F are paths or forks. In Lemma 5.45, we prove that if a component is a fork, it should satisfy the conditions of Point 2 in Definition 5.43. In Lemma 5.46, we prove that if a component is a path, it should satisfy the conditions of Point 1 in Definition 5.43. Finally, in Lemma 5.50, we prove that if two components are weakly favorable, then the outcome is $\mathcal{D}$.

Lemma 5.45. *Assume that $o((F, \{v_0\}, \emptyset), B)) = \mathcal{A}$. Let C be a connected component of S_F which induces a fork. Then C is favorable.*

Proof. Proceeding by contraposition, we assume that C is not favorable. We will prove that $o([AoA] \cup P_C, B) = \mathcal{D}$ which gives the result using Lemma 5.36.

Let c denotes the center of the fork. Let a_1 and a_2 be the two vertices on the path between c and v_0 with a_2 adjacent to v_0, and v_{-1} be the unclaimed vertex of $[AoA]$. Since C is not favorable, it means that either a_2 or a neighbor of c but not a_1 is dominated by Bob in P_C. We consider the two cases.

- If a_2 is already dominated by Bob, then Bob claims c.
  - If Alice replies in v_{-1} then Bob can claim all the neighbors of c distinct from a_1 and Alice is forced to answer the leaf of C adjacent to the vertex claimed by Bob. Afterward, Bob claims a_1, isolating the vertex c, which ensures that Alice will not dominate the graph.
  - otherwise, Bob claims v_{-1} and, therefore, dominates the graph before Alice.
- If Bob already dominates a neighbor b_1 of c (with $b_1 \neq a_1$), then Bob claims all the other neighbors of c distinct from a_1 and b_1, forcing Alice to claim the leaves adjacent to them. Then Bob claims the unclaimed vertex of $[AoA]$. At this point, he needs only one move to dominate the graph that can be either a_1 or a_2, whereas Alice still need to dominate a_1 and b_2, which cannot be dominated in a single move. Thus, Bob will dominate before Alice. □

Lemma 5.46. *Assume that $o((F, \{v_0\}, \emptyset, B)) = \mathcal{A}$, and let C be a connected component of S_F which induces a path. Then C is favorable.*

Proof. Let $P_C = [Ao^nB]^U$. We proceed by contraposition. Assume that C is not favorable. We will prove that $o([AoA] \cup P_C, B) = \mathcal{D}$, which gives the result by using Lemma 5.36. We split the proof into a few claims. We denote by v_{-1} the unclaimed vertex of $[AoA]$ and by $v_1, ..., v_n$ the unclaimed vertices of P_C. We consider for the labeling that P_C is rooted in v_0.

Claim 5.47. *If n is even and $5 \in U$, then $o([AoA] \cup P_C, B) = \mathcal{D}$.*

Proof of the claim. When $n = 6$, Bob claims v_3, which enforces Alice to reply in v_4 (if Alice replies in v_2, then Bob claims v_5 and create a double trap in v_4 and v_6). Then Bob successively claims v_1, forcing Alice to claim v_2, and v_{-1}, therefore dominates before Alice.

If $n \geq 8$, v_7 is labeled by 1. By Lemma 5.26, the position can be reduced to the case $n = 6$, and thus is also a draw. ◇

Claim 5.48. *If $n \geq 4$ and if there exists $i \in U$ such that $i \notin \{2, 3, 5, n-2\}$, then $o([AoA] \cup P_C, B) = \mathcal{D}$.*

Proof of the claim. Assume first that $i = 1$.

- If $n \in \{4, 5\}$, then Bob first claims v_3, (which forces Alice to answer v_4 or v_2). Then Bob claims v_{-1} and dominates before Alice (who does not dominate v_2 or v_4).
- If $n \geq 6$, either v_5 or v_6 is labelled by 1. Then as before, we can apply Lemma 5.26 to reduce to the case $n \in \{4, 5\}$.

We can now assume that $i \geq 4$. There are two cases, according to the parity of the value $n - i$. In each case, Lemma 5.26 is used.

- If $n - i$ is odd,
  - If $n - i = 1$, that is $i = n - 1$, then Bob claims v_{n-3} (this is possible since $4 \leq i < n$),
    - For $n = 5$, Alice has to reply a vertex in $\{v_3, v_4, v_5\}$, otherwise Bob plays v_4 and creates a double trap in v_3 and v_5. But she cannot dominate in a single move, and then Bob claims v_{-1} and directly dominates before Alice.

* For $n \geq 6$, Alice necessarily replies in v_{n-2}. Indeed, by Lemma 5.12, Alice should answer in v_{n-4} or v_{n-2}, but if she claims v_{n-4} then a component $[Bo^{2k+1}B]$ will appear which is a draw by Lemma 5.18. After Alice has claimed v_{n-2}, the resulting position P is then equivalent to a position $[AoA] \cup [Ao^{n-4}B]^{U'} \cup [Ao^2B]^{\{1\}}$, for some set U'. We thus have $o(P, B) \leq o([AoA] \cup ([Ao^{n-4}B] \cup [Ao^2B]^{\{1\}}), B)$. But, by Lemma 5.16, the component $[Ao^{n-4}B]$ can be removed, thus, $o(P, B) \leq o([AoA] \cup [Ao^2B]^{\{1\}}$, which has outcome $\mathcal{D}$ as Bob can dominate in one move while Alice does not dominate.

– If $n - i \geq 3$, then v_{i+2} is labeled by 1 and Lemma 5.26 applies to reduce the instance to the case where $i = n - 1$.

- Assume now that $n - i$ is even. Since $i \notin \{n - 2, n\}$, we have $n - i \geq 4$. If $n - i \geq 6$, then v_{i+5} is labeled by 1, and Lemma 5.26 applies to reduce the instance to the case where $n - i = 4$. Thus, we can assume that $i = n - 4$. Furthermore, since $i \notin \{1, 2, 3, 5\}$, we can assume that $n \geq 8$ and $n \neq 9$.

  – if $n = 8$, then Bob successively claims v_6, v_2, v_{-1} and dominates before Alice. Indeed, she has only claimed two moves to dominate vertices between v_2 and v_7, and the only dominating set of size two on this path contains v_6.

  – if $n = 10$, then Bob successively claims v_8, v_4. As, by Lemma 5.12, Alice has to claim a vertex adjacent to the vertex that Bob has claimed, her moves are almost forced. If she has claimed v_9 and v_3, Bob claims v_6 and creates two traps in v_5 and v_7. Otherwise, he claims v_{-1} and dominates before Alice by a final claim in $\{v_1, v_2\}$ while Alice needs at least two moves to dominate.

  – if $n = 11$, Bob claims v_9. By Lemma 5.12, Alice should answer in $\{v_8, v_{10}\}$. If Alice claims v_8, Bob claims v_5. Alice has to claim either v_4 or v_6. Then, Bob claims v_{-1}. Alice now needs to claim at least two more vertices to dominate: one to dominate $\{v_4, v_6\}$ and one to dominate v_{10}, while Bob will dominate with his next claim in $\{v_1, v_2\}$.

    If Alice claims v_{10}, Bob also claims v_5. Alice has to claim v_6, otherwise v_7 creates a double trap. Now Bob claims v_{-1} and will dominate with his next claim in $\{v_1, v_2\}$, while Alice needs two vertices to dominate.

  – If $n \geq 12$, Bob claims v_{n-9}, which, as before, enforces Alice to reply in v_{n-8} to avoid a component $[Bo^{2k+1}B]$. The position is now equivalent to $([AoA] \cup [Ao^8B]^4 \cup [Ao^{n-10}B], B)$, which is, by Lemma 5.16, equivalent to $([AoA] \cup [Ao^8B]^4, B)$ which corresponds to the case $n = 8$. ◇

Claim 5.49. *Assume that $n \geq 13$, n is odd and $\{3, 5\} \subseteq U$, then $o([AoA] \cup P_C, B) = \mathcal{D}$.*

Proof of the claim. It suffices to prove it for $n = 13$, since, for $n > 13$, Lemma 5.26 applies. Bob claims v_{11}, which enforces Alice to reply in v_{10} (if Alice replies in v_{12}, then Claim 5.47 applies). Then, Bob claims v_8.

- If Alice replies in v_7, then Bob successively claims v_5 and v_{-1}. Afterward, Bob finally succeeds in dominating before Alice by claiming either v_1 or v_2.

- If Alice replies in v_6, then Bob claims v_2, which enforces Alice to claim v_1,v_3, or v_4. Then Bob claims v_{-1}. At this step, Bob threatens to claims v_5 and dominate before Alice. Thus, the reply of Alice is necessarily v_5. Then, Bob claims v_7, which enforces the reply v_9 from Alice. Finally, Bob claims either v_3 or v_4 (one of these vertices is free) and dominates before Alice. ◇

We can now finish the proof of the lemma. Since C is not favorable, we can split the cases according to n as follows:

- If $n = 1$, then the result is obvious, since there is no unfavorable component.

- If $n = 2$ and $1 \in U$, then Bob claims v_{-1} and then dominates the graph before Alice, which is a contradiction.

- If $n = 3$ and $1, 2 \in U$, then Bob claims v_{-1} and then dominates the graph before Alice, which is a contradiction
- If $n \in \{4, 5, 6, 7\}$ then Claim 5.47 and Claim 5.48 give the result.
- If $n \geq 8$ and n is even, the combination of Claims 5.47 and 5.48 gives that $U \subseteq \{2, 3, n-2\}$.
- If n is odd and $n \geq 13$, then the combination of Claims 5.48 and 5.49 gives that either $U \subseteq \{2, 3, n-2\}$ or $U \subseteq \{2, 5, n-2\}$, which gives the result .
- If $n \in \{9, 11\}$, then Claim 5.48 allows to conclude. □

Lemma 5.50. *Assume there are two connected components C, C' of S_F that are weakly favorable. Then $o(F, \{v_0\}, \emptyset, B)) = \mathcal{D}$.*

Proof. Assume first that both C and C' both induce forks. We prove the result for forks with exactly three branches. Indeed, if Bob has a strategy in this case, he will have a strategy for forks with more branches using Lemma 5.26 since all the neighbors of c are labeled by 1. Let c denote the center of the fork induced by C, a_1, b_1, c_1 denote the neighbors of c, in such a way that such that a_1 is between c and v_0, and a_2, b_2, c_2 respectively denote the neighbors of a_1, b_1, c_1 different from c . We define in the same way $c', a'_1, b'_1, c'_1, a'_2, b'_2$ and c'_2 for C'. Bob start by claiming v_{-1}. By symmetry, we can suppose that Alice replies one vertex among $\{c, a_1, a_2, b_1, b_2\}$.

- If Alice replies in c , then Bob successively claims b_1 and c_1 (the replies of Alice are forced). Now Bob claims c', then Bob needs two more moves to dominate (one in (a_1, a_2) and one in (a'_1, a'_2)) whereas Alice needs three moves.
- If Alice replies in a_1, then Bob claims a_2.
  - If Alice replies in a'_1, then Bob claims a'_2. At this step, Alice needs at least four moves to dominate while Bob can dominate in three moves in a lot of manners, with a center of a fork, and two vertices of the other fork. Alice cannot avoid Bob to dominate using three moves, and therefore, before Alice.
  - If Alice replies in c, then Bob successively claims b_1 and c_1. After the forced replies of Alice, Bob claims in c' and dominates before Alice with one last move in (a'_1, a'_2) while Alice cannot dominate in one move. The case where Alice replies in c' can be treated symmetrically.
  - If Alice replies in b_1, then Bob claims successively c_1 and c' and dominates before Alice by claiming one vertex in (c, b_2).
  - If Alice replies in b'_1, then Bob claims c'. At this step, Alice needs at least four moves to dominate while Bob can dominate in three moves by a pairing strategy with the pairs $(a'_1, a'_2), (b_1, b_2)$ and (c_1, c_2).
- If Alice replies in a_2, then Bob claims a_1, and afterward, all is similar to the previous case.
- If Alice replies in b_1 or b_2, then Bob claims c'. Bob can now dominate in three moves by pairing $(a'_1, a'_2), (a_1, a_2)$ and (c, c_1) while Alice needs at least four moves.

Assume now that both C and C' induce paths. Since they satisfy the conditions 1.e of Definition 5.43, their length is 9 or 11. Bob first reduce the paths to length 9 if needed so that both paths have length 9. Let $(v_1, v_2, ..., v_9)$ (respectively $(v'_1, v'_2, ..., v'_9)$) the path induced by C (resp. C'), with v_1 and v'_1 connected to v_0. Since the paths are weakly favorable, Bob already dominates v_3, v_5, v'_3 and v'_5.

First, Bob successively claims v_7 and v'_7. By Lemma 5.12, Alice should answer first v_6 or v_8 and then v'_6 or v'_8. We have three cases according to the replies of Alice.

- If the replies are v_8 and v'_8, then Bob continues by claiming v_3 and v'_3 which enforces Alice to reply in v_4 and v'_4 (if, for instance, Alice does not reply v_4 , then Bob claims v_5 and creates a double trap in v_4 and v_6). Afterward, Bob claims v_{-1}, and achieves to dominate before Alice as he only needs one move in (v_1, v_2) and one in (v'_1, v'_2) while Alice needs to dominate v_2, v'_2, v_6 and v'_6, each of them requiring a different move.

- If the replies are v_6 and v'_8 (note that v_8 and v'_6 is symmetric), then Bob continues by claiming v'_3 which enforces Alice to reply in v'_4, Afterward, Bob claims v_{-1}. Now Bob has a paired dominating set of size 3 $\{(v_1, v_2), (v'_1, v'_2), (v_3, v_4)\}$, but Alice needs to dominate v_2, v_8, v'_2 and v'_6, each of them requiring a different move. Thus, Bob can dominate first.

- If the replies are v_6 and v'_6, Bob claims v_{-1}. At this time Bob and Alice need four moves to dominate, but each set of four moves for Alice contains the pair $\{v_3, v'_3\}$. Thus, after the reply of Alice, Bob can claim one element of the pair $\{v_3, v'_3\}$ and, by this way, dominate before Alice.

Assume now that C induces a path and C' induces a fork. As before, C' can be assumed to have exactly three branches and C nine vertices. We denote the vertices of the fork and the path as in the previous cases.

First, Bob claim v_7. Once again, Alice has to claim either v_6 or v_8, otherwise Bob ensure a draw by Lemma 5.12 There are two cases according to the reply of Alice.

- If Alice replies v_8, then Bob successively claims b_1, c_1 and a_1 (replies of Alice being forced in b_2, c_2 and c). Afterward, Bob claims v_3. Alice has to claim either v_2 or v_5 by Lemma 5.12. If she claims v_2, Bob claims v_5 and creates two traps. If she claims v_4, Bob claims v_{-1}. Now Bob only needs one move in either v_1 or v_2 to dominate while Alice needs at least two, as she does not dominate v_2 nor v_6.

- If Alice replies v_6, then Bob claims v_{-1}

  - If Alice replies c, then Bob successively claims b_1, c_1 (replies of Alice are again forced in b_2 and c_2) and then v_3. At this time, Bob needs two moves to dominate (one in v_1, v_2 and one in (a_1, a_2)), while Alice needs to dominate v_2, v_4 and v_8, and each of them requires a different move. Therefore, Bob dominates before Alice by this way.
  - If Alice replies v_3, then Bob claims c. Now $(a_1, a_2), (v_1, v_2)$ and (v_4, v_5) is a paired dominating set for Bob of size three, while Alice needs to dominated a_2, b_1, c_1 and v_8, each of these vertices requiring a different move.
  - If Alice replies v_1 or v_2, by Lemma 1.78, as $N[v_1] \setminus N[V_t] \subset N[v_2] \setminus N[V_t]$ for $t \in \{A, B\}$, we can suppose, she plays v_2. Then Bob successively claims b_1, c_1, a_1 (replies are forced in b_2, c_2, c respectively) and then v_1. At this time Bob needs one move to dominate in (v_3, v_4), while Alice needs to dominate v_4 and v_8 which she cannot dominate in a single move.
  - If Alice replies a_1 or a_2, by Lemma 1.78, as $N[a_2] \setminus N[V_t] \subset N[a_1] \setminus N[V_t]$ for $t \in \{A, B\}$, we can suppose she claims a_1. Bob claims a_2.
    - If Alice replies c, then Bob successively claims b_1, c_1, (replies are forced) and v_3. At this time Bob can dominate in two moves, one in (v_1, v_2) and one in (v_4, v_5) while Alice cannot, as she still has to dominate v_2, v_4 and v_8.
    - If Alice replies v_1 or v_2, by Lemma 1.78, as $N[v_1] \setminus N[V_t] \subset N[v_2] \setminus N[V_t]$ for $t \in \{A, B\}$, we can suppose she replies v_2. Then Bob successively claims b_1, c_1, (replies are forced for Alice) and v_1. At this time Bob needs one move in (v_3, v_4) to dominate, while Alice needs two moves to dominate v_4 and v_8.
    - If Alice replies elsewhere, then Bob claims c. At this time Bob needs two moves to dominate,one in (v_1, v_2) and one in (v_3, v_4, v_5). By hypothesis, at least one will be available. Alice needs at least three moves to dominate, as she does not dominate at least three of the four vertices v_3, v_8, b_1 and c_1, each of them requiring a different move.

– If Alice replies elsewhere, then Bob claims c. At this time Bob needs three moves to dominate, one in (v_1, v_2), one in (a_1, a_2) and one in (v_3, v_4, v_5). By hypothesis, at least one will be available in each of these sets at any moment. Alice needs at least four moves to dominate, as she does not dominate at least four of the five vertices a_1, b_1, c_1, v_3 and v_8, each of them requiring a different move. □

5.5.2 The converse part

In this final subsection, we prove the reverse part of Theorem 5.44: if all the connected components of S_F are favorable and at most one is weakly favorable, then Alice has a winning strategy. This part is naturally harder than the other one, as Alice cannot force Bob to answer where she would like to. Hence, all the possible answers of Bob must be considered, which was not the case previously, as Alice was often forced to answer locally to a move of Bob.

We first prove that we can remove the strongly favorable components (i.e. corresponding to the cases 1.a to 1.c of Definition 5.43). Then we consider only one weakly favorable component and give a strategy for Alice in this case. Finally, we define a class of positions $\mathcal{C}_1$ that contains the starting positions (without strongly favorable components) and for which Alice can ensure either to win or to stay in this class after a move of Bob. By induction, this will imply that Alice has a winning strategy in this class. Note that one can always consider U to be maximal in Definition 5.43. Indeed, if Alice have a strategy for U maximal, she will have a strategy with any subset of U. Therefore, and to merge some cases in the proofs, sometimes some integers will be in U even if the corresponding vertices do not exist in the graph. In this case, consider that this integer is not really in U.

Removing strongly favorable components

Lemma 5.51. *Let Q be a position where Bob is not dominating. Consider a strongly favorable component $[Ao^nB]^U$. We have*

$$o(Q, B) \leq o(Q \cup [Ao^nB]^U, B).$$

In other words, adding a strongly favorable component can only be favorable to Alice.

Proof. First note that since $[Ao^nB]^U$ is strongly favorable, we have $U \subseteq \{2, 3, n-2\}$.

We prove by induction on the number p of unclaimed vertices of $Q \cup [Ao^nB]^U$ that if $o(Q, B) = \mathcal{A}$, then $o(Q \cup [Ao^nB]^U, B) = \mathcal{A}$. Note that $p \geq n$ Let $v_1, v_2, ...v_n$ denotes the sequence of unclaimed vertices of $[Ao^nB]^U$. First we can assume that $n > 1$. Indeed, if $n = 1$, then $U = \emptyset$ and by Observation 5.6, $o(Q \cup [AoB]^U, B) = o(Q, B)$.

Thus, we can assume that $p \geq 2$ and $n \geq 2$. If $p = n = 2$, Q contains no unclaimed vertices. Since $o(Q, B) = \mathcal{A}$, Alice dominates Q while Bob does not. Thus, in $Q \cup [Ao^2B]^U$ Bob claims v_1 or v_2, and Alice answers by claiming the other one. By this way, Alice dominates $Q \cup [Ao^nB]^U$ while Bob does not. Thus, the resulting position is winning for Alice.

Consider now that $p \geq 3$. Let y be the vertex claimed by Bob. Assume first that y is an unclaimed vertex of Q. Note that Bob cannot dominate Q in one move unless Alice already does, otherwise we would have $o(Q, B) = \mathcal{D}$.

- If Q is not dominated by Alice yet, then Alice claims x according to a winning strategy in Q. Thus, by definition, $o(Q_{x,y}, B) = \mathcal{A}$ and by induction hypothesis $o(Q_{x,y} \cup [Ao^nB]^U, B) = \mathcal{A}$.
- if G is already dominated by Alice:
  - if $2 \leq n \leq 4$, then Alice claims v_3 and dominates the whole graph before Bob;

– if $n = 5$, the Alice claims v_3. We have:

$$o((Q \cup [Ao^n B]^U)_{v_3,y}, B) \geq o([Ao^n B]^U)_{v_3,y}, B) \geq o([Ao^2 A]^{\{2\}} \cup [Ao^2 B]), B) = \mathcal{A},$$

where the first inequality comes from the fact that Q is dominated by Alice, the second inequality comes from Lemma 5.8 and from the fact that $U \subseteq \{2, 3\}$, which is strongly favorable according to Definition 5.43.1.a, and the final equality is obtained by induction hypothesis, using $Q' = [Ao^2 A]^{\{2\}}$.

– if $n = 6$, then Alice claims v_3. We have:

$$o((Q \cup [Ao^n B]^U)_{v_3,y}, B) \geq o([Ao^n B]^U)_{v_3,y}, B) \geq o([Ao^2 A]^{\{2\}} \cup [Ao^3 B])^{\{1\}}, B) = \mathcal{A},$$

where the first inequality comes from the fact that Q is dominated by Alice, the second inequality comes from Lemma 5.8 and from the fact that $U \subseteq \{2, 3, n-2\}$, which is strongly favorable according to Definition 5.43.1.b, and the final equality is obtained by induction hypothesis, using $Q' = [Ao^2 A]^{\{2\}}$.

– if $n \geq 7$, then Alice claims v_3. We have:

$$o((Q \cup [Ao^n B]^U)_{v_3,y}, B) \geq o([Ao^n B]^U_{v_3,y}, B) \geq o([Ao^2 A]^{\{2\}} \cup [Ao^{n-3} B]^{\{n-5\}}, B) = \mathcal{A},$$

where the first inequality comes from the fact that Q is dominated by Alice, the second inequality comes from Lemma 5.8 and from the fact that $U \subseteq \{2, 3, n-2\}$, which is strongly favorable according to Definition 5.43.1.c, and the final equality is obtained by induction hypothesis, using $Q' = [Ao^2 A]^{\{2\}}$.

Assume now that Bob claimed a vertex v_i of $[Ao^n B]^U$.

- If $2 \leq i \leq n-1$, then Alice claims v_{i+1}. We have

$$o((Q \cup [Ao^n B]^U)_{v_i,v_{i+1}}, B) = o(Q \cup [Ao^{i-1} B]^{\{2,3\}} \cup [Ao^{n-i-1} B]^{\{n-i-3\}}, B) = \mathcal{A},$$

where the first equality comes from Lemma 5.8 and the inequality comes from two applications of the induction hypothesis.

- If $i = 1$, and $n \geq 3$ then Alice claims v_3 . We have

$$o((Q \cup [Ao^n B]^U)_{v_3,v_1}, B) \geq o(Q \cup [Ao^{n-3} B]^{\{n-5\}}, B) = \mathcal{A},$$

by induction hypothesis.

If $n = 2$ then Alice claims v_2 . We have $o((Q \cup [Ao^2 B]^U)_{v_2,v_1}, B) = o(Q, B) = \mathcal{A}$, from Observation 5.6

- If $i = n$, then Alice claims v_{n-1}. We have

$$o((Q \cup [Ao^n B]^U)_{v_{n-1},v_n}, B) \geq o(Q \cup [Ao^{n-2} A]^{\{2,3\}}, B) \geq o(Q \cup [Ao^{n-2} B]^{\{2,3\}}, B) = \mathcal{A},$$

by induction hypothesis and Observation 5.41.

Thus, for each claim y of Bob on $Q \cup [Ao^n B]^U$, there exists an answer x of Alice such that $o((Q \cup [Ao^n B]^U)_{x,y}, B) = \mathcal{A}$. This ensures that $o(Q \cup [Ao^n B]^U, B) = \mathcal{A}$. □

Dealing with the weakly favorable component

Lemma 5.52. *Let C be a connected component of S_F that is weakly favorable, then $o(P_C \cup [AoA], B) = \mathcal{A}$. Moreover, Alice can play to ensure that after each of her move, there is an A-pairing.*

Proof. Recall that by Observation 5.40, there is A-pairing in any bounded path $[Xo^nY]$ except if $X = Y = B$ and n is odd.

Assume first that C is a fork. Let c be the center of C, a_1 the neighbor of c on the path to v_0 and a_2 the vertex between a_1 and v_0. Let b_1 be another neighbor of c and b_2 the other neighbor of b_1. We prove the result by induction on the number of branches in the fork. Note that by Lemma 1.78, as $N[a_2]\backslash N[V_t] \subset N[a_1]\backslash N[V_t]$ for $t \in \{A, B\}$, a_1 is always a better move than a_2, thus we can suppose that it will not be played as next move.

- if Bob claims c, then Alice replies in a_1. Since there is a double B-trap and an A-pairing, disjoint from the traps, Alice wins by Lemma 5.14. Moreover, by following the pairing strategy, she will keep the fact there is always an A-pairing.
- If Bob claims b_1, then Alice replies in b_2. If there are strictly more than three branches, the position is still a fork where the center is dominated by Bob (which is still weakly favorable). The result is true by induction. If there were exactly three branches, then the actual position is $((P_C \cup [AoA])_{b_2,b_1}, B) = ([Ao^5B]^{\{2,3\}} \cup [AoA], B)$. By Lemma 5.51, $o((P_C \cup [AoA])_{b_2,b_1}, B) \geq o([AoA], B) = \mathcal{A}$. Furthermore, there is A-pairing by Observation 5.40.
- If Bob claims b_2, then Alice replies in b_1. If there are strictly more than three branches, then Alice follows the strategy with one less branch. Since she dominates one more vertex, it can only be better for her. If there are exactly three branches, we have as before:

$$o((P_C \cup [AoA])_{b_1b_2}, B) \geq o([Ao^5B]^{\{2,3\}} \cup [AoA], B) \geq o([AoA], B) = \mathcal{A}.$$

- If Bob claims a_1 , then Alice replies by claiming c. We have

$$o((P_C \cup [AoA])_{c,a_1}, B) = o([Ao^2B] \cup ... \cup [Ao^2B] \cup [AoA], B) \geq o([AoA], B) = \mathcal{A}.$$

  There is an A-pairing by pairing together the two vertices belonging to the same paths.
- If Bob claims v_{-1}, then Alice replies by claiming c. We have

$$o((P_C \cup [AoA])_{c,v_{-1}}, B) \geq o([Ao^2A]^{\{2\}} \cup [Ao^2B] \cup ... \cup [Ao^2B], B) \geq o([Ao^2A]^{\{2\}}, B) = \mathcal{A}.$$

  Again, there is an A-pairing by pairing together the two vertices belonging to the same paths.

Assume now that P_C is a bounded path. Alice will never let components of the form $[Bo^{2\ell+1}B]$ to Bob and thus there will always be an A-pairing. Assume first that P_C has length 9. We just need to prove that Alice has a strategy in the worst case, that is for $U = \{2, 3, 5, 7\}$. Thus, let $P_C = [Ao^9B]^{\{2,3,5,7\}}$. Note that by Lemma 1.78, as $N[v_1] \setminus N[V_t] \subset N[v_2] \setminus N[V_t]$ for $t \in \{A, B\}$, v_2 is always a better move than v_1, thus we can suppose that it will not be played as next move.

- If Bob claims v_1 or v_2, then Alice replies in v_3. We have

$$o(([Ao^9B]^{\{2,3,5,7\}})_{v_3,v_1} \cup [AoA]), B) \geq o([Ao^6B]^{\{2,4\}} \cup [AoA], B) \geq o([AoA], B) = \mathcal{A},$$

  where the last inequality comes from Lemma 5.51.
- If Bob claims v_3, then Alice replies in v_2. Then $o((P_C \cup [AoA])_{w_2,w_3}, B) = \mathcal{A}$ since there are two B-traps and an A-pairing.
- If Bob claims v_i, $4 \leq i \leq 8$, then Alice replies in v_{i+1}. The resulting position has outcome

$$o([Ao^{i-1}B]^{\{2,3,5,7\}} \cup [Ao^{9-i-1}B]^{7-i-1} \cup [AoA], B) \geq o([AoA], B) = \mathcal{A},$$

  where the last inequality comes from two applications of Lemma 5.51 (since both created paths satisfy hypotheses of Lemma 5.51).

- If Bob claims v_9 , then Alice replies in v_8. The resulting position has outcome

$$o([Ao^7A]^{\{2,3,5,7\}} \cup [AoA], B) \geq o([Ao^7B]^{\{2,3,5\}} \cup [AoA], B) \geq o([AoA], B) = \mathcal{A},$$

  where the last inequality comes from two applications of Lemma 5.51.

- If Bob claims v_{-1}, then Alice replies in v_3. We have

$$o(([Ao^9B]^{\{2,3,5,7\}} \cup [AoA])_{v_3,v_{-1}}, B) \geq o([Ao^6B]^{\{2,4\}} \cup [Ao^2A], B) \geq o([Ao^2A], B) = \mathcal{A},$$

  where the last inequality comes from Lemma 5.51.

Assume now that P_C has length 11. As before, we can assume that $P_C = [Ao^{11}B]^{\{2,3,5,9\}}$.

- If Bob claims v_i, $1 \leq i \leq 8$, then it can be done like for a path of length 9.

- If Bob claims v_9, then Alice replies in v_8,

  - If now Bob claims v_3, then Alice replies in v_2, creating two B-traps, in v_{-1} and v_1. Thus $o(Q_{v_2,v_3}, B) = \mathcal{A}$, by Lemma 5.14,
  - If now Bob claims v_5, then Alice replies in v_6, creating two B-traps, in v_{-1} and v_8. Thus $o(Q_{v_6,v_5}, B) = \mathcal{A}$, by Lemma 5.14
  - If now Bob claims v_{10} (or v_{11}) , then Alice replies in v_{11} (or v_{10}). The resulting position is equivalent to $[Ao^7A]^{\{2,3,5\}} \cup [AoA]$, and, from Lemma 5.51,

$$o([Ao^7A]^{\{2,3,5\}} \cup [AoA], B) \geq o([AoA], B) = \mathcal{A}.$$

    .
  - If now Bob claims v_i , with $i \in \{-1, 1, 2, 4\}$, then Alice replies in v_3. Afterward Alice can dominate with two more claims, one in $\{v_5, v_6\}$, one in $\{v_{10}, v_{11}\}$, whatever the strategy of Bob; therefore Alice dominates before Bob
  - If now Bob claims v_i , with $i \in \{6, 7\}$, then Alice replies in v_5. Afterward Alice can dominate with two more claims, one in $\{v_2, v_3\}$, one in $\{v_{10}, v_{11}\}$, whatever the strategy of Bob; therefore Alice dominates before Bob.

- if Bob claims in v_{10}, then Alice replies in v_{11}, which reduces the problem to the previous case with 9 vertices.

- If Bob claims v_{11}, then Alice replies in v_{10}. The resulting position is equivalent to to $[Ao^9A]^{\{2,3,5,9\}} \cup [AoA]$, and by Observation 5.41, we have

$$o([Ao^9A]^{\{2,3,5,9\}} \cup [AoA], B) \geq o([Ao^9B]^{\{2,3,5\}} \cup [AoA], B) = \mathcal{A},$$

  as seen before.

- If Bob claims v_{-1}, then Alice replies in v_3. We have

$$o(([Ao^{11}B]^{\{2,3,5,9\}} \cup [AoA])_{v_3,v_{-1}}, B) \geq o([Ao^8B]^{\{2,6\}} \cup [Ao^2A], B) \geq o([Ao^2A], B) = \mathcal{A},$$

  where the last inequality comes from Lemma 5.51. □

A stable class of positions

In order to define our stable class $\mathcal{C}_1$, we first define the class $\mathcal{C}_0$, that informally corresponds to the positions derived from a weakly favorable position when Alice follows a winning strategy.

Definition 5.53 ($\mathcal{C}_0$)**.** *The class $\mathcal{C}_0$ is defined recursively as follows. A position P is an element of $\mathcal{C}_0$ if:*

- *$P = P_C$ where C is a weakly favorable component;*
- *there exists $P' \in \mathcal{C}_0$, with two unclaimed vertices x and y, such that $P = P'_{x,y}$ and $o(P \cup [AoA], B) = \mathcal{A}$.*

Next corollary is a direct application of Lemma 5.52.

Corollary 5.54. *For each $P \in \mathcal{C}_0$, we have $o(P \cup [AoA], B) = \mathcal{A}$.*

We can now define the stable class of positions $\mathcal{C}_1$. Intuitively, $\mathcal{C}_1$ corresponds to all the reachable position from skeletons S containing only favorable positions and at most one weakly favorable position, after having removed strongly favorable positions.

Definition 5.55 ($\mathcal{C}_1$)**.** *A position P belongs to the class $\mathcal{C}_1$ if P is the union of the following positions:*

1. *at most one position in $\mathcal{C}_0$.*
2. *a union of positions of the type $[Ao^nB]^{\{2,5,n-2\}}$ $n \geq 9$, n odd.*
3. *a union of k positions of the type $[Bo^{2t}B]^{\{2t-2\}}$, $t \geq 1$,*
4. *a union of k' positions of the type $[Ao^nA]^{\{2,5\}}$, with $k' \geq k$ and $n \geq 1$.*

Remark 5.56. *By definition, each position formed by at most one weakly favorable position and some positions of the alternative 1.d of Definition 5.43 is an element of $\mathcal{C}_1$.*

Next corollary is a direct application of Lemma 5.52 and Observation 5.40.

Corollary 5.57. *For each $P \in \mathcal{C}_1$, P admits an A-pairing.*

We will now prove that the class $\mathcal{C}_1$ (with a trap $[AoA]$ adjoined) is either stable after Alice's answer to Bob's claim, or directly winning for Alice. We will consider the two cases according to whether Bob claims the unclaimed vertex v_{-1} of $[AoA]$ or not. The first case where Bob does not claim v_{-1} is proved by Lemma 5.59 that requires Claim 5.58. The second case is when Bob claims v_{-1} and is solved by Lemma 5.61 that requires Claim 5.60 as a particular case.

Claim 5.58. *Let Q be any position, and t be a positive integer. Let $(w_1, w_2, ..., w_{2t})$ be the unclaimed vertices of $[Bo^{2t}B]^{\{2t-2\}}$. For each w_i, there exists $w_j \neq w_i$ and $0 \leq t' < t$ such that*

$$o(Q \cup [Bo^{2t'}B]^{\{2t'-2\}}, B) \leq o(Q \cup [Bo^{2t}B]^{\{2t-2\}}_{w_j,w_i}, B)$$

(with the convention that $[Bo^0B]^{\{-2\}}$ is empty).

The idea behind this claim is that if a position contains a bounded path $[Bo^{2t}B]^{\{2t-2\}}$ and Bob claims a vertex in it, then Alice can reply in the same bounded path, preserving the global structure of the position.

Proof. We have three cases.

- If $i = 2t$ then Alice claims w_{2t-1}, using Observations 5.41 and 5.42, we have

$$\begin{aligned} o(Q \cup ([Bo^{2t}B]^{\{2t-2\}})_{w_{2t-1},w_{2t}}, B) &= o(Q \cup [Bo^{2t-2}A]^{\{2t-2\}}, B) \\ &\geq o(Q \cup [Bo^{2t-2}B], B) \\ &\geq o(Q \cup [Bo^{2t-2}B]^{\{2t-4\}}, B) \end{aligned}$$

- If $i = 2k$, $1 \leq k < t$, then Alice claims w_{2k-1}. We have

$$o(Q \cup [Bo^{2t}B]^{\{2t-2\}}_{w_{2k-1},w_{2k}}, B) = o(Q \cup [Bo^{2k-2}A] \cup [Bo^{2(t-k)}B]^{\{2(t-k)-2\}}, B).$$

  Lemma 5.51 applies, and we get

$$o(Q \cup [Bo^{2t}B]^{\{2t-2\}}_{w_j,w_i}, B) \geq o(Q \cup [Bo^{2(t-k)}B]^{\{2(t-k)-2\}}, B).$$

- If $i = 2k-1$, $1 \leq k \leq t$, then Alice claims w_{2k}. We have

$$o(Q \cup [Bo^{2t}B]^{\{2t-2\}}_{w_j,w_i}, B) = o((Q \cup [Bo^{2k-2}B] \cup [Ao^{2(t-k)}B]^{\{2(t-k)-2\}}, B),$$

  Lemma 5.51 applies, and we get

$$o(Q \cup (Bo^{2t}B]^{\{2t-2\}}_{w_j,w_i}, B) \geq o(Q \cup [Bo^{2k-2}B], B) \geq o(Q \cup [Bo^{2k-2}B]^{\{2k-4\}}, B).$$

□

Lemma 5.59 (Bob does not claim v_{-1})**.** *Let $P \in \mathcal{C}_1$, such that P is not dominated by Alice. Let $Q = P \cup [AoA]$ and y be an unclaimed vertex of P. There exists an unclaimed vertex $x \neq y$ such that at least one of the following alternatives holds:*

- *$o(Q_{x,y}, B) = \mathcal{A}$;*
- *or there exists $P' \in \mathcal{C}_1$ with at least two unclaimed vertices less than P, such that $o(P' \cup [AoA], B) \leq o(Q_{x,y}, B)$.*

Proof. We denote by v_{-1} the unclaimed element of $[AoA]$ in Q. We assume that Bob claims y. We have several cases, according to the component that contains y.

- Assume first that y is an element of the component P' of P that belongs to $\mathcal{C}_0$. If there exists a free vertex x of P' such that $P'_{x,y} \in \mathcal{C}_0$, then we are done, since $P_{x,y} \in \mathcal{C}_1$.

  Otherwise, by definition of $\mathcal{C}_0$, $(P' \cup [AoA], B)$ is a win for Alice. Since there is no winning answer in P' for Alice to the claim y of Bob, we necessarily have $o((P' \cup [AoA])_{v_{-1},y}, B) = \mathcal{A}$. From the position $(P' \cup [AoA])_{v_{-1},y}$, Bob cannot dominate the whole graph. Alice can follow her strategy on $(P' \cup [AoA])_{v_{-1},y}$ and a pairing strategy on the rest of the graph (that exists by Corollary 5.57). This ensures that $o(Q_{v_{-1},y}, B) = \mathcal{A}$

- Assume now that y is an unclaimed vertex of $[Ao^nB]^{\{2,5,n-2\}}$, with n odd, $n \geq 9$. Let $(w_1, w_2, ..., w_n)$ be the sequence of free vertices of $[Ao^nB]^{\{2,5,n-2\}}$. By Lemma 1.78, as $N[w_1] \setminus N[V_t] \subset N[w_2] \setminus N[V_t]$ for $t \in \{A, B\}$, one can assume that $y \neq w_1$. Let P' be the position such that $P = P' \cup [Ao^nB]^{\{2,5,n-2\}}$ and let $Q' = P' \cup [AoA]$. Note that $P' \in \mathcal{C}_1$.

  - If $y = w_2$ then take $x = w_3$. By Lemma 5.8, Observation 5.6 and Lemma 5.51,

$$o(Q_{x,y}, B) \geq o(Q' \cup [Ao^{n-3}B]^{\{2,n-5\}}, B) \geq o(Q', B).$$

  - If $y = w_3$, then take $x = w_2$. There is double B-trap in Q_{w_2,w_3}. Thus, by Lemmas 5.14 and 5.57, we have $o(Q_{w_2,w_3}, B) = \mathcal{A}$.
  - If $y = w_i$ $i \geq 4$ with i even, then take $x = w_{i+1}$ We have

$$o(Q_{x,y}, B) = o(Q' \cup [Ao^{i-1}B]^{\{2,5\}} \cup [Ao^{n-i}B]^{\{n-i-2\}}, B) \geq o(Q' \cup [Ao^{i-1}B]^{\{2,5\}}, B)$$

    by application of Lemma 5.51. Note that $P' \cup [Ao^{i-1}B]^{\{2,5\}}$ is an element of $\mathcal{C}_1$.

– if $y = w_i$ with i odd and $i \geq 5$ then take $x = w_{i-1}$. We have

$$o(Q_{x,y}, B) = o(Q' \cup [Ao^{i-2}A]^{\{2,5\}} \cup [Bo^{n-i}B]^{\{n-i-2\}}, B).$$

Note that $P' \cup [Ao^{i-2}A]^{\{2,5\}} \cup [Bo^{n-i}B]^{\{n-i-2\}}$ is an element of $\mathcal{C}_1$, since $n - i$ is even and since we add a position $[Ao^{i-2}A]^{\{2,5\}}$.

- If y is an unclaimed vertex of a component of the form $[Bo^{2t}B]^{\{2t-2\}}$, then the position can be reduced by Claim 5.58.

- Assume now that y is an unclaimed vertex of $[Ao^nA]^{\{2,5\}}$. As before, we denote by $(w_1, w_2, ..., w_n)$ the vertices of $[Ao^nA]^{\{2,5\}}$, by P' the position P without the component $[Ao^nA]^{\{2,5\}}$ and by Q' the position $P' \cup [AoA]$. If $n \leq 3$, then take $x = v_{-1}$. We have $o(Q_{v_{-1},y}, B) = \mathcal{A}$ since the position $Q_{v_{-1},y}$ admits a A-pairing by Corollary 5.57 and Bob. Thus, it can be assumed that $n \geq 4$.

  – If $y \in \{w_1, w_2\}$, then take $x = w_3$. We have

  $$o(Q_{w_3,y}, B) \geq o(Q' \cup [Ao^{n-3}A]^{\{2\}}, B).$$

  Note that $P' \cup [Ao^{n-3}A]^{\{2\}}$ is in $\mathcal{C}_1$.

  – If $y = w_3$, then take $x = w_2$. The position Q_{w_2,w_3} admits an A-pairing, by Corollary 5.57. Thus, by lemma 5.14, we have $o(Q_{w_2,w_3}) = \mathcal{A}$.

  – if $y = w_i$, with $i \geq 4$, then take $x = w_{i-1}$. We have

  $$o(Q_{x,y}, B) = o(Q' \cup [Ao^{i-2}A]^{\{2,5\}} \cup [Ao^{n-i}B], B) \geq o(Q' \cup [Ao^{i-2}A]^{\{2,5\}}, B)$$

  according to Lemma 5.51. We are done since $P' \cup [Ao^{i-2}A]^{\{2,5\}}$ is in $\mathcal{C}_1$. □

Claim 5.60. *Let P be any position, and t be a nonnegative integer. We have*

$$o(P \cup [AoA], B) \leq o(P \cup [AooAooA]^{\{2,5\}} \cup [Bo^{2t}B]^{\{2t-2\}}, B)$$

where $[AooAooA]^{\{2,5\}}$ is the position obtained from $[Ao^5A]^{\{2,5\}}$ by adding the central vertex v_3 in the set V_A.

Proof. Let $(w_1, w_2, ..., w_{2t})$ be the sequence of unclaimed vertices of $[Bo^{2t}B]^{\{2t-2\}}$, (v_1, v_2, v_4, v_5) the sequence of unclaimed vertices of $[AooAooA]^{\{2,5\}}$, from left to right, and v_{-1} denote the unclaimed vertex of $[AoA]$. To lighten notation, we also state $R = P \cup [AooAooA]^{\{2,5\}} \cup [Bo^{2t}B]^{\{2t-2\}}$

Assume that $o(P \cup [AoA], B) = \mathcal{A}$. We prove that $o(R, B) = \mathcal{A}$ by induction on the number p of unclaimed vertices of R. For initialization, if $p \in \{4, 5\}$, then $t = 0$ and all the vertices of P are claimed except possibly one. Thus, since $o(P \cup [AoA], B) = \mathcal{A}$, Alice dominates P, and therefore Alice dominates $R = P \cup [AooAooA]^{\{2,5\}}$.

Now assume that $p \geq 6$. We have several alternatives according to the claim y of Bob in R.

- If y is an unclaimed vertex of $[Bo^{2t}B]^{\{2t-2\}}$, then Claim 5.58 applies, and the induction gives the result.

- Assume now that y is an unclaimed vertex of P. If there exists an unclaimed vertex x of P such that $o(P_{x,y} \cup [AoA], B) = \mathcal{A}$. We conclude by the induction hypothesis. If it is not possible, it means that once Bob has claimed y in the position $P \cup [AoA]$, Alice is forced to claim v_{-1}. Then, in R, Alice claims v_2, creating a double B-trap in v_1 and v_4.

  In the position $((P \cup [AoA])_{v_{-1},y}, B)$, Alice has a strategy which allows her to dominate P. Thus, Alice can win in $R_{v_2,y}$ playing component by component, until, the game ends as follows:

  – If Bob claims in P, Alice also claims in P according to the strategy in $((P \cup [AoA])_{v_{-1},y}, B)$.

  – If Bob claims in $[AooAooA]^{\{2,5\}}$, then Alice isolates one vertex using one of the B-trap.

– If Bob claims in $[Bo^{2t}B]^{\{2t-2\}}$, Alice follows a pairing strategy with the A-pairing of this component.

This way, Alice will prevent Bob to dominate since Bob has to claim at some point in $[AooAooA]$. The two other items ensure that Alice while dominate the whole graph.

- Assume finally that y is an unclaimed vertex of $[AooAooA]$. The position (P, A) has outcome $\mathcal{A}$ since it corresponds to the position $(P \cup [AoA], B)$ where Bob has claimed v_{-1}. Let x be an unclaimed vertex of P such that $o(P_x, B) = \mathcal{A}$, where P_x is the position obtained from P when Alice has claimed x. Alice answers x in the game R. We prove that $o(R_{x,y}, B) = \mathcal{A}$. Alice plays as follows:
  - If Bob claims a vertex in P_x, Alice answers in P_x according to her strategy in (P_x, B).
  - If Bob claims a vertex in $[Bo^{2t}B]^{\{2t-2\}}$, then Claim 5.58] can be used.
  - If Bob claims again a vertex in $[AooAooA]$, Alice claims w_1.

  The first item ensures that Alice dominates P before Bob. The second item ensures that the fact that Bob claims in the component $[Bo^{2t}B]^{\{2t-2\}}$ is irrelevant. In the case of third item, one can consider using Lemma 5.51 that the component $[Bo^{2t}B]^{\{2t-2\}}$ disappears. The component $[AooAooA]$ becomes also irrelevant since each player dominates all vertices of this component. Thus, it remains only the component derived from P_x where Alice dominates before Bob. □

Lemma 5.61 (Bob claims v_{-1})**.** *Let $P \in \mathcal{C}_1$ such that P is not dominated by Alice and $Q = P \cup [AoA]$. There exists an unclaimed vertex x of P such that at least one of the following alternatives holds:*

- $o(Q_{x,v_{-1}}, B) = \mathcal{A}$,
- *there exists $P' \in \mathcal{C}_1$ with at least one unclaimed vertex less than P, such that $o(P' \cup [AoA], B) \leq o(Q_{x,v_{-1}}, B)$.*

Proof. We have different cases according to the structure of P.

- If P contains a component $[Ao^nB]^{\{2,5,n-2\}}$, $n \geq 9$, n odd, let $v_1, v_2, ..., v_n$ denote the sequence of free vertices of this component. Take $x = v_2$. Let P' be the position such that $P = P' \cup [Ao^nB]^{\{2,5,n-2\}}$ and let $Q' = P' \cup [AoA]$. Note that $P' \in \mathcal{C}_1$.

  We now have
  $$o(Q_{v_2,v_{-1}}, B) \geq o(Q' \cup [Ao^{n-2}B]^{\{3,n-4\}}, B) \geq o(Q', B)$$
  from Lemma 5.51.

- If P contains a component $[Ao^nA]^{\{2,5\}}$, let $w_1, w_2, ..., w_n$ denote the sequence of vertices of $[Ao^nA]^{\{2,5\}}$. Let P' be the position such that $P = P' \cup [Ao^nA]^{\{2,5\}}$ and let $Q' = P' \cup [AoA]$.
  - If $n \geq 6$, then take $x = w_{n-1}$. We have
    $$o(Q_{w_{n-1},v_{-1}}, B) \geq o(Q' \cup [Ao^{n-2}A]^{\{2,5\}}, B)$$
    and note that $P' \cup [Ao^{n-2}A]^{\{2,5\}}$ is an element of $\mathcal{C}_1$.
  - If $3 \leq n \leq 4$, then take $x = w_2$. We have
    $$o(Q_{w_2,v_{-1}}, B) \geq o(Q' \cup [Ao^{n-2}A], B)$$
    and we conclude as previously.
  - If $n = 1$, then take $x = w_1$, isolating a vertex for Bob. The position $Q_{w_1,v_{-1}}$ admits an A-pairing from Corollary 5.57. Thus, from 5.13, we have $o(Q_{w_1,v_{-1}}, B) = \mathcal{A}$.

– If $p = 5$ and $k \geq 1$, take $x = w_3$, which creates an $[AooAooA]$. We define P'' such that

$$P' = P'' \cup ([Bo^{2t}B]^{\{2t-2\}}).$$

Note that $P'' \in \mathcal{C}_1$ as one component of each type 3 and 4 (in Definition 5.55)) has been removed. Lemma 5.60 applies, thus

$$o(P'' \cup [AoA], B) \leq o(P'' \cup [AooAooA]^{\{2,5\}} \cup [Bo^{2t}B]^{\{2t-2\}}, B) = o(Q_{w_3,v_{-1}}, B)$$

- If P cannot be treated in one of the previous cases, then P only contains some subpositions of the type $[Ao^5A]^{\{2,5\}}$ and at most one position $P_0 \in \mathcal{C}_0$. In this case, we will prove that $o(Q, B) = \mathcal{A}$, which implies the result.

  First we have $o([Ao^5A]^{\{2,5\}}, B) = \mathcal{A}$. Indeed, Bob is forced to claim w_3, since otherwise Alice claims w_3 and wins. After this claim, there exists an A-pairing of size 2, while Bob needs at least two more claims to dominate the graph. Thus $Q = P \cup [AoA]$ is formed by a union of winning positions (when Bob starts): one is $P_0 \cup [AoA]$ (or simply $[AoA]$ if P_0 does not exist) and the other ones are positions $[Ao^5A]^{\{2,5\}}$. Thus, by Observation 5.7, we get $o(Q, B) = \mathcal{A}$. □

Corollary 5.62. *For each position $P \in \mathcal{C}_1$, we have $o(P \cup [AoA], B) = \mathcal{A}$.*

Proof. We prove the result by induction on the number of unclaimed vertices in $Q = P \cup [AoA]$. If Alice dominates P, then we directly have $o(P \cup [AoA], B) = \mathcal{A}$. Otherwise, consider a move x of Bob. If $x \neq v_{-1}$ (respectively $x = v_{-1}$), then by Lemma 5.59 (resp. Lemma 5.61), there exists an unclaimed vertex y of P such that either Alice wins or there exists a position P' of $\mathcal{C}_1$ with less unclaimed vertices such that $o(P' \cup [AoA], B) \leq o(Q_{x,y}, B)$. By induction, $o(P' \cup [AoA], B) = \mathcal{A}$ and thus $o(Q_{x,y}, B) = \mathcal{A}$.

Thus, for any claim x of Bob, Alice can claim a vertex y such that $o(Q_{x,y}, B) = \mathcal{A}$. Therefore, $o(Q, B) = \mathcal{A}$. □

Conclusion

Putting together all the previous results, we can prove the reverse part of Theorem 5.44:

Corollary 5.63. *If all the components of S_F are favorable to Alice and at most one of them is weakly favorable, then $o((F, \{v_0\}, \emptyset), B) = \mathcal{A}$.*

Proof. Let $Q = (F, \{v_0\}, \emptyset)$ be a position such that all the components of S_F are favorable to Alice and at most one of them is weakly favorable. By Lemma 5.51, one can assume that there is no strongly favorable component in S_F. Let P such that $Q = P \cup [AoA]$. By Remark 5.56, P is an element of $\mathcal{C}_1$. Corollary 5.62 leads to the desired result. □

5.6 Further work

Although the provided algorithm and its proof are more complicated than in the Maker-Breaker version of the game, we manage to prove that the winner of the Maker-Maker domination game in forests can be computed in polynomial time. The last result listed in [DGPR20] that currently has no version in Maker-Maker is the polynomial algorithm for computing the winner in cographs. We only know that Alice wins in connected cographs, but since the union is hard to handle in Maker-Maker games, we have not managed to get a result for disconnected cographs. Finally, in Maker-Breaker, the presence of a paired dominating set provides a winning strategy for Maker, while as we saw before, it is not enough in Maker-Maker, even in cycles. However, we proved that if a graph G has a pairing dominating set of order $\gamma(G)$, Alice wins. Since this constraint is very strong, one can wonder if it is possible to find another sufficient condition for having a graph on which Alice wins, but with fewer constraints.

Another direction to pursue the study is to now focus on the parameterized complexity of the Maker-Maker convention, as it was done in the previous chapter for Maker-Breaker. Since the winner of the Maker-Maker domination game can also be computed in polynomial time on trees, considering its parameterized complexity by the feedback edge set should be the next step. Note, however, that it should be more difficult here, since we cannot reduce long induced paths as easily as in the Maker-Breaker convention, since even for one cycle, the outcome changes with this operation.

Chapter 6

Scoring positional games

Keep reading, you're almost done.

Until now, we considered positional games in their standard definition, i.e the game could end whenever Maker filled up a hyperedge. In this chapter, we extend the study of positional games to handle scores. Indeed, several games, as *Dots and boxes*, or the largest connected subgraph game are very close to positional games, but are not, as both players aim to maximize their score instead of just filling up one hyperedge. Therefore, we present in this chapter scoring positional games, which consist in playing on a hypergraph until all the vertices are claimed, and by defining the score as the number of hyperedges a player has filled up. In the Maker-Breaker convention, the score is defined as the number of hyperedges filled up by Maker, whereas in the Maker-Maker convention, it is the difference between the number of hyperedges filled up by the two players.

In Section 6.1, we present the general framework of scoring games. In particular, we present Milnor's universe as the natural universe to handle scoring positional games. In Section 6.2, we give some general results about scoring positional games. In the rest of this chapter, we focus on the game Incidence, the scoring positional game played on 2-uniform hypergraphs. We prove in Section 6.3 that the score of Maker-Maker Incidence is computable in polynomial time. In Section 6.4, we give some general results about Maker-Breaker Incidence, by adapting some known results from general Maker-Breaker games, such as Erdős-Selfridge's criterion or the Super Lemma. In Section 6.5, we prove that, contrary to the Maker-Maker convention, computing the score in Maker-Breaker Incidence is PSPACE-complete. Finally, in Section 6.6 we provide several equivalences on paths, that allow us to compute the score on paths and cycles.

This work was a collaboration with Guillaume Bagan, Quentin Deschamps, Eric Duchêne, Bastien Durain, Brice Effantin, Valentin Gledel and Aline Parreau. It has been published in Discrete Mathematics [BDD+23].

6.1 Scoring combinatorial games

Scoring games have been introduced in the 1950s by Milnor [Mil53] and Hanner [Han59]. Their study was almost forgotten until the 2000s, when different formalisms for such games have been introduced by Ettinger [Ett96], Stewart [Ste12], or Larsson, Nowakowski and Santos [LNS15b]. Therefore, scoring game have been introduced in the combinatorial game theory, and relates more to this structure than to the structure of positional games. The survey paper [LNS15a] summarizes these different approaches.

In scoring games, two players, usually Left and Right, alternate moves with a score adjoined to the game. Each move of a player can modify this score, Left aims at maximizing the score at the end of the game, while Right tries to minimize it. Since scoring games are also finite perfect information games, if both players play optimally, the score at the end of the game is well-defined and only depends on who starts.

Despite the fact that scoring games were less studied, mainly due to the difficulty to build a general framework for them, particular scoring games on graphs have still been introduced recently. One can cite

the game Influence introduced by Duchêne [DGP$^+$21] in 2021 which has been proven PSPACE-complete in 2024 [DOP24], or the largest connected subgraph game, introduced by Bensmail *et al.* [BFMIN22, BFM$^+$23], firstly as a scoring connection game, and then as a Maker-Breaker connection game. Nevertheless, there is currently no general framework on scoring games on graphs, and we introduce here the framework of scoring positional games to develop some general properties to handle some of them.

6.1.1 Definition of scoring positional games

Scoring positional games

Scoring positional games are played on hypergraphs by two players, Left and Right, with the same rules as standard positional games. The only differences are the winning conditions. In a scoring positional game, the game ends when all vertices have been claimed. A player's score is then defined as the number of hyperedges he manages to fill up. In the Maker-Maker convention, each player tries to both maximize his score and to minimize his opponent's score. In the Maker-Breaker convention, Maker (identified as Left) tries to maximize her score while Breaker (identified as Right) tries to minimize the score of Maker. Note that we use Left and Right here instead of Alice and Bob or Maker and Breaker, as these are the usual names of the players in scoring games.

More formally, for any scoring game, two scores are defined depending on which player starts. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. We define the score of $\mathcal{H}$ as follows:

- in the Maker-Maker convention, $Ls(\mathcal{H})$ (resp. $Rs(\mathcal{H})$) is the difference between the number of hyperedges filled up by Left and the number filled up by Right when Left starts (resp. when Right starts) and both players play optimally.
- in the Maker-Breaker convention, $Ls(\mathcal{H})$ (resp. $Rs(\mathcal{H})$) is the number of hyperedges filled up by Left when Left (resp. Right) starts and both players play optimally.

It is well-known in scoring game theory that these notions exist and are well-defined (by considering the game tree of all the possible moves). Note that in the Maker-Maker convention, by symmetry of the roles of both players, we have that $Ls(\mathcal{H}) = -Rs(\mathcal{H})$, so computing $Ls(\mathcal{H})$ will be of sufficient interest. In the Maker-Breaker convention, we have that $Ls(\mathcal{H})$ and $Rs(\mathcal{H})$ are nonnegative values by definition.

In addition, it will be helpful to consider the scores obtained after some vertices have been claimed. Therefore, positions are defined here similarly to their definition in positional games. A *position* of a scoring positional game is a triplet $P = (\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$ such that $\mathcal{X}_L$ and $\mathcal{X}_R$ are disjoint subsets of vertices. The set $\mathcal{X}_L$ corresponds to the vertices claimed by Left whereas $\mathcal{X}_R$ correspond to the vertices claimed by Right. The set of remaining vertices will be generally denoted by $\mathcal{X}_F$. We have $\mathcal{X}_F = \mathcal{X} \setminus (\mathcal{X}_L \cup \mathcal{X}_R)$. For both conventions, we will denote by $Ls(P)$ (resp. $Rs(P)$) the score of $\mathcal{H}$ if Left has already claimed the vertices of $\mathcal{X}_L$, and Right the vertices of $\mathcal{X}_R$, when Left (resp. Right) is the next player to move on P. When $\mathcal{X}_F \neq \emptyset$, the scores at a position P can be recursively defined as follows:

$$Ls(P) = \max_{x \in \mathcal{X}_F} Rs(\mathcal{H}, \mathcal{X}_L \cup \{x\}, \mathcal{X}_R)$$
$$Rs(P) = \min_{x \in \mathcal{X}_F} Ls(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R \cup \{x\}).$$

When $\mathcal{X}_F = \emptyset$, the score depends on the convention. In Maker-Maker convention,

$$Ls(P) = Rs(P) = |\{e \in \mathcal{F} | e \subseteq \mathcal{X}_L\}| - |\{e \in \mathcal{F} | e \subseteq \mathcal{X}_R\}|$$

whereas in Maker-Breaker convention, we have

$$Ls(P) = Rs(P) = |\{e \in \mathcal{F} | e \subseteq \mathcal{X}_L\}|.$$

In the literature, there are few games that can be seen as scoring positional games. The famous Dots and Boxes games [Ber00], that has recently be proven PSPACE-complete by Buchin *et al.* [BHKvM21], could be an example, with the additional constraint that a player is forced to play again each time he gets points. By removing this constraint, we get a pure example of the above definition (in the Maker-Maker convention), and the game is known as Picarête [BDG06]. More recently, the Constructor-Blocker game introduced by Patkos *et al.* [PSV23] in 2023, in which Constructor aims at maximizing the number of copies of a graph H with a forbidden graph F, can be seen as a scoring Maker-Breaker game when F is empty.

6.1.2 Milnor's universe

Properties of Milnor's universe

In 1953 [Mil53], Milnor introduced a universe of scoring games having nice properties. This universe is the one of *dicotic nonzugzwang* games:

- a game is *dicotic* if at any moment of the game, if a player can move, the other player can also move.
- a game is *nonzugzwang* if at any moment of the game, both players have no interest in skipping their turn.

We say that a game *belongs to Milnor's universe* if it satisfies these two properties.

Being in Milnor's universe induces a couple of useful results concerning the sum operator and the equivalence of games. The *disjunctive sum operator* $+$ applied to scoring (positional) games G_1 and G_2 defines the game $G_1 + G_2$ as the game in which a move consists in moving either in G_1 or in G_2. The game ends when the moves in both components of the sum are exhausted. See [DOP24] for the formal definition. Note that the sum of two scoring positional games, with the same convention, is still a scoring positional game with hypergraph the disjoint union of the two hypergraphs. As game sums appear in many games when playing, one could expect to simplify them by replacing large games by smaller ones. This leads to the notion of equivalence of games:

Definition 6.1 (Milnor [Mil53])**.** *Two scoring games G_1 and G_2 are* equivalent *(write $G_1 \equiv G_2$) if for any game G, we have $Ls(G + G_1) = Ls(G + G_2)$ and $Rs(G + G_1) = Rs(G + G_2)$.*

In other terms, one can always exchange G_1 and G_2 in any sum of games if they are equivalent. In particular, games that are equivalent to the empty game can be removed from any sum of games.

Games belonging to Milnor's universe form an Abelian group with the sum operator[Mil53]. In particular, this implies that every game G in Milnor's universe admits an inverse, i.e. a game G' such that $G + G' \equiv 0$ (where 0 is the empty game). More precisely, this inverse corresponds to the *negative* of G, denoted by $-G$, i.e. the game where the roles of Left and Right are exchanged, together with their scores. Moreover, proving equivalence in Milnor's universe is greatly simplified, thanks to the next lemma.

Lemma 6.2 (Milnor [Mil53])**.** *For any games G_1 and G_2 that are dicotic nonzugzwang, we have: $Ls(G_1 - G_2) = Rs(G_1 - G_2) = 0$ if and only if G_1 and G_2 are equivalent.*

More generally, a game G is *a number* $k \in \mathbb{Z}$, and we write $G = k$, if we have $Ls(G) = Rs(G) = k$. The following property holds with numbers:

Lemma 6.3 (Milnor [Mil53])**.** *Let G be a game and k be a number. We have $Ls(G + k) = Ls(G) + k$ and $Rs(G + k) = Rs(G) + k$. Note that on the left side of the equality, k is a game, and on the right side, it is an integer.*

In addition, sums of games in Milnor's universe can be bounded as follows:

Lemma 6.4 (Milnor [Mil53])**.** *Let G_1 and G_2 be two dicotic nonzugzwang games, we have*

$$Rs(G_1) + Rs(G_2) \leq Rs(G_1 + G_2) \leq Ls(G_1) + Rs(G_2) \leq Ls(G_1 + G_2) \leq Ls(G_1) + Ls(G_2).$$

Belonging to Milnor's universe is a very strong property dealing with scoring games, and we prove here that scoring positional games belong to this universe.

Partisan scoring positional games

In what follows, we will show that scoring positional games belong to Milnor's universe. Yet, the negative of a game cannot be defined in the Maker-Breaker convention, as the scores of Maker and Breaker can not be interchanged naturally, by asymmetry of the definition of the score. Therefore, we have decided to embed scoring positional games in a more general family that will be called *Partisan scoring positional games*. The term partisan is derived from standard combinatorial games [Ber00], meaning that Left and Right may have different moves (and also different ways of scoring points).

A partisan scoring positional game is played on a hypergraph $\mathcal{H}$ whose hyperedges are either colored blue, red or green. The two players, Left and Right, alternatively claim vertices of $\mathcal{H}$. The score of Left corresponds to the blue and green hyperedges she claimed, whereas the score of Right corresponds to the red and green ones. As previously, the score of the game ($Ls(\mathcal{H})$ and $Rs(\mathcal{H})$, depending on who starts) is the difference between the score of Left and Right.

Partisan scoring positional games include both Maker-Maker and Maker-Breaker scoring positional games. Even more, the convention can be omitted, as it is deduced by the colors of the hypergraph. Indeed, if all the hyperedges are green, it means that both players can score any hyperedge, which corresponds to the Maker-Maker version. If all the hyperedges are blue, it corresponds to the Maker-Breaker convention, as only Left can get points. According to this definition, the negative of a partisan scoring positional game is well-defined, as it suffices to exchange the colors blue and red in the hyperedges, as well as the vertices already chosen by Left and Right (if any).

We will now give several general results about partisan scoring positional games. By inclusion, these results will also concern scoring positional games. First, we will prove that they belong to Milnor's universe and thus satisfy Lemma 6.2.

Lemma 6.5. *Partisan scoring positional games belong to Milnor's universe.*

Proof. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph with hyperedges colored blue, red and green, and $\mathcal{X}_L, \mathcal{X}_R \subset \mathcal{X}$ be vertices already claimed by Left and Right respectively such that $\mathcal{X}_L \cap \mathcal{X}_R = \emptyset$.

A partisan scoring positional game is dicotic: if $\mathcal{X}_L \cup \mathcal{X}_R = \mathcal{X}$, then no moves are available, neither for Left nor for Right. Otherwise, let $v \in \mathcal{X} \setminus \{\mathcal{X}_L \cup \mathcal{X}_R\}$. Both Left and Right are allowed to play v as it is an unclaimed vertex. Therefore, the game is dicotic.

A partisan scoring positional game is nonzugzwang: We need to prove that $Ls(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) \geq Rs(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$. Let $k = Rs(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$ with $\mathcal{X}_L, \mathcal{X}_R$ vertices already claimed in $\mathcal{H}$ by Left and Right respectively. If $\mathcal{X}_L \cup \mathcal{X}_R = \mathcal{X}$, we have $Ls(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) = Rs(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) = k$ as there is no move available in $\mathcal{H}$. Otherwise, let $\mathcal{S}$ be an optimal strategy for Left when Right starts. We define a strategy $\mathcal{S}'$ for Left when she starts as follows:

- Left considers an arbitrary unclaimed vertex v_0 of the graph, and plays the vertex she would have played in $\mathcal{S}$ if Right plays v_0.
- Whenever, Right plays a vertex w in $\mathcal{X} \setminus \{v_0\}$, she plays the vertex she would have played in $\mathcal{S}$ if Right has played w in $\mathcal{S}$ after having played v_0 on first move.
- If Right plays v_0, she considers an arbitrary unclaimed vertex v_1 in the graph, and continues this strategy by supposing that Right has played v_1 instead of v_0. More generally, when Right claims the vertex v_ℓ, she considers an unclaimed vertex $v_{\ell+1}$ and considers that Right has claimed $v_{\ell+1}$ instead.
- At the end, if she needs to consider that Right has played a vertex v_ℓ and no other vertex is available, she plays v_ℓ.

Following this strategy, all the vertices Left would have played in $\mathcal{S}$ if Right has played the vertices v_is she has considered, have been played in $\mathcal{S}'$ by Left. Similarly, the vertices that Right have played in

$\mathcal{S}'$ are a subset of the one he would have played in $\mathcal{S}$. Therefore, as $\mathcal{S}$ was an optimal strategy in $\mathcal{H}$ when Right starts, this strategy ensures that Left scores at least $k = Rs(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$. Finally, we have $Ls(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) \geq k = Rs(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$, and the game is nonzugzwang.

As the game is nonzugzwang and dicotic, it belongs to Milnor's universe. □

Belonging to Milnor's universe will be the main tool of the proofs of Section 6.6.

6.1.3 Incidence

In most of this chapter, we will mainly focus on an example of scoring positional game that is called Incidence. It corresponds to the game played on a hypergraph where all hyperedges are of size two. In others terms, this game can be defined as follows on a simple graph $G = (V, E)$. Alternately, two players claim an unclaimed vertex of V. When all the vertices have been taken, the score of a player is defined as the number of edges in the subgraph of G induced by the vertices he claimed.

Hence, in both conventions, Left (that is always Maker) aims at collecting points by claiming the two extremities of an edge. The main difference concerns the role of Right, that aims at touching the maximum number of edges (hence prohibiting a maximum number of points for Left) in the Maker-Breaker convention. See Figure 6.1 for an example of computations of the score at the end of a game.

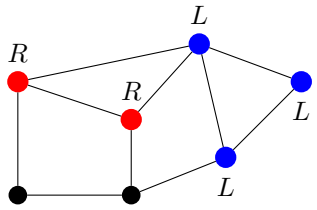


Figure 6.1: A position P of Incidence. In Maker-Maker convention, we have $Ls(P) = 4 - 2 = 2$ and $Rs(P) = 3 - 2 = 1$ as both player wants to claim the unclaimed vertex on the left. In Maker Breaker convention, we have $Ls(P) = 4$ and $Rs(P) = 3$ as Left has already the two extremity of three edges, and she can take one more only if she is the next player to move.

6.2 General results on scoring positional games

In this section, we present the first results on scoring positional games. First, as a more general instance of positional game, we prove that computing the score is PSPACE-complete. Then, we bound the score in the general case.

6.2.1 General complexity of scoring positional games

To deal with the complexity of scoring positional games, we must first formally define what a scoring positional game is.

Definition 6.6 (Maker-Breaker Scoring Positional Game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Maker-Breaker Scoring Positional Game *is played by two players, Left and Right. Left and Right take turns claiming an unclaimed vertex of $\mathcal{X}$ with Left starting. The score of the game is defined as the number of hyperedges filled up by Left.*

Definition 6.7 (Maker-Maker Scoring Positional Game)**.** *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. A* Maker-Maker Scoring Positional Game *is played by two players, Left and Right. Left and Right take turns claiming an unclaimed vertex of $\mathcal{X}$ with Left starting. The score of the game is defined as the difference between the number of hyperedges filled up by Left and the one filled up by Right.*

First, note that in Maker-Breaker, Left scores at least one means that she manages to fill up a hyperedge. The following lemma is a straightforward application of Theorem 1.64. A result for hypergraphs of lower ranks will be provided is Section 6.5.

Lemma 6.8. *Determining whether Maker can score at least k in a Maker-Breaker Scoring Positional Game is* PSPACE*-complete, even restricted to* 6*-uniform hypergraphs and $k = 1$.*

Proof. Let $\mathcal{H}$ be a hypergraph.

Maker wins the Maker-Breaker positional game on $\mathcal{H}$ if and only if Left can score at least 1 on $\mathcal{H}$. By Theorem 1.64, it is PSPACE-complete to determine whether Maker wins on $\mathcal{H}$. □

6.2.2 Bounds on the score

In this section, we provide bounds on the score in both Maker-Maker and Maker-Breaker convention.

Bounds in Maker-Maker convention

We start by providing a bound in Maker-Maker convention, using the maximal degree of the hypergraph. Let $\mathcal{H}$ be a hypergraph. Similarly to graphs, we define the *degree* of a vertex v of $\mathcal{H}$ as the number of hyperedges containing v. We denote by $\Delta(\mathcal{H})$ the maximal degree of $\mathcal{H}$.

Lemma 6.9. *Let $\mathcal{H}$ be a hypergraph. In the Maker-Maker scoring positional game on $\mathcal{H}$, we have $-\Delta(\mathcal{H}) \leq Rs(\mathcal{H}) \leq 0 \leq Ls(\mathcal{H}) \leq \Delta(\mathcal{H})$.*

Proof. As noticed in Section 6.1.1, we have $Ls(\mathcal{H}) = -Rs(\mathcal{H})$ in the Maker-Maker convention since players have symmetric roles. Since the game is nonzugzwang, we also have $Ls(\mathcal{H}) \geq Rs(\mathcal{H})$ which implies that $Rs(\mathcal{H}) \leq 0 \leq Ls(\mathcal{H})$.

To prove the upper bound with $\Delta(\mathcal{H})$, we just need to prove that $Ls(\mathcal{H}) \leq \Delta(\mathcal{H})$. Let v_0 be the first vertex played in an optimal strategy. Consider the hypergraph $\mathcal{H}'$ obtained from $\mathcal{H}$ by removing v_0 and all the hyperedges containing it. If the second player applies the optimal strategy for $\mathcal{H}'$ during the rest of the game, he will score at least $Rs(\mathcal{H}') \leq 0$ on it and the final score will be at most $|\{e|v_0 \in e\}| + Rs(\mathcal{H}')$. Thus, we have $Ls(\mathcal{H}) \leq deg(v_0) + Rs(\mathcal{H}') \leq \Delta(\mathcal{H})$. □

We do not think that the upper bound in Lemma 6.9 is tight if the hypergraph is simple (i.e. there are no two hyperedges that contain exactly the same vertices). Actually, the best example we know in this case is a hypergraph $\mathcal{H}$ having a universal vertex x, a hyperedge with x alone and $\Delta - 1$ hyperedges of size 2 containing x and another unique vertex, see Figure 6.2. For this hypergraph, $Ls(\mathcal{H}) = \lfloor \frac{\Delta(\mathcal{H})+1}{2} \rfloor$. Furthermore, we will prove that for 2-uniform hypergraphs (i.e. graphs), the score is at most $\Delta(\mathcal{H})/2$ (see Corollary 6.15). We believe that this bound remains true in any hypergraph:

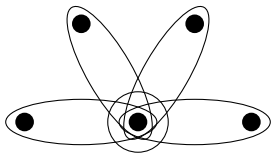


Figure 6.2: A hypergraph satisfying $Ls(H) = \lfloor \frac{\Delta(H)+1}{2} \rfloor$ in Maker-Maker convention

Conjecture 6.10. *Let $\mathcal{H}$ be a simple hypergraph. In the Maker-Maker scoring positional game on $\mathcal{H}$, we have $Ls(\mathcal{H}) \leq \frac{\Delta(\mathcal{H})+1}{2}$.*

Bounds in Maker-Breaker

In Maker-Breaker convention, the bound from Lemma 6.9 is not valid anymore. Indeed, the score can actually be linear with the number of vertices of the hypergraph, even if the maximal degree is constant. Next, we derive a general tight bound, based on the same principle used to prove the Erdös-Selfridge criterion [ES73]. Some tight examples will be given in Section 6.4 for 2-uniform hypergraphs (see Corollary 6.16).

The main idea to prove Theorem 1.25 (from Erdős and Selfridge) is that if the hyperedges are large enough, Breaker will have the time to play in all of them before Maker can fill one. A similar idea can be introduced when dealing with scores by computing how many hyperedges Breaker can touch. The strategy used relies on a greedy strategy by introducing a potential function, as it was done by Erdős and Selfridge. Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. We denote by $\ell(\mathcal{H})$ the maximum number of hyperedges that contain a fixed pair of vertices. More formally, $\ell(\mathcal{H}) = \max_{x,y \in \mathcal{X}^2} |\{e \in \mathcal{F} | x, y \in e\}|$.

Theorem 6.11. *Let $\mathcal{H} = (\mathcal{X}, \mathcal{F})$ be a hypergraph. In the Maker-Breaker convention, we have $Ls(\mathcal{H}) \geq \sum_{e \in \mathcal{F}} 2^{-|e|} - \frac{n\ell(\mathcal{H})}{8}$, and $Rs(\mathcal{H}) \leq \sum_{e \in \mathcal{F}} 2^{-|e|}$.*

Proof. Let $(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$ be any position of a Maker-Breaker scoring positional game. We introduce the potential function:

$$P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) = \sum_{e \in \mathcal{F}, e \cap \mathcal{X}_R = \emptyset} 2^{-|e \setminus \mathcal{X}_L|}.$$

In this function, only hyperedges not played by Right are considered, and we only count the number of free vertices in the edge. Note that at the beginning of the game, $P(\mathcal{H}, \emptyset, \emptyset) = \sum_{e \in \mathcal{F}} 2^{-|e|}$. At the end of the game, $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_R$ and $P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) = |\{e \in \mathcal{F} | e \cap \mathcal{X}_R = \emptyset\}|$ is the final score. Furthermore, when a vertex v is played by Maker (respectively Breaker), the potential is increasing (resp. decreasing) by the quantity

$$\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v) = \sum_{e | e \cap \mathcal{X}_R = \emptyset, v \in e} 2^{-|e \setminus \mathcal{X}_L|}.$$

Let $\mathcal{S}$ be a strategy for Maker consisting in maximizing P at each move, i.e. Maker chooses the vertex v that maximizes $\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v)$. We prove that this strategy provides the desired bound. Suppose first that Maker starts. Suppose $\mathcal{X}_L$ and $\mathcal{X}_R$ have already been played by Maker and Breaker respectively. Let v_L the vertex played by Maker according to $\mathcal{S}$ and v_R the vertex played by Breaker after this move. As Maker has played v_L and not v_R, we have, before v_L was played, $\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_L) \geq \delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_R)$.

However, $\delta_P(\mathcal{H}, \mathcal{X}_L \cup \{v_L\}, \mathcal{X}_R, v_R)$ might be larger than $\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_R)$ after v_L was played if there exist some hyperedges that contain both v_L and v_R. We actually have:

$$\begin{aligned}\delta_P(\mathcal{H}, \mathcal{X}_L \cup \{v_L\}, \mathcal{X}_R, v_R) &= \delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_R) + \sum_{e \cap \mathcal{X}_R = \emptyset, v_L, v_R \in e} 2^{-|e \setminus \mathcal{X}_L|} \\ &\leq \delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_R) + \frac{\ell(\mathcal{H})}{4}.\end{aligned}$$

Last inequality comes from the fact that $e \setminus \mathcal{X}_L$ must contain v_L and v_R and thus has size at least 2. Therefore, we have

$$\begin{aligned}P(\mathcal{H}, \mathcal{X}_L \cup \{v_L\}, \mathcal{X}_R \cup \{v_R\}) &= P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) + \delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_L) - \delta_P(\mathcal{H}, \mathcal{X}_L \cup \{v_L\}, \mathcal{X}_R, v_R) \\ &\geq P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) - \frac{\ell(\mathcal{H})}{4}.\end{aligned}$$

As there is n moves in the game by applying this step $\frac{n}{2}$ times for each pair of moves (recall that we consider here that Maker starts), we have at the end of the game $Ls(\mathcal{H}) \geq P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) \geq P(\mathcal{H}, \emptyset, \emptyset) - \frac{n}{2} \times \frac{\ell(\mathcal{H})}{4}$, as required.

Suppose now that Breaker starts and considers this strategy for him (i.e. choosing the vertex v that maximizes $\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v)$). Suppose $\mathcal{X}_L$ and $\mathcal{X}_R$ have already been played by Maker and Breaker respectively. Let v_R be the vertex played by Breaker according to $\mathcal{S}$ and let v_L be the vertex answered by Maker. We have $\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_R) \geq \delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R, v_L)$. Note that here, $\delta_P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R \cup \{v_R\}, v_L)$ cannot increase after the move of Right, as it does not change the size of the hyperedges (it can only decrease if some edges containing v_L also contains v_R). Therefore, after these two moves, we obtain $P(\mathcal{H}, \mathcal{X}_L \cup \{v_L\}, \mathcal{X}_R \cup \{v_R\}) \leq P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R)$. By applying this result from $\mathcal{X}_L = \mathcal{X}_R = \emptyset$ to the end of the game, we obtain $P(\mathcal{H}, \mathcal{X}_L, \mathcal{X}_R) \leq P(\mathcal{H}, \emptyset, \emptyset)$ for any sets $\mathcal{X}_L$ and $\mathcal{X}_R$ obtained after Right applies $\mathcal{S}$. In particular, when the game ends, this strategy ensures that $Rs(\mathcal{H}) \leq P(\mathcal{H}, \emptyset, \emptyset) = \sum_{e \in \mathcal{H}} 2^{-|e|}$. □

6.3 Maker-Maker Incidence is polynomial

From now on and until the end of the chapter, we will focus on the game Incidence, defined in Section 6.1.3.

In this section, we provide a linear time algorithm to compute the score of Maker-Maker Incidence. A natural idea, while playing Incidence, is that high degree vertices are interesting to play first, as they enable to score many points with their multiple adjacent edges. Therefore, a simple strategy for both players would be to play greedily by always picking an available vertex of highest degree. We prove here that this strategy is optimal.

Theorem 6.12. *Let G be a graph with n vertices. Let $d_1 \geq ... \geq d_n$ be the degree of the vertices in decreasing order. For the game Maker-Maker Incidence played on G, we have*

$$Ls(G) = \frac{1}{2}\left(\sum_{i \text{ odd}} d_i - \sum_{i \text{ even}} d_i\right).$$

In particular, the score can be computed in linear time.

Proof. Let $G = (V, E)$ be a graph. Denote by $v_1, \ldots, v_n$ the vertices of G of degree $d_1, \ldots, d_n$ respectively, and arranged such that $d_1 \geq d_2 \geq \cdots \geq d_n$. Denote by $s = \frac{1}{2}(\sum_{i \text{ odd}} d_i - \sum_{i \text{ even}} d_i)$. We will prove that $Ls(G) = s$. Before proving the value of the score, we prove the following claim:

Claim 6.13. *Denote by V_L the vertices claimed by Left, and by V_R the vertices claimed by Right at the end of a game played on G. The score obtained is $\frac{1}{2}(\sum_{v_l \in V_L} d_l - \sum_{v_r \in V_R} d_r)$.*

Proof of the claim. Denote by e_L (resp. e_R) the number of edges where both endpoints were claimed by Left (resp. Right) and by e_0 the number of edges which have one extremity claimed by each player.

By definition, the score is $e_L - e_R$. Now, by a double counting argument, we have $\sum_{v_l \in V_L} d_l = 2e_L + e_0$, and $\sum_{v_r \in V_R} d_r = 2e_R + e_0$. Therefore, the score of the game is $e_L - e_R = \frac{1}{2}(\sum_{v_f \in V_L} d_l - \sum_{v_r \in V_R} d_r)$. ◇

Now we provide a strategy for Left that proves that $Ls(G) \geq s$. The same argument works for Right and leads to $Ls(G) \leq s$. Consider that Left claims at each turn the free vertex of highest degree. During her first turn, she claims a vertex of degree d_1, during the second turn, she claims either a vertex of degree d_2 or d_3, both having a value of at least d_3, ..., during here k-th turn, she will claim a vertex of degree $d_k, d_{k+1}, \ldots$ or d_{2k-1}, each of them have a value of at least d_{2k-1}. In the end, she will have played $\lceil \frac{n}{2} \rceil$ vertices, and the k-th of them will be of degree at least d_{2k-1}. Reciprocally, the highest degree played by Right has value at most d_2, the second highest has value at most d_4 and so on. Therefore, by using the result of the claim, the score obtained by this strategy is at least s.

The above score can be computed in linear time because it does not require to sort the list of the vertices, but only to know the number of vertices of any degree, which is bounded by $n - 1$. □

Corollary 6.14. *Let $n \in \mathbb{N}$. Denote by P_n the path of order n. In Maker-Maker Incidence, we have $Ls(P_n) = -Rs(P_n) = 0$ if n is even and $Ls(P_n) = -Rs(P_n) = 1$ if n is odd.*

Proof. P_n has exactly $n-2$ vertices of degree 2 and two vertices of degree 1. Therefore, if n is even, an optimal strategy gives $\frac{n}{2}-1$ vertices of degree two and one vertex of degree one to each player, which provides a draw. If n is odd, Left has one more vertex of degree 2 to play, and her score is then 1. □

Corollary 6.15. *Let G be a graph of maximal degree Δ. In Maker-Maker Incidence, we have $Ls(G) \leq \frac{\Delta}{2}$.*

Proof. Let G be a graph of maximal degree Δ. Up to adding an isolated vertex, suppose it has an even number of vertices. Denote by $d_1, d_2, \ldots, d_{2n}$ its degrees written in decreasing order. We have $Ls(G) = \frac{1}{2}\sum_{i=1}^{n}(d_{2i-1} - d_{2i}) = \frac{\Delta}{2} - \sum_{i=1}^{n}(d_{2i} - d_{2i+1})$, by setting $d_{2n+1} = 0$. For any $1 \leq i \leq n$, we have $d_{2i} \geq d_{2i+1}$. Hence, each term of the sum is nonnegative, and finally, we have $Ls(G) \leq \frac{\Delta}{2}$. □

6.4 General results on Maker-Breaker Incidence

In the rest of the chapter, we focus on the Maker-Breaker version of Incidence. Contrary to the Maker-Maker version of this game, a greedy strategy is not always optimal. Thus, studying this game is much more challenging. In this section, we give some general results on this version. We start with a direct application of the bound given for general scoring positional games in Theorem 6.11.

Corollary 6.16. *Let G be a graph with n vertices and m edges. In the Maker-Breaker Incidence game, $Ls(G) \geq \frac{m}{4} - \frac{n}{8}$, and $Rs(G) \leq \frac{m}{4}$.*

These bounds are tight for arbitrarily large values of n and m.

Proof. This is a direct application of Theorem 6.11. Since the hypergraph is 2-uniform and simple, for each pair of vertices, there is at most one edge containing the two vertices. Thus we have $\ell(G) = 1$. Furthermore, each edge has size 2, thus $\sum_{e \in G} 2^{-|e|} = \frac{m}{4}$.

For tightness, consider first a graph G that is a complete graph of order $8k$, with $k \in \mathbb{N}$. The lower bound gives $Ls(G) \geq \frac{\binom{8k}{2}}{4} - k = \binom{4k}{2}$. By playing randomly, Left takes $4k$ vertices and each pair of vertices scores one point. Thus $Ls(G) = \binom{4k}{2}$

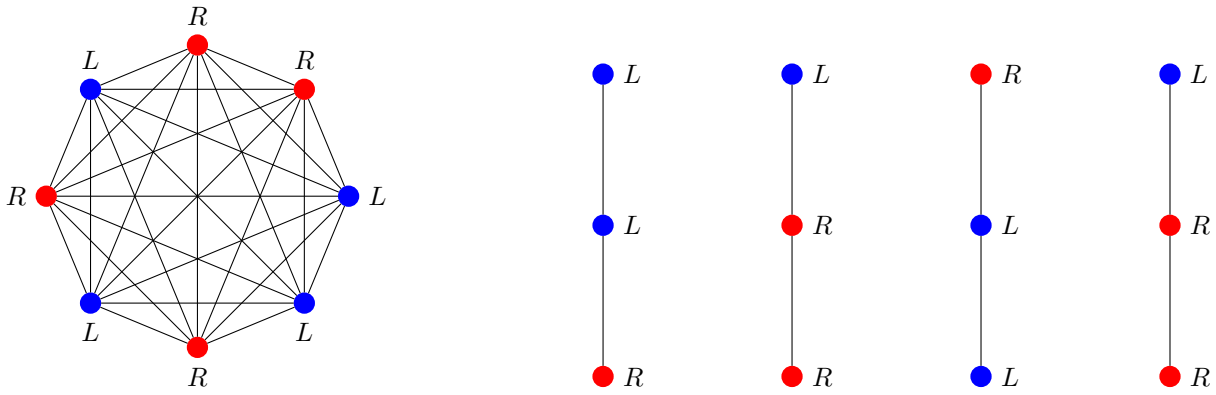
Consider the graph H made by a disjoint union of $2k$ paths on three vertices. Left playing second can take k central vertices and one leaf for each central vertex he has taken. This strategy gives at most k points to Left which is equal to the upper bound $\frac{m}{4}$ given in the statement.

The graphs satisfying these tight bounds are depicted in Figure 6.3 □

The Super Lemma can also be applied in scoring positional games. Indeed, its proof does not use the fact that the game ends after a hyperedge has been filled up. By consistency, even if the two proofs are similar, its scoring version will still be proved here. We first introduce *equivalent vertices.*

Definition 6.17. *let $G = (V, E)$ be a graph, $P = (G, V_L, V_R)$ some position of the game on G and $V_F = V \setminus (V_L \cup V_R)$ the set of free vertices. Let $v_1, v_2 \in V_F$ be two free vertices. v_1, v_2 are said to be* equivalent *in P if and only if we have $N(v_1) \cap V_F \setminus \{v_2\} = N(v_2) \cap V_F \setminus \{v_1\}$ and $|N(v_1) \cap V_L| = |N(v_2) \cap V_L|$. Note that the first equality is a set equality, while the second one only is on cardinals. In practice, the first equality states that both vertices are adjacent to the same unclaimed vertices, and thus create the same threats, while the second one states that Left scores the same amount of points by playing v_1 or v_2. Equivalent vertices are depicted in Figure 6.5.*

Lemma 6.18 (Super Lemma, scoring version)**.** *Let $G = (V, E)$ be a graph and let $P = (G, V_L, V_R)$ be a position of the game. Let v_1, v_2 be equivalent vertices in P. In Maker-Breaker Incidence, we have $Ls(P) = Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$ and $Rs(P) = Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$.*



(a) Graph satisfying $Ls(G) = \frac{m}{4} - \frac{n}{8} = \frac{28}{4} - \frac{8}{8} = 6$ (b) Graph satisfying $Rs(G) = \frac{m}{4} = \frac{8}{4} = 2$

Figure 6.3: Example of tight bounds for Corollary 6.16. The position depicted is one obtained after optimal moves from both players.

Proof. We prove both results by induction on $|V_F| = |V \setminus (V_L \cup V_R)|$, the number of free vertices. The result is clear if there are only two free vertices v_1 and v_2 as each player will claim one of them, and they will have the same number of neighbors in V_L at the end. Let $P = (G, V_L, V_R)$ be a position with $|V_F| \geq 3$, and let $v_1, v_2 \in V_F$ be equivalent vertices in P.

We first prove that $Ls(P) = Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. Let x be an optimal move for Left in P. If $x \in \{v_1, v_2\}$, we have $Ls(P) = Rs(G, V_L \cup \{v_1\}, V_R)$. Indeed, exchanging the roles of v_1 and v_2 is possible since they will score exactly the same number of points at the end. Using the recursive definition of the scores we have, $Rs(G, V_L \cup \{v_1\}, V_R) \leq Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. Otherwise, we have $Ls(P) = Rs(G, V_L \cup \{x\}, V_R)$. Vertices v_1 and v_2 are still equivalent in $(G, V_L \cup \{x\}, V_R)$. By induction, $Rs(G, V_L \cup \{x\}, V_R) = Rs(G, V_L \cup \{v_1, x\}, V_R \cup \{v_2\})$. According to the recursive definition of the score, $Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\}) \geq Rs(G, V_L \cup \{v_1, x\}, V_R \cup \{v_2\})$. Finally, in both cases, $Ls(P) \leq Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$.

We now prove the other inequality. Let x be an optimal move for Left in $(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. We have $Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\}) = Rs(G, V_L \cup \{v_1, x\}, V_R \cup \{v_2\})$. By induction, since v_1 and v_2 are still equivalent in $(G, V_L \cup \{x\}, V_R)$, we have $Rs(G, V_L \cup \{v_1, x\}, V_R \cup \{v_2\}) = Rs(G, V_L \cup \{x\}, V_R)$. Using the recursive definition of the score, $Ls(P) \geq Rs(G, V_L \cup \{x\}, V_R)$, which leads to $Ls(P) \geq Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. Finally, we have proved $Ls(P) = Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$.

We now turn to the proof of $Rs(P) = Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. Let x be an optimal move for Right in P. If $x \in \{v_1, v_2\}$, we have $Rs(P) = Ls(G, V_L, V_R \cup \{v_2\})$. Indeed, exchanging the roles of v_1 and v_2 is possible since they will score exactly the same number of points at the end. Using the recursive definition of the scores, we have $Ls(G, V_L, V_R \cup \{v_2\}) \geq Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. Otherwise, we have $Rs(P) = Ls(G, V_L, V_R \cup \{x\})$. Vertices v_1 and v_2 are still equivalent in $(G, V_L, V_R \cup \{x\})$. By induction, $Ls(G, V_L, V_R \cup \{x\}) = Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2, x\})$. According to the recursive definition of the score, $Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\}) \leq Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2, x\})$. Finally, in both cases, $Rs(P) \geq Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$.

We now prove the other inequality. Let x be an optimal move for Right in $(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. We have $Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\}) = Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2, x\})$. By induction, since v_1 and v_2 are still equivalent in $(G, V_L, V_R \cup \{x\})$, we have $Ls(G, V_L \cup \{v_1\}, V_R \cup \{v_2, x\}) = Ls(G, V_L, V_R \cup \{x\})$. Using the recursive definition of the score, $Rs(P) \leq Ls(G, V_L, V_R \cup \{x\})$, which leads to $Rs(P) \leq Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$.

Finally, we have proved $Rs(P) = Rs(G, V_L \cup \{v_1\}, V_R \cup \{v_2\})$. □

Note that this result is only true for equivalent vertices. In general, a good move for Left is not necessarily

a good move for Right. For instance, in Figure 6.4, if Left starts by playing u, the score is 4, and if she starts by playing any other vertex, the score is at most 3, thus her only optimal move is u. If Right starts by playing v, the score is 2, but if he starts by playing any other vertex, the score is at least 3. Hence, his only optimal move is v.

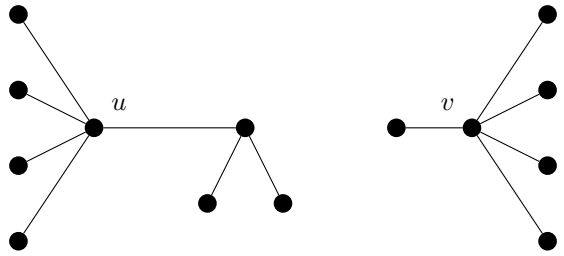


Figure 6.4: A graph G for which $Ls(G) = 4$ with unique optimal move u and $Rs(G) = 2$ with unique optimal move v.

The Super Lemma 6.18 is actually very useful to deal with similar vertices. We illustrate its power by computing the score for complete binary trees. A complete binary tree of depth k is a rooted tree such that each vertex at depth $j < k$ has exactly two children (and by definition of the depth, each vertex at depth k is a leaf).

Corollary 6.19. *Let T_k be a complete binary tree of depth $k \geq 1$. The scores in Maker-Breaker Incidence are $Ls(T_k) = 2^{k-1}$ and $Rs(T_k) = 2^{k-1} - 1$.*

Proof. Let T_k be a complete binary tree of depth k. Its leaves are pairwise equivalent. By the Super Lemma 6.18, we can assume that for any two leaves connected to a same vertex, one leaf can be given to Right, and the second one to Left. Then, the parents of the leaves have only one unclaimed neighbor and one neighbor claimed by Left. Therefore, any two parents of leaves with the same unclaimed neighbor are equivalent. Thus, we can again apply the Super Lemma 6.18 and assign one vertex of each pair to each player. By iterating this process from the leaves to the root, for any pair of vertices having the same parent, Maker and Breaker both get one of them. The game is then equivalent to the game where only the root is unclaimed, and thus the first player claims it. Finally, the number of edges taken by Maker satisfies $Ls(T_k) = Ls(T_{k-1}) + Rs(T_{k-1}) + 1$ (when Maker starts), and $Rs(T_k) = Ls(T_{k-1}) + Rs(T_{k-1})$ (when Breaker starts). Since $Ls(T_0) = Rs(T_0) = 0$, we get the result by induction. □

The application of Corollary 6.19 is depicted in Figure 6.5.

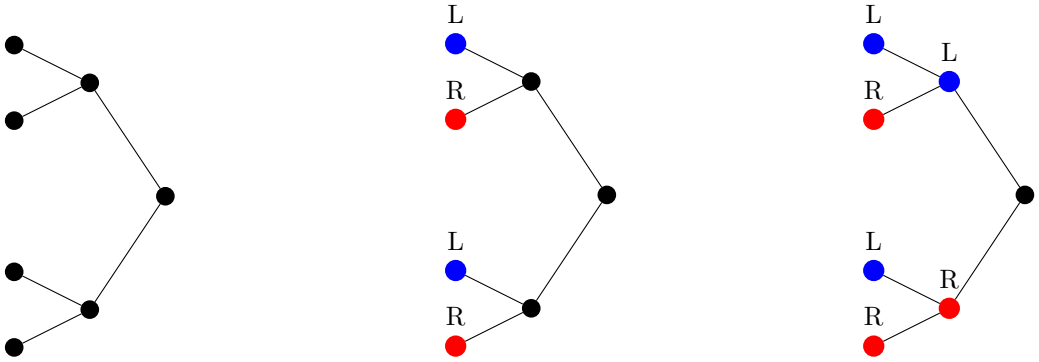
6.5 Complexity of Maker-Breaker Incidence

6.5.1 PSPACE-completeness of Incidence

Reductions in Maker-Breaker positional games are often made from POS CNF (as it was done in Chapter 3). In our cases, we need to deal with scores and not only a structure. To handle this problem, we use a quantified version of Max-2-SAT that we proved to be PSPACE-complete using 3-QBF.

Problem 6.20 (Quantified Max 2 SAT)**.**
Input : an integer k and a Quantified CNF Formula φ of the form $\varphi = \exists x_1 \forall x_2 \dots \exists x_{2n-1} \forall x_{2n} \bigwedge_{1 \leq i \leq m} C_i$ where each C_i contains at most 2 literals.
Output : true if at least k clauses in φ are satisfied, false otherwise.

Theorem 6.21. Quantified Max 2 SAT *is* PSPACE*-complete.*



(a) Apply the Super Lemma to the leaves

(b) We can now apply the Super Lemma to their parents.

(c) The resulting graph after application of the Super Lemma.

Figure 6.5: Application of the Super Lemma to compute the score in complete binary trees. Here $Ls(T_2) = 2$ and $Rs(T_2) = 1$

Proof. The proof of PSPACE-completeness of Quantified Max 2 SAT is similar to the proof of NP-completeness of Max-2-SAT from Papadimitriou [Pap94].

First, Quantified Max 2 SAT is in PSPACE, as any valuation can be computed in polynomial space. Therefore, by a min-max argument, it is possible to compute the number of satisfied clauses in polynomial space.

We provide a reduction from 3-QBF. Let $\phi = \exists x_1, \ldots, \forall x_n \quad \psi(x_1, x_2, ...x_n)$ be a 3-QBF formula on m clauses. For each clause $c_i = l_1^i \vee l_2^i \vee l_3^i$ of ψ, we introduce a new variable d_i and construct a set $\mathcal{C}_i$ of 10 clauses $C_i^1, \ldots, C_i^{10}$ of size at most 2:

$$\mathcal{C}_i = \{(l_1), (l_2), (l_3), (d_i), (\neg l_1 \vee \neg l_2), (\neg l_1 \vee \neg l_3), (\neg l_2 \vee \neg l_3), (\neg d_1 \vee l_1), (\neg d_1 \vee l_2), (\neg d_1 \vee l_3)\}$$

Claim 6.22. *Given any valuation of the literals l_i's, if c_i is satisfied, then there exists a valuation of d_i such that exactly seven clauses in $\mathcal{C}_i$ are satisfied. Otherwise, at most six clauses of $\mathcal{C}_i$ are satisfied for any valuation of d_i*

Proof of the claim. The proof of the claim is a case analysis depending on the number of literals l_i that are true in c_i (since the literals play a symmetric role). The following table gives the number N_C of clauses in $\mathcal{C}_i$ that are satisfied depending on the number N_L of literals l_i that are true and the valuation of d_i.

N_L	0	0	1	1	2	2	3	3
d_i	F	T	F	T	F	T	F	T
N_C	6	4	7	6	7	7	6	7

◇

Let $\varphi = \exists x_1, \ldots, \forall x_n, \exists d_1, \ldots, \exists d_n, \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{10} C_i^j$ and let $k = 7m$. Note that dummy variables can be added to preserve the alternation between $\exists$ and $\forall$ if needed.

If ϕ is true, then, for any valuation obtained by the Q_i's that makes ψ true, there exists a valuation for each d_j such that there are exactly seven clauses satisfied in each set $\mathcal{C}_j$. Thus, by taking this valuation for each d_j, we have that $k = 7m$ clauses satisfied in φ.

Reciprocally, if ϕ is false, then for any valuation provided by the Q_is, there exists a clause C_j that is not satisfied. Therefore, at most six clauses in $\mathcal{C}_j$ are satisfied. For the other clauses, at most seven of them are satisfied. Thus, the total number of satisfied clauses in φ is at most $7m - 1 = k - 1$.

Finally, the formula φ of Quantified Max 2 SAT has at least $7m$ clauses satisfied if and only if ϕ is True.

Up to adding a variable in all the clauses of size 1 and quantifying it with a $\forall$, we can suppose that all the clauses of φ have size 2. □

We now turn to the main proof of this section, that is the proof of the complexity of Maker-Breaker Incidence.

Theorem 6.23. *Determining whether Left can score at least k in Maker-Breaker Incidence is* PSPACE*-complete.*

The construction provided in the proof will require some tools to order the moves of both player. First we introduce the scoring version of Lemma 1.78: Let $P = (G, V_L, V_R)$ be a game position of Incidence. Let u and v be free vertices. We say that v *dominates* u in P and write $v \geq_P u$ if in any position obtained from P, it is always more interesting to play v than u. More formally, $v \geq_P u$ if for any V'_L, V'_R such that $V_L \subset V'_L$ and $V_R \subset V'_R$, $V'_L \cap V'_R = \emptyset$ and $u, v \notin V'_L \cup V'_R$, we have $Rs(G, V'_L \cup \{u\}, V'_R) \geq Rs(G, V'_L \cup \{v\}, V'_R)$ and $Ls(G, V'_L \cup \{u\}, V'_R) \leq Ls(G, V'_L \cup \{v\}, V'_R)$.

Lemma 6.24. *Let $G = (V, E)$ be a graph and $P = (G, V_L, V_R)$ a position of Maker-Breaker Incidence. Let u, v be two free vertices such that $|N(v) \cap V_L| \geq |N(u) \cap V_L| + |N(u) \setminus N(v) \cap V_F|$. Then $v \geq_P u$.*

Proof. Let $\mathcal{S}$ be a strategy in (G, V_L, V_R) that plays u before v. We define a strategy $\mathcal{S}'$ that plays v before u as follows:

- While $\mathcal{S}$ wants to claim a vertex $w \neq u$, claim w.
- If $\mathcal{S}$ wants to claim u while v is unclaimed, claim v instead, and still consider that u is claimed in $\mathcal{S}$.
- When $\mathcal{S}$ wants to claim v, if it is already claimed, claim u instead. If the opponent has claimed u, consider that he has claimed v, and continue to follow $\mathcal{S}$.

Following this strategy, according to the moves of the opponent, all the vertices claimed by $\mathcal{S}$ are claimed by $\mathcal{S}'$, with only a difference on u and v if they are not claimed by the same player.

If $\mathcal{S}$ was a strategy for Left, by following $\mathcal{S}'$, each edge that does not contain u nor v that was claimed by $\mathcal{S}$ is claimed by $\mathcal{S}'$, and reciprocally. Concerning the edges containing u or v, Left has scored at most $|N(u) \cap V_L| + |N(u) \cap V_F|$ points on them with $\mathcal{S}$ and $|N(v) \cap V_L| + |N(v) \cap V_F|$ by following $\mathcal{S}'$. Therefore, as $|N(v) \cap V_L| \geq |N(u) \cap V_L| + |N(u) \setminus N(v) \cap V_F|$, Left has score at least the same number of edges following $\mathcal{S}'$.

The same argument shows that Right will have more edges with a vertex claimed by him by playing v instead of u. □

Proof of Theorem 6.23. First, computing the score in Maker-Breaker Incidence is in PSPACE as the game last at most $|V|$ moves and the score is at most $|E|$. Thus, it can be computed in polynomial space, according to Section 6.1 in [HD09].

We prove that determining whether left can score k in Maker-Breaker Incidence is PSPACE-complete by a reduction from Quantified Max 2 SAT. In this proof, we consider a quantified formula as a two-player game. We first assume that the formula has the form $\exists x_{2n} \forall x_{2n-1} \exists x_{2n-2}, \dots \forall x_1 \ \psi$, i.e. that the quantifiers $\exists, \forall$ are alternating and starting with a quantifier $\exists$. This can be done for any quantified formulas by adding some vertices with the desired quantifier that are put in no clause, and thus that does not change the number of clauses that are satisfied. The first player, Satisfier, tries to satisfy the largest number of clauses by choosing the values of the even variables x_{2k} (i.e. that are quantified by an $\exists$-quantifier) while the second player, Falsifier, tries to spoil the formula and turn the largest number of clauses to false by choosing the values of the odd variables x_{2k-1} (i.e. that are quantified by a $\forall$-quantifier).

Denote $\psi = \bigwedge_{j=1}^{m} (l_1^j \vee l_2^j)$ for l_1^j, l_2^j some literals. We build a graph $G = (V, E)$ as follows (see Figure 6.6):

- For each variable x_i, we create $6mi + 3$ vertices. These vertices induce three stars of center v_i, $\overline{v_i}$ and $\widetilde{v_i}$, and with $2mi$ leaves each. We will denote by V_i the set $\{v_i, \overline{v_i}, \widetilde{v_i}\}$.

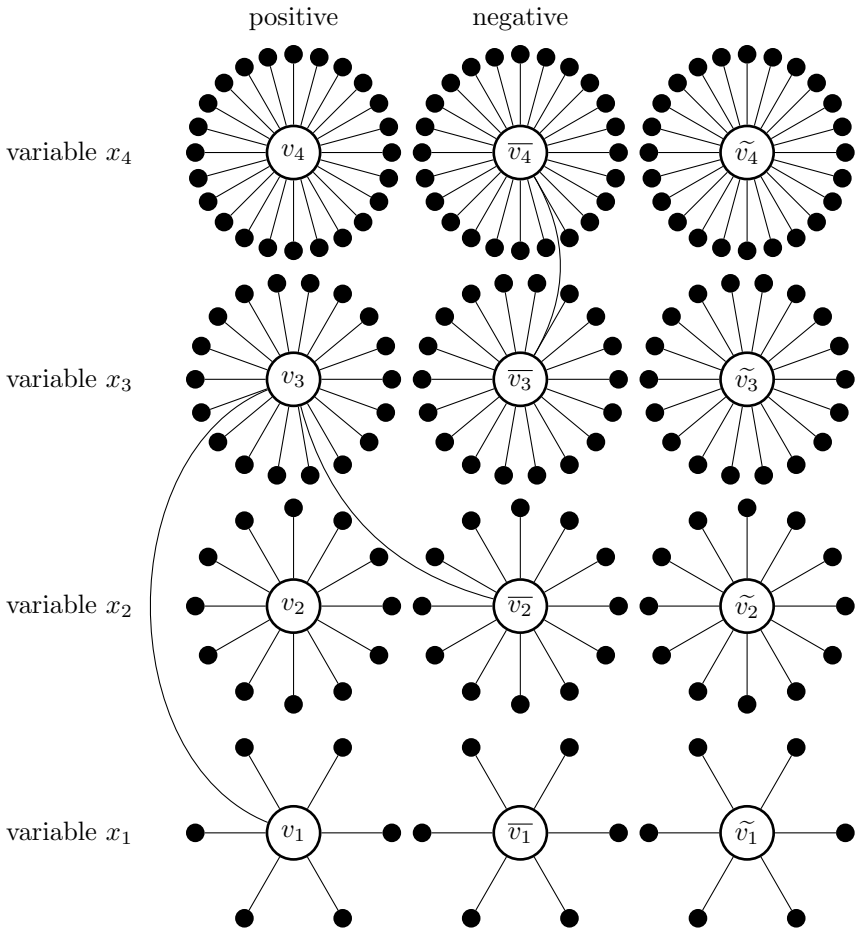


Figure 6.6: Reduction of $\exists x_4 \forall x_3 \exists x_2 \forall x_1 (\neg x_2 \vee x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_3 \vee \neg x_4)$

- We consider a function f defined by $f(x_i) = v_i$ and $f(\neg x_i) = \overline{v_i}$. For each clause $C_j = l_1^j \vee l_2^j$, we add an edge $e_j = (f(l_1^j), f(l_2^j))$.

The number of vertices outside sets V_i (i.e. the number of leaves) is $N = \sum_{i=1}^{2n} 6mi = 6mn(2n+1)$. Thus the total number of vertices in G is $N + 6n$ and the total number of edges is $N + m$, which is polynomial in the size of φ. An example of reduction is provided in Figure 6.6 with $m = 3$ and $n = 2$.

Consider a game of Maker-Breaker Incidence on G with Right starting. Using the Super Lemma 6.18, for every $1 \leq i \leq 2n$, the leaves connected to vertices v_i, $\overline{v_i}$ and $\widetilde{v_i}$ respectively, are equivalent. Thus, half of them can be given to Left and the other half to Right. Since there are an even number of leaves for each star, the only free vertices after this operation are the $6n$ vertices in sets V_i for $1 \leq i \leq n$. Let $P^0 = (G, V_L^0, V_R^0)$ be this position, and denote by V_F^0 the set of free vertices in this position. By the Super Lemma 6.18, we have $Rs(G) = Rs(P_0)$.

Now, if $1 \leq j < i \leq 2n$, for any $v_i^* \in V_i$ and $v_j^* \in V_j$, we have $|N(v_i^*) \cap V_L^0| = mi$, $|N(v_j^*) \cap V_L^0| = mj$ and $|N(v_j^*) \cap V_F^0| \leq m$. Therefore, by Lemma 6.24 we have $v_i^* \geq_{P_0} v_j^*$. Moreover, as $N(\widetilde{v_i}) \cap V_F^0 = \emptyset$, we also have $v_i \geq_{P_0} \widetilde{v_i}$ and $\overline{v_i} \geq_{P_0} \widetilde{v_i}$.

Hence, in any optimal strategy in P_0 with Right starting, the vertices are played in n rounds, from round $\ell = n$ to $\ell = 1$, with the following six steps in each round:

1. One vertex chosen by Right among $\{v_{2\ell}, \overline{v_{2\ell}}\}$

2. The other vertex among $\{v_{2\ell}, \overline{v_{2\ell}}\}$ is taken by Left.

3. The vertex $\widetilde{v_{2\ell}}$ is taken by Right.

4. One vertex among $\{v_{2\ell-1}, \overline{v_{2\ell-1}}\}$ is taken by Left.

5. The second vertex in $\{v_{2\ell-1}, \overline{v_{2\ell-1}}\}$ is taken by Right.

6. The vertex $\widetilde{v_{2\ell-1}}$ is taken by Left.

This way, Left will obtain exactly $N' = \sum_{\ell=1}^{n}(2\ell m + 2(2\ell-1)m) = 3mn(n+1) - 2mn$ edges in the stars and maybe some other edges in the clause edges. Let $k' = N' + m - k + 1$.

We will prove that $Rs(G) \geq k'$ in Maker-Breaker Incidence if and only if Falsifier wins on (φ, k) in Quantified Max 2 SAT.

Claim 6.25. *If Satisfier has a strategy to satisfy k clauses in φ, then $Rs(G) < k'$.*

Proof of the claim. We suppose that Satisfier has a winning strategy $\mathcal{S}$ in (φ, k). We consider that both Right and Left play optimally in G and thus we can assume that the game is played in P_0 and respects the previous order.

Consider the following strategy for Right. At each round ℓ from $\ell = n$ to $\ell = 1$, Right takes a decision only at Step 1. If Satisfier would turn x_{2i} to True in the game played on φ, then Right plays v_{2i}, otherwise, he plays $\overline{v_{2i}}$. Then, Steps 2 and 3 are determined. At Step 4, if Left plays v_{2i-1} then Right considers that Falsifier has turned x_{2i-1} to False, otherwise he considers she has turned it to True. Then again, Steps 5 and 6 are determined. By following this strategy, the underlying value obtained for φ is exactly the value that Satisfier would obtain by playing according to $\mathcal{S}$. Thus, at least k clauses are satisfied in φ.

Note that for a literal l^j, the vertex $f(l^j)$ is taken by Right if and only if l^j is True in the game of Quantified Max 2 SAT. Let $C_j = l_1^j \vee l_2^j$ be a clause. If Left has claimed the two extremities of e_j, it means that Left has played $f(l_1^j)$ and $f(l_2^j)$. Therefore, the underlying values of l_1^j and of l_2^j are both False, and C_j is not satisfied in ψ. Hence, Left claims at most $m - k$ edges e_j. Finally, Left claimed at most $k' - 1$ edges and we have $Rs(G) < k'$. ◇

Claim 6.26. *If Falsifier has a strategy such that at most $k-1$ clauses are satisfied in ϕ, then $Rs(G) \geq k'$.*

Proof of the claim. We now suppose that Falsifier has a winning strategy $\mathcal{S}$ in (φ, k). We consider that both Right and Left play optimally in G and thus we can assume that the game is played in P_0 and respects the previous order. Consider the following strategy for Left. At each round ℓ from $\ell = n$ to $\ell = 1$, Left takes a decision only at Step 4. At Step 1, if Right plays $v_{2\ell}$ then Left considers that Satisfier has turned $x_{2\ell}$ to True, otherwise she considers he has turned it to False. Then, Steps 2 and 3 are determined. At Step 4, if Falsifier would turn x_{2i-1} to False in the game played on φ, then Left plays v_{2i-1}, otherwise, she plays $\overline{v_{2i-1}}$. Then again, Steps 5 and 6 are determined.

By following this strategy, the underlying value obtained for φ is exactly the value that Falsifier would obtain by playing according to $\mathcal{S}$. Thus, it would satisfy at most $k-1$ clauses in φ. As before, if a clause $l_1^j \vee l_2^j$ is not satisfied in φ it means that both vertices $f(l_1^j)$ and $f(l_2^j)$ are taken by Left and thus Left got the edge. Thus, Left claims at least $N' + m - k + 1$ edges in the game G and $Rs(G) \geq k'$. ◇ □

Remark 6.27. *Note that up to adding a useless variable in φ, φ could start by a $\forall$-quantifier, implying that computing whether Left can score at least k in Maker-Breaker Incidence is* PSPACE*-complete even if Left starts.*

Corollary 6.28. *Computing whether Left can score at least k more than Right in Maker-Maker Scoring Positional Game is* PSPACE*-complete, even restricted to 3-uniform hypergraphs.*

Proof. The proof is similar to the proof of Corollary 1.65. From a graph $G = (V, E)$ of Maker-Breaker Incidence, we consider the instance of 3-uniform Maker-Maker Scoring Positional Game obtained by adding a universal vertex v_0. Consider the hypergraph $\mathcal{H} = (V \cup \{v_0\}, \{e \cup \{v_0\} | e \in E\})$. When Left starts, any optimal strategy starts by playing v_0, otherwise Right plays it and the score will be at most 0. Then we are left to a Maker-Breaker position as Right cannot score any point, but starts. Finally the Left score of $\mathcal{H}$ in Maker-Maker convention is equal to the Right score of G in Maker-Breaker convention, which is PSPACE-complete to compute. □

6.5.2 Complexity parameterized by the neighborhood diversity

As the game is PSPACE-hard, considering its parameterized complexity is a way to obtain positive results. The Super Lemma 6.18 enables us to know how equivalent vertices will be claimed. A particular case of equivalent vertices are vertices of the same type. We recall that two vertices u and v have the same *type* if $N(v) \setminus \{u\} = N(u) \setminus \{v\}$. This leads to an FPT algorithm for the neighborhood diversity.

Theorem 6.29. *Maker-Breaker Incidence parameterized by the neighborhood diversity w has a cubic kernel.*

The proof is divided into four steps. We first give the main idea behind each one:

1. We divide the graph into w sets of vertices of the same type, and we use the Super Lemma 6.18 to leave at most one vertex unplayed in each set.
2. As we are in the Maker-Breaker convention, we remove the vertices played by Right, as it is done in positional games (Lemma 1.74). We cannot remove Left's vertices because they may contribute to edges later, but we can remove edges claimed by Left by updating the score.
3. If some vertices have many neighbors already claimed by Left, they dominate the others, and we know that they will be played first. So, we remove some of their neighbors and update the score.
4. We replace the vertices claimed by Left with a fewest number that preserves the number of adjacent vertices already played for each unclaimed vertex.

Proof. In this proof, we will consider as instances of Maker-Breaker Incidence pairs (P, k) where P is a position of Maker-Breaker Incidence played on G (i.e. some vertices are already played) and k is the aimed score of Left. Note that this does not change the complexity of the problem. Indeed, from any position $P = (G, V_L, V_R)$ one can obtain a graph G' with no vertices played for which the games are equivalent. First remove all the vertices in V_R of the graph. Then, duplicate each vertex in V_L by creating a twin vertex having the same neighborhood and free the vertices in V_L. By Lemma 6.18, one can assume that both players will take one vertex in each pair of twins.

Let $G = (V, E)$ be a graph of neighborhood diversity w. Consider a partition $(V_1, \dots, V_w)$ of V such that the vertices in each part are all of the same type. We provide the following kernelization algorithm. Let $I = ((G, \emptyset, \emptyset), k)$ be an instance of Maker-Breaker Incidence. An example of the different steps is provided in Figure 6.7.

Step 1: While there exists a part V_i, $1 \leq i \leq w$ such that there are at least two free vertices $u, v \in V_i$, add u to V_L and v to V_R. By Lemma 6.18, this transformation does not change the outcome of the game. At the end of Step 1, there are at most w free vertices in G. In Figure 6.7b, it consists in distributing vertices of same type between Left and Right.

Step 2: Remove all the edges included in V_L and set $k \leftarrow k - |e \subset V_L|$. Then remove from G all the vertices in V_R that cannot count for any point. This transformation do not change the outcome of I. At this moment, G only contains free vertices or vertices claimed by Left, and any edge has at least one free extremity. In Figure 6.7c, it consists in removing the 16 edges on which the two endpoints are claimed by Left, and to remove the red vertices and their incident edges. Therefore, k is decreased from 30 to 14.

Step 3: Let r the number of free vertices in P, we have $r \leq w$. Let $v_1, \dots, v_r$ be these vertices. For $1 \leq i \leq r$, let $p_i = |N(v_i) \cap V_L|$ and order the vertices such that $p_1 \geq p_2 \geq \cdots \geq p_r$. While there exists

an integer i such that $p_i > p_{i+1} + r$ (with $p_{r+1} = 0$), by Lemma 6.24, there exists an optimal strategy in which the vertices $v_1, \dots, v_i$ are played before the vertices $v_{i+1}, \dots, v_r$. On these vertices, Left will score at least p_i at each Left move. Therefore, we can do the following transformation. Let $s = p_i - p_{i+1} - r$ for any $1 \leq j \leq i$, set $p_j \leftarrow p_j - s$ and set $k \leftarrow k - s\left\lceil \frac{i}{2} \right\rceil$. Repeat Step 3 until we have $p_i \leq p_{i+1} + r$ for all $1 \leq i \leq r$. In particular, we have after these operations $p_1 \leq r^2$. In Figure 6.7d, it happens only once, as $p_1 = 8$, $p_2 = 3$ and $w' = 4$. Therefore, we set $p_1 = 7$ and k is decreased from 14 to 13.

Step 4: Let $U = \{u_1, \dots, u_{p_1}\}$ be p_1 new vertices and transform $(G, V_L, \emptyset)$ into $((G \setminus V_L) \cup U, U, \emptyset)$, and, for $1 \leq i \leq r$, connect the vertex v_i to any p_i vertices in U. This transformation do not change the outcome of the game, since only the number of neighbors in V_L matters when a vertex is played. In Figure 6.7e, we have $p_1 = 7$. Thus, U contains seven vertices and each remaining uncolored vertex v_i is connected to p_i of these seven vertices.

Finally, if $k \geq r^3$, as there are at most r^3 edges in the final graph, we can just transform P into a trivial False instance like the empty graph with $k = 1$. Thus, we can assume that $k \leq r^3$.

The instance obtained has $p_1 + r \leq r^2 + r \leq w^2 + w$ vertices, at most $r * p_1 \leq r^3 \leq w^3$ edges, $k \leq r^3 \leq w^3$ and the same outcome as the input. Finally, this new instance has cubic size in w and thus Maker-Breaker Incidence has a cubic kernel. □

Corollary 6.30. *Let G be a graph of order n and neighborhood diversity w. In Maker-Breaker Incidence $Ls(G)$ and $Rs(G)$ can be computed in time $O(w^2 w! + n^2)$*

Proof. We can compute the kernel in time n^2, and then try all the possible games by testing all the moves in time $w^2 w!$. □

Note that the cubic size of the kernel is mostly due to the w^2 vertices that are already claimed by Left. As these vertices cannot be played any longer, by giving weight to the vertices, it is possible to have a quasilinear kernel by storing only the number of neighbors of each vertex that are already claimed by Left instead of vertices themselves.

6.6 Paths and cycles

We here give the exact values of the score for Maker-Breaker Incidence played on paths and cycles. For that purpose, we will consider the equivalence properties of Milnor's universe detailed in Section 2. In particular, the notion of negative will be required, implying to consider the partisan version of Incidence. More precisely, in this section, instances of Maker-Breaker Incidence will correspond to paths or cycles where the edges are either colored all blue (i.e. only Left can get points) or all red (i.e. only Right can get points). The notations are defined as follows:

- P_n^L: path of order n where all the edges are colored blue. We denote the vertices of P_n^L by $\{v_0, \dots, v_{n-1}\}$
- P_n^R: path of order n where all the edges are colored red. We denote the vertices of P_n^R by $\{v'_0, \dots, v'_{n-1}\}$

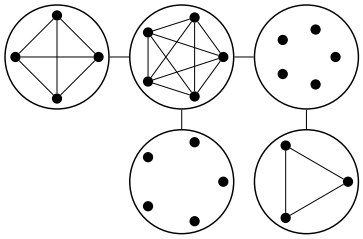
By definition, we have that $P_n^L = -P_n^R$.

6.6.1 Equivalences of paths

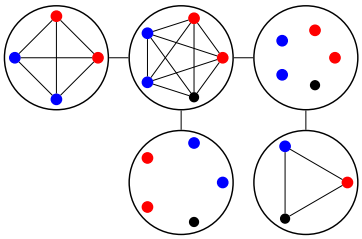
We first give the main result about the equivalence between paths modulo 5. We recall that, for $k \in \mathbb{Z}$, we define by k the game with no option and where Left has a score of k points. Thus, in Maker-Breaker Incidence, the game 1 is equivalent to P_2^L in which Left has claimed the two vertices and -1 is equivalent to P_2^R in which Right has claimed the two vertices. The main theorem of this section states that paths of order at least 6 are equivalent to paths having five vertices less, with a difference of one in the score. This result remains true if an extremity of the path is already colored.

Theorem 6.31. *Let $n \geq 1$ be an integer. We have $P_{n+5}^L \equiv P_n^L + 1$ and $P_{n+5}^R \equiv P_n^R - 1$.*
Let $n \geq 2$ be an integer. We have $(P_{n+5}^L, \{v_0\}, \emptyset) \equiv (P_n^L, \{v_0\}, \emptyset) + 1$ and $(P_{n+5}^R, \emptyset, \{v'_0\}) \equiv (P_n^R, \emptyset, \{v'_0\}) - 1$.

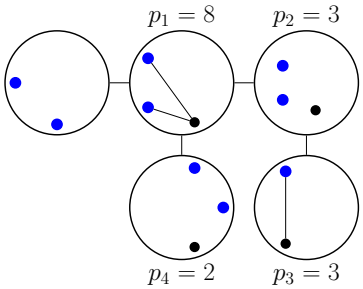
The rest of this subsection will be dedicated to the proof of this theorem.



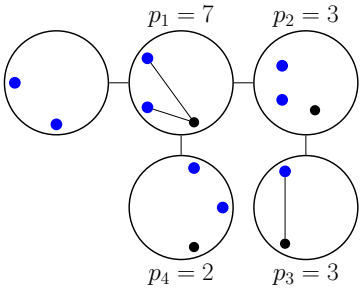
(a) A graph to kernelize. Set $k = 30$.



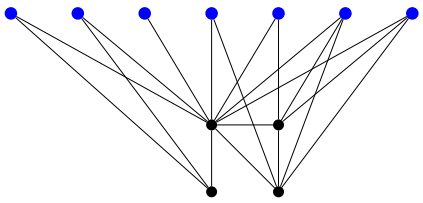
(b) Step 1, $k = 30$. Pairwise equivalent vertices are distributed between Left and Right.



(c) Step 2, 16 edges removed, $k = 30 - 16 = 14$. The Vertices claimed by Right are removed.



(d) Step 3, the unclaimed vertex in p_1 dominates the other. p_1 has decreased by 1. $k = 13$.



(e) Step 4, each vertex v_i has p_i blue neighbors.

Figure 6.7: Example of a kernelization. Vertices in the same circle have same type. An edge between two circles means that all the edges between the vertices of the two circles are in the graph. Blue and red vertices are given to Left and Right respectively. We start with $n = 22$ and after Step 1 $r = 4$.

Strategy for Left when Right starts

Lemma 6.32. *Let $n \geq 1$ be an integer. In Maker-Breaker Incidence, we have $Rs(P^L_{n+5} + P^R_n) \geq 1$.*
Let $n \geq 2$ be an integer. In Maker-Breaker Incidence, we have $Rs(P^L_{n+5} + P^R_n, \{v_0\}, \{v'_0\}) \geq 1$.

This proof will be done by induction. Therefore, to handle the small cases, the scores of first paths will be required. They are recorded in Figure 6.8 and Figure 6.9 and can be easily checked by hand.

n	1	2	3	4	5	6	7	8	9	10
$Ls(P^L_n)$	0	0	1	1	1	1	1	2	2	2
$Rs(P^L_n)$	0	0	0	0	0	1	1	1	1	1

Figure 6.8: First scores in short paths

n	1	2	3	4	5	6	7	8	9	10	11
$Ls((P^L_n, \{v_0\}, \emptyset))$	0	1	1	1	1	2	2	2	2	2	3
$Rs((P^L_n, \{v_0\}, \emptyset))$	0	0	0	0	1	1	1	1	1	2	2

Figure 6.9: First scores in short paths with an extremity claimed by Left

Proof. In order to prove that $Rs(P^L_{n+5} + P^R_n) \geq 1$ (resp. $Rs(P^L_{n+5} + P^R_n, \{v_0\}, \{v'_0\}) \geq 1$), we provide a strategy for Left by induction. An example of move is provided in Figure 6.10 If $1 \leq n \leq 5$ (resp. $2 \leq n \leq 6$), a computation can verify that the result is true.

If $n \geq 6$ (resp. $n \geq 7$), we consider the first move of Right:

- If Right plays a vertex v'_i for $0 \leq i \leq n-1$ (resp. $1 \leq i \leq n-1$), Left answers by playing the vertex v_i. The resulting position is $(P^L_{n+5} + P^R_n, \{v_i\}, \{v'_i\})$ (resp. $(P^L_{n+5} + P^R_n, \{v_0, v_i\}, \{v'_0, v'_i\})$), which is equivalent to $(P^L_{i+1} + P^R_{i+1}, \{v_i\}, \{v'_i\}) + (P^L_{n+5-i} + P^R_{n-i}, \{v_0\}, \{v'_0\})$ (resp. $(P^L_{i+1} + P^R_{i+1}, \{v_0, v_i\}, \{v'_0, v'_i\}) + (P^L_{n+5-i} + P^R_{n-i}, \{v_0\}, \{v'_0\})$). As we have $(P^L_{i+1} + P^R_{i+1}, \{v_i\}, \{v'_i\}) \equiv 0$ (resp. $(P^L_{i+1} + P^R_{i+1}, \{v_0, v_i\}, \{v'_0, v'_i\}) \equiv 0$) and $(P^L_{n+5-i} + P^R_{n-i}, \{v_0\}, \{v'_0\})$ satisfies the induction hypothesis, and therefore the score is at least one.

- If Right plays a vertex v_i for $0 \leq i \leq n-1$ (resp. $1 \leq i \leq n-1$), Left answers by playing the vertex v'_i. The resulting position is $(P^L_{n+5} + P^R_n, \{v'_i\}, \{v_i\})$ (resp. $(P^L_{n+5} + P^R_n, \{v_0, v'_i\}, \{v'_0, v_i\})$), which is equivalent to $(P^L_i + P^R_i) + (P^L_{n+5-(i+1)} + P^R_{n-(i+1)})$ (resp. $(P^L_i + P^R_i, \{v_0\}, \{v'_0\}) + (P^L_{n+5-(i+1)} + P^R_{n-(i+1)})$). As we have $(P^L_i + P^R_i) \equiv 0$ (resp. $(P^L_i + P^R_i, \{v_0\}, \{v'_0\}) \equiv 0$) and $(P^L_{n+5-(i+1)} + P^R_{n-(i+1)})$ satisfies the induction hypothesis, the score is at least one.

- If Right plays a vertex v_i for $n \leq i \leq n+4$. Left answers by playing v'_{i-5}, which exists as $n \geq 6$ (resp. $n \geq 7$). The resulting position is $(P^L_{n+5} + P^R_n, \{v'_{i-5}\}, \{v_i\})$ (resp. $(P^L_{n+5} + P^R_n, \{v_0, v'_{i-5}\}, \{v'_0, v_i\})$), which is equivalent to $(P^L_i + P^R_{i-5}) + (P^L_{n-1-i} + P^R_{n-1-i})$ $((P^L_i + P^R_{i-5}, \{v_0\}, \{v'_0\}) + (P^L_{n-1-i} + P^R_{n-1-i})$. Here, we have $(P^L_{n-1-i} + P^R_{n-1-i}) \equiv 0$ and $(P^L_i + P^R_{i-5})$ (resp. $(P^L_i + P^R_{i-5}, \{v_0\}, \{v'_0\})$) satisfies the induction hypothesis as $n+4 \geq i \geq n \geq 6$ (resp. $i \geq n \geq 7$) and therefore the score is at least one.

This strategy ensures that $Rs(P^L_{n+5} + P^R_n) \geq 1$ (resp. $Rs(P^L_{n+5} + P^R_n, \{v_0\}, \{v'_0\}) \geq 1$). □

Strategy for Right when Left starts

When Left starts, the induction made in the previous proof cannot be applied. Indeed, from the position $(P^L_{n+5} + P^R_n, \{v_0\}, \{v'_0\})$, Left can in one move make the position be $(P^L_{n+5} + P^R_n, \{v_0, v_{n+3}\}, \{v'_0\})$ and no move of right can transform it into a position handled by the induction hypothesis. Therefore, another strategy is required. We will consider a strategy for Right that consists, for the leftmost vertices of both

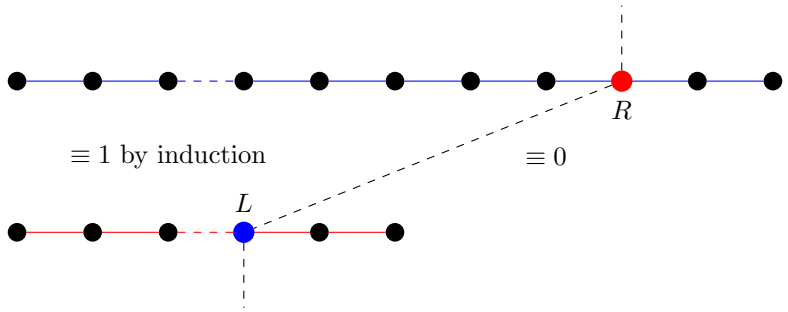


Figure 6.10: Example of answer for Left. Only Left can score on the path at the top. Only Right can score on the path at the bottom. Here, Right played v_{n+2}, and Left answered v'_{n-3}

paths, in mimicking any move of Left on the other path, and that ensures some minimal properties on the moves played on the rightmost vertices. We introduce the following lemma to handle the rightmost vertices.

Lemma 6.33. *Consider the graph $G = P_6^L + \{v'_0\}$. Let v_0 be an extremity of P_6^L. In Maker-Breaker Incidence, Right has a strategy, going second, such that Left claims either v_0 and v'_0 without any point, or at most one of $\{v_0, v'_0\}$ and she scores at most one point on G.*

Proof. Let $G = P_6^L + \{v'_0\}$. This graph represents the seven vertices on the left of the dashed line in Figure 6.11. Recall that $v_0, \dots, v_5$ are the vertices of P_6^L. We will describe a strategy for Right playing second such that Left scores no point or such that she does not claim both v_0 and v'_0 with at most one point.

- If Left plays v_0, Right answers v_1,
  - if Left plays v'_0, Right plays v_3 and pairs v_4 and v_5. Left cannot score a point.
  - If Left plays v_2 (resp. v_5), Right plays v_3 (resp. v_4) and pairs (v_4, v_5) (resp. (v_2, v_3)). This way, Left cannot score a point.
  - If Left plays in v_3 (resp. v_4), Right plays v_4 (resp. v_3) and pairs v_2 (resp. v_5) with v'_0. Either Left scores a point or claims both v_0 and v'_0.
- If Left plays v_1, Right answers v_0. He has claimed one of (v_0, v'_0). He then pairs (v_2, v_3) and (v_4, v_5). With this pairing, Left can score at most one point.
- If Left plays v_2, Right answers v_3. He then pairs $(v_0, v_1$ and $v_4, v_5)$. The only one edge outside the pairing (and therefore that can be claimed by Left) is v_1, v_2 but with this pairing, Right then plays v_0 and claim one of v_0, v'_0. Otherwise, Left scores no point.
- If Left plays v_3 (resp. v_5), Right answers v_4, he then pairs (v_0, v'_0) and (v_1, v_2). This way, Left scores at most one point on the edge (v_2, v_3) or (v_0, v_1) but she cannot take both. And Right will be able to take one of v_0 or v'_0.
- If Left plays v_4, Right answers v_3
  - If Left plays v_0, Right plays v_1 and pairs v'_0 with v_5. Either Left claims v'_0, and then by claiming v_5, Right ensures that Left scores no point, or Left claims v_5 and scores one point, but Right claims $v'_0 \in \{v_0, v'_0\}$.
  - If Left plays v_1 (resp. v_2), Right plays v_0 and pairs v_2 (resp. v_1) and v_5. By claiming one of them, Left scores one point but Right claims the second one, and therefore, Right ensures that Left scores only one point and does not claim both v_0 and v'_0.

– If Left plays v_5 (resp. v'_0), Right plays v_0 and pairs (v_1, v_2). Then, Left cannot score a second point (resp. can score at most one point by playing v_5), and Right has already claimed one of v_0, v'_0.

- If Left plays v'_0, Right answers v_0. He has already claimed one of v_0, v'_0, and the remaining graph is equivalent to P_5^L for which we already know that Left gets at most 1 when she starts.

□

Lemma 6.34. *Let $n \geq 1$ be an integer. In Maker-Breaker Incidence, we have $Ls(P_{n+5}^L + P_n^R) \leq 1$.*
Let $n \geq 2$ be an integer. In Maker-Breaker Incidence, we have $Ls(P_{n+5}^L + P_n^R, \{v_0\}, \{v'_0\}) \leq 1$.

Proof. The proof below holds for the two cases, i.e. if the vertices v_0 and v'_0 are already colored or not.

Recall that $v_0, \ldots, v_{n+4}$ are the vertices of P_{n+5}^L and $v'_0, \ldots, v'_{n-1}$ are the vertices of P_n^R. We provide here a strategy for Right to ensure that the score is at most 1 as follows:

- If Left plays a vertex in a pair (v_i, v'_i) with $0 \leq i \leq n-2$, Right answers the second vertex of this pair.
- If Left plays another vertex, Right follows the strategy of Lemma 6.33 with $P_6^L = \{v_0 = v_{n-1}, \ldots, v_{n+4}\}$ and $v'_0 = v'_{n-1}$.

This strategy is depicted in Figure 6.11

According to this strategy, Right ensures that Left scores the same number of points as him on the subgraph induced by the vertices v_i, v'_i with $0 \leq i \leq n-2$. On the rest of the graph, from Lemma 6.33, either Left takes the two vertices v_0, v'_0 and gets no point, which can yield her overall at most one point with the edge (v_{n-2}, v_{n-1}) of P_{n+5}^L. Otherwise, she takes v'_0 or the extremity v_0 of the P_6^L and scores one point. In this case, if this extremity corresponds to v'_{n-1} of P_n^R she does not score a second point, and if this extremity is v_{n-1}, she can score a point if she also takes v_{n-2}. But in this case, Right has claimed both v'_{n-2} by the pairing strategy and v'_{n-1} as he has also claimed the other extremity. Thus, Right also scores one point. Finally, Right ensures that the score is at most 1 with this strategy, and we have $Ls(P_{n+5}^L + P_n^R) \leq 1$. □

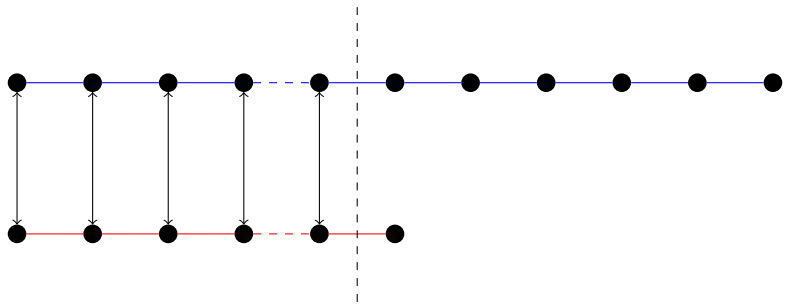


Figure 6.11: Only left can score on the path at the top, and only Right can score on the path at the bottom. We present a strategy for Right to ensure that the score is at most one. On the left of the dashed line, the pairing is depicted ensuring a zero score. On the right, Lemma 6.33 ensures that Right either claims the two extremities connected to the left part, or scores one with at most one of the two extremities, ensuring a score at most one.

Proof of Theorem 6.31 and score on paths

Now we can prove Theorem 6.31.

Proof. By symmetry, as $P_n^L = -P_n^R$ for any n, we only need to prove the result for P_n^L.

As our game is in Milnor's universe, according to Lemma 6.2, it is sufficient to prove that $P_{n+5}^L - P_n^L - 1 \equiv 0$ (resp. $(P_{n+5}^L - P_n^L, \{v_0\}, \{v_0'\}) - 1 \equiv 0$), i.e. $Ls(P_{n+5}^L + P_n^R) = Rs(P_{n+5}^L + P_n^R) = 1$ (resp. $Ls(P_{n+5}^L + P_n^R, \{v_0\}, \{v_0'\}) = Rs(P_{n+5}^L + P_n^R, \{v_0\}, \{v_0'\}) = 1$).

As the game is nonzugzwang, and according to Lemma 6.32 and Lemma 6.34, we have proven $1 \geq Ls(P_{n+5}^L + P_n^R) \geq Rs(P_{n+5}^L + P_n^R) \geq 1$ (resp. $1 \geq Ls(P_{n+5}^L + P_n^R, \{v_0\}, \{v_0'\}) \geq Rs(P_{n+5}^L + P_n^R, \{v_0\}, \{v_0'\}) \geq 1$), which corresponds to the desired result. □

From Theorem 6.31, and since the score on small paths is provided by Figure 6.8, the score of any path can be computed as follows:

Corollary 6.35. *Let $n \geq 1$ be an integer. Denote by $n = 5q + r$ with q and r the quotient and the rest of n divided by 5. In Maker-Breaker Incidence, on the one hand, we have $Ls(P_n^L) = -Rs(P_n^R) = q$ if $0 \leq r \leq 2$, and $Ls(P_n^L) = -Rs(P_n^R) = q + 1$ if $3 \leq r \leq 4$. On the other hand, we have $Rs(P_n^L) = -Ls(P_n^R) = q - 1$ if $r = 0$, $Rs(P_n^L) = -Ls(P_n^R) = q$ if $1 \leq r \leq 4$.*

6.6.2 Union of paths and cycles

We will denote cycles as follows:

- C_n^L: cycle of length n where all the edges are colored blue.
- C_n^R: cycle of length n where all the edges are colored red.

Now that the equivalences of paths are known, union of paths can easily be reduced to union of paths of order at most 5. Yet, to deal with unions of paths, it is not sufficient in general to compute the score on each of them, since the score on a union is not in general the sum of the scores. The problem can be solved by considering new equivalences between small paths.

Lemma 6.36. *In Maker-Breaker Incidence, we have the following equivalences:*

$$P_1^L \equiv P_2^L \equiv 0 \tag{6.1}$$

$$2P_3^L \equiv 1 \tag{6.2}$$

$$P_4^L \equiv P_3^L \tag{6.3}$$

$$2P_5^L + P_3^L \equiv 2 \tag{6.4}$$

Proof. Recall that given a graph G and an integer k, in order to prove that $G \equiv k$, it is sufficient to prove $k \geq Ls(G)$ and $Rs(G) \geq k$.

1. We have $Ls(P_1^L) = Rs(P_1^L) = 0$ and $Ls(P_2^L) = Rs(P_2^L) = 0$ as in both games no edges are taken by a player. This proves, by Lemma 6.2, that $P_1^L = P_2^L = 0$
2. We prove $Ls(2P_3^L) = Rs(2P_3^L) = 1$. To do that, we first apply the Super Lemma 6.18 on the two extremities of each path. Only their middle vertices are now unclaimed, thus Left will claim one and Right the other one, proving $Ls(2P_3^L) = Rs(2P_3^L) = 1$.
3. As $-P_3^L = P_3^R$, we will prove $P_4^L + P_3^R = 0$. Denote by (v_0, v_1, v_2, v_3) the vertices of P_4^L and by (v_0', v_1', v_2') the vertices of P_3^R
   - Suppose Left starts. If she plays in P_3^R, Right plays v_1 and pairs (v_2, v_3) to ensure that Left cannot score an edge. If Left plays v_0 or v_1 (resp. v_2 or v_3), Right plays v_2(resp. v_1) and pairs (v_0', v_2') and v_1' with the available vertex in $\{v_0, v_1\}$ (resp. in $\{v_2, v_3\}$). This way, Left and Right scores the same number of edges and this proves $Ls(P_3^R + P_4^L) \leq 0$
   - Suppose Right starts. Left considers the pairing (v_0, v_0'), (v_1, v_1'), (v_2, v_2'). This way, any point scored by Right is scored by Left. Therefore $Rs(P_4^L + P_3^R) \geq 0$.

4. Let $G = 2P_5^L + P_3^L$. Denote by $v_0, \dots, v_4$ and $v'_0, \dots, v'_4$ the vertices of the two copies of P_5^L and by (u_0, u_1, u_2) the vertices of P_3^L. Let first prove $Rs(G) \geq 2$. Up to consider only 3 vertices of one copy of P_5^L, we can suppose that the first move of Right is in a P_5^L and we will prove that Left scores 2 on $P_5^L + P_3^L$. Suppose Right has played a vertex v'_i with $0 \leq i \leq 4$. Left plays v_2 and continues as follows:

   - If Right plays v_0 or v_1 (resp. v_3 or v_4), Left plays v_3 (resp. v_1) and pairs (v_4, u_1) (resp. (v_0, u_1)) and (u_0, u_2).
   - If Right plays u_0, u_1 or u_2, Left plays v_1 and pairs (v_0, v_3).

   In both cases, Left scores at least two points. Now we prove $Ls(G) \leq 2$. After the first move of Left, at least one of the two copies of P_5^L has its 5 vertices available. Suppose it is $v'_0, \dots, v'_4$. Right plays v'_2 and pairs (v'_0, v'_1) and (v'_3, v'_4), ensuring Left won't score any point on this copy of P_5^L. Left plays a second move:

   - If v_2 has not been played yet, Right plays v_2. Left plays a third move. If the three moves of Left are in P_3^L, Right pairs (v_0, v_1) and (v_3, v_4), ensuring Left does not score any other point. If at least one of them is not in P_3, Right plays any vertex of P_3, and know that at least one vertex of $(v_0, v_1, v_3, v_4, u_0, u_1, u_2)$ will be available for his next move. Thus, Left cannot score more than two points on the rest of them.
   - If Left has played v_2, at least one of v_1 or v_3 is available. Right plays it. By symmetry, suppose it is v_1. After the next move of Left, at least one of v_3, v_4, u_1 will be available. Right plays it, ensuring again that Left cannot score more than 2.

□

We can now state the equivalence theorem for union of paths.

Corollary 6.37. *Let $P_1, \dots, P_N$ be paths of lengths $n_1, \dots, n_N$.*

Let $q_1, \dots, q_N$ be positive integers and $1 \leq r_1, \dots, r_N \leq 5$ be integers such that for any $1 \leq i \leq N$, we have $n_i = 5q_i + r_i$.

Denote for $1 \leq i \leq 5$ by N_i the number of r_j equal to i. In Maker-Breaker Incidence, we have:

$$\sum_{i=1}^{N} P^L_{5q_i+r_i} \equiv \sum_{i=1}^{N} q_i + \left\lfloor \frac{N_3 + N_4}{2} \right\rfloor + 3 \left\lfloor \frac{N_5}{4} \right\rfloor + (N_3 + N_4 \mod 2)P_3 + (N_5 \mod 4)P_5$$

Therefore, $Ls(\sum_{i=1}^{N} P_i)$ and $Rs(\sum_{i=1}^{N} P_i)$ are computable in linear time.

Proof. By Theorem 6.31, any path $P^L_{5q_i+r_i}$ is equivalent to $q_i + P_{r_i}$. Then, by Lemma 6.36, we have $P_3 \equiv P_4$, $2P_3 \equiv 1$, and $2(2P_5 + P_3) \equiv 4P_5 + 2P_3 \equiv 4P_5 + 1 \equiv 4$. Thus $4P_5 \equiv 3$. Note that these computations are possible thanks to Milnor's universe. □

Note that we consider $1 \leq r_i \leq 5$ and not $0 \leq r_i \leq 4$, so q_i and r_i are not exactly the quotient and the rest of the size of the path by 5.

Corollary 6.38. *Let $n \geq 1$. In Maker-Breaker Incidence, there exists a linear time algorithm to compute $Ls(C_n^L)$ and $Rs(C_n^L)$.*

Proof. First, note that $Rs(C_n^L) = Ls(P_{n-1}^L)$.

To compute $Ls(C_n^L)$, note that all the vertices are symmetric. Therefore, we can suppose that Left first plays any of them. The next move of Right will make the graph equivalent to $(P_k^L, \{v_0\}, \emptyset) + (P_{k'}^L, \{v'_0\}, \emptyset)$ with v_0, v'_0 extremities of P_k^L and $P_{k'}^L$ and with $k + k' = n$. The score on these graphs can be computed in linear time by using Corollary 6.37, and therefore, $Ls(C_n^L)$ too as, by Theorem 6.31, at most 5 values are to be considered for the pair (k, k') according to the equivalences. □

6.7 Further work

In this chapter, we introduced scoring positional games as a general framework and then focused on Incidence, which corresponds to the case of 2-uniform hypergraphs. As we just opened a new area of studies, several questions are relevant. We have solved Maker-Breaker Incidence on union of paths using game equivalences, as a generalization of this result, one can look for a polynomial algorithm on trees and forests.

Focusing on scoring positional games in general, and not just Incidence, we have proved that computing the score in Maker-Maker Scoring Positional Game is PSPACE-complete even for 3-uniform hypergraphs but provided a linear algorithm for 2-uniform hypergraphs. It might be interesting to look at particular 3-uniform hypergraphs. For example, is it possible to compute the score in the scoring version of the Triangle game (where players choose edges of a graph and try to construct triangles)? The hypergraph of this game has the particularity to be linear. A more general question would be to find the complexity of computing the score in Maker-Maker Incidence on linear 3-uniform hypergraphs.

In terms of parameterized complexity, we proved that computing the score in Maker-Breaker Incidence is fixed-parameter tractable using the neighborhood diversity. It would be interesting to find other parameters for which the problem is FPT. For example, we expect some FPT algorithm parameterized by the modular-width with similar arguments to the one used in Chapter 4. Moreover, the most natural parameter in positional games is the number of moves, and the most natural parameter in scoring game is the score. The study of these two parameters on scoring positional games would make sense and seems to be the next step of the parameterized study of Incidence.

Conclusion

The end.

In this thesis, I have presented several results on the complexity of positional games, both on general positional games, and on specific ones. I summarize here the main open problems that could be the next step of the studies.

As far as I am concerned, the study of the conventions of positional games, other than Maker-Breaker, is probably the one that deserves more attention. We recall in Table 6.1 the different results obtained about the complexity of the different conventions, but the complexities of some conventions are still unknown. Among the classes for which no results are known yet, I think that the most affordable are 3-uniform Waiter-Client and Client-Waiter, which should be solvable in polynomial time.

We also recall that most of the community conjectures that 4- and 5-uniform Maker-Breaker games to be PSPACE-complete (see Conjecture 1.67). This conjecture can be extended to Avoider-Enforcer and Client-Waiter as the reduction provided for Theorem 2.1 and Theorem 2.10 uses only one gadget hyperedge of rank 6, the others being of rank 4.

On the positive side, on particular games, as in Chapter 5, we proved that the Domination game is polynomial in trees even in the Maker-Maker convention, it would be natural to consider this problem under the conventions Avoider-Enforcer or Waiter-Client. To the best of my knowledge, the only results known about them are in Gledel's thesis [Gle19].

Rank	2	3	4, 5	6	7+
Maker-Breaker	P [Folklore]	P [Gal23]	Open	PSPACE-c [RW21]	PSPACE-c [RW21]
Maker-Maker	P [Folklore]	Open	Open	Open	PSPACE-c [RW21, Bys04]
Avoider-Avoider	PSPACE-c [BH19]	PSPACE-c [BH19]	PSPACE-c [BH19]	PSPACE-c [BH19]	PSPACE-c [BH19]
Avoider-Enforcer	P [Folklore]	Open	Open	PSPACE-c Thm 2.1	PSPACE-c Thm 2.1
Client-Waiter	P Thm 1.69	Open	Open	PSPACE-c Thm 2.10	PSPACE-c Thm 2.10
Waiter-Client	P Thm 1.70	Open	Open	Open	Open

Table 6.1: The complexity of the different conventions

In Chapter 3, we started the study of the complexity of classical games on the edges graphs. This ongoing project still has a lot of questions open. First, the complexity of the hamiltonicity game in graphs is still unknown, even if I conjectured that it is PSPACE-complete. Related to the H-game, the two natural directions to consider would be the path game and the clique game.

From the point of view of parameterized complexity, the Super Lemma was a powerful tool that allowed to prove several results. You probably guessed it, but for me this lemma is the most important result of my thesis, because of its numerous applications, and its simplicity: if two vertices are twins, Maker and Breaker will each claim one in an optimal strategy. This lemma makes it possible to handle symmetries of the graph, and we can wonder if a similar application of this lemma might occur in non-positional games, or even more, in other graph theory problems. Moreover, its application in Chapter 4 provided an FPTalgorithm for the modular width, and we can wonder if applying the Super Lemma to other games could provide the same result.

Chapter 6 was a natural way to extend the study of positional games to handle some scoring games, but other extensions should be interesting. For example, one can add constraints on the moves. The famous board game *Connect* 4 is not handled by positional games because of the gravity. What happens if we add a partially ordered set to our hypergraph, so that a vertex can only be claimed if all its predecessors have already been claimed ? We recently started to study this type of games with Guillaume Bagan, Eric Duchêne, Florian Galliot, Valentin Gledel, Mirjana Mikalački, Aline Parreau and Miloš Stojaković.

Among game problems whose complexity is still open Schaefer [Sch78] introduced two games in 1978: Node Kayles and Arc Kayles. Both games are played by two players on a graph. In the first game, the players take turns removing a vertex and all its neighbors, and in the second game, they take turns removing two adjacent vertices. The first player who cannot move loses. Schaefer directly proved that determining the winner of Node Kayles is PSPACE-complete, but the complexity of Arc Kayles is still open. With Kyle Burke and Antoine Dailly [BDO24], we started working on this problem and proved that this game is PSPACE-complete when we add the constraint that the players are not allowed to disconnect the graph. We would like to continue this work and maybe succeed in proving that Arc Kayles is PSPACE-complete without restrictions.

Bibliography

[ABBS02] Noga Alon, József Balogh, Béla Bollobás, and Tamás Szabó. Game domination number. *Discrete Mathematics*, 256(1-2):23–33, 2002.

[ADGV15] Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic nintendo games are (computationally) hard. *Theoretical Computer Science*, 586:135–160, 2015. Fun with Algorithms.

[AvdHH96] L. Victor Alis, Jaap van der Herik, and M. P. H. Huntjens. Go-moku solved by new search techniques. *Computational Intelligence*, 12, 1996.

[BDD+23] Guillaume Bagan, Quentin Deschamps, Eric Duchêne, Bastien Durain, Brice Effantin, Valentin Gledel, Nacim Oijid, and Aline Parreau. Incidence, a scoring positional game on graphs. *Discrete Mathematics*, in press, 2023.

[BDG06] Lois Blanc, Eric Duchêne, and Sylvain Gravier. A deletion game on graphs:"le pic arête". *Integers: Electronic Journal of Combinatorial Number Theory*, 6(G02):G02, 2006.

[BDO24] Kyle Burke, Antoine Dailly, and Nacim Oijid. Complexity and algorithms for arc-kayles and non-disconnecting arc-kayles, 2024.

[Bec82] József Beck. Remarks on positional games. i. *Acta Mathematica Hungarica*, 40(1-2):65–71, 1982.

[Bec02] József Beck. Positional games and the second moment method. *Combinatorica*, 22:169–216, 2002.

[Bec08] József Beck. *Combinatorial games: tic-tac-toe theory*, volume 114. Cambridge University Press Cambridge, 2008.

[Ber00] Elwyn R Berlekamp. *The dots and boxes game: sophisticated child's play*. CRC Press, 2000.

[BF12] Afshin Behmaram and Shmuel Friedland. Upper bounds for perfect matchings in pfaffian and planar graphs. *The Electronic Journal of Combinatorics*, 20, 2012.

[BFM+23] Julien Bensmail, Foivos Fioravantes, Fionn Mc Inerney, Nicolas Nisse, and Nacim Oijid. The maker-breaker largest connected subgraph game. *Theoretical Computer Science*, 943:102–120, 2023.

[BFMIN22] Julien Bensmail, Foivos Fioravantes, Fionn Mc Inerney, and Nicolas Nisse. The largest connected subgraph game. *Algorithmica*, 84(9):2533–2555, sep 2022.

[BGL+17] Édouard Bonnet, Serge Gaspers, Antonin Lambilliotte, Stefan Rümmele, and Abdallah Saffidine. The Parameterized Complexity of Positional Games. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in*

Informatics (LIPIcs), pages 90:1–90:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BH19] Kyle Burke and Bob Hearn. Pspace-complete two-color placement games. *International Journal of Game Theory*, 48, 06 2019.

[BHKvM21] Kevin Buchin, Mart Hagedoorn, Irina Kostitsyna, and Max van Mulken. Dots & boxes is pspace-complete. *arXiv preprint arXiv:2105.02837*, 2021.

[BHOP] Guillaume Bagan, Mathieu Hilaire, Nacim Oijid, and Aline Parreau. private communication.

[BJS16] Édouard Bonnet, Florian Jamain, and Abdallah Saffidine. On the complexity of connection games. *Theoretical Computer Science*, 644:2–28, 2016. Recent Advances in Computer Games.

[BKR10] Boštjan Brešar, Sandi Klavzar, and Douglas Rall. Domination game and an imagination strategy. *SIAM J. Discrete Math.*, 24:979–991, 01 2010.

[BL00] Malgorzata Bednarska and Tomasz Luczak. Biased positional games for which random strategies are nearly optimal. *Combinatorica*, 20:477–488, 04 2000.

[BM76] John A. Bondy and Uppaluri S. R. Murty. *Graph Theory with Applications*. Elsevier, New York, 1976.

[Bys04] Jesper Makholm Byskov. Maker-maker and maker-breaker games are pspace-complete. *BRICS Report Series*, 11(14), 2004.

[CE78] Vašek Chvátal and Paul Erdös. Biased positional games. In B. Alspach, P. Hell, and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 221–229. Elsevier, 1978.

[CMP09] András Csernenszky, C. Ivett Mándity, and András Pluhár. On chooser–picker positional games. *Discrete Mathematics*, 309(16):5141–5146, 2009.

[CMP11] András Csernenszky, Ryan Martin, and András Pluhár. On the complexity of chooser-picker positional games. *Integers*, 2012:427–444, 11 2011.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[Cse10] András Csernenszky. The chooser-picker 7-in-a-row-game. *arXiv preprint arXiv:1004.2460*, 2010.

[DDO+24] Eric Duchêne, Arthur Dumas, Nacim Oijid, Aline Parreau, and Eric Rémila. The maker–maker domination game in forests. *Discrete Applied Mathematics*, 348:6–34, 2024.

[DF95] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer-Verlag New York, 1999.

[DGI+23] Eric Duchene, Valentin Gledel, Fionn Mc Inerney, Nicolas Nisse, Nacim Oijid, Aline Parreau, and Miloš Stojaković. Complexity of maker-breaker games on edge sets of graphs. *arXiv preprint arXiv:2302.10972*, 2023.

[DGP+21] Eric Duchene, Stéphane Gonzalez, Aline Parreau, Eric Rémila, and Philippe Solal. influence: a partizan scoring game on graphs. *Theoretical Computer Science*, 878:26–46, 2021.

[DGPR20] Eric Duchêne, Valentin Gledel, Aline Parreau, and Gabriel Renault. Maker-Breaker domination game. *Discrete Mathematics*, 343, 2020.

[DOP24] Eric Duchêne, Nacim Oijid, and Aline Parreau. Bipartite instances of influence. *Theoretical Computer Science*, 982:114274, 2024.

[ES73] Paul Erdös and John L. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3):298–301, 1973.

[Ett96] John Mark Ettinger. *Topics in combinatorial games*. The University of Wisconsin-Madison, 1996.

[FG01] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. ACM*, 48(6):1184–1206, nov 2001.

[Gal23] Florian Galliot. *Hypergraphes et jeu Maker-Breaker : une approche structurelle*. PhD thesis, Université Grenoble Alpes, 2023. Thèse de doctorat dirigée par Gravier, Sylvain et Sivignon, Isabelle Mathématiques et informatique Université Grenoble Alpes 2023.

[GGGP] Florian Galliot, Valentin Gledel, Sylvain Gravier, and Aline Parreau. private communication.

[GIK19] Valentin Gledel, Vesna Irsic, and Sandi Klavzar. Fast winning strategies for the maker-breaker domination game. In Gabriel Coutinho, Yoshiharu Kohayakawa, Vinícius Fernandes dos Santos, and Sebastián Urrutia, editors, *Proceedings of the tenth Latin and American Algorithms, Graphs and Optimization Symposium, LAGOS 2019, Belo Horizonte, Brazil, June 2-7, 2019*, volume 346 of *Electronic Notes in Theoretical Computer Science*, pages 473–484. Elsevier, 2019.

[Gle19] Valentin Gledel. *Couverture de sommets sous contraintes*. PhD thesis, Université Claude Bernard, Lyon 1, 2019. Thèse de doctorat dirigée par Eric Duchêne et Aline Parreau Informatique Lyon 2019.

[GLO13] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In Gregory Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation*, pages 163–176, Cham, 2013. Springer International Publishing.

[GO23] Valentin Gledel and Nacim Oijid. Avoidance Games Are PSPACE-Complete. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science (STACS 2023)*, volume 254 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GOT] Valentin Gledel, Nacim Oijid, and Sébastien Tavenas. private communication.

[GSV84] Yuri Gurevich, Larry Stockmeyer, and Uzi Vishkin. Solving np-hard problems on graphs that are almost trees and an application to facility location problems. *J. ACM*, 31(3):459–473, jun 1984.

[Hal35] Philip Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10(1):26–30, 1935.

[Han59] Olof Hanner. Mean play of sums of positional games. *Pacific Journal of Mathematics*, 9(1):81–99, 1959.

[HD09] Robert A. Hearn and Erik D. Demaine. *Games, puzzles, and computation*. CRC Press, 2009.

[Hef07] Dan Hefetz. *Positional games on graphs*. PhD thesis, Citeseer, 2007.

[HJ63] Robert I. Hales and Alfred W. Jewett. Regularity and positional games. *Trans. Am. Math. Soc*, 106:222–229, 1963.

[HKS07a] Dan Hefetz, Michael Krivelevich, and Tibor Szabó. Avoider-Enforcer games. *Journal of Combinatorial Theory, Series A*, 114(5):840–853, 2007.

[HKS07b] Dan Hefetz, Michael Krivelevich, and Tibor Szabó. Bart–Moe games, JumbleG and discrepancy. *European Journal of Combinatorics*, 28(4):1131–1143, 2007.

[HKSS07] Dan Hefetz, Michael Krivelevich, Miloš Stojaković, and Tibor Szabó. Fast winning strategies in positional games. *Electronic Notes in Discrete Mathematics*, 29:213–217, 2007.

[HKSS08] Dan Hefetz, Michael Krivelevich, Miloš Stojaković, and Tibor Szabó. Planarity, colorability, and minor games. *SIAM Journal on Discrete Mathematics*, 22(1):194–212, 2008.

[HKSS10] Dan Hefetz, Michael Krivelevich, Miloš Stojaković, and Tibor Szabó. Avoider–enforcer: The rules of the game. *Journal of Combinatorial Theory, Series A*, 117(2):152–163, 2010.

[HKSS14] Dan Hefetz, Michael Krivelevich, Miloš Stojaković, and Tibor Szabó. *Positional games*, volume 44. Springer, 2014.

[HKT16] Dan Hefetz, Michael Krivelevich, and Wei En Tan. Waiter–client and client–waiter planarity, colorability and minor games. *Discrete Mathematics*, 339(5):1525–1536, 2016.

[Jan13] Bart M.P. Jansen. *Power of Data Reduction: Kernels for Fundamental Graph Problems*. Dissertation, Utrecht University, The Netherlands, 2013.

[Kar72] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[KF04] Martin Kutz and Stefan Felsner. Weak positional games on hypergraphs of rank three. *2005 European Conference on Combinatorics, Graph Theory and Applications (EuroComb '05), DMTCS, 31-36 (2005)*, DMTCS Proceedings vol. AE,..., 01 2004.

[Kno12] Fiachra Knox. Two constructions relating to conjectures of beck on positional games. *arXiv preprint arXiv:1212.3345*, 2012.

[Kri11] Michael Krivelevich. The critical bias for the hamiltonicity game is (1+o(1))n/ln n. *Journal of the American Mathematical Society*, 24:125–131, 01 2011.

[Lam10] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. In *European Symposium on Algorithms*, pages 549–560. Springer, 2010.

[Leh64] Alfred Lehman. A Solution of the Shannon Switching Game. *Journal of the Society for Industrial and Applied Mathematics*, 12(4):687–725, 1964.

[LNS15a] Urban Larsson, Richard J. Nowakowski, and Carlos P. Santos. Scoring combinatorial games: the state of play. *Games of No Chance*, 2015.

[LNS15b] Urban Larsson, Richard J. Nowakowski, and Carlos P. Santos. When waiting moves you in scoring combinatorial games. *arXiv preprint arXiv:1505.01907*, 2015.

[Lu91] Xiaoyun Lu. A matching game. *Discret. Math.*, 94(3):199–207, 1991.

[Mil53] John Milnor. Sums of positional games. *Contributions to the Theory of Games II*, 28:291–301, 1953.

[MS22] Tillmann Miltzow and Miloš Stojaković. Avoider-enforcer game is np-hard. *arXiv preprint arXiv:2208.06687*, 2022.

[Nen23] Rajko Nenadov. Probabilistic intuition holds for a class of small subgraph games. *Proceedings of the American Mathematical Society*, 151(04):1495–1501, 2023.

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[PSV23] Balázs Patkós, Milos Stojakovic, and Máté Vizer. The constructor-blocker game. *Applicable Analysis and Discrete Mathematics*, pages 22–22, 01 2023.

[Rei81] Stefan Reisch. Hex ist PSPACE-vollständig. *Acta Inf.*, 15(2):167–191, 1981.

[RW21] Md Lutfar Rahman and Thomas Watson. 6-uniform maker-breaker game is pspace-complete. In *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2021.

[Sch78] Thomas J. Schaefer. On the Complexity of Some Two-Person Perfect-Information Games. *Journal of computer and system Sciences*, 16:185–225, 1978.

[Sim68] Gustavus J. Simmons. The Game of SIM. In *Mathematical Solitaires & Games*, pages 50–50. Routledge, 1968.

[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 J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 1–9, 1973.

[SS05] Miloš Stojaković and Tibor Szabó. Positional games on random graphs. *Random Structures & Algorithms*, 26(1-2):204–223, 2005.

[Ste12] Fraser Stewart. Scoring play combinatorial games. *Games of No Chance*, 5:447–467, 2012.

[TCHP08] Marc Tedder, Derek Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, pages 634–645, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Zet80] T.G.L. Zetters. Problems dedicated to emory p.starke. *The American Mathematical Monthly*, 87(7):574–576, 1980.

Index