Pontifícia Universidade Católica
do Rio de Janeiro

Handel Scholze Marques

Combinatorial Games and the Neighborhood Conjecture

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Matemática, do Departamento de Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Matemática.

Advisor: Prof. Simon Griffiths

Rio de Janeiro
April 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Handel Scholze Marques

Combinatorial Games and the Neighborhood Conjecture

Dissertation presented to the Programa de Pós–graduação em Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Matemática. Approved by the Examination Committee:

Prof. Simon Griffiths
Advisor
Departamento de Matemática – PUC-Rio

Prof. Nicolau Corção Saldanha
Departamento de Matemática – PUC-Rio

Prof. Taísa Lopes Martins
Universidade Federal Fluminense – UFF

Prof. Maurício de Lemos Rodrigues Collares Neto
Universidade Federal de Minas Gerais – UFMG

Rio de Janeiro, April the 30th, 2021

Handel Scholze Marques

Majored in mathematics by Pontífica Universidade Católica do Rio de Janeiro (Rio de Janeiro, Brazil)

Bibliographic data

Scholze, Handel

Combinatorial Games and the Neighborhood Conjecture / Handel Scholze Marques; advisor: Simon Griffiths. – 2021.

50 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, 2021.

Inclui bibliografia

1. Matemática – Teses. 2. Matemática – Teses. 3. Jogos Combinatórios. 4. Problema de SAT. 5. Método Probabilístico. 6. Hipergrafos. I. Griffiths, Simon. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. III. Título.

CDD: 510

To all the really precious people that I call as friends.
In particular, to my mother, for all the love and attention.

Acknowledgments

First, I would like to thank my advisor Simon Griffiths for all the attention and patience. Without him this work would never go beyond the first page.

Then, I wish to thank professor Josimar Silva, from my high school, that made me fall in love with math.

I thank all the jury members for accepting the invitation and for the attention they gave to this work.

Also, thanks to all my dear friends, in special Rafael Serpa that gave me a home where I could be alone with my thoughts.

And I would like to thank my mother, that always worked so hard to secure that I could be a happy and healthy child.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, thanks for the support.

Finally, thanks for the scholarship that PUC gave me.

Abstract

Scholze, Handel; Griffiths, Simon (Advisor). **Combinatorial Games and the Neighborhood Conjecture**. Rio de Janeiro, 2021. 50p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

The theory of Combinatorial Games is the study of games with perfect information. This means that all players have knowledge of all possible moves, also there isn't luck or skill to perform a move, so, in theory perfect play is possible. Examples of games like these are tic-tac-toe, chess, checkers, Nim... the list goes on. In this dissertation we focus on the Maker-Breaker game. It has two players that pick a vertex from a hypergraph. The goal of Maker is to claim all vertices of an edge and the goal of Breaker is to prevent it. To understand in which types of hypergraphs does Maker or Breaker win and what are the winning strategies, we make use of SAT, Probability, general Graph Theory and more.

Keywords

Combinatorial Games; SAT; Probabilistic Method ; Hypergraphs.

Resumo

Scholze, Handel; Griffiths, Simon. **Jogos Combinatórios e a Conjectura da Vizinhança**. Rio de Janeiro, 2021. 50p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

A teoria dos Jogos Combinatórios é o estudo de jogos com informação completa. Isso é, todos os jogadores conhecem todos os possíveis movimentos, além disso, temos que não há sorte ou a habilidade de realizar um movimento, então, em teoria jogar perfeitamente é possível. Exemplos de jogos assim são jogo da velha, xadrez, damas, Nim... a lista continua. Nessa dissertação focamos no jogo Maker-Breaker. Ele tem dois jogadores que sequencialmente escolhem um vértice de um hipergrafo. O objetivo de Maker é escolher todos os vértices de uma aresta e o objetivo de Breaker é prevenir isso. Para entender em quais tipos de hipergrafos Maker ou Breaker ganha e quais são as estratégias de vitória utilizamos SAT, probabilidade, teoria dos grafos em geral e mais.

Palavras-chave

Jogos Combinatórios; Problema de SAT; Método Probabilístico; Hipergrafos.

Table of contents

We must know.
We shall know.

David Hilbert

1
Introduction

Combinatorial Game Theory is the study of sequential games with perfect information. This means that each player has their own turn and there is no luck or difficulty to perform a move and the skill of a player relates to knowing which move will be better to get to a winning position. So, in theory, perfect play is possible, and the outcome of the game can be predicted if we assume both players play optimally.

Some games like these are chess, Go and checkers. The last is solved by a giant amount of computations that last for 18 years, to the conclusion that the game must be a draw if played optimally. For the other two games, pessimists could say that we will never have the complete solution due to the combinatorial explosion we have when analysing each. On the other hand, chess and Go enthusiasts may be glad that some mystery remains.

Another combinatorial game is Nim, this one was the first I experienced with a more mathematical approach. A book that grew my interest in the subject was "Winning Ways for Your Mathematical Plays" [2].

A simpler combinatorial game, known to many people, is tic-tac-toe. Again, if both players plays optimally the game will end in a draw. József Beck wrote an amazing book [1] with the study of these games, with a great focus on generalizations of the tic-tac-toe game. The book inspired many mathematicians to look at this world of combinatorial games which contains many challenging and interesting problems.

In this dissertation we will focus on the Maker-Breaker combinatorial game, which is played on a k-uniform hypergraph. A hypergraph is a generalization of the concept of a graph in which each edge has size k. The game has two players, Maker and Breaker. Each player in their turn claims a vertex of the hypergraph that will belong to that player until the end of the game. Maker wins if he claims all the vertices of any edge. Breaker wins if he prevents this, that is, if in the end of the game, when there are no more vertices to be claimed, Maker's vertices do not contain all the vertices of any edge.

In this dissertation we say that a player has a winning strategy if he has

a strategy which will lead to a win whatever way his opponent plays. We say that a Maker-Breaker game is *Maker win* if Maker has a winning strategy, and we say that the game is a *Breaker win* if Breaker has a winning strategy. We note that draws do not occur and so every game is Maker win or Breaker win. Also, Maker will be the first to play unless we specify otherwise.

Something you may notice is that the Maker-Breaker game is kind of a generalization of tic-tac-toe. Make the vertices of your Maker-Breaker hypergraph the spaces of the 3x3 grid of the tic-tac-toe game, and the edges of the Maker-Breaker hypergraph the winning crosses of the grid (the rows, columns and diagonals). The difference is that a draw favors the second player (Breaker), and that Breaker does not play to earn a winning crossing, but to prevent Maker doing so.

In 1970, Erdős and Selfridge published a paper [3] which was one of the origins of the Maker-Breaker game. They showed that 2^{k-1} is the minimum number of edges a k-uniform hypergraph must have to be a Maker win. The minimum is attained by a hypergraph which contains a vertex of degree 2^{k-1}. This inspired Beck to conjecture that this was essentially tight. He asked if every Maker-Breaker game with maximum neighborhood less than 2^{k-1} should be a Breaker win. This conjecture is known as "The Neighborhood Conjecture" [1].

This problem remained open for a long time until Gebauer [4] disproved it. However, this is not the end of the story. We may ask for the behaviour of the function $f(k)$, which is defined to be the smallest integer such that there exists an Maker win k-uniform hypergraph with maximum degree $f(k)$. It is still not known whether $f(k)$ grows linearly with k, or exponentially, or somewhere in between.

1.1 Overview of the dissertation

In Chapter 2, we introduce some fundamental results. In particular, we state and prove Hall's Theorem, the Lovász Local Lemma and the Erdős-Selfridge Theorem.

In Chapter 3, studying the results from [8] we consider $f(k)$ for small values of k. In particular, we show that $f(3) = 2$ and $f(4) = 3$.

In Chapter 4, we prove a general lower bound on $f(k)$. That is, we prove that $f(k) > k/2$ for all k. This result is well known due the pairing argument that can be found in [6], an article of Hales and Jewett from 1963 that is

important in the history of the combinatorial games. We also show that if $f(k') > k'/2 + 1$ for some k' then the above bound may be improved for all large k. As far as we know this result has not been published elsewhere.

In Chapter 5, until the end of the dissertation we study the results of Gebauer in [5]. We prove a connection between the function $f(k)$ and a satisfiability (SAT) problem. The SAT problem may then be related to a problem related to binary trees. This enables us to bound $f(k)$ in terms of a function $f_{tree}(k)$ related to the tree problem.

In Chapter 6, we prove an upper bound on $f_{tree}(k)$. This completes the proof of the upper bound on $f(k)$.

2
Some Fundamental Results

In this chapter we will talk about the basic theorems and lemmas that we will use in further chapters.

2.1
Tools and Basic concepts

A set M of independent edges of a graph G is called a matching. If a vertex $v \in V(G)$ is in some edge of M then we say that v is matched by M. Vertices that are not incident to any edge from M are unmatched vertices. We say that M is an A-perfect matching for some $A \subset G$ if every vertex of A is matched by M.

For a vertex $v \in G$ we write $N(v)$ as the neighborhood of v. And for a set of vertices S, we write $N(S)$ for $\bigcup_{v \in S} N(v)$.

Theorem 2.1.1 (Hall's marriage theorem)**.** *Let G be a bipartite graph with bipartition (A, B). Then G has an A-perfect matching if and only if for every $S \subset A$ we have that $|S| \leq |N(S)|$.*

Proof. The easy direction is that if we have an A-perfect matching then we have for every $S \subset A$, $|S| \leq |N(S)|$. Let M be such a matching, and take any $S \subset A$. For each $s \in S$ we have an edge that links s to a unique $s' \in B$. Therefore for each vertex in S we have a vertex in B that belongs to $N(S)$, giving us $|S| \leq |N(S)|$.

Now we prove that if we have the basic condition that $|S| \leq |N(S)|$ holds for all $S \subset A$, then we have an A-perfect matching for G.

We recall some basic concepts about graphs. Let M be an arbitrary matching. An *alternating path* with respect of M is a path that begins in an unmatched vertex $a \in A$ and then continues alternating edges from $E(G) \setminus M$ and M. An alternating path that ends in an unmatched vertex $b \in B$, is also called an *augmenting path*. If such an augmenting path exists then we can make a matching larger than M, since we can swap each edge of the path that is matched with the unmatched ones, that are by construction one more.

Now, we prove the second part of the theorem by contradiction, assuming that there is no A-perfect matching even when we have $\forall S \subset A, |S| \leq |N(S)|$.

Let M be a maximum matching, let $a_0 \in A$ be a vertex that M leaves unmatched. Let $a_0, b_1, a_1, b_2, a_2, ...$ be a maximum sequence of distinct vertices where $a_i \in A$ and $b_i \in B$ for all i, where $a_i b_i \in M$ and b_i is adjacent to some $a_{f(i)}$ with $f(i) \in \{0, 1, ..., i-1\}$.

The sequence can not end at an $a_r \in A$, because if the sequence has r elements from A, their neighborhood must have size at least r, so it would have an element in the neighborhood that is not in the sequence, violating the maximality of the sequence.

So, let $b_r \in B$ be our last vertex of the sequence. Then we have an alternating path

$$P = b_r a_{f(r)} b_{f(r)} a_{f^2(r)} b_{f^2(r)} ... a_{f^n(r)}$$

where $f^n(r) = 0$.

We now observe that P is an augmenting path. It is clearly an alternating path by definition, so we just need that the two endpoints are unmatched. This follows from the definition for one endpoint, a_0. For the other endpoint, b_r, we note that if b_r were matched we could extend M using this edge, and this would contradict the maximality of M. Therefore M is an augmenting path.

Using the augmenting path we may produce a matching M' strictly larger than M, which gives the required contradiction. □

Let $B_1, B_2, ..., B_k$ be events in a probability space, we say that a digraph D is a dependency graph for these events if for all $i \in \{1, ..., k\}$, B_i is independent from the family of events $\{B_j : \vec{ij} \notin E(D)\}$.

Theorem 2.1.2 (Lovász Local Lemma)**.** *Let $B_1, B_2, ..., B_k$ be events and D a dependency graph for them. If there exist numbers $x_i \in [0, 1)$ such that*

$$\mathbb{P}(B_i) \leq x_i \prod_{\vec{ij} \in E(D)} (1 - x_j) \; \forall i = 1, ..., k$$

then, the probability that all the complement events B_i^c happen is positive.

Moreover we have $\mathbb{P}\left(\bigcap_{j=1}^{k} B_j^c\right) \geq \prod_{j=1}^{k} (1 - x_j)$

Proof. For each $i = 1, ..., k$, $\forall S \subset [k] \setminus i$ we have

$$\mathbb{P}\left(B_i \middle| \bigcap_{j\in S} B_j^c\right) \leq x_i$$

We prove this by induction on the size of S.

For $|S| = 0$ we have trivially by the hypothesis of the theorem that $\mathbb{P}(B_i) \leq x_i \prod_{\vec{ij}\in E(D)} (1 - x_j) \leq x_i$ since $(1 - x_j) \leq 1$ for all j.

Now we do the induction step.

For i and S, we define $S_1 = \{j : \vec{ij} \in E(D)\}$, the index of the events that are dependent of B_i, and define $S_2 = \{j : \vec{ij} \notin E(D)\} = S \setminus S_1$, the index of the events that are independent of B_i.

Then, using conditional probability we have

$$\begin{aligned}\mathbb{P}\left(B_i \middle| \bigcap_{j\in S} B_j^c\right) &= \frac{\mathbb{P}\left(B_i \cap \bigcap_{j\in S_1} B_j^c \cap \bigcap_{j\in S_2} B_j^c\right)}{\mathbb{P}\left(\bigcap_{j\in S_1} B_j^c \cap \bigcap_{j\in S_2} B_j^c\right)} \\ &= \frac{\mathbb{P}\left(B_i \cap \bigcap_{j\in S_1} B_j^c \middle| \bigcap_{j\in S_2} B_j^c\right)}{\mathbb{P}\left(\bigcap_{j\in S_1} B_j^c \middle| \bigcap_{j\in S_2} B_j^c\right)} \times \frac{\mathbb{P}\left(\bigcap_{j\in S_2} B_j\right)}{\mathbb{P}\left(\bigcap_{j\in S_2} B_j\right)} \qquad (*)\end{aligned}$$

Now we analyze the numerator and denominator of (*) separately.

For the numerator, as we have B_i independent of $\{B_j : j \in S_2\}$,

$$\mathbb{P}\left(B_i \cap \bigcap_{j\in S_1} B_j^c \middle| \bigcap_{j\in S_2} B_j^c\right) \leq \mathbb{P}\left(B_i \middle| \bigcap_{j\in S_2} B_j^c\right) = \mathbb{P}(B_i) \leq x_i \prod_{\vec{ij}\in E(D)} (1 - x_j)$$

Since the first inequality is trivial, the equality follows from independence and the last inequality is given by the theorem hypothesis.

For the denominator of (*) we have 1 if $|S| = 0$. For $|S| = r \geq 1$ write $S = \{j_1, j_2, ..., j_r\}$ then we have

$$
\begin{aligned}
&\mathbb{P}\left(\bigcap_{j\in S_1} B_j^c \middle| \bigcap_{j\in S_2} B_j^c\right)\\
&= \mathbb{P}\left(B_{j_1}^c \middle| \bigcap_{j\in S_2} B_j^c\right)\mathbb{P}\left(\bigcap_{i=2}^{r} B_{j_i}^c \middle| B_{j_1}^c \cap \bigcap_{j\in S_2} B_j^c\right)\\
&= \mathbb{P}\left(B_{j_1}^c \middle| \bigcap_{j\in S_2} B_j^c\right)\mathbb{P}\left(B_{j_2}^c \middle| B_{j_1}^c \cap \bigcap_{j\in S_2} B_j^c\right)\mathbb{P}\left(\bigcap_{i=i_3}^{r} B_{j_i}^c \middle| B_{j_1}^c \cap B_{j_2}^c \cap \bigcap_{j\in S_2} B_j^c\right)\\
&= \ \ldots \ =\\
&= \mathbb{P}\left(B_{j_1}^c \middle| \bigcap_{j\in S_2} B_j^c\right)\mathbb{P}\left(B_{j_2}^c \middle| B_{j_1}^c \cap \bigcap_{j\in S_2} B_j^c\right)\ldots\mathbb{P}\left(B_{j_r}^c \middle| \bigcap_{i=1}^{r-1} B_{j_i}^c \cap \bigcap_{j\in S_2} B_j^c\right)\\
&= \left(1-\mathbb{P}\left(B_{j_1} \middle| \bigcap_{j\in S_2} B_j^c\right)\right)\ \ldots\ \left(1-\mathbb{P}\left(B_{j_r} \middle| \bigcap_{i=1}^{r-1} B_{j_i}^c \cap \bigcap_{j\in S_2} B_j^c\right)\right)\\
&\geq (1-x_{j_1})(1-x_{J_2})\ldots(1-x_{j_r}) = \prod_{j\in S_1}(1-x_j),
\end{aligned}
$$

where the inequality holds by the induction hypothesis.

We have

$$
\mathbb{P}\left(B_i | \bigcap_{j\in S} B_j^c\right) = \frac{\mathbb{P}\left(B_i \cap \bigcap\limits_{j\in S_1} B_j^c \middle| \bigcap\limits_{j\in S_2} B_j^c\right)}{\mathbb{P}\left(\bigcap\limits_{j\in S_1} B_j^c \middle| \bigcap\limits_{j\in S_2} B_j^c\right)} \leq \frac{x_i \prod\limits_{\vec{ij}\in E(D)}(1-x_j)}{\prod\limits_{j\in S_1}(1-x_j)} \leq x_i,
$$

finishing the induction step.

Now,

$$
\mathbb{P}\left(\bigcap_{j=1}^{k} B_j^c\right) = \mathbb{P}(B_1^c)\mathbb{P}(B_2^c|B_1^c)\ldots\mathbb{P}\left(B_k^c \middle| \bigcap_{j=1}^{k-1} B_j^c\right) \geq \prod_{j=1}^{k}(1-x_j),
$$

as desired.

□

2.2
Maker and Breaker

In 1970, Erdős and Selfridge published a paper [3] which was one of the origins of the Maker-Breaker game. In the paper they do not even use the vocabulary of graph theory, but they established a fundamental condition on the number of edges for a game to be a Maker win. Here we replicate the result.

We require the following definition.

Definition 2.2.1. Given an positive integer k, let $m^*(k)$ be the smallest integer for which there is a Maker win k-uniform hypergraph with $m^*(k)$ edges.

Theorem 2.2.2. $m^*(k) = 2^{k-1}$

Proof. First we show a hypergraph H on $2k$ vertices with 2^{k-1} edges which is a Maker win. Let V be a set of $2k$ vertices and partition the vertices of V in two sets, $A = \{a_1, a_2, ..., a_k\}$ and $B = \{b_1, b_2, ..., b_k\}$. Now we create sets recursively that in the kth step will be our edge set, let $E_1 = a_1$ and $E'_1 = b_1$, define $E_{i+1} = (E_i + \{a_{i+1}\}) \cup (E'_i + \{a_{i+1}\})$ and $E'_{i+1} = (E_i + \{b_{i+1}\}) \cup (E'_i + \{b_{i+1}\})$. Note that in each iteration we double the number of sets inside E_i, so E_k has 2^{k-1} sets in it. Also, in each iteration we incremented by 1 the size of each set inside E_i, so E_k has only sets with size k.

Set E_k as the edge set of H. We claim that H is Maker win.

The winning strategy for Maker is as follows. Begin claiming a_k, then whenever Breaker claims a_i, Maker claim b_i, and whenever Breaker claims b_i, Maker claims a_i. In each turn Breaker blocks only half of the remaining unblocked edges, so on the kth turn Maker will have an entire edge and win the game.

Now we must show that for any k-uniform hypergraph H with edge set $|E| < 2^{k-1}$, Breaker has a winning strategy.

We associate an integer value with each edge as follows:

(i) If Breaker has not played in this edge, and Maker has claimed j vertices so far, then we associate value 2^j with this edge,

(ii) If Breaker has played in this edge we associate value 0 with this edge.

This value can be thought of as the "danger" of an edge. And for the vertices, give each a value equal to the sum of the danger of the edges in which it belongs. Let D be the sum of the danger of all edges. Maker wins if he has all the vertices of some edge. Therefore Maker can only win if D is eventually at least 2^k.

In order to show a Breaker win strategy we give a strategy for Breaker such that the total danger D is always less than 2^k. The strategy of Breaker is to always pick one of the vertices with maximum value.

We begin with $D = |E| < 2^{k-1}$, as all edges have danger equal to 1. When Maker picks a vertex, it doubles the danger of all edges to which it

belongs. And so, in the first turn, whichever vertex Maker picks, the danger will at most double, and so, D will still be less than 2^k.

Before the next move from Maker, D will be less than $2^k - R$, where R is the sum of the danger of the edges just blocked by Breaker, that is, the value of the vertex taken by Breaker. Now on Maker's move, he will double the danger of the edges that contain the vertex he chooses. That is, he adds R' (the sum of the previous dangers of these edges) to D. However, we have $R' \leq R$, since R was chosen to be maximum. And so, the danger in this step the danger has changed from D to $D - R + R'$ which is at most D, and so is still less than 2^k.

In this way, the danger will always be less than 2^k, and so Breaker wins the game. □

As we said in the Introduction, this result motivated Beck to pose his conjecture. If we think about the maximum neighborhood of the graph that we constructed in Theorem 2.2.2, we would notice that an edge of the vertex a_1 intersects all edges of the graph, showing that there is a neighborhood of size 2^{k-1}. So, Beck conjectured that any graph with maximum neighborhood less than 2^{k-1} would be a Breaker win, and like we said before this was proved wrong.

3
On $f(3)$ and $f(4)$

It is very common in mathematics to try first to solve "small cases" before going on to tackle the general problem. Based on [8], we will construct examples that give us the first values for the f function. We shall see that $f(3) = 2$ and $f(4) = 3$. With this we will really immerse ourselves in the world of Maker and Breaker.

3.1
About the f function

In the previous chapters we discussed the Neighbourhood Conjecture. Even though the original statement has been proved wrong, it is still natural to study the function $f(k)$, which is the smallest integer for which there exists a k-uniform hypergraph H of maximum degree at most $f(k)$ which is Maker win.

Let us begin by analysing f for small k. As k is the size of the edges, we may begin with $k = 2$. For graphs with maximum degree 1, we can think of the graph with only two vertices linked by an edge, it is clear that Maker cannot win this game. Other graphs with maximum degree 1 are just disconnected copies of this one, and potentially isolated vertices that have no influence in the game.

In the case in which the maximum degree is 2, any graph will be Maker win. Indeed, he may just begin by claiming any vertex with degree 2, Breaker could then block one of the neighbours of the claimed vertex, however now Maker just picks the other one and wins. So we have $f(2) = 2$.

For $n = 3$ we also have $f(3) = 2$. Again it is easy to see that any 3-uniform graph with maximum degree 1 cannot be a Maker win. With $\Delta(G) = 2$ the graph G_3 (see Figure 3.1) is an example where Maker has a winning strategy as follows.

We, as Maker, start claiming v_1. Note that v_1 splits the graph in two, so whichever side Breaker plays, Maker claims the v vertex from the other side.

Let us say that Breaker got some vertex from the v_3 side, then we take v_2 and now Breaker has only 6 meaningful claims, choose a_1, a_2, a_3, b_1, b_2 or b_3. The idea here to win is if Breaker claims a_i then we will win by claiming the edge $b = \{b_1, b_2, b_3\}$ with b_i being our last claim, analogously we win by a_i if Breaker claims b_i. For example, suppose Breaker claimed a_1, then we claim b_3, forcing Breaker to claim a_3, otherwise we win by choosing a_3 for ourselves, given that we have v_2 and b_3, then we take b_2 forcing again Breaker to claim a_2, and then we claim b_1 and win.

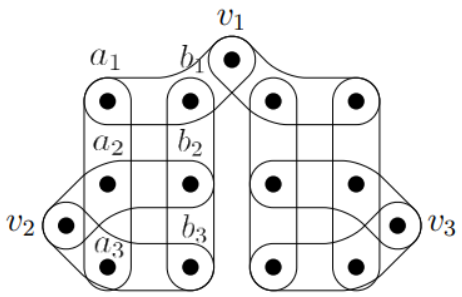

Figure 3.1: Graph G_3. Image from [8] (slightly modified).

3.2 $f(4)$ is equal to 3

From G_3 we can construct a 4-graph with max degree 4 which Maker wins. To do this, take each edge $e \in G_3$ and create two vertices, x_e and y_e, replace e by e_x and e_y where e_x is e with x_e added and e_y is e with y_e added (see figure 3.2). The winning strategy is simple: use the old strategy to claim an edge e of G_3 and then claim e_x or e_y. If Breaker makes an early move on the newer vertices, just take the other vertex of the pair right after.

With this, we have that $f(4) \leq 4$, and by the pairing argument that we will see in the next chapter, we have $f(4) > 4/2 = 2$, then the only possible values left for $f(4)$ are 3 or 4. So, we will show an example, that we will call by G_4, of an 4-graph with $\Delta = 3$ where Maker wins, implying $f(4) = 3$.

To construct our example we will do it in three stages, fist we create a 3-graph Γ which Maker wins, where the main structure that makes Maker win belongs. Then we derive a Γ' hypergraph with mixed edge sizes and from Γ' we finally achieve G_4.

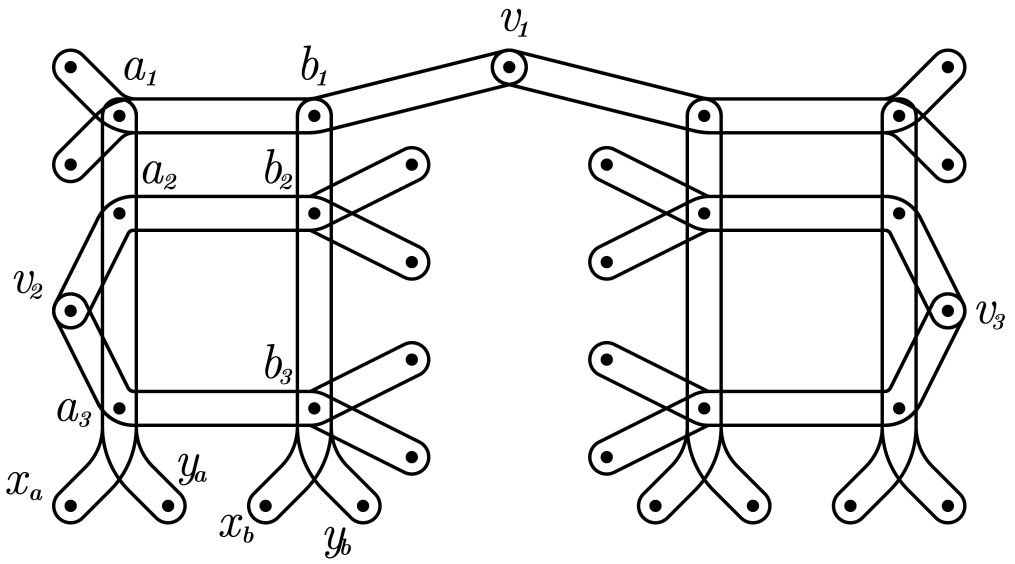

Figure 3.2: Graph G'_3

3.2.1

The Γ graph

Continuing, we have the construction of Γ (figure 3.3). The vertices of Γ are the union of three disjoint sets
$W = \{w_i \mid i \in [5]\}$ with size 5
$X = \{x_{ij} \mid i \in [5],\ j \in [3]\}$ with size 15
$T = \{\, t_{ij} \mid i \in [5],\ j \in [3]\}$ with size 15
so, we have $V(\Gamma) = W \cup X \cup T$ with size 35.
For the edges let $e_i = x_{i1}x_{(i+2)2}x_{(i+3)3}$ where $i \in [5]$ so define
$E(\Gamma) = \{w_i x_{ij} t_{ij} \mid i \in [5],\ j \in [3]\} \cup \{e_1, e_2, e_3, e_4, e_5\}$ giving us 20 edges in total. Note that the vertices w_i have degree 3, the vertices t_{ij} have degree 1 and the vertices x_{ij} have degree 2. So, we have that $\Delta(\Gamma) = 3$, what is left is to show how Maker wins in this graph.

Lemma 3.2.1. *Maker has a winning strategy for the Maker-Breaker game on* Γ*, where Breaker goes first.*

Proof. The idea of Maker's strategy is to first play on the vertices $\{\omega_i \mid i \in [5]\}$, by claiming a w_i where t_{ij} and x_{ij} are unclaimed with $j \in [3]$. Maker threatens claiming x_{ij} forcing Breaker to claim t_{ij} and this way Maker gains an advantage in the fight on the $e_1, ..., e_5$ edges that is where he will make his win.

Case 1: Breaker claims w_i for some i. Without loss of generality by the symmetry of Γ we can assume that Breaker claims w_1. Maker then claims w_2.

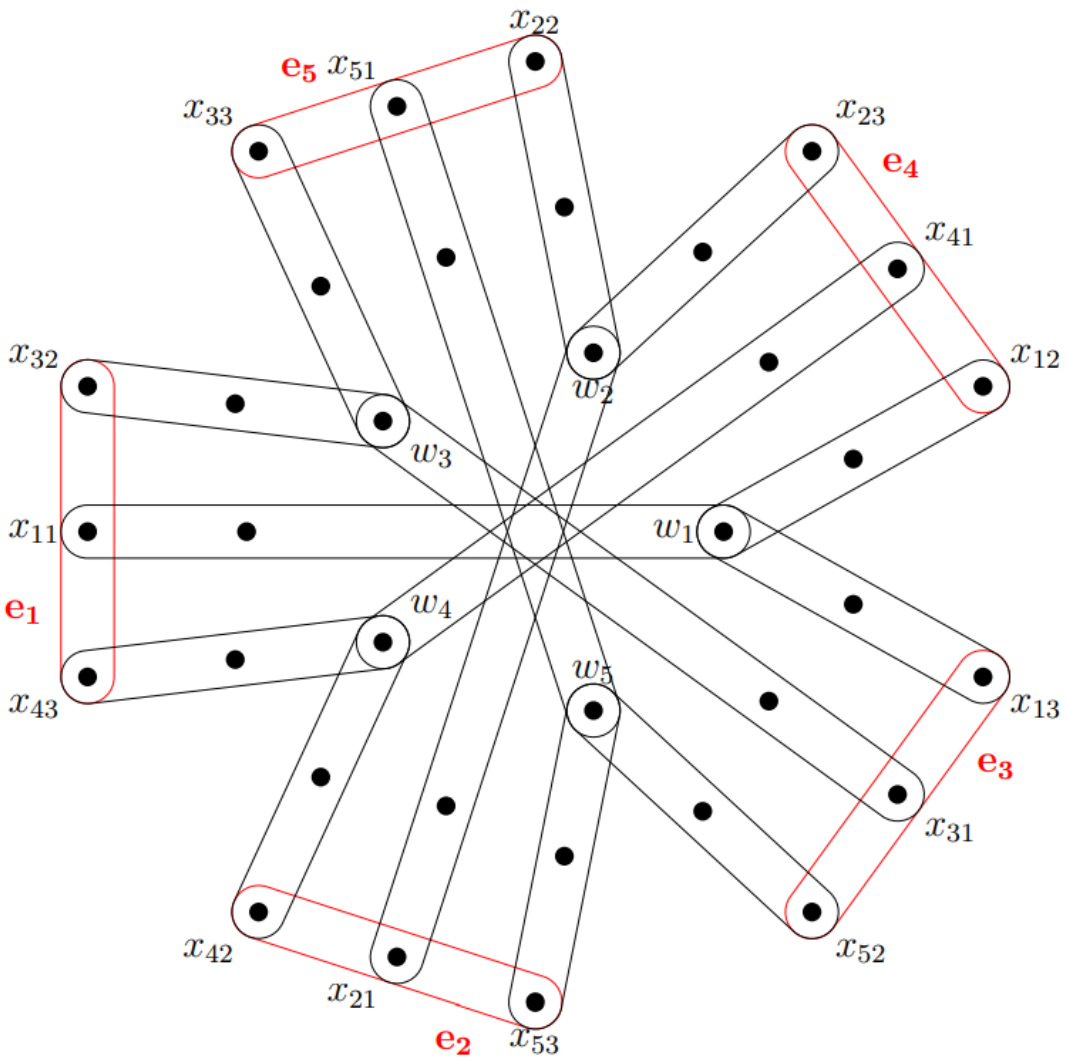

Figure 3.3: Graph Γ. Image from [8].

Now Maker is threatening their win by claiming e_2 with the following forced victory. Maker claims x_{21} forcing Breaker to claim t_{21}, then x_{23} forcing t_{23}, then x_{41}. Breaker now is forced to claim x_{12} to prevent Maker to claim e_4. Then Maker claims w_4 forcing Breaker to claim t_{41}, then claims x_{42} forcing t_{42}, then claims x_{53} and wins by claiming e_2. So, in the second turn Breaker must claim a vertex to prevent this strategy.

Case 1.1: Breaker claims w_4 or a vertex of e_4 on his second turn. In this case Maker just do the same as before but using w_5 instead of w_2. The moves are: Maker again claims x_{21}, forcing t_{21}, then x_{22} (instead of x_{21}), forcing t_{22}, then x_{51}. Breaker now is forced to claim x_{33} to prevent Maker to claim e_5. Then Maker can now claim w_5, forcing t_{51}, then he claims x_{53}, forcing t_{53} and then x_{42} claiming the edge e_2 and winning.

Case 1.2: Breaker claims a vertex of e_2 on his second turn. In this case Maker will win by claiming e_5, for this he will use w_3 and w_4. The moves are

the following: Maker claims x_{22} forcing t_{22}, then claims x_{23} forcing t_{23}, then x_{41} forcing x_{12}, then he claims w_4 forcing t_{41}, then x_{43} forcing t_{43}, then x_{32} forcing x_{11}, to then claim w_3 with Breaker being forced to claim t_{32}, then Maker claims x_{33} forcing t_{33} and finally wins by claiming x_{51} getting the edge e_5.

The more careful reader must notice that we didn't cover the case where Breaker claims a t vertex that would help him in the above games, this is solved by Maker using the rule that whenever Breaker claims a vertex t_{ij} that Maker has not claimed both w_i and x_{ij}, Maker plays as if Breaker had claimed w_i or x_{ij}, whichever is free (or arbitrarily if neither is free). This rule still applies to the following cases.

Case 2: Breaker claims x_{i1} for some i. Without loss of generality by the symmetry of Γ we can assume that Breaker claims x_{11}. Maker then claims w_2. If Breaker skips his turn then Maker can do the same strategy as in Case 1, since Breaker having x_{11} does not interfere.

Case 2.1: Breaker claims w_4 or a vertex of e_4 on his second turn. Then Maker wins as in Case 1.1, since Breaker's move on x_{11} does not prevent this.

Case 2.2: Breaker claims a vertex of e_2 on his second turn. Then Maker claims x_{22}, forcing t_{22}, then x_{23}, forcing t_{23} and then x_{12} Breaker is forced to claim x_{41} or Maker wins immediately by playing there and claiming e_4. Then Maker claims w_1 forcing t_{12}, x_{13} forcing t_{13} and then x_{52}. Breaker is forced to claim x_{31} or Maker wins immediately by playing there and claiming e_3. Then Maker claims w_5 forcing t_{52}, x_{51} forcing t_{51} and then x_{33}, claiming the edge e_5 and winning.

Case 3: Breaker claims x_{i2} or x_{i3} for some i. Without loss of generality by the symmetry of Γ we can assume that Breaker claims x_{32}. Again we can use the standard strategy of the Case 1.

Case 3.1 Breaker claims w_4 or a vertex of e_4 on his second turn. Then Maker wins as in Case 1.1.

Case 3.2 Breaker claims a vertex of e_2 on his second turn. Then Maker wins as in Case 2.2.

□

3.2.2
Γ' from Γ

Continuing to our goal, the graph G_4, now we will derive a non-uniform hypergraph with maximum degree 3 Γ' from Γ with edges of size 3 and 4. Add

new vertices (y_{ijk}) $k \in [6]$ and (z_{ijk}) $k \in [4]$ to Γ for every $i \in [5]$ and $j \in [3]$. Then for every $i \in [5]$ and $j \in [3]$, writing $w = w_i$, $x = x_{ij}$, $t = t_{ij}$, $y_k = y_{ijk}$ for $k \in [6]$ and $z_k = z_{ijk}$ for $k \in [4]$, we replace the edge wxt by the edges

$$wty_1y_2,\ xty_3y_4,\ xty_5y_6 \text{ and } y_1y_3y_5z_1,\ y_1y_3y_5z_2,\ y_2y_4y_6z_3,\ y_2y_4y_6z_4.$$

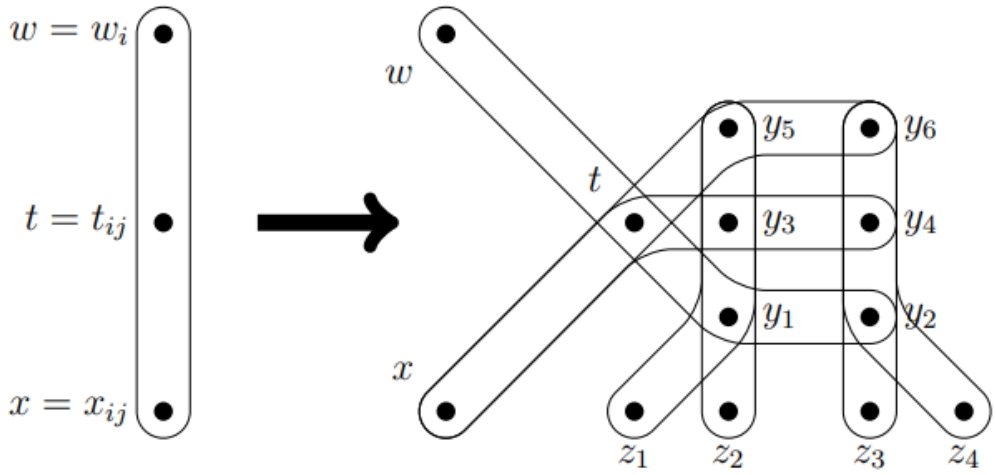

Figure 3.4: From Γ to Γ'. Image from [8].

Each 3 vertex set wtx became a set of 13 vertices, this means that for each wtx we added 10 vertices to our graph. Since we have 15 edges of this type then we added 150 vertices resulting in a graph of 185 vertices. For the edges each wtx makes 7 edges, again we have 15 edges of this type, then it has 105 edges plus the 5 e_i edges, resulting in a graph of 110 edges.

Note that all the edges of Γ' has size 4 except for the e_i edges with size 3, since all other original edges were replaced to ones with size 4 that we just added. Also, Γ' has maximum degree 3, each edge incident to the w_i vertices were replaced by only one edge of size 4, keeping the degree of those vertices equals 3. For the x_{ij} vertices, each had degree 2, the e_i edge that was left untouched, and the wtx edge, that became 2 edges (xty_5y_6 and xty_3y_4), resulting in a vertex with degree equals 3. The remaining vertices to analyse are clearly checked with the figure 3.4, the z_i vertices has degree 1 and the others has degree 3.

Lemma 3.2.2. *Maker has a winning strategy for the Maker-Breaker game on Γ', where Breaker goes first.*

Proof. Initially Maker plays on the vertices $\{w_i \mid i \in [5]\}$ and $\{x_{ij} \mid i \in [5],\ j \in [3]\}$ and plays according to the strategy described in Lemma 3.2.1. If Breaker

claims y_{ijk} or z_{ijk} for some i, j, k then Maker plays as if Breaker had claimed t_{ij}, also, if Breaker has already claimed t_{ij} then we choose a free x_{ij} arbitrarily and play as if Breaker had played there instead, let us call this the * rule. Playing following the strategy from Lemma 3.2.1 implies that Maker will claim an edge of Γ. If it is some e_i then Maker wins because it is also an edge of Γ', the other option is that Maker will claim $w_i x_{ij} t_{ij}$ for some $i \in [5]$ and $j \in [3]$. Note that Maker having t_{ij} implies that y_{ijk} and z_{ijk} are unclaimed, by the * rule. Now Breaker has a turn, assume that Breaker claims y_2, z_3 or z_4 (other choices by Breaker are similarly covered). Then Maker claims y_5 forcing y_6, claims y_3 forcing y_4, claims y_1 and Maker wins by claiming z_1 or z_2 and Breaker cannot lock both.

□

3.2.3

G_4 from Γ'

Finally the G_4 graph is made as follows: Take 3 disjoint union of three copies Γ'_1, Γ'_2 and Γ'_3 of Γ', let $e_{i1}, ...e_{i5}$ be the edges of size 3 of Γ'_i for each $i \in [3]$. Let $v_1, v_2, v_3, v_4, s_1, s_2, s_3$ be new vertices, and replace the edges e_{i1} and e_{i2} by $e_{i1} \cup v_i$ and $e_{i2} \cup v_i$, also, replace e_{i3}, e_{i4} and e_{i5} by $e_{i3} \cup s_i, e_{i4} \cup s_i$ and $e_{i5} \cup s_i$ for each $i \in [3]$, and add the edge $v_1 v_2 v_3 v_4$ (Figure 3.5).

G_4 is a 4-uniform graph, since all 3 edges from Γ' we make into a 4 edge, G_4 has maximum degree 3, since Γ' has maximum degree 3 and all the vertices that we add also has maximum degree 3. G_4 has 562 vertices (185 for each Γ' plus the 7 added vertices) and 331 edges (110 for each Γ' plus one edge).

Theorem 3.2.1. *Maker has a winning strategy for the Maker-Breaker game on G_4, where Maker goes first.*

Proof. If Maker claims v_i and s_i where Breaker has not claimed any vertex from Γ'_i then Maker just wins by following the Lemma 3.2.2 strategy where Breaker plays first. Since claiming any e_{ij} implies Maker claiming $e_{ij} \cup v_i$ or $e_{ij} \cup s_i$ which are G_4 edges, or instead, Maker will claim any other edge of Γ'_i and also win since it will be an G_4 edge too. So if Maker begins claiming v_1, Breaker is forced to claim s_1 to prevent Maker from winning, then Maker claims v_2 forcing Breaker to claim s_2 for the same reason, then Maker claims v_3 forcing the s_3 move from Breaker and then Maker claims v_4 and wins by having the edge $v_1 v_2 v_3 v_4$. □

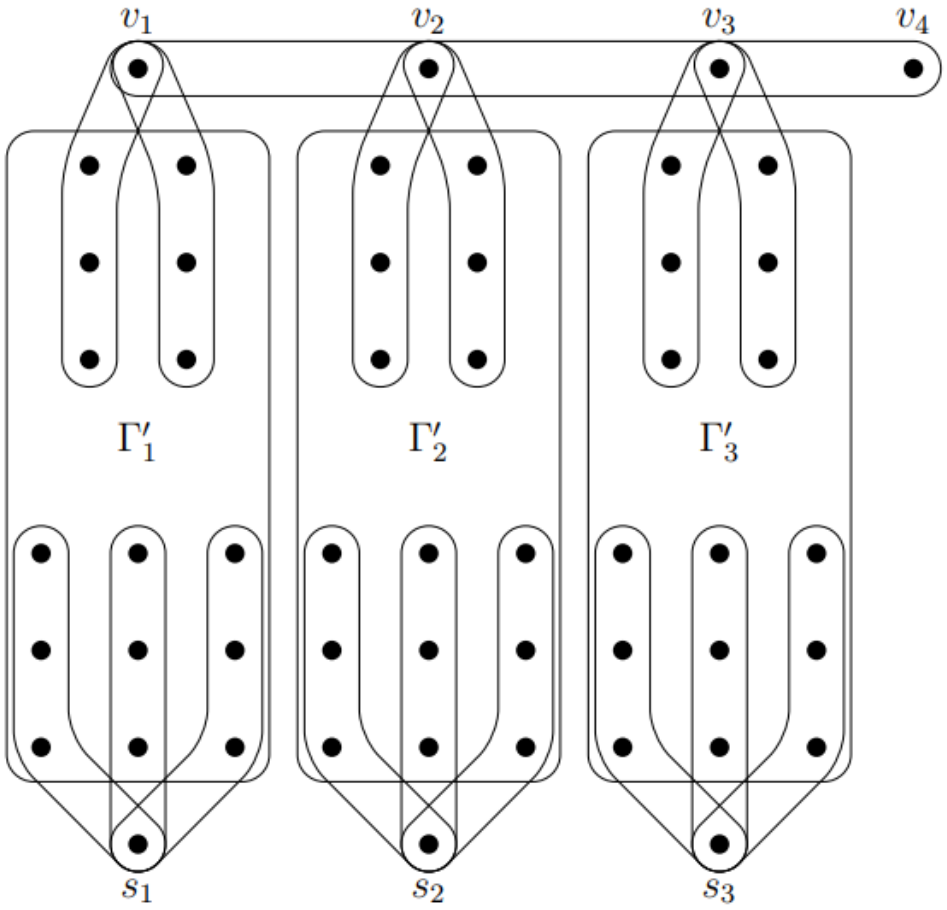

Figure 3.5: The Graph G_4. Image from [8].

4
Lower bounds on the function $f(k)$

In this chapter we prove the lower bound $f(k) > k/2$. The argument is based on a Breaker winning strategy that involves pairs, which may be found using Hall's Theorem.

We also show that if $f(k') > k'/2 + 1$ for some k' then the bound may be improved for all large k. As far as we are aware this result does not appear in any published article.

4.1
The lower bound of the f function

Theorem 4.1.1. *Let H be a k-uniform hypergraph, with $k \geq 2$, if the maximum degree of H is at most $k/2$ then H is Breaker win.*

Proof. We will explicitly show a winning strategy for Breaker. To do so, we create a bipartite graph $G = A \cup B$ with vertex sets $A = 2 \cdot |E(H)|$ and $B = |V(H)|$. This way each vertex from H will have a representative in B and each edge of H will have two representatives in A.

The edges of G are as follows, for each u in A we connect it to a vertex v in B if and only if v is in u in the hypergraph H.

Our goal is to find an A-perfect matching M in G, the matching describes the winning strategy for Breaker. Indeed, given such a matching we identify two distinct representatives $\{v_e, v'_e\}$ for each edge e. Now Breaker plays by a simple rule: whenever Maker plays a vertex in such a pair Breaker then plays the other. This makes it impossible for Maker to claim an entire edge.

What is left is to show that this M matching exists, we recall Hall's marriage theorem. Let S be a subset of A, and $N(S)$ the neighborhood of S, we just need to prove $|N(S)| \geq |S|$. Let us begin by counting the number of

edges between S and $N(S)$. We have

$$e(S, N(S)) = k \cdot |S| \,,$$

as each vertex of S is representing an edge of H that has size k. It follows that

$$k \cdot |S| \leq \sum_{v \in N(S)} d(v) \,.$$

We now observe that $d(v)$, the degree of v in the bipartite graph is exactly double $d_H(v)$, its degree in H. And so

$$k \cdot |S| \leq 2 \cdot \Delta(H) \cdot |N(S)| \,.$$

By assumption $\Delta(H) \leq k/2$ and so we have

$$k \cdot |S| \leq k \cdot |N(S)| \,.$$

This proves that Hall's condition holds.

□

We say for an integer k and c that (k, c) is a Maker win if every k-uniform hypergraph with maximum degree c is a Maker win. Conversely, we say that (k, c) is a Breaker win if the same class of graphs is always a Breaker win.

We now show that if (k, c) is Breaker win for any pair (k, c) with $c > k/2$ then we may strengthen the above bound for all large n.

Theorem 4.1.2. *If (k, c) is Breaker win, then $f(n) > \dfrac{c \cdot n}{k}$.*

Proof. Let H be a n-hypergraph which maximum vertex degree is at most $\dfrac{c \cdot n}{k}$.

The same way as above we create an auxiliary bipartite graph $G = A \cup B$, this time with $A = k \cdot |E(H)|$ and $B = c \cdot |V(H)|$. The edges are defined the same way as before.

Let S be a subset of A, counting the edges between S and its neighborhood $N(S)$ give us

$$e(S, N(S)) = c \cdot n \cdot |S|$$

Now we have the c factor since each vertex e from S, that must be connected to a vertex v from $N(S)$, will also be connected to the c copies of v.

Again it follows that

$$c \cdot n \cdot |S| \leq \sum_{v \in N(S)} d(v)$$

We now observe that $d(v)$, the degree of v in the bipartite graph is $d_H(v)$, its degree in H, multiplied by k, since we made k copies of each edge. And so

$$c \cdot n \cdot |S| \leq k \cdot \Delta(H) \cdot |N(S)|$$

By assumption $\Delta(H) \leq c \cdot n/k$ and so we have

$$c \cdot n \cdot |S| \leq c \cdot n \cdot |N(S)|$$

So as before, using the Hall's marriage theorem, we have an A-matching M. This way for each edge e from $E(H)$ there are k vertices inside e marked by our matching M, however, now these vertices can be the mark of another edges as well, at most c edges to be more precise. This reveals a structure of an (k, c) hypergraph inside our bigger H hypergraph, where Breaker can use to win the game. □

5
A little help from our friends (SAT and binary trees)

In this Chapter we shall relate Maker-Breaker problems to boolean satisfiability problems. We shall see that this allows to prove upper bounds on the function $f(k)$.

More explicitly, we shall define a function $f_{tree}(k)$ related to a problem on binary trees, and via satisfiability problems we shall show that $f(k) \leq 2f_{tree}(k-2)$. In Chapter 6, we bound $f_{tree}(k)$.

5.1
SAT

The boolean satisfiability problem, called SAT, is to determine if a boolean formula can have its boolean variables assigned in a way that the result is set as true.

A CNF formula $F = C_1 \wedge C_2 \wedge C_3 \wedge ... \wedge C_n$, known as Conjunctive Normal Form, is a boolean formula which is a conjunction of clauses (C_i). A conjunction is a statement formed by adding statements with the connector AND ($\wedge$), and a clause $C = l_1 \vee l_2 \vee l_3 \vee ... \vee l_k$ is a disjunction of literals, where a disjunction is a statement formed by adding statements with the connector OR ($\vee$). Literals are either the affirmation of a boolean variable (p) or the negation of the same ($\neg p$), called as positive literal and negative literal respectively. One example of a CNF formula is

$$(\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (x_2 \vee \neg x_4) \wedge (x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_4).$$

This one is satisfiable, that is, there exists an assignment for the variables as true or false such that the resulting value of the formula is true. A possible assignment is to set x_1 as false, x_2 as true, x_3 as true and x_4 as true. Note that in each clause it is only necessary to have one true literal to make the whole clause be true, since we have an OR operation.

We use the notation $x_i \in C_j$ to denote the fact that either x_i or $\neg x_i$ occurs in the clause C_j.

A k-CNF formula is a CNF formula where each clause has exactly k literals.

A (k, s)-CNF formula is a k-CNF formula where each literal appears in at most s clauses.

Theorem 5.1.1. *Every (r, r)-CNF formula is satisfiable.*

Proof. Let $x_1, x_2, x_3, ..., x_n$ be the variables and $C_1, C_2, C_3, ..., C_m$ be the clauses of our (r, r)-CNF formula, where $n \leq m$. We construct a bipartite graph $G = A \cup B$, where one of the vertex sets, called A, corresponds to the clauses and the other, called B, corresponds to the variables. We join by an edge a clause C_j with a variable x_i if and only if $x_i \in C_j$. Now we observe that if we take k clauses, the neighborhood of these clauses in the bipartite graph will have at least k different variables, since each clause has r distinct variables and each variable appears in at most r clauses. Then Hall's condition holds, giving us an A-perfect matching that describes which variable will be used to set which clause as true. □

Note that this implies that every (r, s)-CNF formula with $r \leq s$ is satisfiable.

A natural question we may ask is for a fixed k, when does (k, s)-CNF have unsatisfiable formulas? Define $f_{CNF}(k)$ as the largest number such that all $(k, f_{CNF}(k))$-CNF formulas are satisfiable, in other words, $f_{CNF}(k)$ is the smallest number such that there are unsatisfiable $(k, f_{CNF}(k) + 1)$-CNF formulas.

It is natural to ask whether there are other natural properties which imply that a k-CNF formula is satisfiable. We now show another property which implies satisfiability.

We may associate a graph D with each k-CNF formula as follows. The vertices of D are the clauses of the k-CNF formula, and two clauses C and C' are adjacent in D if they share a variable. Note that, with this definition, the neighborhood of C is

$$N(C) = \{C' : \exists i \quad x_i \in C \text{ and } x_i \in C'\}.$$

We shall see that an upper bound on neighborhood sizes may imply satisfiability.

Let $\ell_{CNF}(k)$ be the largest integer ℓ such that the condition that all clauses have $|N(C)| \leq \ell$ implies satisfiability.

We now prove a lower bound of $\ell_{CNF}(k)$ using the Lovász Local Lemma, a result that can be further studied in [7].

Proposition 5.1.1. $\ell_{CNF}(k) \geq \left\lfloor \frac{2^k}{e} \right\rfloor - 1$

Proof. Let $F = \bigwedge_{i=1}^{n} C_i$ be any k-CNF formula with maximum neighborhood of size $\left\lfloor \frac{2^k}{e} \right\rfloor - 1$.

Set each variables of F as $true$ or $false$ with probability $1/2$ independently and define B_i as the event "C_i is false". In this way, we have $P(B_i) = 2^{-k}$. We also claim that the graph[1] D defined above is a dependency graph for the events B_i. To see this we note that revealing all the information about the clauses not adjacent to B_i tells us nothing about the truth value of the literals which occur in C_i.

What we must show is that there exist $x_i, i = 1 \dots n$ where

$$P(B_i) \leq x_i \prod_{\overrightarrow{B_i B_j} \in E(D)} (1 - x_j)$$

Let $x_i = e \cdot P(B_i)$ then we have

$$2^{-k} \leq e\, 2^{-k} \prod_{\overrightarrow{B_i B_j} \in E(D)} (1 - \frac{e}{2^k}), \text{ que segue de, } e(1 - \frac{e}{2^k})^{\frac{2^k}{e} - 1} > ee^{-1} = 1$$

So, applying the Local Lemma we have that $P(\bigcap_{i=1}^{n} B_i^c) > 0$, giving us that the probability of all clauses be satisfiable is positive. □

We have an easy upper bound for $\ell_{CNF}(k)$, take a k-CNF formula with k variables and 2^k clauses, where the clauses are all the possible arrangements of the variables (the positive literal and the negative literal for each variable), such k-CNF formula has maximum neighborhood of $2^k - 1$, actually, every clause of this formula has a neighborhood of size $2^k - 1$. With that, we obtain that $\ell_{CNF}(k)$ must be less than $2^k - 1$.

[1] Technically, we should relabel the vertex C_i as B_i

One thing to notice is that the lower bound of $\ell_{CNF}(k)$ gives us an easy lower bound for $f_{CNF}(k)$. Take a $(k, f_{CNF}(k)+1)$-CNF formula that is unsatisfiable (we know that it is possible because of the definition of f_{CNF}). In such formula each clause has k variables, that each appears at most in $f_{CNF}(k)$ more clauses, so, each clause has a neighborhood of size at most $kf_{CNF}(k)$, this gives us that $\ell_{CNF}(k) < kf_{CNF}(k)$ implying

$$f_{CNF}(k) \geq \left\lfloor \frac{\ell_{CNF}(k)}{k} \right\rfloor + 1 \geq \left\lfloor \frac{2^k}{ek} \right\rfloor.$$

5.2 Trees

One tool that will help us with our upper bounds are the (k, d)-trees, first described by H. Gebauer in [4]. These trees are proper binary trees, i.e., rooted trees in which all nodes (except leaves) have exactly two children.

We say for two vertices u, w that u is an ancestor of w if u appears on the path between w and the root. We say that u, w are *k-close* if their graph distance is at most k and u is an ancestor of w.

Definition 5.2.1 (**(k, d)-tree**)**.** A proper binary tree is defined to be a (k, d)-*tree* if it satisfies the following two properties:

(i) Every leaf has depth at least k.
(ii) For every node u of the tree, there are at most d leaves k-close to u.

For each k we consider the minimum d such that there exits a (k, d)-tree.

Theorem 5.2.2. *Let $f_{tree}(k)$ be the smallest integer d such that a (k, d)-tree exists. Then,*

$$f_{tree}(k) \leq \left(\frac{2}{e} + O\left(\frac{1}{\sqrt{k}}\right)\right)\frac{2^k}{k}.$$

This result was the main subject of [5]. We postpone its proof to Chapter 6. We remark that a lower bound

$$f_{tree}(k) > \left\lfloor \frac{2^{k+1}}{e(k+1)} \right\rfloor$$

is also proved in the same article using the lopsided local lemma.

Given a proper binary tree T with leaves of depth at least k we can construct a k-CNF formula $F_k(T)$ associated with T. For each non-leaf node

$u \in V(T)$ we know that it has two children, so we create a variable x_u and label its children with the positive literal x_u and the negative literal $\neg\, x_u$ respectively.

Now for every leaf $w \in V(T)$ we will create a clause C_w, defined to be the disjunction of the literal at w and the first $k-1$ literals which occur as we walk from l towards the root. The conjunction of all the clauses C_w (for w a leaf of T) will be $F_k(T)$, our k-CNF formula.

Proposition 5.2.1. *Let T be a proper binary tree with leaves of depth at least k. Then $F_k(T)$ is unsatisfiable.*

Proof. Let α be any truth assignment of the variables of $F_k(T)$. Now we will describe a path from the root to some leaf. Beginning at the root we create the path by selecting, at each step, the literal which is assigned "false" by α. In other words, given the current path from the root to a vertex u we extend the path to the child labelled x_u if x_u is assigned "false" by α, and we extend to $\neg\, x_u$ otherwise. The path halts when it arrives at a leaf l. We observe that the clause C_l evaluates as $false$ under α as it is a disjunction of literals assigned as false. And so α is not a valid assignment for $F_k(T)$. Since α is arbitrary, it follows that $F_k(T)$ is unsatisfiable. □

5.3
The upper bound of the f function

For any CNF formula F we can create a hypergraph $H(F)$ from it. Let the vertices of H be all the possible literals of F, and let the hyperedges be the clauses of F.

For example, given the CNF formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$ we would associate with it the hypergraph with vertices $\{x_1, x_2, x_3, \neg x_1, \neg x_2, \neg x_3\}$ and edges $\{x_1, x_2, x_3\}$ and $\{x_1, \neg x_2, \neg x_3\}$.

Proposition 5.3.1. *If F is unsatisfiable, then $H(F)$ is Maker win*

Proof. Since going second can only harm Maker, we can assume that Maker will be going second. Maker's strategy will be as follows: whenever Breaker picks a literal u, Maker picks $\neg\, u$. Once the game is complete, we may consider the following truth assignment: all Breaker's literals are *true*. Since F is unsatisfiable, under this assignment F must be false. And so at least one of the clauses of F is false, which means that one entire edge of $H(F)$ does not have vertices from Breaker, which means that Maker has the entire edge in question. □

Theorem 5.3.1. $f(k) \leq 2f_{tree}(k-2) = \left(1 + O\left(\frac{1}{\sqrt{k}}\right)\right)\frac{2^k}{ek}$

In the proof of Theorem 5.3.1 we will define a tree T and then obtain the hypergraph H which gives us the required bound by:

(i) Considering $F_k(T)$, the k-CNF formula associated with the tree, and then

(ii) we obtain H as $H(F_k(T))$, the hypergraph associated with this k-CNF formula.

It is clear that this is a valid strategy for creating a Maker win hypergraph as Proposition 5.2.1 will give us that $F_k(T)$ is unsatisfiable and Proposition 5.3.1 will then give us that the hypergraph is Maker win.

We recall that in a $(k-2,d)-tree$ all leaves have depths at least $k-2$. We will choose T to be a $(k-2,d)$-tree. However, in order to apply Proposition 5.2.1 we will require that all its leaves have depth at least k, rather than $k-2$.

Proof. We begin by constructing a $(k-2, f_{tree}(k-2))$-tree T in which each leaf has depth at least k.

Let T_1, T_2, T_3 and T_4 be four copies of an arbitrary $(k-2, f_{tree}(k-2))$-tree. We will construct T using these four trees and three new vertices v_1, v_2 and r. Attach v_1 to the root vertex of T_1 and also to the root vertex of T_2, attach v_2 to the root vertex of T_3 and to the root vertex of T_4. Note that with this we have two $(k-2, f_{tree}(k-2))$-tree where each leaf has depth at least $k-1$, since with the addition of v_1 we incremented the depth of each node from T_1 and T_2 by 1, and the same holds with v_2, T_3 and T_4. Finally, attach r to v_1 and v_2. We consider r to be the root of the resulting tree T. Clearly T is a $(k-2, f_{tree}(k-2))$-tree in which each leaf has depth at least k.

Now that we have defined T we may follow the strategy discussed above. We may take $F_k(T)$ to be the k-CNF formula associated with T, and $H = H(F_k(T))$ to be the associated hypergraph. We have that $F_k(T)$ is unsatisfiable by Proposition 5.2.1 and that H is Maker win by Proposition 5.3.1.

Now, all that remains is to prove that the degrees in H are all at most $2f_{tree}(k-2)$. We begin by recalling that a vertex v of H corresponds to a literal l of $F_k(T)$ which in turn corresponds to a node n of T. On the other hand, an edge containing v in H, corresponds to a clause containing l in $F_k(T)$ which in turn corresponds to a path of k nodes in T which ascends from some leaf w of T. Thus, the degree of v in H is simply given by the number of $k-1$-close leaves from the corresponding node n of T.

We now complete the proof by showing that for every node n of T there are at most $2f_{tree}(k-2)$ leaves $(k-1)$-close to n. This is immediate for the root r as it has distance at least k from all leaves. Let n be any other node and note that any leaf $(k-1)$-close to n is $(k-2)$-close to one of its children. Since these children necessarily belong to the trees T_i which are $(k-2, f_{tree}(k-2))$-trees, each child node is $(k-2)$-close to at most $f_{tree}(k-2)$ leaves. It follows that each node n is $(k-1)$-close to at most $2f_{tree}(k-2)$ leaves. This proves the bound we need.

□

Using the same construction given in the proof of Theorem 5.3.1, if we analyse how many edges intersect each other we come up with the statement

$$\Delta(L(H)) = (k-1)f_{tree}(k-2) = \left(1 + O(k^{-1/2})\right)\frac{2^{k-1}}{e},$$

where $L(H)$ denotes the *line-graph* of H, so $\Delta(L(H))$ denotes the maximum neighborhood of the hypergraph. Note that this is a counterexample to the Neighborhood Conjecture, since the conjecture claims that any hypergraph H with $\Delta(L(H)) < 2^{k-1}$ is a Breaker win.

6
The upper bound on f_{tree}

We now discuss the upper bound on $f_{tree}(k)$, which appears in [5] and which we stated as Theorem 5.2.2 in the previous chapter.

6.1
Preliminaries tools and definitions

The key characteristic about (k, d)-trees is their leaf distribution, more precisely how many leaves it has at a certain depth. Hence, when constructing one, it will be useful to introduce the concept of a leaf-vector. For a given node v in an arbitrary tree T, a vector $\vec{x}$ of non-negative integers $(x_0, x_1, ..., x_k)$ is a *leaf-vector* for v if v has at most x_i leaf-descendants at distance exactly i for each $0 \leq i \leq k$. For example, a leaf-vector of a leaf could be $(1, 0, ..., 0)$, a leaf-vector for the root r of a full binary tree with height s could be $\vec{y} = (\underbrace{0, ..., 0}_{s}, 2^s, 0, ..., 0)$, or indeed any vector $\vec{z} = (z_0, z_1, ..., z_s, ...z_k)$ with non-negative entries and $z_s \geq 2^s$. Set $|\vec{x}| = \sum_{i=0}^{k} x_i$, so now in our previous example we could say that $\vec{y}$ is the smallest leaf-vector for r, as it minimizes $|\vec{y}|$ among all possible leaf-vectors. Note that every node v in a (k, d)-tree has a leaf-vector $\vec{x}$ with $|\vec{x}| \leq d$.

Definition 6.1.1 ($(k, d, \vec{x})$-tree)**.** Given $\vec{x} \in \mathbb{N}^{k+1}$, a $(k, d, \vec{x})$-tree is a tree where

(i) $\vec{x}$ is a leaf-vector for the root.

(ii) For every node u of the tree, there are at most d leaves k-close to u.

Definition 6.1.2 ((k, d)-constructible)**.** Given $\vec{x} \in \mathbb{N}^{k+1}$, $k \in \mathbb{N}$, $d \in \mathbb{N}$, $\vec{x}$ is (k, d)-constructible if a $(k, d, \vec{x})$-tree exists.

Now with this definition, in the discussion that we had earlier we could say that $(1, 0, ..., 0)$ is (k, d)-constructible for any $k \geq 1$ and $d \geq 2$. Also, we have that $(\underbrace{0, ..., 0}_{s}, 2^s, 0, ..., 0)$ is (k, d)-constructible with $2^s \leq d$.

Now we show the trivial link between (k, d)-trees and constructible vectors.

Proposition 6.1.1. *Given $k \in \mathbb{N}, d \in \mathbb{N}$, a (k, d)-tree exists if and only if the vector $(0, ..., 0, d)$ is constructible.*

Proof. $(0, ..., 0, d)$ is constructible if and only if a $(k, d, (0, ..., 0, d))$-tree exists. So we must show that a (k, d)-tree and a $(k, d, (0, ..., 0, d))$-tree are equivalent. We recall that both are trees with two conditions, the conditions (ii) are the same. The condition that $(0, ..., 0, d)$ is a leaf-vector for the root implies that no leaf has depth less than $k - 1$, while a tree where every leaf has depth at least k implies that $(0, ..., 0, x)$ must be a leaf-vector for the root for some x, applying (ii) for the root we have that it holds for $x = d$. So conditions (i) are equivalent.

□

The next proposition will allow to handle how the leaf-vectors change as we go to a child node.

Proposition 6.1.2. *Let $\vec{x} = (x_0, x_1, ..., x_k)$ and $\vec{y} = (y_0, y_1, ..., y_k)$ be (k, d)-constructible vectors and take $\vec{z} = (0, x_0 + y_0, x_0 + y_0, ..., x_{k-1} + y_{k-1})$ with $|\vec{z}| \leq d$. Then $\vec{z}$ is also (k, d)-constructible.*

Proof. Let T' be a $(k, d, \vec{x})$-tree with root r' and T'' be a $(k, d, \vec{y})$-tree with root r'', create a vertex r, and attach it to r' and r'', name the resulting tree as T. The root r is not a leaf, so there are no leaves at distance 0 from r. A leaf of T at distance $i \geq 1$ is either a leaf of T' with distance $i - 1$ from r' or a leaf of T'' with distance $i - 1$ from r'', this gives us that $\vec{z}$ is a leaf-vector for r.

Finally, let us observe that no node in T is k-close to more than d leaves. This holds for nodes in T' and T'', by condition (ii) of the definition. For r, the root vertex we have introduced, this is true by the assumption that $|z| \leq d$.

□

Define the *weight* of a vector $\vec{x} = (x_0, x_1, ..., x_k)$ as $w(\vec{x}) = \sum_{i=0}^{k} \frac{x_i}{2^i}$.

Lemma 6.1.1. *Let $\vec{x} \in \mathbb{N}^{k+1}$ with $|\vec{x}| \leq d$. If $w(\vec{x}) \geq 1$ then $\vec{x}$ is (k, d)-constructible.*

Proof. We will recursively construct a binary tree that is a $(k, d, \vec{x})$-tree.

Begin with only the root at step 0.

Then, given the tree after i steps, we ask whether $\sum_{j=0}^{i} x_j/2^j < 1$. If it is the case that $\sum_{j=0}^{i} x_j/2^j < 1$ then choose x_i nodes to be leaves and add two children to all the other $2^i(1-\sum_{j=0}^{i} x_j/2^j)$ nodes. On the other hand, if at step i we have $\sum_{j=0}^{i} x_j/2^j \geq 1$ we stop and set all remaining nodes as leaves.

The fact that $w(\vec{x}) \geq 1$ guarantees that our procedure stops. In fact the tree we produce has depth $k' \leq k$ where k' is minimal such that $\sum_{j=0}^{k'} x_j/2^j \geq 1$. The total number of leaves is $\sum_{j=0}^{k'} x_j \leq |\vec{x}| \leq d$, and so condition (ii) holds. Also the number of leaves at distance j from the root is at most x_j. So, $\vec{x}$ is a leaf-vector for the root. Since we have constructed a tree with the two required conditions (i) and (ii) we have a $(k, d, \vec{x})$-tree.

□

For a vector $\vec{x} = (x_0, x_1, ..., x_k)$ define

$$E'(\vec{x}) = (x_1/2, x_2/2, ..., x_k/2, d/2).$$

The idea behind the definition of E' is how a leaf-vector can be passed to its children. When constructing a $(k, d, \vec{x})$-tree, we could begin with only one node r that will have $\vec{x}$ as its leaf-vector in the end of our construction. So, adding the two children of r one possibility is to split the future leaf decedents of r evenly between the children, giving them each $E'(\vec{x})$ as the promised leaf-vector. Note if we have $|\vec{x}| \leq d$ then, as discussed before, condition (ii) of a (k, d)-tree holds and also for $E'(\vec{x})$.

With the same idea, let $r, l \in \mathbb{N}$ with $1 \leq l \leq r \leq k$ define

$$\vec{x}' = (x_l, x_{l+1}, ..., x_r, 0, ..., 0).$$
$$\vec{x}'' = (\underbrace{0, ..., 0}_{r-l+1}, \frac{x_{r+1}}{2^l - 1}, \frac{x_{r+2}}{2^l - 1}, ..., \frac{x_k}{2^l - 1}, \underbrace{0, ..., 0}_{l}).$$

Here the operation would be to include below r a complete binary tree with height l and then we choose one leaf to set $\vec{x}'$ as the promised leaf-vector and for the $2^l - 1$ remaining leaves we set $\vec{x}''$ as the promised leaf-vector. The idea is to set all the leaves in the $[l, r]$ range to one node and for the other ones we do a split similar with the E' split. Notice that with this split for each x'_i with $l \leq i \leq r$ we have its weight multiplied by 2^l, also for $\vec{x}''$ we have a tiny weight gain, a factor of $2^i/(2^i - 1)$ since we shifted each x_i with $r + 1 \leq i \leq k$ by l and divided by $2^l - 1$. The idea with this operation is to be able to use Lemma 6.1.1.

Now we give a more precise definition for the operations discussed above.

Let d, k, l be positive fixed integers, set

$$d' = d\Big(1 - \frac{1}{2^l - 1}\Big)$$

For a vector $\vec{x} = (x_0, x_1, ..., x_k)$ we define

$$E(\vec{x}) = (\lfloor x_1/2 \rfloor, \lfloor x_2/2 \rfloor, ..., \lfloor x_k/2 \rfloor, \lfloor d'/2 \rfloor)$$

And for an integer m

$$E^m(\vec{x}) = \left(\left\lfloor \frac{x_m}{2^m} \right\rfloor, \left\lfloor \frac{x_{m+1}}{2^m} \right\rfloor, ..., \left\lfloor \frac{x_k}{2^m} \right\rfloor, \left\lfloor \frac{d'}{2^m} \right\rfloor, \left\lfloor \frac{d'}{2^{m-1}} \right\rfloor, ..., \left\lfloor \frac{d'}{2} \right\rfloor\right)$$

is the resulting vector when we apply E m times. Note that with each E application we left shift the coordinates by 1 and divide them by 2.

For $l \leq r \leq k$ define

$$C_r(\vec{x}) = \Big(\underbrace{0, ..., 0}_{r-l+1}, \underbrace{\left\lfloor \frac{x_{r+1}}{2^l - 1} \right\rfloor, ..., \left\lfloor \frac{x_k}{2^l - 1} \right\rfloor}_{k-r}, \underbrace{\left\lfloor \frac{d'}{2^l} \right\rfloor, \left\lfloor \frac{d'}{2^{l-1}} \right\rfloor, ..., \left\lfloor \frac{d'}{2} \right\rfloor}_{l}\Big)$$

$$C_r^*(\vec{x}) = (\underbrace{x_l, x_{l+1}, ..., x_r}_{r-l+1}, \underbrace{0, ..., 0}_{k-r+l})$$

Lemma 6.1.2. *Let k, d, l be positive integers and $\vec{x} \in \mathbb{N}^{k+1}$ with $|\vec{x}| \leq d$ then*

(a) $|E(\vec{x})| \leq d$. If $E(\vec{x})$ is (k, d)-constructible, then $\vec{x}$ is (k, d)-constructible.

(b) For $l \leq r \leq k$ we have $|C_r(\vec{x})| \leq d$ and $|C_r^*(\vec{x})| \leq d$. Furthermore, if $C_r(\vec{x})$ and $C_r^*(\vec{x})$ are (k, d)-constructible and $|C_r^*(\vec{x})| \leq d/2^l$, then $\vec{x}$ is (k, d)-constructible.

Proof. (a)

$$|E(\vec{x})| \leq \sum_{i=1}^{k} \left\lfloor \frac{x_i}{2} \right\rfloor + \frac{d'}{2} \leq \frac{|\vec{x}|}{2} + \frac{d'}{2} < \frac{d}{2} + \frac{d}{2} = d.$$

If $E(\vec{x})$ is (k, d)-constructible, there exists a $(k, d, E(\vec{x}))$ tree T. Take two copies of T with roots r' and r''. Create a vertex r and attach it to r' and r'', the resulting tree is a $(k, d, \vec{x})$-tree, so $\vec{x}$ is (k, d)-constructible.

(b) For C_r^* we have trivially that $|C_r^*(\vec{x})| = \sum_{i=l}^{r} x_i \leq \sum_{i=0}^{k} x_i = |\vec{x}| \leq d$ since $x_i \geq 0$ for every i. For C_r we have

$$|C_r(\vec{x})| \leq \sum_{i=r+1}^{k} \frac{x_i}{2^l - 1} + \sum_{i=1}^{l} \frac{d'}{2^i}$$

using $\sum_{i=r+1}^{k} \frac{x_i}{2^l - 1} \leq \frac{\vec{x}}{2^l - 1} \leq \frac{d}{2^l - 1}$ and $\sum_{i=1}^{l} \frac{d'}{2^i} \leq d' \sum_{i=1}^{\infty} \frac{1}{2^i} = d'$ we have

$$|C_r(\vec{x})| \leq \frac{d}{2^l - 1} + d' = d' \leq d.$$

Let T' be a $(k, d, C_r(\vec{x}))$-tree and T^* be a $(k, d, C_r^*(\vec{x}))$-tree, let T be a complete binary tree with height l, attach T^* to one of its leaves, and a copy of T' to each of the remaining $2^l - 1$ leaves. We claim that the resulting tree is a $(k, d, \vec{x})$-tree.

Now as always we need to check conditions (i) and (ii) of a $(k, d, \vec{x})$-tree for T.

(i) Note that we do not have leaves at depth less than l, for leaves at distance $l \leq j \leq r$ we only have leaves from T^* and for leaves at distance $r < j \leq k$ we have them all inside the $2^l - 1$ copies of T'. Therefore we have

$$\vec{y} = \left(0, ..., 0, x_l, x_{l+1}, ..., x_r, (2^l - 1)\left\lfloor \frac{x_{r+1}}{2^l - 1} \right\rfloor, ..., (2^l - 1)\left\lfloor \frac{x_k}{2^l - 1} \right\rfloor\right)$$

as a leaf-vector for the root of T. Since we have $y_i \leq x_i$ for every $0 \leq i \leq k$, $\vec{x}$ is also a leaf-vector for the root of T.

(ii) Since $|\vec{x}| \leq d$, condition (ii) holds for the root of T.

For vertices with depth at least l, we have vertices that belong to T^* or to a copy of T', so, for them the condition (ii) follows from the equivalent condition for the trees T'.

Finally, for nodes at depth $0 < j < l$ we have two types of nodes. Nodes with 2^{l-j} copies of T' below them, or nodes with $2^{l-j} - 1$ copies of T' and the one copy of T^* below them.

Lets say that v is a node of the first case, then the number of leaves k-close to v is

$$\begin{aligned}
2^{l-j} \cdot |C_r(\vec{x})| &= 2^{l-j}\Big(\frac{x_{r+1} + ... + x_k}{2^l - 1} + \frac{d'}{2^l} + \frac{d'}{2^{l-1}} + ... + \frac{d'}{2^{l-j+1}}\Big) \\
&\leq \frac{2^{l-j}}{2^l - 1} \cdot d + d'\Big(1 - \frac{1}{2^j}\Big) \\
&= \frac{d}{2^l - 1}\Big(2^{l-j} + (2^l - 2)\Big(1 - \frac{1}{2^j}\Big)\Big) \\
&\leq \frac{d}{2^l - 1}(2^l - 1) \\
&= d\,.
\end{aligned}$$

Note that the term $\frac{d'}{2^{l-j+1}}$ are leaves at distance k from v.

For the second case the number of leaves k-close to v is

$$\begin{aligned}
&(2^{l-j} - 1)\Big(\frac{x_{r+1} + ... + x_k}{2^l - 1} + \frac{d'}{2^l} + \frac{d'}{2^{l-1}} + ... + \frac{d'}{2^{l-j+1}}\Big) + |C_r^*(\vec{x})| \\
&\quad\leq (2^{l-j} - 1)\frac{d}{2^{l-j}} + \frac{d}{2^l} \\
&\quad\leq d.
\end{aligned}$$

□

6.2
The proof

Now we are prepared to prove the upper bound of the f_{tree}. To do so we will show the constructability of the vector $(0, ..., 0)$ for a large enough k and $d = \left(\frac{2}{e} + O\left(\frac{1}{\sqrt{k}}\right)\right)\frac{2^k}{k}$. This is sufficient as the constructibility of $(0, ..., 0)$ implies the constructibility of $(0, ..., 0, d)$ which in turn shows that a (k, d)-tree exists by Proposition 6.1.1.

We will prove that $\vec{0} = (0, ..., 0)$ is constructible using the above lemmas, Lemma 6.1.1 and Lemma 6.1.2. With these lemmas in mind, we show that there exists a sequence of the operations C and E which, when applied to $\vec{0}$, produce a vector with weight at least 1. This final vector is then constructible by Lemma 6.1.1. We use this as the start point of a backwards induction that allows us to deduce that $\vec{0}$ is constructible.

Proof. Let $d = \lfloor 2^{k+1}/(ek) + 100 \cdot 2^{k+1}/k^{3/2} \rfloor$.

Set $l = \lfloor \log(k)/2 \rfloor$, this way we have $2^l \sim \sqrt{k}$.

Define recursively vectors $\vec{x}^{(t)} = (x_0^{(t)}, x_1^{(t)}, ..., x_k^{(t)}) \in \mathbb{N}^{k+1}$ starting with $\vec{x}^{(0)} = E^{k-2l}(\vec{0})$, and for $t \geq 0$ set

$$\vec{x}^{(t+1)} = E^{r_t-3l}(C_{r_t}(\vec{x}^{(t)}))$$

where r_t is the the smallest integer $3l \leq r_t \leq k$ with $\sum_{j=0}^{r_t} x_j^{(t)}/2^j \geq 2^{-l}$, namely the smallest integer such that if we do the weight calculation for $\vec{x}^{(t)}$ (recall that $w(\vec{x}) = \sum_{j=0}^{k} x_j/2^j$), summing from 0 to k, r_t would be the first index where the weight becomes equal or greater than 2^{-l}.

The more attentive reader should question if the total weight of $\vec{x}^{(t)}$ could be smaller than 2^{-l}. In this case r_t would not be defined. We will show that this is not the case and that the $\vec{x}^{(t)}$ sequence is indeed well-defined.

Some things to notice:

When applying E we shift each coordinate by 1, divide it by 2 and add $d'/2$ to the last coordinate, so, $\vec{x}^{(0)}$ looks like $(\underbrace{0, ..., 0}_{2l+1}, \lfloor d'/2^{k-2l} \rfloor, ..., \lfloor d'/2 \rfloor)$, given that for any $a \in \mathbb{R}$ and $b \in \mathbb{N}$, $(\lfloor \frac{\lfloor a \rfloor}{b} \rfloor = \lfloor \frac{a}{b} \rfloor)$.

For $C_r(\vec{x})$, the first $r-l+1$ coordinates are set to 0, so we have that the first r_t-l+1 coordinates of $C_{r_t}(\vec{x}(t))$ are zeros, when we apply E^{r_t-3l} we lose the first r_t-3l coordinates due to the left shift, keeping $(r_t-l+1)-(r_t-3l) = 2l+1$ zeros in the beginning of the vector. So for all integer $t \geq 0$ we have that the first $2l+1$ coordinates of $\vec{x}^{(t)}$ are zeros.

The other entries are obtained by taking an integer and dividing it by a power of 2 or by $(2^l - 1)$ and then taking the floor, so we can write them as $\lfloor \frac{d'}{2^i(2^l-1)^c} \rfloor$ for some non-negative integers i and c, if we think that we could always divide by 2 and multiply by $\alpha = 2^l/(2^l - 1)$ whenever it should be a division by $(2^l - 1)$ instead of 2 we would arrive at $\lfloor \frac{d'}{2^{i+lc}} \alpha^c \rfloor$.

Now for each j such that $2l < j \leq k$ we may write $\vec{x}_j^{(t)}$ as

$$\vec{x}_j^{(t)} = \left\lfloor \frac{d'}{2^{c'(t,j)}} \alpha^{c(t,j)} \right\rfloor \tag{6.1}$$

our task is to determine the functions c' and c.

So (6.1) can be written as

$$\vec{x}_j^{(t)} = \left\lfloor \frac{d'}{2^{k-j+1}} \alpha^{c(t,j)} \right\rfloor \tag{6.2}$$

To understand $c'(t, j)$ we note that this simply counts the total number of leftward shifts that occurred to move from the last position to the current

position (as we divide by 2 with each shift) plus one (as the kth position is $d'/2$). As $\vec{x}_j^{(t)}$ is in position j and the last position is k, we have $c'(t,j) = k - j + 1$.

We now consider $c(t,j)$. One factor of α is introduced for each application of C. So $c(t,j)$ counts the number of iterations which have been applied to the entry $\vec{x}_j^{(t)}$.

To answer this we must think about where $\vec{x}_j^{(t)}$ came from. In the last iteration (from $t-1$ to t) we shifted entries leftwards by l positions (with the application of C_{r_t}) and further leftwards by $r_t - 3l$ positions (with the application of E^{r_t-3l}). In this sense $\vec{x}_j^{(t)}$ "comes from" the position $j + r_t - 2l$. We set $q_t = r_t - 2l$, which represents the distance shifted.

Since the vector ends with its kth entry, the number of iterations involving $\vec{x}_j^{(t)}$ is the largest c such that $\sum_{i=t-c+1}^{t} q_i \leq k - j$. And so

$$c(t,j) = \max\left\{ c \in \{1,\ldots,t+1\} : \sum_{i=t-c+1}^{t} q_i \leq k - j \right\}.$$

We are now ready to plan the remainder of the proof. The main steps are

(i) Show that the sequence $\vec{x}^{(t)}$ is well defined,

(ii) Show that the sequence of vectors $\vec{x}^{(t)}$ converges to a vector $\vec{x}$,

(iii) Show that this convergence occurs in at most d steps, so that $\vec{x}^{(t)} = \vec{x}$ for all $t \geq d$,

(iv) Show that $w(\vec{x}^{(d)}) \geq 1$, which gives us that $\vec{x}^{(d)}$ is constructible, and finally,

(v) Use this as the base case of a backwards induction which shows that $\vec{x}^{(t)}$ is constructible for each $0 \leq t \leq d$.

We begin with task (i). That is, we must show that $\vec{x}^{(t)}$ is well defined. This requires us to show that r_t exists for all t. And so it suffices to show that $w(\vec{x}^{(0)}) \geq 2^{-l}$ and that $w(\vec{x}^{(t)})$ is increasing in t. We shall in fact prove that both $x_j^{(t)}$ and the sequence $c(t,j)$ increases monotonically with t for each fixed $2l < j \leq k$ and q_t decreases monotonically in t.

Let us do induction on t. We have $c(0,j) = 0$ for all j, so $c(1,j) \geq c(0,j)$. If we have that $c(t+1,j) \geq c(t,j)$ for all j, then we have $\vec{x}_j^{(t+1)} \geq \vec{x}_j^{(t)}$ for all j by (6.2), note that $c'(t,j) = k - j + 1$ is independent of t. It follows that

$r_{t+1} \leq r_t$ since we will sum up to 2^{-l} with the same number or fewer terms. We also have $q_{t+1} \leq q_t$, as we recall that $q_t = r_t - 2l$. Since $q_0 \geq q_1 \geq ... \geq q_{t+1}$, by the definition of $c(i,j)$ we have $c(t+2,j) \geq c(t+1,j)$.

Now that $\vec{x}_j^{(t)}$ is increasing in t, we do also have that the weight of $\vec{x}^{(t)}$ is increasing. We now prove that $w(\vec{x}^{(0)}) \geq 2^{-l}$.

Calculating $w(\vec{x}^{(0)})$

$$\begin{aligned}\sum_{j=2l+1}^{k} \frac{\left\lfloor \frac{d'}{2^{k+1-j}} \right\rfloor}{2^j} &> \sum_{j=2l+1}^{k} \frac{\frac{d'}{2^{k+1-j}} - 1}{2^j} = \sum_{j=2l+1}^{k} \frac{d'}{2^{k+1}} - 2^{-j} \\ &> (k-2l)\frac{d'}{2^{k+1}} - 2^{-2l} = (k-2l)\frac{d(1-\frac{1}{2^l-1})}{2^{k+1}} - 2^{-2l} \\ &> (k-\log k)\frac{1-\frac{1}{2^l-1}}{ek} - 2^{-\log k+2} \longrightarrow \frac{1}{e}\end{aligned}$$

In the last inequality we used $d = \lfloor 2^{k+1}/(ek) + 100 \cdot 2^{k+1}/k^{3/2} \rfloor > 2^{k+1}/(ek)$. Then we have for large enough k, $w(\vec{x}^{(0)}) > 2^{-l}$.

We now prove (ii), that the sequence $\vec{x}^{(t)}$ converges. The sequence $\vec{x}^{(t)}$ is well-defined and monotonic increasing in each coordinate, and each vector was obtained applying E and C many times to the zero vector $\vec{0}$, so by Lemma 6.1.2 we have $|\vec{x}^{(t)}| \leq d$ for all t. It follows that each coordinate of $\vec{x}^{(t)}$ is increasing and bounded above and so converges. Thus the vector $\vec{x}^{(t)}$ converges.

We shall now show (iii), that the convergence is rapid. Note that, as soon as two consecutive vectors agree on all coordinates, then all future vectors will be the same. Since we are dealing with integers, and the sum of entries is at most d, the sequence must converge in at most d steps. More formally, there exists a vector $\vec{x} = (x_0, x_1, ..., x_k)$ such that $\vec{x}^{(t)} = \vec{x}$ for all $t \geq d$.

With this we also have that q_t converges to some q, and that $q_t = q$ for all $t \geq d$. It now follows that equation (6.1) can be written for some $t > d + k$ as

$$x_j = \left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{\lfloor (k-j)/q \rfloor} \right\rfloor \tag{6.3}$$

Note that $q \geq l$ since $q = q_t = r_t - 2l$, where $3l \geq r_t \geq k$.

Before moving on to (iv) we prove the following claim.

Claim: $q = l$

Proof of Claim: We will show this by contradiction, assume $q > l$.

So, we have by the minimality of r_t that

$$\begin{aligned}
2^{-l} &> \sum_{j=0}^{r_t-1} \frac{x_j}{2^j} \\
&= \sum_{j=2l+1}^{2l+q-1} \frac{\left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{\lfloor (k-j)/q \rfloor} \right\rfloor}{2^j} \quad \text{(the first } 2l+1 \text{ entries of } x_j \text{ are 0)} \\
&> \sum_{j=2l+1}^{2l+q-1} \frac{\frac{d'}{2^{k+1-j}} \alpha^{\frac{k-j}{q}-1} - 1}{2^j} \\
&= \sum_{j=2l+1}^{2l+q-1} \left(\frac{d'}{2^{k+1}} \alpha^{\frac{k-j}{q}-1} - 2^{-j} \right)
\end{aligned}$$

using that $\frac{j}{q} \leq \frac{2l+q}{q} = 1 + \frac{2l}{q} \leq 1 + \frac{2l}{l} = 3$ we have

$$\geq \frac{d'}{2^{k+1}} \alpha^{\frac{k}{q}-4} \left(\sum_{j=2l+1}^{2l+q-1} 1 \right) + \left(\sum_{j=2l+1}^{2l+q-1} -2^{-j} \right)$$

using that $\sum_{j=2l+1}^{2l+q-1} 2^{-j} < \sum_{j=2l+1}^{\infty} 2^{-j} = 2 \cdot 2^{-2l-1}$, we arrive at

$$2^{-l} > (q-1) \frac{d'}{2^{k+1}} \alpha^{\frac{k}{q}-4} - 2^{-2l}$$

rearranging the terms we have

$$2^{-l}(1 + 2^{-l}) \alpha^4 \frac{2^{k+1}}{d'} > (q-1) \alpha^{\frac{k}{q}}$$

Now we will minimize $g(q) = (q-1)\alpha^{\frac{k}{q}}$ over real numbers $q > 2$. Taking the derivative dg/dq we obtain

$$\alpha^{k/q} - (q-1) \frac{\alpha^{k/q} k \ln \alpha}{q^2}.$$

And setting this equal to 0 gives

$$q \cdot \frac{q}{q-1} = k \ln(\alpha).$$

So we have that the minimum of $g(q)$ is obtained with $k \ln \alpha - 2 \leq q \leq k \ln \alpha - 1$, and for that minimum we have $g(q) > (k \ln \alpha - 3)e$. Using $\alpha = 2^l/(2^l - 1) > e^{2^{-l}}$ we have $k \ln \alpha \geq k/2^l$, so applying our observations

$$2^{-l}(1+2^{-l})\alpha^4\frac{2^{k+1}}{d'} > \frac{ke}{2^l} - 3e$$

Note that we arrived at this inequality by taking the lowest value possible of $(q-1)\alpha^{k/q}$.

Rearranging again we have

$$(1+2^{-l})\alpha^4\frac{2^{k+1}}{d'} - ke + 3e2^l > 0$$

Now substituting $d, d' = d\left(1-\frac{1}{2^l-1}\right), \alpha = \left(1+\frac{1}{(2^l-1)}\right), l = \lfloor \log(k)/2 \rfloor$ and k large enough we have that the right hand side is at most

$$\begin{aligned}
&\left(1+\frac{2}{\sqrt{k}}\right)\left(1+\frac{4}{\sqrt{k}}\right)^4\frac{2^{k+1}}{d\left(1-\frac{4}{\sqrt{k}}\right)} - ek + 3e\sqrt{k} \\
\leq &\left(1+\frac{19}{\sqrt{k}}\right)\frac{ek}{\left(1-\frac{4}{\sqrt{k}}\right)\left(1+\frac{99e}{\sqrt{k}}\right)} - ek + 3e\sqrt{k} \\
\leq &\left(1-\frac{200}{\sqrt{k}}\right)ek - ek + 3e\sqrt{k} \\
= &-200e\sqrt{k} + 3e\sqrt{k} < 0
\end{aligned}$$

which is a contradiction, implying $q = l$. This completes the proof of the Claim.

Now we have all the tools we require complete steps (iv) and (v). As a result, we shall prove that all $\vec{x}^{(t)}$ are constructible and this completes the proof of the theorem. We do this by downward induction on t.

Starting with $t = d$ and using (6.3) with $q = l$ we have

$$\begin{aligned} w(\vec{x}^{(d)}) &= \sum_{j=0}^{k} \frac{\vec{x}_j^{(d)}}{2^j} \geq \frac{x_{2l+1}^{(d)}}{2^{2l+1}} \\ &\geq \frac{d'}{2^{k+1}} \alpha^{\frac{k-2l-1}{l}} - 1 \\ &\geq \frac{d'}{2^{k+1}} \alpha^{\frac{k}{2l}} - 1 \\ &\geq \frac{d/2}{2^{k+1}} \left(1 + \frac{1}{\sqrt{k}}\right)^{\frac{k}{\log k}} - 1 \\ &\geq \frac{1}{2ek} e^{\frac{k}{2\sqrt{k}} \log k} - 1 > 1 \end{aligned}$$

so by Lemma 6.1.1 $\vec{x}^{(d)}$ is (k, d)-constructible.

Having completed the base case ($t = d$) we now prove the induction step. Assuming that $\vec{x}^{(t+1)}$ is constructible for some $t \leq d-1$, we will show that $\vec{x}^{(t)}$ is constructible.

To show this we recall that $\vec{x}^{(t+1)} = E^{r_t - 3l}(C_{r_t}(\vec{x}^{(t)}))$. Using part (a) of Lemma 6.1.2 repeatedly we conclude that $C_{r_t}(\vec{x}^{(t)})$ is constructible. Now the only thing left is to show that $C^*_{r_t}(\vec{x}^{(t)})$ is constructible and $|C^*_{r_t}(\vec{x}^{(t)})| \leq d/2^l$ to be able to apply part (b) of Lemma 6.1.2 finishing the proof.

Constructibility of $C^*_{r_t}(\vec{x}^{(t)})$ will follow from the definition of r_t, which gives us that

$$\sum_{j=l}^{r_t} \frac{\vec{x}_j^{(t)}}{2^j} = \sum_{j=0}^{r_t} \frac{\vec{x}_j^{(t)}}{2^j} \geq 2^{-l}\,,$$

where the equality comes from the fact that $\vec{x}^{(t)}$ has its first $2l+1$ coordinates as zeros.

And the weight of $C^*_{r_t}(\vec{x}^{(t)})$ can be written as

$$w(C^*_{r_t}(\vec{x}^{(t)})) = \sum_{j=l}^{r_t} \frac{\vec{x}_j^{(t)}}{2^{j-l}} = 2^l \sum_{j=l}^{r_t} \frac{\vec{x}_j^{(t)}}{2^j} \geq 1$$

giving us that $C^*_{r_t}(\vec{x}^{(t)})$ is constructible by Lemma 6.1.1.

For $|C^*_{r_t}(\vec{x}^{(t)})|$, using the monotonicity of $\vec{x}_j^{(t)}$, we have that

$$|C^*_{r_t}(\vec{x}^{(t)})| = \sum_{j=2l+1}^{r_t} x_j^{(t)} \leq \sum_{j=2l+1}^{r_t} x_j\,.$$

We have by the monotonicity of r_t that $r_t \leq r_0 \leq k/2$ for large enough k,

recall that $\vec{x}^{(0)} = (0, ..., 0, \lfloor d'/2^{k-2l} \rfloor, ..., \lfloor d'/2 \rfloor)$. Using $q = l$ and (6.3) we have

$$\begin{aligned}
|C^*_{r_t}(\vec{x}^{(t)})| &\leq d' \sum_{2l+1}^{r_t} \frac{1}{2^{k+1-j}} \alpha^{\frac{k-j}{l}} \\
&\leq d \sum_{j=2l+1}^{r_t} \left(\frac{\alpha}{2}\right)^{k-j} \\
&\leq d \left(\frac{3}{4}\right)^{k-r_t} \cdot 4 \\
&\leq d \left(\frac{3}{4}\right)^{\frac{k}{2}} \cdot 4 \\
&< \frac{d}{\sqrt{k}} \\
&\leq \frac{d}{2^l}
\end{aligned}$$

And now being able to apply part (b) of Lemma 6.1.2 we finish our proof.

□

Bibliography

[1] BECK, J.. **Combinatorial Games: Tic-Tac-Toe Theory**. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2008.

[2] BERLEKAMP, E.; CONWAY, J. ; GUY, R.. **Winning Ways for Your Mathematical Plays**. Número v. 2. Taylor & Francis, 2003.

[3] ERDŐS, P.; SELFRIDGE, J.. **On a combinatorial game**. Journal of Combinatorial Theory, Series A, 14(3):298–301, 1973.

[4] GEBAUER, H.. **Disproof of the neighborhood conjecture with implications to sat**, 2009.

[5] GEBAUER, H.; SZABO, T. ; TARDOS, G.. **The local lemma is asymptotically tight for sat**, 2016.

[6] HALES, A. W.; JEWETT, R. I.. **Regularity and positional games**. Transactions of the American Mathematical Society, 106(2):222–229, 1963.

[7] KRATOCHVÍL, J.; SAVICKÝ, P. ; TUZA, Z.. **One more occurrence of variables makes satisfiability jump from trivial to np-complete**. SIAM J. Comput., 22:203–210, 1993.

[8] KNOX, F.. **Two constructions relating to conjectures of beck on positional games**, 2012.