Analytic Methods
and
Combinatorial Plants

by

Jeremy Chizewer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics & Optimization

Waterloo, Ontario, Canada, 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The thesis is based on the following papers completed in whole or in part during my time in the Master of Mathematics program at the University of Waterloo.

- Enumeration and Succinct Encoding for AVL Trees, joint work with Stephen Melczer, J. Ian Munro, and Ava Pun [11].
- The hat guessing number on cactus graphs and cycles, joint work with I.M.J. McInnis, Mehrdad Sohrabi, and Shriya Kaistha [10].
- On restricted intersections and the sunflower problem [9] (building on research conducted as an undergraduate at Princeton University).

Abstract

Combinatorial structures have broad applications in computer science, from error-correcting codes to matrix multiplication. Many analytic tools have been developed for studying these structures. In this thesis, we examine three applications of these tools to problems in combinatorics. By coincidence, each problem involves a combinatorial structure named for a plant–AVL trees, cactus graphs, and sunflowers–which we refer to collectively as combinatorial plants.

In our first result, we use a novel decomposition to create a succinct encoding for tree classes satisfying certain properties, extending results of Munro, Nicholson, Benkner, and Wild. This has applications to the study of data structures in computer science, and our encoding supports a wide range of operations in constant time. To analyze our encoding, we derive asymptotics for the information-theoretic lower bound on the number of bits needed to store these trees. Our method characterizes the exponential growth for the counting sequence of combinatorial classes whose generating functions satisfy certain functional equations, and may be of independent interest. Our analysis applies to AVL trees (a commonly studied self-balancing binary search tree in computer science) as a special case, and we show that about 0.938 bits per node are necessary and sufficient to encode AVL trees.

Next, we study the hat guessing game on graphs. In this game, a player is placed on each vertex v of a graph G and assigned a colored hat from $h(v)$ possible colors. Each player makes a deterministic guess on their hat color based on the colors assigned to the players on neighboring vertices, and the players win if at least one player correctly guesses his assigned color. If there exists a strategy that ensures at least one player guesses correctly for every possible assignment of colors, the game defined by $\langle G, h\rangle$ is called winning. The hat guessing number of G is the largest integer q so that if $h(v) = q$ for all $v \in G$ then $\langle G, h\rangle$ is winning. We determine whether $\langle G, h\rangle$ is winning for any h whenever G is a cycle, resolving a conjecture of Kokhas and Latyshev in the affirmative and extending it. We then use this result to determine the hat guessing number of every cactus graph, in which every pair of cycles shares at most one vertex.

Finally, we study the sunflower problem. A sunflower with r petals is a collection of r sets over a ground set X such that every element in X is in no set, every set, or exactly one set. Erdős and Rado [15] showed that a family of sets of size n contains a sunflower if there are more than $n!(r-1)^n$ sets in the family. Alweiss et al. [5], and subsequently Rao [37] and Bell et al. [6], improved this bound to $(O(r\log(n))^n$. We study the case where the pairwise intersections of the set family are restricted. In particular, we improve the

best-known bound for set families when the size of the pairwise intersections of any two sets is in a set L. We also present a new bound for the special case when the set L is the nonnegative integers less than or equal to d, using techniques of Alweiss et al. [5].

Acknowledgements

I would like to thank:

- Stephen Melczer for his help and guidance. I am extremely grateful for all the time he dedicated to helping me succeed and all the encouragement and feedback on countless papers, talks, and applications.
- My co-authors for their collaboration and contributions to the works in this thesis.
- J. Ian Munro and Olya Mandelshtam for reading my thesis.
- Andrew Odlyzko for discussions on the asymptotic behavior of AVL trees and the growth constant α.
- Noga Alon for connecting me to my co-authors on hat guessing.
- Ryan Alweiss, Noga Alon, and Robert Sedgewick for helpful suggestions on the sunflower problem, Lutz Warnke for telling me about [6], and The Fifty Five Fund for Senior Thesis Research (Class 1955) Fund, Princeton University for partially supporting the portion of this research conducted during my undergraduate program.
- NSERC and the Department of Combinatorics and Optimization at Waterloo for financial support.
- The Women in Mathematics committee for supporting the directed reading program which resulted in some of the work in this thesis
- My friends and family for their love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Generating Functions and Enumeration

Given a set $\mathcal{C}$, and a *weight* function $w : \mathcal{C} \to \mathbb{N}$ mapping objects in $\mathcal{C}$ to the nonnegative integers such that there are a finite number of elements in $\mathcal{C}$ of any weight, the *counting sequence* $c_0, c_1, \ldots$ of the *combinatorial class* defined by $\mathcal{C}$ and w is the sequence defined by

$$c_n = |\{c \in \mathcal{C} : w(c) = n\}|$$

for all n. In other words, c_n is the number of objects in $\mathcal{C}$ with *weight* n.

The *generating function* $C(z)$ for a combinatorial class $\mathcal{C}$ with counting sequence $c_0, c_1, \ldots$ is the formal power series

$$C(z) = \sum_{n \geq 0} c_n z^n.$$

Often combinatorial classes admit symbolic equations relating the objects in the class. For example, the class $\mathcal{B}$ of binary trees can be written as

$$\mathcal{B} = \circ + \mathcal{B} \times \bullet \times \mathcal{B} \tag{1.1}$$

where $\circ$ symbolizes a leaf node, $\bullet$ symbolizes an internal node, $+$ symbolizes union, and $\times$ symbolizes the cartesian product. Indeed, every binary tree is either the empty tree, or a node with left and right child binary trees. If we are interested in a counting sequence for binary trees where the weight function indicates the number of non-empty nodes in the

tree, then we see that $w(\bullet) = 1$ and $w(\circ) = 0$. Using this weight function, we can transfer the symbolic equation (1.1) to a function equation for the generating function

$$B(z) = 1 + B(z) \cdot z \cdot B(z). \tag{1.2}$$

Generating functions are a powerful tool for analyzing the asymptotic growth of a sequence from equations such as (1.2). We use standard notation when discussing the asymptotics of a sequence: a sequence $f_n = O(g_n)$ if there exists a constant $c > 0$ such that $f_n \leq cg_n$ for all $n \in \mathbb{N}$, we say that $f_n = o(g_n)$ if

$$\lim_{n\to\infty} \frac{f_n}{g_n} = 0,$$

and we write $f_n \sim g_n$ if

$$\lim_{n\to\infty} \frac{f_n}{g_n} = 1.$$

When $f_n = O(g_n)$ and $g_n = O(f_n)$, we say $f_n = \Theta(g_n)$ and when $f_n = o(g_n)$, we say $g_n = \omega(f_n)$.

When a generating function F represents a convergent series, there are strong links between the behavior of the series and the analytic behavior of F.

Theorem 1.1.1 (Vivanti-Pringsheim)**.** *If $F(z) = \sum_{n\geq 0} f_n z^n$ and $f_n \geq 0$ for all n then the radius of convergence R of this series equals the minimum modulus of a singularity of $F(z)$.*

From this theorem, and the root test for series convergence, we can conclude that the exponential growth of a counting sequence is determined by the reciprocal moduli of the generating function's singularities closest to the origin, which we call the *dominant singularities*.

In other words (cf. e.g. [18]), if the series defining $F(z)$ has dominant singularities of modulus R then

$$\limsup_{n\to\infty} |f_n|^{1/n} = \frac{1}{R}.$$

Returning to our binary trees example, we can solve Equation (1.2) to get

$$B(z) = (1 - \sqrt{1 - 4z})/2z,$$

from which it is clear that there is a singularity at $z = 1/4$ since there is a zero inside of a square root (note that the singularity at $z = 0$ is "removable" and thus does not affect the

asymptotic growth), and the counting sequence b_n grows like 4^n times a sub-exponential factor.

In Chapter 2 we consider generating functions that satisfy more complicated functional equations where the location of the dominant singularity is not easily determined. We develop new tools to determine the location of the dominant singularities in this case.

1.2 The Probabilistic Method

The probabilistic method is a powerful combinatorial tool. Roughly speaking, the method involves defining a probability space over some class of combinatorial objects and then showing a desired property holds with positive probability, proving the existence of an object in the class with the property. Consider the following example appearing in [4] giving a lower bound on the *Ramsey number* $R(k,k)$, the smallest integer n such that any two coloring of the edges of the complete graph K_n contains a monochromatic K_k subgraph.

Lemma 1.2.1 ([4])**.** *For any integer k, the Ramsey number $R(k,k) > \lfloor 2^{k/2} \rfloor$.*

Proof. We start by defining a probability space over the two colorings of the edges of K_n. In this case, we take a uniformly random two-coloring of the edges of K_n where every edge independently takes the color red or blue, each with probability $1/2$.

The probability that a given k vertex complete subgraph G is monochromatic is

$$\mathbb{P}(G \text{ monochromatic}) = 2^{1-\binom{k}{2}},$$

and taking the union bound over all such k vertex subgraphs gives

$$\mathbb{P}(\exists G \text{ monochromatic}) < \binom{n}{k} 2^{1-\binom{k}{2}}.$$

For $n = \lfloor 2^{k/2} \rfloor$ some algebra shows that

$$\binom{\lfloor 2^{k/2} \rfloor}{k} 2^{1-\binom{k}{2}} < 1,$$

and therefore with positive probability there are no monochromatic k vertex complete subgraphs. Hence, there exists a two coloring of $K_{\lfloor 2^{k/2} \rfloor}$ with no monochromatic k vertex complete subgraphs and

$$R(k,k) > \lfloor 2^{k/2} \rfloor . \qquad \square$$

In Chapter 4 we will apply the probabilistic method to the sunflower problem, following a strategy used in [5].

1.3 Organization

Using generating functions, analytic techniques, and the probabilistic method, we study problems including succinct representation of AVL trees, the hat guessing number of graphs, and the sunflower problem. The results and presentation of Chapter 2 are taken from [11]. The results and presentation of Chapter 3 are taken from [10]. The results and presentation of Chapter 4 are taken from [9].

Chapter 2

AVL Trees

This chapter is adapted from Chizewer, Melczer, Munro, and Pun [11].

2.1 Preliminaries

AVL trees [1] (named for their discoverers, G. Adelson-Velsky and E. Landis) are a subclass of binary search trees with logarithmic height, a property they maintain with updates during insertions and deletions in logarithmic time. Indeed, AVL trees are the oldest class of binary search trees maintaining logarithmic height and are characterized by the key property that any pair of sibling subtrees differ in height by at most 1. Here we are interested in the amount of storage needed to encode AVL trees with n nodes, a property intimately related to the number of AVL trees on n nodes. Odlyzko [36] gave a conjectural form for the number of AVL trees on n nodes in the 1980s, anticipating a forthcoming proof, but this proof did not appear in the literature.

If $\mathcal{C} = \bigsqcup_{n=0}^{\infty} \mathcal{C}_n$ is a family of objects, with $\mathcal{C}_n$ denoting the objects of size n in $\mathcal{C}$, then a representation of $\mathcal{C}$ is called *succinct* if it uses $\log_2 |\mathcal{C}_n| + o(\log |\mathcal{C}_n|)$ bits to store the objects of $\mathcal{C}_n$. A succinct representation is thus one whose space complexity asymptotically equals, up to lower-order terms, the information-theoretic lower bound. A *succinct data structure* [34, 35] for $\mathcal{C}$ is a succinct representation of $\mathcal{C}$ that supports a range of operations on $\mathcal{C}$ under reasonable time constraints.

2.2 Representations of Trees

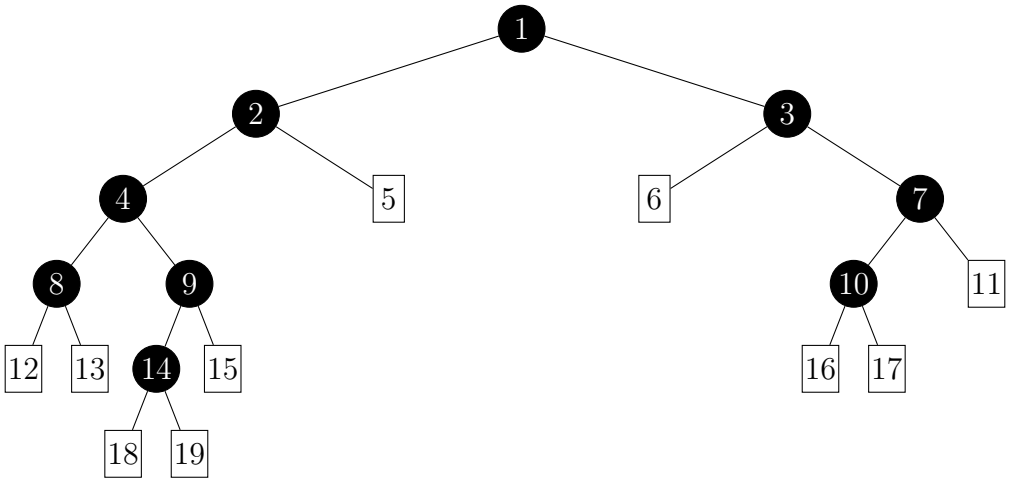
The theory of succinct data structures has a long history, much of it focused on representations of trees. We first describe some important classes of trees in this context, and then discuss our main results.

Binary Search Trees

Let $\mathcal{B}$ be the class of rooted binary trees, so that the number $|\mathcal{B}_n|$ of objects in $\mathcal{B}$ of size n is the nth Catalan number $b_n = \frac{1}{n+1}\binom{2n}{n}$. The class $\mathcal{B}$ lends itself well to storing ordered data in a structure called a *binary search tree*. The general idea is that for each node in the tree, the data stored in its left subtree will be smaller than the data at that node, and the data stored in the right subtree will be larger. To retrieve elements, one can recursively navigate through the tree by comparing the desired element to the current node, and moving to the left or right subtree if the element is respectively smaller or larger than the current node. As a result, it is desirable to efficiently support the navigation operations of moving to parent or child nodes in whatever representation is used.

A naive representation of $\mathcal{B}$ gives each node a label (using roughly $\log_2 n$ space) and stores the labels of each node's children and parent. The resulting data structure supports operations like finding node siblings in constant time, but is *not* succinct as it uses $\Theta(n \log n)$ bits while the information-theoretic lower bound is only $\log_2(b_n) = 2n + o(n)$. Somewhat conversely, a naive space-optimal representation of $\mathcal{B}$ is obtained by listing the objects of $\mathcal{B}_n$ in any canonical order and referencing a tree by its position $\{1, \ldots, b_n\}$ in the order, but asking for information like the children or parents of a node in a specific tree is then expensive as it requires building parts of the tree.

Practical succinct representations of binary trees supporting efficient navigation date back to Jacobson [23], who encoded a tree by storing the binary string of length $2n + 1$ obtained by adding *external* vertices so that every node has exactly two children, then taking a level-order traversal of the tree and recording a 1 for each original *internal* node encountered and a 0 for each external node encountered (see Figure 2.1). If each node is labelled by its position in a level-order traversal then, for instance, the children of the node labelled x in the tree encoded by the string σ have labels $2\,\mathrm{rank}_x(\sigma)$ and $2\,\mathrm{rank}_x(\sigma)+1$ where $\mathrm{rank}_x(\sigma)$ is the number of ones in σ up to (and including) the position x. By storing $o(n)$ bits, the rank operation (and similar supporting operations used to retrieve information about the trees) can be implemented in $O(1)$ time. Jacobson's results allow finding a



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Level-order bitmap:	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0

Figure 2.1: A binary tree and its level-order bitmap representation.

parent or child using $O(\log_2 n)$ bit inspections; Clark [12] and Munro [32] improved this to $O(1)$ inspections of $\log_2 n$ bit words.

AVL Trees

Because the time taken to access elements in a binary search tree typically depends on the height of the tree, many data structures *balance* their trees as new data is added. The balance operation requires rearranging the tree while preserving the underlying property that, for each node, the elements in the left subtree are smaller and the elements in the right subtree are larger. One of the most popular balanced tree structures – for theoretical study and practical application – are *AVL trees* [1]. Roughly speaking, AVL trees have balancing rules that force the subtrees rooted at the children of any node differ in height by at most one. Throughout this chapter we let $\mathcal{A}$ denote the class of AVL trees, so that $\mathcal{A}_n$ consists of all binary trees on n vertices such that the subtrees of any vertex differ in height by at most one (including empty subtrees).

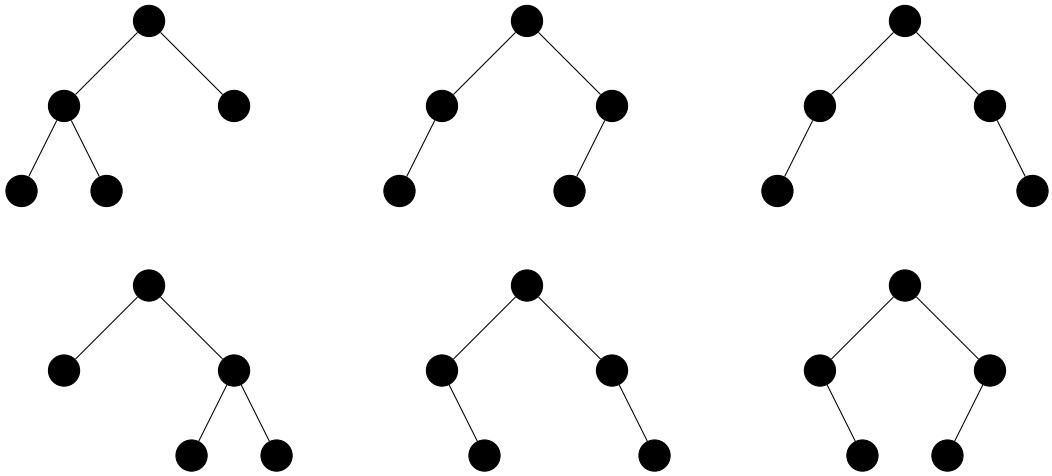


Figure 2.2: The six types of AVL trees with $n = 5$ nodes.

Due to the way they are constructed, AVL trees have mainly been enumerated under height restrictions, and enumeration by number of vertices (which is crucial for determining space-efficient representations, but not as important for other applications) is less studied. A 1984 paper [36] of Odlyzko describes the behaviour of a family of trees whose generating functions satisfy certain equations. It ends by stating that the generating function of AVL trees 'appears not to satisfy any simple functional equation, but by an intensive study... it can be shown' that $|\mathcal{A}_n| \sim n^{-1}\alpha^{-n}u(\log n)$ where $\alpha = 0.5219\ldots$ is 'a certain constant' and u is a periodic function, referencing for details a paper that was planned to be published but was never written.[1]

Efficiently Representing Tree Classes

Let B be a function satisfying $B(n) = \Theta(\log n)$. In [33] the authors give a method to construct a succinct encoding, and corresponding data structure, for any class of binary trees $\mathcal{T}$ satisfying the following four conditions.

1. *Fringe-hereditary*: For any tree $\tau \in \mathcal{T}$ and node $v \in \tau$ the *fringe subtree* $\tau[v]$, which consists of v and all of its descendants in τ, also belongs to $\mathcal{T}$.

[1] The current authors thank Andrew Odlyzko for discussions on the asymptotic behaviour of AVL trees and the growth constant α.

2. *Worst-case B-fringe dominated*: Most nodes in members of $\mathcal{T}$ do not generate large fringe subtrees, in the sense that

$$\left|\{v \in \tau : \left|\tau[v]\right| \geq B(n)\}\right| = o(n/\log B(n))$$

for every binary tree τ in the subset $\mathcal{T}_n \subset \mathcal{T}$ containing the members of $\mathcal{T}$ with n nodes.

3. *Log-linear*: There is a constant $c > 0$ and a function $\vartheta(n) = o(n)$ such that

$$\log|\mathcal{T}_n| = cn + \vartheta(n). \tag{2.1}$$

4. *B-heavy twigged*: If v is a node of any $\tau \in \mathcal{T}$ with $|\tau[v]| \geq B(n)$, and $\tau_\ell[v]$ and $\tau_r[v]$ are the left and right subtrees of v in τ, then $|\tau_\ell[v]|, |\tau_r[v]| = \omega(1)$.

We present a new construction that gives a succinct encoding for all classes of trees defined by a property satisfying only the first three conditions. By using constant time rank and select operations already supported by a succinct encoding for binary trees, we can also eliminate the use of so-called "portal nodes" and thus relax the second condition to the following property.

2′. *Worst-case weakly fringe dominated:* Most nodes in members of $\mathcal{T}$ do not generate large fringe subtrees, in the sense that there is a $B'(n)$ satisfying $B'(n) = d\log n + o(\log n)$ for some $d < 1$ such that

$$\left|\{v \in \tau : \left|\tau[v]\right| \geq B'(n)\}\right| = o(n) \tag{2.2}$$

for every binary tree $\tau \in \mathcal{T}_n$.

Adopting terminology similar to that of [33], we call a class of binary trees *weakly tame* if it is fringe-hereditary, worst-case weakly fringe dominated, and log-linear. As in [33], our encoding is *static*–it cannot be updated efficiently. Constructing a succinct *dynamic encoding*–one that can be updated efficiently–for any class of weakly tame trees is postponed to a future work.

Theorem 2.2.1. *There exists a succinct encoding for any weakly tame class $\mathcal{T}$ that supports the operations in Table 2.1 in $O(1)$ time using the $(\log n)$-bit word RAM model.*

Proof. See Section 2.3. □

Corollary 2.2.2. *There exists a succinct encoding for AVL trees that supports the operations in Table 2.1 in $O(1)$ time using the $(\log n)$-bit word RAM model.*

Proof. AVL trees are weakly tame (see [33, Example F.2]) so the result follows immediately from Theorem 2.2.1. □

Remark 2.2.3. In [33] the log-linearity of AVL trees is inferred from the stated exponential growth of a_n in Odlyzko [36]. This growth is proven in Theorem 2.2.5 below.

A minor modification of the arguments in [33] show that *Left-Leaning AVL (LLAVL) Trees*, which are AVL trees with the added restriction that at every node the height of the left subtree is at least the height of the right subtree, are also weakly tame, giving the following.

Corollary 2.2.4. *There exists a succinct encoding for LLAVL trees that supports the operations in Table 2.1 in $O(1)$ time using the $(\log n)$-bit word RAM model.*

To characterize how much space is required by a succinct encoding, we derive an asymptotic bound on the number of AVL trees using techniques from *analytic combinatorics* [18, 31]. To this end, let $a_n = |\mathcal{A}_n|$ be the counting sequence of $\mathcal{A}$ and let $A(z) = \sum_{n\geq 0} a_n z^n$ be its associated generating function. The key to enumerating AVL trees is to let $A_h(z)$ be the generating function for the subclass of AVL trees with *height* h. The balance condition on subtrees implies that an AVL tree of height $h+2$ is a root together with a subtree of height $h+1$ and a subtree of height either $h+1$ or h, giving rise to the recursive equation

$$A_{h+2}(z) = A_{h+1}(z)(A_{h+1}(z) + 2A_h(z)) \tag{2.3}$$

for all $h \geq 0$. This recursion, along with the initial conditions $A_0(z) = z$ (encoding the only AVL tree with height zero, which is a single vertex) and $A_1(z) = z^2$ (encoding the only AVL tree with height one, which is a root with two children) uniquely determines $A_h(z)$ for all h. Summing over all possible heights gives the generating function

$$A(z) = \sum_{h=0}^{\infty} A_h(z)$$

for AVL trees.

Equation (2.3) implies that $A_h(z)$ is a non-constant polynomial with positive coefficients for all h, so the equation $A_h(z) = 1/3$ has a unique positive solution for all $h \in \mathbb{N}$ (see Figure 2.3 for values of these solutions). We prove the following.

$\texttt{parent}(v)$	the parent of v, same as $\texttt{anc}(v, 1)$
$\texttt{degree}(v)$	the number of children of v
$\texttt{left\_child}(v)$	the left child of node v
$\texttt{right\_child}(v)$	the right child of node v
$\texttt{depth}(v)$	the depth of v, i.e., the number of edges between the root and v
$\texttt{anc}(v, i)$	the ancestor of node v at depth $\texttt{depth}(v) - i$
$\texttt{nbdesc}(v)$	the number of descendants of v
$\texttt{height}(v)$	the height of the subtree rooted at node v
$\texttt{LCA}(v, u)$	the lowest common ancestor of nodes u and v
$\texttt{leftmost\_leaf}(v)$	the leftmost leaf descendant of v
$\texttt{rightmost\_leaf}(v)$	the rightmost leaf descendant of v
$\texttt{level\_leftmost}(\ell)$	the leftmost node on level ℓ
$\texttt{level\_rightmost}(\ell)$	the rightmost node on level ℓ
$\texttt{level\_pred}(v)$	the node immediately to the left of v on the same level
$\texttt{level\_succ}(v)$	the node immediately to the right of v on the same level
$\texttt{node\_rank}_X(v)$	the position of v in the X-order, $X \in \{\texttt{PRE, POST, IN}\}$, i.e., in a preorder, postorder, or inorder traversal of the tree
$\texttt{node\_select}_X(i)$	the ith node in the X-order, $X \in \{\texttt{PRE, POST, IN}\}$
$\texttt{leaf\_rank}(v)$	the number of leaves before and including v in preorder
$\texttt{leaf\_select}(i)$	the ith leaf in preorder

Table 2.1: Operations discussed in [20, 33] which can be done in $O(1)$ time in the $(\log n)$-bit word RAM model in a succinct encoding of a binary tree.

Theorem 2.2.5. *If α_h is the unique positive solution to $A_h(z) = 1/3$ then the limit*

$$\alpha = \lim_{h\to\infty} \alpha_h = 0.5219\ldots$$

is well-defined. Furthermore,

$$\log_2(a_n) = \underbrace{n \log_2(\alpha^{-1})}_{n(0.938\ldots)} + \log \theta(n)$$

for a function θ growing at most sub-exponentially (meaning $\theta(n) = o(\kappa^n)$ for all $\kappa > 1$).

Proof. The result follows immediately from applying Theorem 2.4.5 below with $f(x_1, x_2) = x_1^2 + 2x_1x_2$, since the unique positive solution to $f(C, C) = C$ is $C = 1/3$. □

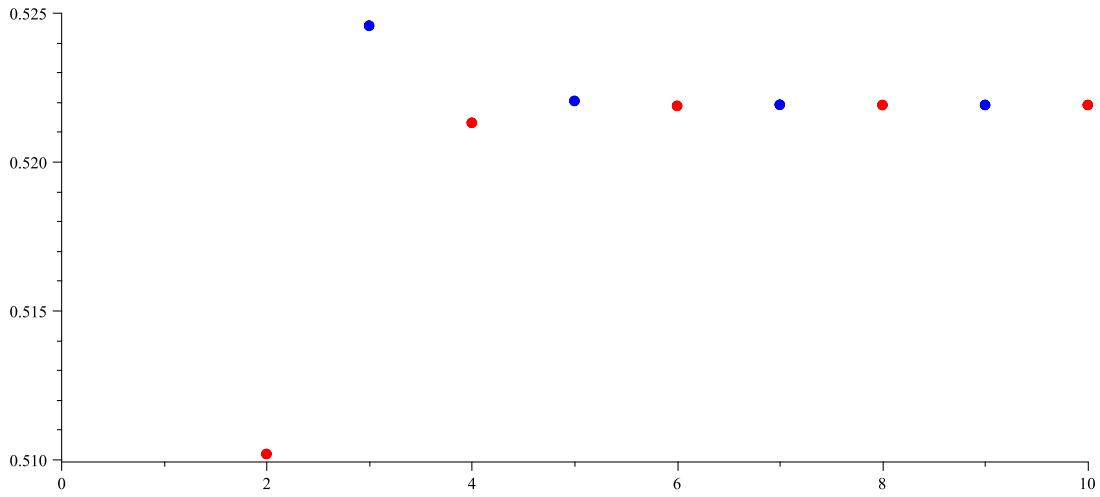


Figure 2.3: Values α_h converging to $\alpha = 0.5219\ldots$ monotonically from below among even h (red) and monotonically from above among odd h (blue).

Remark 2.2.6. A full proof of the claimed asymptotic behaviour $a_n \sim n^{-1}\alpha^{-n}u(\log n)$ in Odlyzko [36], which characterizes *sub-dominant* asymptotic terms for the bitsize, requires a more intense study of the recursion (2.3) and is outside the scope of this discussion. It is postponed to future work.

Our approach derives asymptotics for a family of generating functions satisfying recursive equations similar to (2.3). For instance, if $L_h(z)$ is the generating function for LLAVL trees with height h then

$$L_{h+2}(z) = L_{h+1}(z)(L_{h+1}(z) + L_h(z)) \tag{2.4}$$

for all $h \geq 0$, as an LLAVL tree of height $h+2$ is a root together with a left subtree of height $h+1$ and a right subtree of height $h+1$ or h. Note that the only difference between this recurrence and the recursive equation (2.3) for AVL trees is the coefficient of $L_h(z)$, since there is now only one way to have an unbalanced pair of subtrees.

Theorem 2.2.7. *If γ_h is the unique positive solution to $L_h(z) = 1/2$ then the limit*

$$\gamma = \lim_{h\to\infty} \gamma_h = 0.67418\ldots$$

is well-defined. Furthermore, the number ℓ_n of LLAVL trees on n nodes satisfies

$$\log_2(\ell_n) = \underbrace{n \log_2(\gamma^{-1})}_{n(0.568\ldots)} + \log\theta(n)$$

for a function θ growing at most sub-exponentially.

Proof. The result follows by applying Theorem 2.4.5 below with $f(x_1, x_2) = x_1^2 + x_1 x_2$, since the unique positive solution to $f(C, C) = C$ is $C = 1/2$. □

2.3 A New Succinct Encoding for Weakly Tame Classes

We now prove Theorem 2.2.1, first describing our encoding and then showing it has the stated properties.

2.3.1 Our encoding

Let $\mathcal{E}$ denote a succinct data structure representing all binary trees that supports the operations in Table 2.1, and denote the encoding of a binary tree τ in this data structure by $\mathcal{E}(\tau)$. We now fix a weakly tame class of binary trees $\mathcal{T}$ and, given a binary tree $\tau \in \mathcal{T}$ of size n, set

$$\tau' = \left\{ v \in \tau : \left|\tau[v]\right| \geq d \log n \right\}$$

where d is a constant such that $B'(n) = d \log n + o(\log n)$ satisfies (2.2) in the definition of worst-case weakly fringe dominated.

Our succinct data structure for $\mathcal{T}$ is constructed as follows.

1. We call τ' the *upper tree* of τ and simply copy the encoding $\mathcal{E}(\tau')$ for these nodes.
2. For every $1 \leq j \leq d \log n$ we write down a lookup table mapping the trees in $\mathcal{T}_j$ (with j nodes) to their corresponding $\mathcal{E}$ encoding. We can do this, for example, by enumerating the $\mathcal{T}_j$ in lexicographic order by the $\mathcal{E}$ encoding using integers of bitsize $\log |\mathcal{T}_j| = cj + o(j)$, where c is the constant in the definition of log-linearity (2.1).
3. For each leaf node $\ell \in \tau'$ the tree $\tau[\ell]$ has size $|\tau[\ell]| < d \log n$ by definition of τ'. We call these trees *lower trees*, and write them down using their encoding in a lookup table in `leaf_rank` order of their roots in τ', storing the root locations in an indexable dictionary.
4. Lastly, we store additional information in fully indexable dictionaries and indexable dictionaries to support operations like `node_rank/select`, `level_succ/pred`, and

`leaf_rank/select`. For instance, for `node_rank/select` we store a fully indexable dictionary that maps the `node_rank` for a node in τ' to the `node_rank` of the node in τ. The techniques to support the other operations are similar, and are analogous to constructions used in [20, 17].

2.3.2 Proof of Size and Operation Time Bounds

Navigation through the upper tree follows standard navigation using $\mathcal{E}$, which supports the desired operations in constant time. When a leaf node ℓ is reached in the upper tree, the operation $x = \texttt{leaf\_rank}(\ell)$ gives the index of the child tree in the indexable dictionary. Then the operation $\texttt{select}(x)$ gives the location of the string encoding the child tree. Finally, using the table mapping our encoding to the $\mathcal{E}$ encoding gives us the ability to perform all the navigation operations on the smaller tree. In order to perform the lookup using the mapping, it is necessary to know the size of the tree. This can be inferred from the space in memory allocated to the naming, which can be calculated by the operation $\texttt{select}(x+1)$ in the indexable dictionary to find the starting location of the next child tree. To navigate back to the upper tree from a child tree, we use the reverse operations of $y = \texttt{rank}(x)$ in the indexable dictionary followed by $\texttt{select\_leaf}(y)$ in the upper tree.

To get the `node_rank` of a node in τ' we use the fully indexable dictionary, and to get the `node_rank` of a node not in τ' we simply get the `node_rank` of the root of the child tree and the `node_rank` of the node within the child tree and perform the appropriate arithmetic depending on the desired rank order (`pre, post, in`). For `node_select`, if the node is in τ' then selecting using the indexable dictionary is sufficient. Otherwise, the node is in a child tree and the initial `node_select` will return the predecessor node in τ' which will be the root of the child tree when using `preorder` (the argument is similar for `postorder` and `inorder`). Using the rank of this root and appropriate arithmetic, we can then select the desired node in the child tree. Implementing the other operations is analogous. It is clear that all of these operations are supported in constant time, since they involve a constant number of calls to the constant-time operations in the existing data structures, and lookups using $(\log n)$-bit words.

Space Complexity The space used by $\mathcal{E}(\tau')$ is $o(n)$ by the weakly tame property. The space used by the lookup tables is $O(n^d \log n) = o(n)$ by definition of τ' and d, and the space used by all of the encodings of the child trees is $cn + o(n)$ by log-linearity. Lastly, the space needed for the indexable dictionaries is $o(n)$ for each [17, Lemmas 1 and 2].

Summing these requirements shows that the total storage required is $cn + o(n)$ many bits, so the encoding is succinct. □

2.4 Asymptotics for a Family of Recursions

We derive the asymptotic behaviour of a family of generating functions which includes Theorem 2.2.5 as a special case. Let $\mathcal{F}$ be a combinatorial class decomposed into a disjoint union of finite subclasses $\mathcal{F} = \bigsqcup_{h=0}^{\infty} \mathcal{F}_h$ whose generating functions $F_h(z)$ are non-constant and satisfy a recursion

$$F_h(z) = f(F_{h-1}(z), F_{h-2}(z), \ldots, F_{h-c}(z)) \quad \text{for all} \quad h \geq c, \tag{2.5}$$

where c is a positive integer and f is a multivariate polynomial with non-negative coefficients.

Remark 2.4.1. The elements of $\mathcal{F}_h$ are usually *not* the objects of $\mathcal{F}$ of size h (in our tree applications they contain trees of height h, not trees with h nodes). The fact that each $\mathcal{F}_h$ is finite implies that the $F_h(z)$ are polynomials with non-negative coefficients.

We assume that there exists a (necessarily unique) positive real solution C to the equation $C = f(C, C, \ldots, C)$, and for each $h \geq 0$ we let α_h be the unique positive real solution to $F_h(z) = C$. In order to rule out degenerate cases and cases where the counting sequence has periodic behaviour, we need another definition.

Definition 2.4.2. (recursive-dependent) We call the polynomial f *recursive-dependent* if there exists a constant k (depending only on f) such that for any indices $i, j \geq c$ with $i \geq j + k$ there exists a sequence of applications of the recurrence (2.5) resulting in a polynomial P with $F_i = P(F_{\ell_1}, \ldots, F_{\ell_m})$ for some $0 \leq \ell_1 < \cdots < \ell_m \leq i$ where $\frac{\partial P}{\partial F_j} \neq 0$.

Example 2.4.3. The polynomial $f(x, y) = y$ is not recursive-dependent because it leads to the recursion $F_h(z) = F_{h-2}(z)$, meaning that the values of F_h when h is even can be independent of those where h is odd.

Lemma 2.4.4. *If f is recursive-dependent with non-negative coefficients and a positive fixed point then the limit $\alpha = \lim_{h\to\infty} \alpha_h$ exists.*

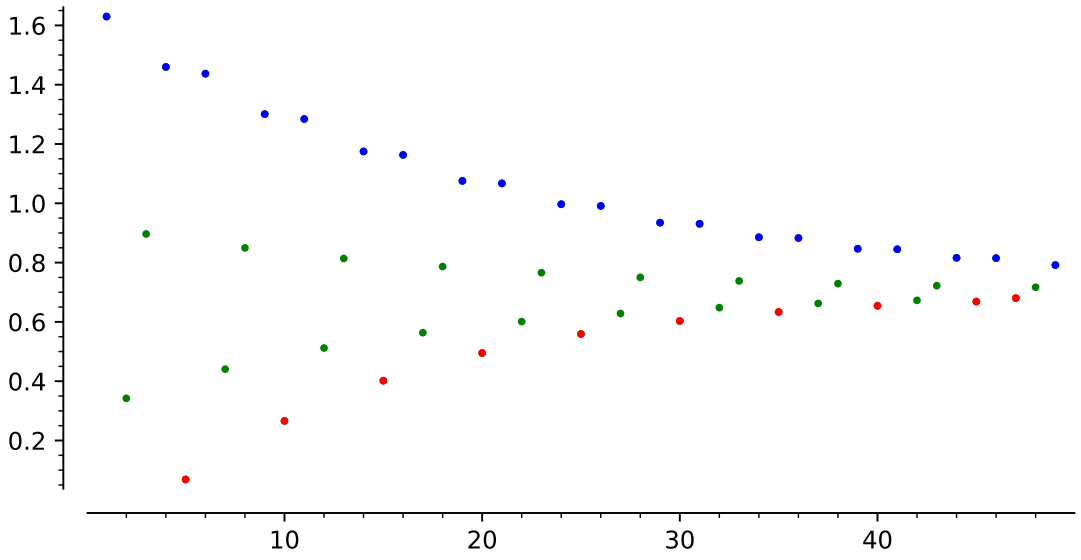


Figure 2.4: Values α_i converging with u_is shown in blue and ℓ_js shown in red.

Proof. We start by defining two subsequences of α_h to give upper and lower bounds on its limit, then prove that these are equal. First, we let

- u_0 be the smallest index $j \in \{0, \ldots, c-1\}$ such that $\alpha_j = \max\{\alpha_0, \ldots, \alpha_{c-1}\}$

and for all $i \geq 0$ let

- u_{i+1} be the smallest index $j \in \{u_i{+}1, \ldots, u_i{+}c\}$ such that $\alpha_j = \max\{\alpha_{u_i+1}, \ldots, \alpha_{u_i+c}\}$,

so that the u_i denote the indices of the maximum values of the α_h as h ranges over intervals of size at most c. Conversely, we let

- ℓ_0 be the index $j \in \{0, \ldots, c-1\}$ such that $\alpha_j = \min\{\alpha_0, \ldots, \alpha_{c-1}\}$

and for all $j \geq 0$ let

- ℓ_{i+1} be the index $j \in \{u_i + 1, \ldots, u_i + c\}$ such that $\alpha_j = \min\{\alpha_{u_i+1}, \ldots, \alpha_{u_i+c}\}$,

so that the ℓ_j denote the indices of the minimum values of the α_h as h ranges over intervals of size at most c. Figure 2.4 is a graph of the sequence α_i, where the u_i's have been colored blue and the ℓ_js have been colored red. The remaining values of α_i which are not part of the sequence of u_is or ℓ_js have been colored green. From the picture, we can see that the u_is and ℓ_js bound the sequence from above and below respectively.

We claim that the subsequence α_{u_i} is non-increasing. To establish this, we fix $i \geq 1$ and consider α_{u_i}. By definition, $\alpha_{u_i} \geq \alpha_{u_j}$ for all $j \in \{u_{i-1}+1, \ldots, u_{i-1}+c\}$. Thus, if $u_{i+1} \in \{u_{i-1}+1, \ldots, u_{i-1}+c\}$ then $\alpha_{u_i} \geq \alpha_{u_{i+1}}$ as claimed. If, on the other hand, $u_{i+1} > u_{i-1}+c$ then repeated application of the recursion (2.5) implies

$$\begin{aligned} F_{u_{i+1}}(\alpha_{u_i}) &= f\Big(F_{u_{i+1}-1}(\alpha_{u_i}), \ldots, F_{u_{i+1}-c}(\alpha_{u_i})\Big) \\ &\vdots \\ &= Q\Big(F_{u_{i-1}+1}(\alpha_{u_i}), \ldots, F_{u_{i-1}+c}(\alpha_{u_i})\Big), \end{aligned}$$

where Q is a multivariate polynomial with non-negative coefficients such that $Q(C, \ldots, C) = C$. All the F_h are monotonically increasing as non-constant polynomials with non-negative coefficients, so $F_j(\alpha_{u_i}) \geq F_j(\alpha_{u_j}) = C$ for all $j \in \{u_{i-1}+1, \ldots, u_{i-1}+c\}$ and

$$F_{u_{i+1}}(\alpha_{u_i}) \geq Q\Big(C, \ldots, C\Big) = C.$$

Since $F_{u_{i+1}}$ is monotonically increasing and $F_{u_{i+1}}(\alpha_{u_{i+1}}) = C$, we once again see that $\alpha_{u_i} \geq \alpha_{u_{i+1}}$. As i was arbitrary, we have proven that α_{u_i} is non-increasing. The same argument, reversing inequalities, proves that the subsequence α_{ℓ_j} is non-decreasing.

As α_{ℓ_j} is non-decreasing and α_{u_i} is non-increasing, either $\alpha_{\ell_j} \leq \alpha_{u_i}$ for all $i, j \geq 0$ or $\alpha_{\ell_j} > \alpha_{u_i}$ for all sufficiently large i and j. The second case implies the existence of indices $a, b > 0$ such that $\alpha_{\ell_b} > \alpha_{u_a}$ but $\ell_b \in \{u_{a-1}+1, \ldots, u_{a-1}+c\}$ so that u_a is not the maximum index of α_j in this range, giving a contradiction. Thus, $\alpha_{\ell_j} \leq \alpha_{u_i}$ for all $i, j \geq 0$ and the limits

$$\alpha_u = \lim_{i \to \infty} \alpha_{u_i} \quad \text{and} \quad \alpha_\ell = \lim_{j \to \infty} \alpha_{\ell_j}$$

exist. To prove that the limit of α_h exists as $h \to \infty$, it is now sufficient to prove that $\alpha_u = \alpha_\ell$.

Suppose toward contradiction that $\alpha_u \neq \alpha_\ell$, and define $a = \alpha_u - \alpha_\ell > 0$. For any $\epsilon > 0$, we pick i, j, k sufficiently large so that $\ell_j > u_i > \ell_k + c$ and $|\alpha_{u_i} - \alpha_u|, |\alpha_{\ell_j} - \alpha_\ell|, |\alpha_{\ell_k} - \alpha_\ell| < \epsilon$.

Then by recursive-dependence we can recursively decompose F_{ℓ_j} in terms of F_{u_i}, and possibly some other terms $F_{h_1}, \ldots, F_{h_r}$ where each $|h_n - u_i| \leq c$, to get

$$C = F_{\ell_j}(\alpha_{\ell_j}) = P(F_{u_i}(\alpha_{\ell_j}), F_{h_1}(\alpha_{\ell_j}), \ldots, F_{h_r}(\alpha_{\ell_j}))$$

where $P(F_{u_i}, F_{h_1}, \ldots, F_{h_r})$ is a polynomial with non-negative coefficients that depends on F_{u_i} and satisfies $P(C, \ldots, C) = C$. Because P is monotonically increasing in each coordinate, and $\alpha_{\ell_k} + \epsilon > \alpha_\ell \geq \alpha_{\ell_j}$, we see that

$$C \leq P(F_{u_i}(\alpha_{\ell_k} + \epsilon), F_{h_1}(\alpha_{\ell_k} + \epsilon), \ldots, F_{h_r}(\alpha_{\ell_k} + \epsilon)).$$

Furthermore, each $\alpha_{h_n} \geq \alpha_{\ell_k}$ so

$$\begin{aligned} C &\leq P(F_{u_i}(\alpha_{\ell_k} + \epsilon), F_{h_1}(\alpha_{h_1} + \epsilon), \ldots, F_{h_r}(\alpha_{h_r} + \epsilon)) \\ &\leq P(F_{u_i}(\alpha_{\ell_k} + \epsilon), C + \text{poly}(\epsilon), \ldots, C + \text{poly}(\epsilon)). \end{aligned}$$

Finally, $\alpha_{u_i} - a \geq \alpha_{\ell_k}$ so

$$C \leq P(F_{u_i}(\alpha_{u_i} - a + \epsilon), C + \text{poly}(\epsilon), \ldots, C + \text{poly}(\epsilon)).$$

Because a is fixed, P is monotonically increasing in each variable, and $F_{u_i}(\alpha_{u_i}) = C$, taking $\epsilon \to 0$ shows that the right-hand side of this last inequality is strictly less than $P(C, \ldots, C) = C$, a contradiction. Thus, $a = 0$ and the limit $\alpha = \alpha_u = \alpha_\ell$ exists. □

Theorem 2.4.5. *If f is recursive-dependent with non-negative coefficients and a positive fixed point, then the number a_n of objects in $\mathcal{F}$ of size n satisfies*

$$a_n = \alpha^{-n}\,\theta(n),$$

where α is the limit described in Lemma 2.4.4 and $\theta(n)$ is a function growing at most sub-exponentially.

Proof. We prove that the generating function $F(z)$ is analytic for $|z| < \alpha$ by showing that the series $\sum_{h=0}^{\infty} F_h(z)$ converges for these values of z. Because $|F(z)| \to \infty$ as $z \to \alpha$, the point $z = \alpha$ is then a singularity of $F(z)$ of smallest modulus, and thus (by the root test for series convergence) the reciprocal of the exponential growth of a_n.

First, assume that there exists some $k \geq 0$ and $0 < \lambda < 1$ such that $F_h(z) < \lambda C$ for every $h \in \{k, k+1, \ldots, k+c-1\}$. Let A be the sum of the coefficients of all degree 1 terms of f. Since f has non-negative coefficients and a positive real fixed point, we must have $A < 1$.

Let $g(x_1, \ldots, x_c)$ be the function created by removing all degree one terms from f. Observe that $C = AC + g(C, \ldots, C)$, and thus $g(\lambda C, \ldots, \lambda C) \leq \lambda^2 g(C, \ldots, C) = \lambda^2(1-A)C$, so that

$$f(\lambda C, \ldots, \lambda C) \leq A\lambda C + \lambda^2(1-A)C.$$

Algebraic manipulation shows that $A\lambda + \lambda^2(1-A) \leq \lambda$, and since f has non-negative coefficients we can conclude that for every $h \in \{k+c, k+1+c, \ldots, k+2c-1\}$ we have $F_h(z) \leq A\lambda C + \lambda^2(1-A)C$. Let $\lambda_0 = \lambda$ and define $\lambda_i = \lambda_{i-1}(A + \lambda_{i-1} - A\lambda_{i-1})$ for all $i \geq 1$. By the above argument we have

$$F_{ch+k}(z) \leq \lambda_h C,$$

so it remains to show that $\sum_{i=0}^{\infty} \lambda_i$ converges. We will show that $\lambda_i \leq \lambda(A + \lambda - A\lambda)^i$ by induction on i. The result holds by definition for $i = 1$. If the result holds for some $j \geq 1$ then

$$\begin{aligned}
\lambda_{j+1} &= \lambda_j(A + \lambda_j - A\lambda_j) \\
&\leq \lambda(A + \lambda - A\lambda)^j(A + \lambda_j - A\lambda_j) \\
&\leq \lambda(A + \lambda - A\lambda)^{j+1},
\end{aligned}$$

where the last inequality follows from the fact that $\lambda_j < \lambda$ since $A + \lambda - A\lambda < 1$. The sum $\sum_{i=0}^{\infty} \lambda(A + \lambda - A\lambda)^i$ converges as a geometric series, and thus $\sum_{h=0}^{\infty} F_h(z)$ converges.

It remains to show that if $|z| < \alpha$ then such a k and λ exist. For any $|z| < \alpha$ there is some N sufficiently large such $|z| < \alpha_n$ for all $n \geq N$. By the definition of α_n, and since the coefficients of F_n are all positive, we must have $F_n(z) < C$. Hence $F_n(z) < \lambda_n C$ for some $0 < \lambda_n < 1$. Taking $k = N$ and letting λ be the largest λ_n for $n \in \{N, N+1, \ldots, N+c-1\}$ proves our final claim. $\square$

Chapter 3

The Hat Guessing Game

This chapter is adapted from Chizewer, McInnis, Sohrabi, and Kaistha [10].

3.1 Preliminaries

Given a simple undirected graph G and a function $h : V(G) \to \mathbb{N}$ mapping the vertices of G to the positive integers, the *hat guessing game* $\langle G, h\rangle$ is a puzzle in which for every vertex $v \in G$ a player is placed and assigned a hat from $h(v)$ possible colors. Each player attempts to guess the color of his hat according to a predetermined strategy that takes as input the colors assigned to players at neighboring vertices. The game $\langle G, h\rangle$ is known in advance by the players, so they may use all information about G and h when devising a set of deterministic strategies. The players are a team and are said to *win* for a given coloring if at least one player guesses correctly. The hat guessing game $\langle G, h\rangle$ is said to be *winning* if there exists some set of strategies for the players so that for every possible color assignment, the players win. The players *lose* on a coloring if no player guesses correctly; if there exists such a coloring for every possible set of guessing strategies, the game is called *losing*.

We refer to the function h as the *hatness function*, and for a vertex $v \in G$ we call the value $h(v)$ the *hatness* of v. A *subgame* $\langle G', h'\rangle$ of a hat guessing game $\langle G, h\rangle$ is a game where $G' \subseteq G$ is a subgraph, and the hatness function $h' = h|_{v\in G'}$ is the hatness function h restricted to the vertices of G in G'. Moreover, we call a subgame *proper* if G' is a proper subgraph of G. The *hat guessing number* of a graph, G, denoted $HG(G)$, is the largest integer q such that $\langle G, \star q\rangle$ is a winning game, where $\star q$ denotes the constant

function $h(\cdot) = q$. We call a vertex v *deletable* if the subgame on $G\backslash\{v\}$ is winning. Note that if a game has a winning subgame then it is winning, and if a game is winning then decreasing the hatness of any vertex results in a winning game.

The hat guessing number was introduced in [8]. Previous results have attempted to bound the hat guessing number of graphs, but few give exact answers. Some exact results include the hat guessing number of trees (folklore), cycles [38], pseudotrees [27], complete graphs (folklore), and some windmills and book graphs [21]. There are also upper and lower bounds (that are somewhat far apart) on complete bipartite graphs [8, 19, 2], upper bounds on outerplanar graphs [7, 26], and lower bounds on planar graphs [3, 24, 29].

Example 3.1.1. The hat guessing game $\langle K_n, \star q\rangle$ is winning if and only if $q \leq n$

There are several proofs for Example 3.1.1 in the literature. To prove that $\langle K_n, \star q\rangle$ is winning for $q \leq n$–and more generally to prove that a particular game is winning–a common approach is to specify a winning strategy. The following is a winning strategy for $q = n$. Number the players and colors $0, 1, \ldots, n-1$ and let player i guess that his color would cause the sum of all the colors to equal $i \pmod n$. This gives a valid strategy since every player sees every other player and so can calculate the unique integer $0, 1, \ldots, n-1$ which would result in the desired sum. As the sum must take a value $0, 1, \ldots, n-1 \pmod n$, there will always be a player who guesses correctly. Using the fact above about winning subgames, it is easy to see that the result holds for $q < n$ as well. There are several techniques for proving that a particular game is losing. One approach is to use the probabilistic method to show that for an arbitrary strategy, there exists a coloring for which no player guesses correctly. Consider the game $\langle K_n, \star q\rangle$ for $q > n$. Color each player uniformly and independently at random from the q possible colors. Since a player's guess cannot depend on his own color, the probability that a given player guesses correctly is $1/q$. By the union bound, the probability that at least one player guesses correctly is at most $n/q < 1$. Hence, with positive probability no player guesses correctly, so there exists a coloring in which no player guesses correctly. By definition, this means the game is losing. □

In this chapter, we study the hat guessing problem for cycles and cactus graphs. Our first result resolves [28, Conjecture 5.2] affirmatively and goes further, showing exactly which games on cycles are winning. The second result adds cactus graphs to the growing list of graph classes for which the hat guessing number is known exactly.

Theorem 3.1.2. *Let C be a cycle on n vertices such that the hatness h of each vertex v in C satisfies $h(v) \geq 2$. Then $\langle C, h\rangle$ is winning if and only if C satisfies at least one of the following conditions:*

1. *The length n is 4 or divisible by 3 and $h(v) \leq 3$ for all $v \in C$.*
2. *The length n is 3 and $\sum_{v \in C} \frac{1}{h(v)} \geq 1$.*
3. *The game $\langle C, h \rangle$ contains a winning proper subgame.*
4. *The hatness function satisfies both of the following properties:*
   (a) *There exists a sequence of adjacent vertices in C with hatnesses $(2, 3, 3)$ or $(3, 2, 3)$.*
   (b) *For all $v \in C$, $h(v) \leq 4$.*

The forward implication for Condition 1 (that condition 1 implies $\langle C, h \rangle$ is winning) is [38, Theorem 1]; for Condition 2, it is [28, Theorem 2.1]; for Condition 3, it follows directly from definitions; and for Condition 4, it is [28, Theorem 5.1]. We prove the reverse implications—that $\langle C, h \rangle$ is losing if none of the above conditions hold—in Section 3.2.

Our next result determines the hat guessing number of cactus graphs. A *cactus graph* is a connected graph in which every pair of cycles share at most one vertex. Figure 3.1 shows an example of a cactus graph (on the left). A useful subset of the class of cactus graphs is the class of *pseudotrees*, which are connected graphs with at most one cycle. The hat guessing number of cactus graphs was first studied in [7], which derived[1] a bound of 16 for any cactus graph. We apply Theorem 3.1.2 to exactly determine the hat guessing number of every cactus graph.

Theorem 3.1.3. *Let G be a cactus graph.*

1. *$HG(G) = 4$ if and only if G contains at least two triangles.*
2. *$HG(G) = 3$ if and only if G contains at least two cycles or a cycle of length 4 or divisible by 3, and G contains fewer than two triangles.*
3. *$HG(G) = 2$ if and only if G is a pseudotree with at least one edge and no cycle of length 4 or divisible by 3.*

The lower bound for Statement 1 of the theorem follows from Theorem 3.2.1 by gluing together two triangles, each with hatnesses $(2, 4, 4)$ to either end of a path (possibly of length 0) with hatnesses $(2, 4, \ldots, 4, 2)$ at the vertices with hatness 2 (as shown on the right

[1]The bound of 16 was mentioned in the preprint (arXiv:2109.13422) but was removed from the publication

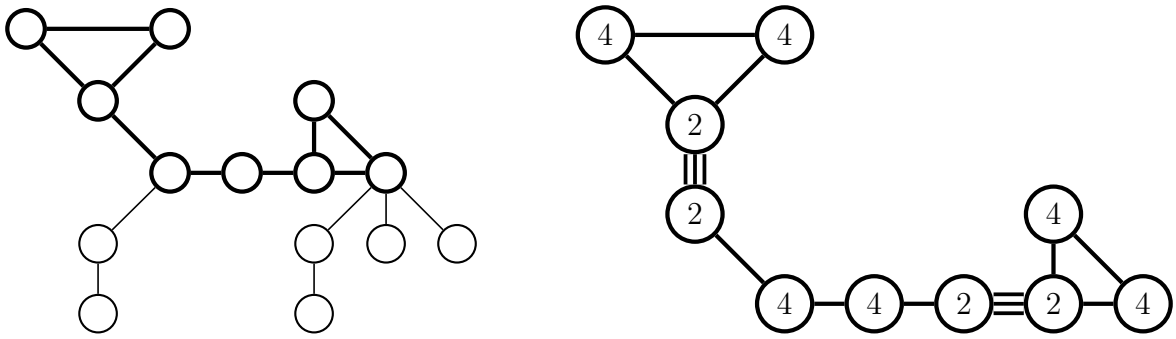


Figure 3.1: A cactus graph with two triangles (left) and the construction that shows that the hat guessing number is at least 4 (right). The triple lines indicate vertices that are glued together in the construction.

of Figure 3.1). Since this forms a winning subgame of $\langle G, \star 4\rangle$, the game $\langle G, \star 4\rangle$ is winning. The lower bound for Statement 2 in the case of two cycles follows analogously by gluing two winning cycles with hatnesses $(2, 3, \ldots, 3)$ to either end of a winning path (possibly of length 0) with hatnesses $(2, 3, \ldots, 3, 2)$ and decreasing all hatnesses in the resulting game to 3 to give a winning subgame of $\langle G, \star 3\rangle$. The lower bound for Statement 2 in the one cycle case follows from [38, Theorem 1], and the lower bound for Statement 3 is trivial. We prove the upper bounds in Section 3.3.

3.2 Cycles

Throughout this section and the next we use *constructors*—rules for building winning or losing games from smaller games that are easier to analyze—to simplify our proofs. Given two graphs G_1 and G_2 with vertices v_1 and v_2, respectively, the graph denoted $G = G_1 +_{v_1,v_2} G_2$ formed by *gluing* G_1 and G_2 at v_1 and v_2 is given by the union of vertices and edges $G = G_1 \cup G_2$ where v_1 and v_2 are identified as a single vertex in G which is incident to all vertices in the union $N_{G_1}(v_1) \cup N_{G_2}(v_2)$ of their respective neighbor sets in the original graphs. The following theorems, which concern the gluing of winning and losing graphs, are used as building blocks to prove our main results.

Theorem 3.2.1 ([28, Theorem 3.1])**.** *Let G_1 and G_2 be graphs with $v_1 \in G_1$ and $v_2 \in G_2$ and let $G = G_1 +_{v_1,v_2} G_2$. Let h_1 and h_2 be hatness functions for the graphs G_1 and*

G_2 respectively, and suppose that $\langle G_1, h_1\rangle$ and $\langle G_2, h_2\rangle$ are both winning. Then $\langle G, h\rangle$ is winning, where

$$h(v) = \begin{cases} h_1(v) & \text{if } v \in G_1 \backslash \{v_1\} \\ h_2(v) & \text{if } v \in G_2 \backslash \{v_2\} \\ h_1(v_1) \cdot h_2(v_2) & \text{if } v = v_1 = v_2 \end{cases}.$$

Theorem 3.2.2 ([25, Theorem 4.1])**.** *Let G_1 and G_2 be graphs with $v_1 \in G_1$ and $v_2 \in G_2$ and let $G = G_1 +_{v_1,v_2} G_2$. Let h_1 and h_2 be hatness functions for the graphs G_1 and G_2 respectively, and suppose that $\langle G_1, h_1\rangle$ and $\langle G_2, h_2\rangle$ are both losing with $h_1(v_1) \geq h_2(v_2) = 2$. Then $\langle G, h\rangle$ is losing, where*

$$h(v) = \begin{cases} h_1(v) & \text{if } v \in G_1 \\ h_2(v) & \text{if } v \in G_2 \backslash \{v_2\} \end{cases}.$$

With these results in mind, we are ready to prove the reverse implications in Theorem 3.1.2: if none of the conditions hold then the game is losing. If none of the conditions hold and $n = 3$, the result follows from [28, Theorem 2.1]. Hence, we may assume $n > 3$ for the rest of the proof. Furthermore, if there are no vertices of hatness 2 then the result holds by [38, Theorem 1], so we may also assume for the rest of the proof that there is at least one vertex a with $h(a) = 2$. We split the proof into two parts, depending on whether Condition 4a or Condition 4b fails.

3.2.1 Proof of Theorem 3.1.2 when Condition 4a Fails

We begin by assuming that Conditions 1, 2, 3, and 4a all fail, and Condition 4b holds.

Proof. Given these assumptions, let C be a cycle of length $n > 3$ such that for every $v \in C$ we have $2 \leq h(v) \leq 4$, and let $h(a) = 2$ for some vertex $a \in C$. As a reminder, our assumptions also imply $\langle C, h\rangle$ does not contain a winning subgame and the hatness sequences $(3, 2, 3)$ and $(2, 3, 3)$ do not appear. Note that a is the only vertex with hatness 2 as otherwise Condition 3 would be satisfied by repeated application of Theorem 3.2.1 to the game $\langle K_2, \star 2\rangle$.

Based on the supposition, we have two possible minimal cases, as suggested in [28]. It suffices to prove the claim for these minimal cases as increasing the hatnesses of any vertex cannot cause a losing game to become winning. The first case is that the hatnesses of a's neighbors are each 4, and every other hatness is 3. The second case is that a has

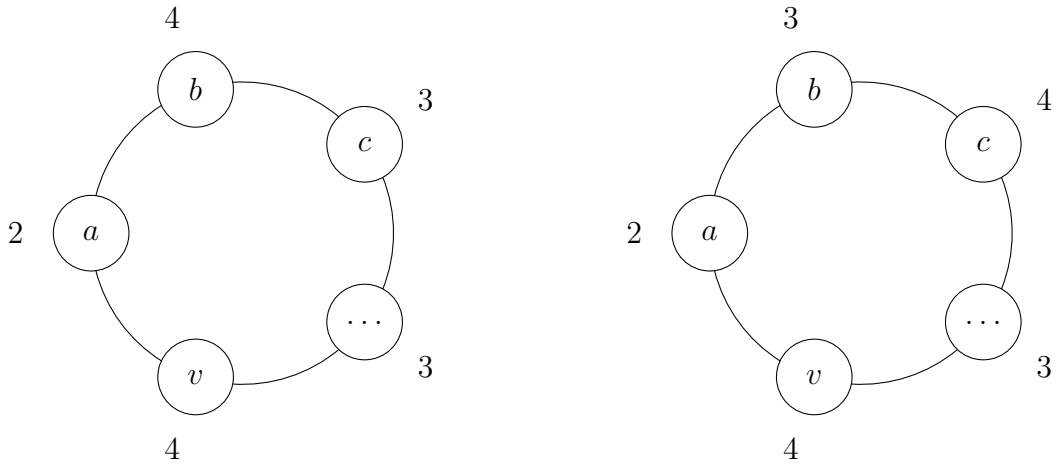


Figure 3.2: The two cases described in the proof of Theorem 3.1.2, where the ellipsis indicates an arbitrary (possibly 0) number of vertices with hatness 3.

one neighbor b with hatness 3 and one with hatness 4, and the second neighbor of b has hatness 4. Any other vertices in the cycle have hatness 3 if they exist. We prove the result for these two possibilities separately, although the technique is quite similar.

In the first case, let v, a, b, and c be a sequence of vertices in the cycle with

$$h(v) = 4,\ h(a) = 2,\ h(b) = 4,\ h(c) = 3,$$

and any other vertices have hatness 3, as shown on the left side of Figure 3.2. Let x_b be the color guessed least frequently by vertex b (breaking ties arbitrarily). By the pigeonhole principle, b can only guess x_b at most once. Let x_a and x_c be the colors of a and c, respectively, when b guesses x_b (if x_b is never guessed then a and c can be chosen arbitrarily). Let X_v be the set of colors for v so that when b is colored x_b, and v is colored from X_v, a guesses color x_a. Suppose toward a contradiction that $|X_v| \geq 2$. Then we commit to coloring a with the color which is not x_a, b with the color x_b, c with any of the three colors, and v with a color from X_v, so a and b guess incorrectly. Hence, $\langle C, h\rangle$ is winning only if there exists a winning strategy for the game on a path of length at least 2 where every vertex has hatness 3 except for one vertex with hatness 2, as depicted in the right side of Figure 3.3. Theorem 3.2.2 implies that this path is losing, hence in any winning strategy we may assume that $|X_v| \leq 1$. Thus, we can commit to coloring a with x_a, v with one of at least 3 colors not in X_v, coloring b with x_b, and coloring c with one of the two colors not equal to x_c. This ensures again that a and b each guess incorrectly, and we again reduce to

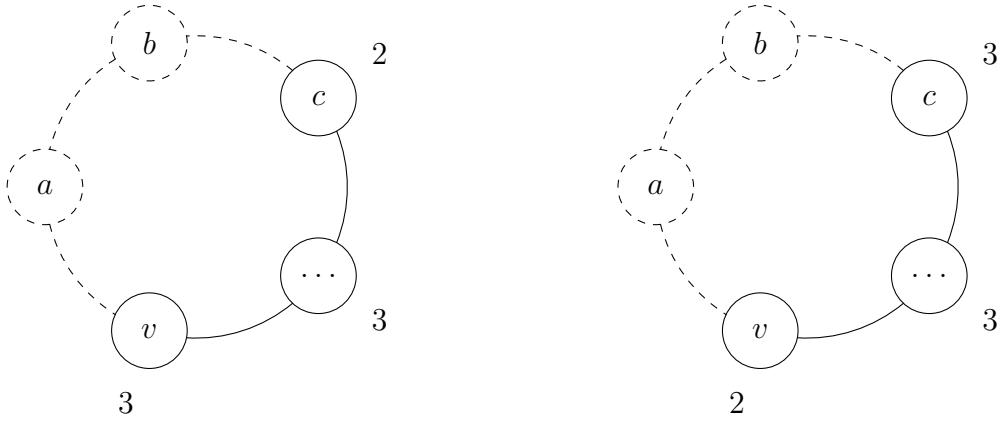


Figure 3.3: The two cases described in the proof of Theorem 3.1.2, after coloring a and b and committing to a subset of colors for c and v. The ellipsis indicates an arbitrary (possibly 0) number of vertices with hatness 3.

the path where every vertex has hatness 3 except one with hatness 2 (in this case depicted on the left side of Figure 3.3), which is losing. Hence, C is losing.

Next, we prove the result in the second case, where we have a sequence of vertices v, a, b, c in the cycle with

$$h(v) = 4,\ h(a) = 2,\ h(b) = 3,\ h(c) = 4,$$

and any other vertices in the cycle have hatness 3 if they exist (as shown on the right side of Figure 3.2). As before, we let x_b be the least frequently guessed color for b, breaking ties arbitrarily. By the pigeonhole principle, x_b is guessed at most twice. Let (x_a, x_c) and (y_a, y_c) be the colorings of (a, c) for which b guesses x_b. If x_b is guessed at most once then we can choose arbitrary values for any undetermined pairs. Now, there are three possibilities to consider. The first case is $x_c = y_c$; the second is $x_a = y_a$; the third is $x_a \neq y_a$ and $x_c \neq y_c$.

In the first case, we commit to coloring b with the color x_b, and c with a color other than x_c. This reduces to a path where the first vertex has hatness 2, the second vertex has hatness 4, and the remaining vertices have hatness 3, which is losing by Theorem 3.2.2.

In the second case, we proceed as in the $(4, 2, 4, 3)$ case. Let X_v be the set of colors for v such that a guesses x_a. As before, if $|X_v| \geq 2$ we color a with the color not equal to x_a, v from X_v, and c arbitrarily; otherwise, we color a with x_a, v from colors not in X_v, and

c from colors not x_c or y_c. As before, these choices ensure that a and b guess incorrectly and leave losing paths. (Shown on the right and left sides of Figure 3.3, respectively; we actually have 4 possible colors for c in the first case, but only 3 are needed.)

Finally, in the third case, assume without loss of generality that a guesses x_a at most as frequently as y_a when b is colored x_b. Then we can commit to coloring b with the color x_b, vertex a with the color x_a, and c with a color that is not x_c, of which there are 3 choices. Since x_a is guessed at most as frequently as y_a, there are at least 2 colors for v such that a does not guess x_a, and we can again reduce the game to a losing path (in this case, the one shown on the right in Figure 3.3). Thus, $\langle C, h\rangle$ is losing in all cases when Conditions 1, 2, 3, and 4a fail. □

It remains to show that $\langle C, h\rangle$ is losing when Conditions 1, 2, 3, and 4b fail.

3.2.2 Proof of Theorem 3.1.2 when Condition 4b Fails

We begin with lemmas to simplify the argument for this case. Our first lemma allows us to delete two vertices of a cycle, leaving behind a path that is winning if the original cycle is winning. An application of the lemma is depicted in Figure 3.4.

Lemma 3.2.3. *Let $C = (V, E)$ be a cycle of length at least 4, let $t, u, v, w \in V$ be consecutive vertices in the cycle so that $(t, u), (u, v), (v, w) \in E$, and let $h(\cdot)$ be a hatness function with $h(v) > h(u)$. We define*

$$h'(x) = \begin{cases} \left\lceil h(w)\left(1 - \left\lceil \frac{h(v)}{h(u)} \right\rceil^{-1}\right)\right\rceil & \text{if } x = w \\ h(t) - \left\lfloor \frac{h(t)}{h(u)} \right\rfloor & \text{if } x = t \\ h(x) & \text{if } x \in V \backslash \{t, u, v, w\} \end{cases}.$$

If $\langle C, h\rangle$ is winning then $\langle C \backslash \{u, v\}, h'\rangle$ is winning.

Proof. The first step is to show that there exists a color $x_v \in [h(v)]$ such that for each color $x_u \in [h(u)]$ there is a set $X_w(x_u, x_v) \subseteq [h(w)]$ of colors for w with

$$|X_w(x_u, x_v)| \geq \left\lceil h(w)\left(1 - \left\lceil \frac{h(v)}{h(u)} \right\rceil^{-1}\right)\right\rceil, \tag{3.1}$$

so that if v is colored x_v, u is colored x_u, and w is colored $x_w \in X_w(x_u, x_v)$ then v guesses incorrectly. Indeed, suppose toward a contradiction that no such x_v exists. Then, for

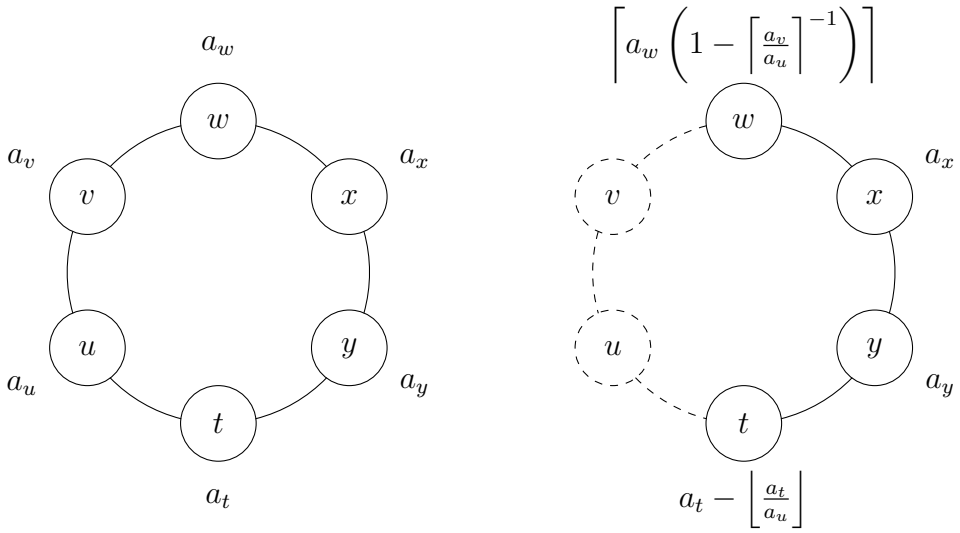


Figure 3.4: A cycle before and after applying Lemma 3.2.3

each color $x_v \in [h(v)]$, there exists a color $x_u \in [h(u)]$ that violates Equation (3.1). That is, there exists a set $Y_w(x_u, x_v) \subseteq [h(w)]$ of colors for w such that v correctly guesses x_v whenever w is colored from $Y_w(x_u, x_v)$ and u is colored x_u, and

$$|Y_w(x_u, x_v)| > h(w) - \left\lceil h(w) \left(1 - \left\lceil \frac{h(v)}{h(u)} \right\rceil^{-1}\right)\right\rceil.$$

By the pigeonhole principle, there is some $y_u \in [h(u)]$ which is used to generate $Y_w(y_u, x_v)$ for a subset $Y_v \subseteq [h(v)]$ of colors with $|Y_v| \geq \left\lceil \frac{h(v)}{h(u)} \right\rceil$. Summing over this set, we get

$$\begin{aligned}
\sum_{y_v \in Y_v} |Y_w(y_u, y_v)| &> \left\lceil \frac{h(v)}{h(u)} \right\rceil \left(h(w) - \left\lceil h(w) \left(1 - \left\lceil \frac{h(v)}{h(u)} \right\rceil^{-1}\right)\right\rceil\right) \\
&\geq \left\lceil \frac{h(v)}{h(u)} \right\rceil \left(h(w) \left\lceil \frac{h(v)}{h(u)} \right\rceil^{-1}\right) \\
&= h(w).
\end{aligned}$$

This is a contradiction, since for a fixed color y_u there are exactly $h(w)$ guesses made by v. Hence, there exists some $z_v \in [h(v)]$ with the desired property.

Fix the color of v to be z_v. With z_v fixed, fix the color of u to be the least frequently guessed color of u, denoted z_u. By our choice of z_u there are at least $h(t) - \left\lfloor \frac{h(t)}{h(u)} \right\rfloor$ colors remaining for t, all of which ensure that u guesses incorrectly. Finally, there are $\left\lceil h(w)\left(1 - \left\lceil \frac{h(v)}{h(u)} \right\rceil^{-1}\right)\right\rceil$ colors for w that ensure v guesses incorrectly. Hence, after the vertices of C commit to a guessing strategy, we can commit to coloring the vertices of $C\backslash\{u, v\}$ using $h'(x)$ colors for each $x \in C\backslash\{u, v\}$ in a way that guarantees u and v guess incorrectly. Hence, if $\langle C, h\rangle$ is winning then the corresponding winning strategy must give a winning strategy on $\langle C\backslash\{u, v\}, h'\rangle$. □

Our next lemma handles a step present in a few cases in Theorem 3.1.2. Equivalent results have been shown in other works, but we reprove the result here in terms that are useful for our arguments. This lemma can be thought of as a "vertex deletion" lemma as it allows us to delete a vertex with hatness 5 lying on a path without changing the winningness of the game.

Lemma 3.2.4. *Let P be a path and let h be a hatness function with $h(v) = 5$ for some $v \in P$. The game $\langle P, h\rangle$ is winning if and only if there is a connected proper subgraph of P which is winning.*

Proof. Let P be a path and let h be a hatness function with $h(v) = 5$ for some $v \in P$. If P contains a winning connected proper subgraph then P is winning. Suppose P does not contain a winning connected proper subgraph and let L, R be the resulting disjoint paths when v is deleted from P. Then L and R are connected proper subgraphs of P, so they are losing with their given hatness functions. The three vertex path with hatnesses $(2, 5, 2)$ is also losing. By Theorem 3.2.2, we can glue together this path with L on one endpoint and R on the other endpoint to get a new losing path where the hatnesses are inherited from the endpoints of L and R. Note that this is exactly the game $\langle P, h\rangle$, so P is losing. □

We are now ready to prove Theorem 3.1.2 when Conditions 1, 2, 3, and 4b all fail. In particular, when Condition 4b fails, there is a vertex with hatness at least 5.

Proof. Let C be a cycle of length at least 4 with at least one vertex of hatness 5 and no winning proper subgraph. Observe that if there are no vertices with hatness 2, the game is losing by [38, Theorem 1]. Hence, we may assume there is at least one vertex with hatness 2. We may also assume without loss of generality that every vertex has hatness 2, 3, or 5 since decreasing the hatness of a vertex does not make a winning game losing. Finally,

on any path between two vertices of hatness 2 we assume there is at most one vertex with hatness 5 for the same reason.

In order to apply Lemma 3.2.3 we need to work with an explicit sequence of hatnesses. Given the above restrictions, we can narrow down the sequences that need to be considered to the following cases:

- There exists a 2 with two adjacent 3's giving the sequences $(3, 2, 3, 3)$ or $(3, 2, 3, 5)$,
- There exists a 2 with exactly one adjacent 3 giving the sequences $(5, 2, 3, 3)$ or $(5, 2, 3, 5)$,
- There are no 2's with any adjacent 3's and we can find the sequence $(2, 5, 2)$.

The conditions in last bullet follows from the restrictions above. We now handle each case.

For the sequence $(2, 5, 2)$, we can choose a color for the vertex of hatness 5 that is never guessed, leaving a path that induces proper subgame of $\langle C, h\rangle$. Hence, this path must be losing and the result holds.

For the sequence $(5, 2, 3, 5)$, we apply Lemma 3.2.3 with

$$h(t) = 5,\ h(u) = 2,\ h(v) = 3,\ h(w) = 5$$

to get a path with endpoints t and w with $h'(t) = 3$ and $h'(w) = 3$. Observe that t and w have degree one in the new graph, and so they are deletable by Theorem 3.2.2. This leaves a proper subgame that must must be losing. Hence, $\langle C, h\rangle$ is losing.

For the sequence $(5, 2, 3, 3)$, we again apply Lemma 3.2.3 with

$$h(t) = 5,\ h(u) = 2,\ h(v) = 3,\ h(w) = 3$$

to get a path with endpoints t and w with $h'(t) = 3$ and $h'(w) = 2$. Observe that t has degree one in the new graph, so it is deletable. Suppose towards a contradiction that the new path is winning. Then, by Theorem 3.2.1, we can glue one end of a path $(2, 3, 2)$ which is winning to w and create a new graph that is still winning but where w has hatness 4. Then we can decrease the hatness of w to 3, and the graph is still winning. However, this new graph is a proper subgraph of the original cycle C, which yields a contradiction. Hence, $\langle C, h\rangle$ is losing.

For the sequence $(3, 2, 3, 3)$, we apply Lemma 3.2.3 with

$$h(t) = 3,\ h(u) = 2,\ h(v) = 3,\ h(w) = 3$$

to get $h'(t) = 2$ and $h'(w) = 2$. Suppose toward a contradiction that the resulting path is winning. Then we can glue a copy of the winning path with hatnesses $(2, 2)$ to t and $(2, 3, 2)$ to w, which gives a new winning path where t and w both have hatness 4 by Theorem 3.2.1. Decreasing the hatnesses for t and w to 3 preserves the winningness of the path. Note that this path contains a vertex of hatness 5. By Lemma 3.2.4 there must be a winning connected proper subgraph. However, any connected proper subgraph of this path cannot contain both endpoints and is thus also a proper subgraph of the original cycle, yielding a contradiction. Hence, $\langle C, h\rangle$ is losing.

For the sequence $(3, 2, 3, 5)$, we apply Lemma 3.2.3 with

$$h(t) = 3,\ h(u) = 2,\ h(v) = 3,\ h(w) = 5$$

to get $h'(t) = 2$ and $h'(w) = 3$. Since w has hatness 3 in the new graph, w is deletable. After deleting w, suppose toward a contradiction that the graph is winning. Then we can glue an edge with hatnesses $(2, 2)$ to t, creating a winning graph where t has hatness 4. We can then decrease the hatness of t to 3. This graph is now a proper subgame of $\langle C, h\rangle$, which must be losing, a contradiction. Hence, $\langle C, h\rangle$ is losing. □

3.3 Cactus Graphs

We start by proving the general upper bound that every cactus graph has hat guessing number at most 4, proving Statement 1 of Theorem 3.1.3. Then, we show how the same argument can be modified to prove Statement 2. Finally, we observe that Statement 3 follows immediately from [27].

Definition 3.3.1. Let G be a cactus graph with a cycle C. We say that C is a *leaf cycle* if deleting the edges of C leaves at most 1 connected component with a cycle (so all other connected components are acyclic).

Figure 3.5 depicts the leaf cycles of a cactus graph in bold.

Proof of Theorem 3.1.3. First, we prove the general upper bound of $HG(G) \leq 4$ for all cactus graphs G by induction on the number of vertices. The base case is trivial. Suppose that the result holds for all cactus graphs on $n \geq 1$ vertices, and let G be a cactus graph with $n + 1$ vertices. We show that $HG(G) < 5$. If G has a vertex v with degree 1 then [2, Theorem 1.8] shows that $HG(G) = HG(G\backslash\{v\})$, and the result holds by induction. Thus, we may assume that G has no vertices of degree 1. If G has no cycles then G is a tree,

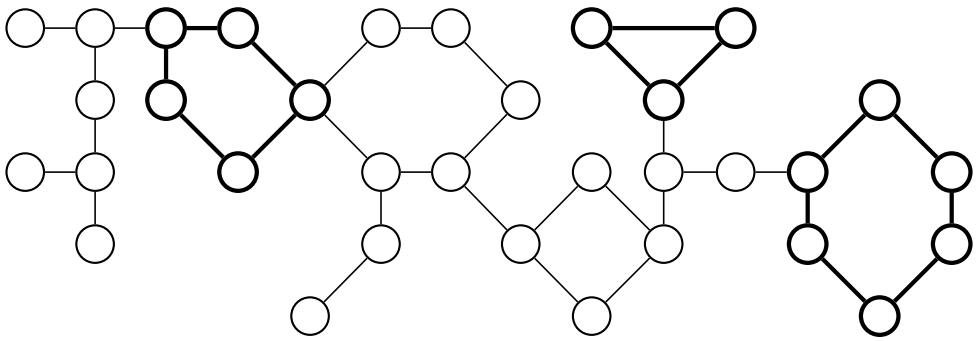


Figure 3.5: A cactus graph with the leaf cycles in bold.

and $HG(G) = 2$. Hence, we may also assume G has at least one cycle. In particular, this implies G has a non-empty leaf cycle C. We can decompose G as $G = C +_{v,v} G'$ at a vertex $v \in C \cap G'$, where G' is a cactus graph on fewer than $n + 1$ vertices. By the induction hypothesis $HG(G') < 5$, so the game $\langle G', \star 5\rangle$ is losing. Theorem 3.1.2 shows that $\langle C, h\rangle$ is losing where $h(v) = 2$ and $h(w) = 5$ for all $w \neq v$. Then $\langle G, \star 5\rangle$ is losing by Theorem 3.2.2, so $HG(G) < 5$ and the result holds.

The same argument can easily be modified to prove the upper bound for Statement 2, by assuming $HG(G) < 4$ in the induction hypothesis for graphs satisfying Statement 2. If the graph has exactly one cycle, then it is a pseudotree and the result holds by [27]. Otherwise, G has at least two cycles, and thus at least two leaf cycles. Statement 2 ensures at most one triangle, so we can choose a non-triangle leaf cycle C in G and write $G = C +_v G'$ where G' is a cactus graph on fewer than $n + 1$ vertices. Then $\langle C, h\rangle$ is losing where $h(v) = 2$ and $h(w) = 4$ for all $w \neq v$, and $\langle G', \star 4\rangle$ is losing by the induction hypothesis, so $\langle G, \star 4\rangle$ is losing by Theorem 3.2.2.

As Statement 3 follows from [27], the proof of Theorem 3.1.3 is complete. □

3.4 Open Problems

Theorem 3.1.2 (along with some lemmas about leaf deletion) can extend the result for pseudotrees [27] to arbitrary hatness functions [30]. Naturally, we would like to do the same for graphs with more than one cycle. The treelike structure of cactus graphs, which this chapter leveraged, could help make them the next step. One challenge that must be overcome is that the hatness for any individual vertex of a winning cactus graph may be

arbitrarily large. For example, we can take n copies of the triangle with hatnesses $(2, 4, 4)$ and glue them together at the vertices of hatness 2. The resulting graph is a winning cactus with a vertex of hatness 2^n. Even worse, we can glue multiple copies of this graph together to get arbitrarily many vertices with hatness 2^n.

Instead, we could turn to *theta graphs*, consisting of three internally vertex disjoint paths that share common endpoints. Thetas are the "simplest" non-cactus graphs, and yet we lack even an analogue to Theorem 3.1.3. Every theta has maximum degree 3, so the hat guessing number of a theta is trivially at most 8 by [16]. Careful analysis using a generalized version of Lemma 3.2.3 shows that thetas have hat guessing number at most 4. We postpone a full characterization of thetas for both constant and general hatness functions to a future work.

Chapter 4

The Sunflower Problem

This chapter is adapted from Chizewer [9].

4.1 Preliminaries

A *set family* $\mathcal{F}$ over a finite set X is a collection of subsets of X. We say that a set family is *n-uniform* if every set in the family has size n.

Definition 4.1.1 (Sunflower)**.** An r-sunflower is a collection of sets $S_1, \ldots, S_r$ such that

$$S_i \cap S_j = S_1 \cap S_2 \cap \cdots \cap S_r = K \text{ for all } i \neq j.$$

We call the set K the *core* and the sets $S_i \backslash K$ the *petals.*

Erdős and Rado [15] originally referred to sunflowers as Δ-systems and proved that given an n-uniform set family $\mathcal{F}$ with $|\mathcal{F}| = n!(r-1)^n$ there exists an r-sunflower contained in $\mathcal{F}$. Sunflowers were renamed by Deza and Frankl in [14], with this new name now dominant in the literature. The sunflower problem has been studied in several papers, including [15, 13, 14, 5, 37, 22], and Erdős and Rado conjectured that a stronger bound holds.

Conjecture 4.1.2 (Erdős and Rado [15])**.** *Let $\mathcal{F}$ be an n-uniform set family. There exists some constant $C = C(r)$ depending only on r such that $\mathcal{F}$ contains an r-sunflower whenever*

$$|\mathcal{F}| > C^n.$$

Recently, Alweiss et al. [5], and subsequently Rao [37] and Bell et al. [6], made progress toward this bound by showing there exists some constant C such that $\mathcal{F}$ contains an r-sunflower whenever $|\mathcal{F}| > (Cr\log n)^n$. We study the sunflower problem with added restrictions on the pairwise intersections.

Definition 4.1.3. Let $\mathcal{F}$ be a set family. We call $\mathcal{F}$ an *L-intersecting family* if there exists a set $L \subset \mathbb{N}$ such that

$$|F_i \cap F_j| \in L \text{ for every } F_i, F_j \in \mathcal{F}, \text{ with } i \neq j.$$

The problem of sunflowers in L-intersecting set families was first studied in [22]. These authors show that given an L-intersecting set family $\mathcal{F}$ with $|L| = s$, the family $\mathcal{F}$ contains a 3-sunflower whenever $|\mathcal{F}| > (n^2-n+1)8^{s-1}2^{(1+\sqrt{5}/5)n(s-1)}$. We improve their bound, and extend the result to all $r \geq 3$ in the following theorem, which is one of the main results of this chapter.

Theorem 4.1.4. *Let $\mathcal{F}$ be an L-intersecting, n-uniform set family, for some set $L \subset \mathbb{N}$ with $|L| = s \geq 1$. Let $m = \max\{r-1, n^2-n+1\}$. Then $\mathcal{F}$ contains an r-sunflower whenever*

$$|\mathcal{F}| > 2^{n\log_2(s+1)+s\log_2(m)}.$$

We also consider the special case where $L = \{0, 1, \ldots, d\}$ for some $d \in \mathbb{N}$. In this case, we call the set family *d-intersecting*. Using the techniques of [5, 37] we achieve the following bound, our second main result.

Theorem 4.1.5. *Let $\mathcal{F}$ be a d-intersecting, n-uniform set family. There exists an absolute constant C such that for every $r, n \geq 3$, the family $\mathcal{F}$ contains an r-sunflower whenever*

$$|\mathcal{F}| > (4r)^n(Cr\log(rd))^d.$$

The results of [6] can be used to get $Cr\log(d)$ instead of $Cr\log(rd)$ by paying an extra factor of 2^n.

Corollary 4.1.6. *Let $\mathcal{F}$ be an n-uniform, r-sunflower-free family for $3 \leq r \leq \log n$. For any $C > 1$ there exists $c = c(r) > 0$, depending only on C and r, such that if $|\mathcal{F}| \geq (C4r)^n$ then there exists $F_1, F_2 \in \mathcal{F}$ with*

$$|F_1 \cap F_2| \geq cn/\log\log n.$$

Corollary 4.1.6 follows immediately from Theorem 4.1.5 by setting $d = cn/\log\log n$. The following question naturally arises.

Question 4.1.7. *Do there exist constants $C, c > 0$ depending only on r, and possibly each other, such that if $\mathcal{F}$ is an n-uniform, r-sunflower-free family of size $|\mathcal{F}| \geq C^n$ then there exist $F_1, F_2 \in \mathcal{F}$ with $|F_1 \cap F_2| \geq cn$?*

Corollary 4.1.6 is of interest because the statement in Question 4.1.7 is equivalent to Conjecture 4.1.2. The proof of this fact follows the same argument that will be used throughout this chapter. We begin, towards a contradiction, with a sunflower-free family of size $(2C)^{n/c}$ (where C and c are given by Question 4.1.7) and identify the largest subfamily with no pairwise intersection of size at least cn. The size of this subfamily is bounded by the statement in Question 4.1.7. Among the remaining sets, we can find $(2C)^{n/c-n}$ of them that all contain the same subset of size cn using the pigeonhole principle and a basic counting argument. Deleting this subset from these sets and applying induction to the family formed by the new sets completes the proof. The reverse direction holds vacuously.

The remainder of the chapter is organized as follows. Theorem 4.1.4 will be proved in Section 4.2, and Theorem 4.1.5 in Section 4.3. We briefly discuss the results in Section 4.4.

4.2 Proof of Theorem 4.1.4

We proceed using similar ideas to the original proof of Erdős and Rado [15], and the subsequent result of Hegedűs [22], beginning with the following lemma.

Lemma 4.2.1 (Deza [13])**.** *Let $\mathcal{F}$ be an L-intersecting, n-uniform set family where $L = \{t\}$ for some $0 \leq t < n$. If $\mathcal{F} \geq n^2 - n + 2$ then $\mathcal{F}$ is a sunflower.*

Lemma 4.2.1 provides the base case for our induction argument. We also state the following definition, which will be useful in this section and the next one.

Definition 4.2.2. Given a family $\mathcal{F}$ over X and a set $T \subseteq X$, the *link* of $\mathcal{F}$ at T, denoted $\mathcal{F}_T$, is defined as

$$\mathcal{F}_T = \{F \setminus T : F \in \mathcal{F}, T \subseteq F\}.$$

Proof of Theorem 4.1.4. We use induction on a slightly stronger statement than that of the theorem.

Claim 4.2.3. *Let $\mathcal{F}$ be an L-intersecting, n-uniform set family for $L = \{\ell_1, \ldots, \ell_s\}$ with $0 \leq \ell_1 < \ell_2 < \cdots < \ell_s < n$ and $s \geq 1$. Let $m = \max\{r-1, n^2 - n + 1\}$. Then $\mathcal{F}$ contains an r-sunflower whenever*

$$|\mathcal{F}| > \frac{n! m^s}{(\ell_1+1)!(\ell_2-\ell_1)!(\ell_3-\ell_2)!\cdots(\ell_s-\ell_{s-1})!(n-\ell_s-1)!}. \tag{4.1}$$

Proof of Claim 4.2.3. Fix m as above. We proceed by induction on s. Indeed, for $s = 1$ we apply Lemma 4.2.1 to get the result immediately. Suppose the result holds for $0 < j < s$, and let $\mathcal{F}$ be an n-uniform, L-intersecting family (with L as above) with $|L| = s \geq 2$. Suppose that $\mathcal{F}$ satisfies Inequality (4.1). Let $\mathcal{S} \subseteq \mathcal{F}$ be a maximal subset of $\mathcal{F}$ such that for every $S_i, S_j \in \mathcal{S}$ if $i \neq j$ then $|S_i \cap S_j| = \ell_1$. If $|\mathcal{S}| > m$ then, by Lemma 4.2.1, $\mathcal{S}$ (and hence $\mathcal{F}$) contains an r-sunflower. Thus, without loss of generality, we can assume that $|\mathcal{S}| \leq m$. By maximality of $\mathcal{S}$, every set in $\mathcal{F}$ intersects at least one set of $\mathcal{S}$ in at least $\ell_1 + 1$ elements. Let $S \in \mathcal{S}$ be the set which intersects the most elements of $\mathcal{F}$ in at least $\ell_1 + 1$ elements, and let

$$\mathcal{F}' = \{F \in \mathcal{F} : |F \cap S| \geq \ell_1 + 1\}.$$

By the pigeonhole principle $|\mathcal{F}'| \geq |\mathcal{F}|/m$. There are $\binom{n}{\ell_1+1}$ subsets of S of size $\ell_1 + 1$, and every set in $\mathcal{F}'$ contains at least one such subset, so again by the pigeonhole principle there exists a set $S' \subseteq S$ such that $|S'| = \ell_1 + 1$ and the link at S' in $\mathcal{F}'$ satisfies

$$|\mathcal{F}'_{S'}| \geq \frac{|\mathcal{F}|}{m\binom{n}{\ell_1+1}} > \frac{(n-\ell_1-1)! m^{s-1}}{(\ell_2-\ell_1)!(\ell_3-\ell_2)!\cdots(\ell_s-\ell_{s-1})!(n-\ell_s-1)!}. \tag{4.2}$$

Let $L' = \{\ell_2 - \ell_1 - 1, \ldots, \ell_s - \ell_1 - 1\}$. We observe that $\mathcal{F}'_{S'}$ is an $(n-\ell_1-1)$-uniform, L'-intersecting family, and $|L'| = s - 1$. Thus, by Inequality (4.2) and induction $\mathcal{F}'_{S'}$ contains an r-sunflower (note that m still satisfies the requirements of the induction hypothesis). If $F_1, \ldots, F_r \in \mathcal{F}'_{S'}$ is an r-sunflower then taking $F_1 \cup S', \ldots, F_r \cup S' \in \mathcal{F}$ gives an r-sunflower. □

Theorem 4.1.4 follows immediately from Claim 4.2.3 using the bound $\binom{n}{m_1,\ldots,m_k} \leq k^n$ for multinomial coefficients. □

4.3 Proof of Theorem 4.1.5

We proceed using a similar argument to the main theorem of [5]. We start by stating some definitions from [5], so that we may apply their results.

Definition 4.3.1. An n-uniform family $\mathcal{F}$ over X is κ-*spread* if $|\mathcal{F}| \geq \kappa^n$ and for all $T \subseteq X$ with $|T| \leq n$ we have $|\mathcal{F}_T| \leq \kappa^{-|T|}|\mathcal{F}|$.

We also introduce weight functions, so that we can deal with multiset families (this is not strictly necessary, but it makes it easier to apply the results of Alweiss et al. [5]).

Definition 4.3.2. A function $\sigma : \mathcal{F} \to \mathbb{Q}$ is a *weight function* on a set family $\mathcal{F}$ if it maps each set in $\mathcal{F}$ to a rational weight, such that not all sets have weight zero. Moreover we define $\sigma(\mathcal{S}) = \sum_{S \in \mathcal{S}} \sigma(S)$ for a set family $\mathcal{S} \subseteq \mathcal{F}$.

Our next definition generalizes the idea of κ-spread using weight functions.

Definition 4.3.3. A set family $\mathcal{F}$ over X, and corresponding weight function $(\mathcal{F}, \sigma)$, is $\mathbf{s}$-*spread* if $\mathbf{s} = (s_0; s_1, \ldots, s_n)$ satisfies $s_0 \geq s_1 \geq \cdots \geq s_n \geq 0$ with $\sigma(\mathcal{F}) \geq s_0$ and for every set $T \subseteq X$ the subfamily $\mathcal{T} = \{F \in \mathcal{F} : T \subseteq F\}$ satisfies $\sigma(\mathcal{T}) \leq s_{|T|}$.

Our final definition is used in our probabilistic arguments below. We write $R \sim \mathcal{U}(X, \alpha)$ whenever $R \subseteq X$ is generated by taking each element of X uniformly and independently at random with probability $0 \leq \alpha \leq 1$.

Definition 4.3.4. Let $0 < \alpha, \beta < 1$. A family $\mathcal{F}$ is (α, β)-*satisfying* if given $R \sim \mathcal{U}(X, \alpha)$,

$$\mathbb{P}_R(\exists S \in \mathcal{F}, S \subseteq R) > 1 - \beta.$$

A family $\mathcal{F}$ is $\mathbf{s}$-*spread* if there exists a weight function σ such that $(\mathcal{F}, \sigma)$ is $\mathbf{s}$-spread, and a weight profile $\mathbf{s}$ is (α, β)-*satisfying* if any $\mathbf{s}$-spread family $\mathcal{F}$ is (α, β)-satisfying.

Using these definitions, we can now state the following lemmas.

Lemma 4.3.5 ([5, Lemma 1.6])**.** *If $\mathcal{F}$ is a $(1/r, 1/r)$-satisfying family, and $\emptyset \notin \mathcal{F}$, then $\mathcal{F}$ contains r pairwise disjoint sets.*

Lemma 4.3.6 ([37, Lemma 4])**.** *Let $0 < \alpha, \beta < 1/2$. There exists a universal constant $C > 1$ such that if $\kappa = \kappa(n, \alpha, \beta) = C \log(n/\beta)/\alpha$ and a multiset family $\mathcal{F}$ over X is a κ-spread, n-uniform family then $\mathcal{F}$ is (α, β)-satisfying.*

The next lemma, which is the main technical result of this section, will allow us to use Lemma 4.3.6 on sets of size d for a d-intersecting family.

Lemma 4.3.7. *Let $\mathcal{F}$ be a d-intersecting, n-uniform set family that is $\mathbf{s}$-spread, such that $\mathbf{s} := (|\mathcal{F}|; s_1, \ldots, s_d, 1, \ldots, 1)$. Let $p, \delta > 0$, and suppose that $\mathbf{s}' = ((1-\delta)|\mathcal{F}|; s_1, \ldots, s_d)$ is (α', β')-satisfying. Then $\mathcal{F}$ is (α, β)-satisfying for*

$$\alpha = p + (1-p)\alpha', \text{ and } \beta = \beta' + (2/p)^n/(\delta|\mathcal{F}|)$$

Before proving Lemma 4.3.7, we define a notion of "good" and "bad" set pairs. Bounding the number of bad pairs is the key idea in this proof.

Definition 4.3.8. Let $\mathcal{F}$ be an n-uniform family over X and let $W \subseteq X$. Given $S \in \mathcal{F}$ and $w \in [n]$ we call the set pair $(W, S)_w$ *good* if there exists a set $S' \in \mathcal{F}$ (possibly with $S' = S$) such that

$$S' \setminus W \subseteq S \setminus W, \text{ and } |S' \setminus W| \leq w$$

We call S' a *witness* to the goodness of $(W, S)_w$. We call a set pair *bad* otherwise.

We use Definition 4.3.8 with $w = d$ for our purposes.

Proof of Lemma 4.3.7. Let $\mathcal{F}$ be a d-intersecting, n-uniform set family over X, with $|X| = x$. We begin by bounding the number of bad set pairs using an encoding inspired by [5]. Suppose $(W, S)_d$ is a bad pair for $W \subseteq X$ where $|W| = px$ and $S \in \mathcal{F}$. First, we consider all possible sets $W \cup S$. Since $|S| = n$ and $|W| = px$, we know that $px \leq |W \cup S| \leq px + n$. Hence, there are

$$\sum_{i=0}^{n} \binom{x}{px+i} \leq \sum_{i=0}^{n} \left(\frac{1-p}{p}\right)^i \binom{x}{px} \leq \left(\frac{1-p}{p} + 1\right)^n \binom{x}{px} = p^{-n}\binom{x}{px}$$

possible sets for $W \cup S$. Let $W \cup S$ be the first piece of information in the encoding. There are 2^n possible values for $W \cap S$ since $|S| = n$. Let $W \cap S$ be the second piece of information in the encoding. Now we claim that given these two pieces of information, and the additional information that the corresponding set pair $(W, S)_d$ is bad, we can reconstruct $(W, S)_d$. Indeed, if we knew S in addition to this information, we could clearly reconstruct $(W, S)_d$. Let $S' \in \mathcal{F}$ and suppose that $S' \subseteq W \cup S$, so that $S' \setminus W \subseteq S \setminus W$. Since $(W, S)_d$ is bad, it must be that $|S' \setminus W| > d$, hence $|S' \cap S| > d$. Since $\mathcal{F}$ is d-intersecting, $S' = S$ and there is a unique set S for a given $W \cup S$, which can be computed by taking the unique set $S \subseteq W \cup S$. Since we also know $W \cap S$, we can compute W. Therefore, there are at most $(2/p)^n\binom{x}{px}$ bad set pairs. Since there are $\binom{x}{px}$ possible sets W, the expected number of bad set pairs for a given W is $(2/p)^n$. Let

$\mathcal{S}(W) = \{S \in \mathcal{F} : (W,S)_d \text{ is bad}\}$. By Markov's inequality, the probability over W drawn uniformly from $\binom{X}{px}$, the set of subsets of X with size px, satisfies

$$\mathbb{P}_W(|\mathcal{S}(W)| \geq \delta|\mathcal{F}|) \leq \frac{(2/p)^n}{\delta|\mathcal{F}|}. \tag{4.3}$$

When $|\mathcal{S}(W)| \leq \delta|\mathcal{F}|$ we define a new d-uniform multiset family $\mathcal{F}'$ over $X \setminus W$ which is $\mathbf{s}' = ((1-\delta)|\mathcal{F}|; s_1, \ldots, s_d)$-spread. The rest of the proof follows immediately from the arguments in Section 2.1 of [5]. □

Proof of Theorem 4.1.5. We roughly follow the argument used in [5, 37]. Let $\mathcal{F}$ be a d-intersecting, n-uniform family over X of size $|\mathcal{F}| > (4r)^n(Cr\log(rd))^d$, for C to be chosen later. Let $T \subseteq X$ be the largest set with $|T| \leq d$ (possibly $|T| = 0$) such that

$$|\mathcal{F}_T| \geq (Cr\log(rd))^{-|T|}|\mathcal{F}|.$$

We claim that $\mathcal{F}_T$ is $\kappa = Cr\log(rd)$-spread. Indeed, if $|T| = d$ then $\mathcal{F}_T$ is a family of pairwise disjoint sets, and otherwise we can find a link at $T' \subseteq X \setminus T$ such that $|T'| > 0$ and

$$(\mathcal{F}_T)_{T'} \geq (Cr\log(rd))^{-|T'|}|\mathcal{F}_T|.$$

But then, taking $T' \cup T$ gives a larger set with

$$|\mathcal{F}_{T\cup T'}| \geq (Cr\log(rd))^{-|T\cup T'|}|\mathcal{F}|,$$

a contradiction. Hence, $\mathcal{F}_T$ is $\mathbf{s}$-spread and $(d-|T|)$-intersecting for weight profile

$$\mathbf{s} = (|\mathcal{F}_T|; |\mathcal{F}_T|/\kappa, \ldots, |\mathcal{F}_T|/\kappa^{d-|T|}, 1, \ldots, 1),$$

taking $\sigma(F) = 1$ for all $F \in \mathcal{F}_T$.

Let $\mathbf{s}' = (|\mathcal{F}_T|/2; |\mathcal{F}_T|/\kappa, \ldots, |\mathcal{F}_T|/\kappa^{d-|T|})$. As in [5], we observe that if a family is $(|\mathcal{F}_T|/2; |\mathcal{F}_T|/\kappa, \ldots, |\mathcal{F}_T|/\kappa^{d-|T|})$-spread, then it is also $\mathbf{s}'' = (|\mathcal{F}_T|; |\mathcal{F}_T|/\kappa', \ldots, |\mathcal{F}_T|/\kappa'^{(d-|T|)})$-spread for $\kappa' = \kappa/2$. By Lemma 4.3.6, $\mathbf{s}''$ (and hence also $\mathbf{s}'$) is $(\frac{1}{2r}, \frac{1}{2r})$-satisfying for C chosen sufficiently large. Hence, by Lemma 4.3.7 with $\delta = 1/2$ and $p = \frac{1}{2r}$, we know that $\mathbf{s}$ is $(1/r, 1/r)$-satisfying choosing C sufficiently large according to the result of the lemma. Therefore, by Lemma 4.3.5, this implies $\mathcal{F}_T$ contains r pairwise disjoint sets. Let $F_1, \ldots, F_r \in \mathcal{F}_T$ be pairwise disjoint. Then $F_1 \cup T, \ldots, F_r \cup T \in \mathcal{F}$ is an r-sunflower. □

4.4 Discussion

It is easy to see that Theorem 4.1.5 implies a stronger statement than the best known bound whenever $d = o(n)$. The original bound of Erdős and Rado can be directly applied to d-intersecting sets families to achieve a bound of $(r-1)^{d+1}n!/(n-d)!$, and it is natural to ask when Theorem 4.1.5 improves this trivial bound. There is some constant $c > 0$ such that

$$\frac{(r-1)^{d+1}n!}{(n-d)!} \geq \left(\frac{(r-1)n}{2}\right)^{cn/\log\log n} \geq (4r)^n (Cr\log(rcn/\log\log n))^{cn/\log\log n}$$

for $d \geq cn/\log\log n$ and n sufficiently large. Hence, in this regime, we improve the best known bound. In particular, this motivates Corollary 4.1.6 and Question 4.1.7.

References

[1] G. Adelson-Velsky and E. Landis. An algorithm for the organization of information. In *Proceedings of the USSR Academy of Sciences (in Russian)*, 1962.

[2] N. Alon, O. Ben-Eliezer, C. Shangguan, and I. Tamo. The hat guessing number of graphs. *J. of Combin. Theory, Ser. B*, 144:119–149, 2020.

[3] N. Alon and J. Chizewer. On the hat guessing number of graphs. *Discrete Math.*, 345:112785, 2022.

[4] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, 2016.

[5] R. Alweiss, S. Lovett, K. Wu, and J. Zhang. Improved bounds for the sunflower lemma. *Ann. of Math.*, 194:795–815, 2021.

[6] T. Bell, S. Chueluecha, and L. Warnke. Note on sunflowers. *Discrete Math.*, 344:112367, 2021.

[7] P. Bradshaw. On the hat guessing number of a planar graph class. *J. of Combin. Theory, Ser. B*, 156:174–193, 2022.

[8] S. Butler, M. Hajiaghayi, R. Kleinberg, and T. Leighton. Hat guessing games. *SIAM J. Discrete Math.*, 22:592–605, 2008.

[9] J. Chizewer. On restricted intersections and the sunflower problem. *Graphs and Combin.*, 40(31), 2024.

[10] J. Chizewer, I.M.J McInnis, M. Sohrabi, and S. Kaistha. The hat guessing number of cactus graphs and cycles. *preprint*, arXiv:2312.00928, 2024.

[11] J. Chizewer, S. Melczer, J. Ian Munro, and A. Pun. Enumeration and succinct encoding of avl trees. *preprint*, arXiv:2311.15511, 2024.

[12] D. Clark. *Compact PAT trees*. PhD thesis, University of Waterloo, 1997.

[13] M. Deza. Solution d'un problème de erdős-lovász. *J. Comb. Theory, Ser. B*, 16:166–167, 1974.

[14] M. Deza and P. Frankl. Every large set of equidistant $(0, +1, -1)$-vectors forms a sunflower. *Combinatorica*, 1:225–231, 1981.

[15] P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960.

[16] M. Farnik. *A hat guessing game*. PhD thesis, Jagiellonian University, 2015.

[17] A. Farzan and J. Ian Munro. A uniform paradigm to succinctly encode various families of trees. *Algorithmica*, 68(1):16–40, Jan 2014.

[18] P. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[19] M. Gadouleau and N. Georgiou. New constructions and bounds for winkler's hat game. *SIAM J. Discrete Math.*, 29:823–834, 2015.

[20] Meng He, J. Ian Munro, and S. Srinivasa Rao. Succinct ordinal trees based on tree covering. In *Automata, Languages and Programming*, pages 509–520. Springer, 2007.

[21] X. He, Y. Ido, and B. Przybocki. Hat guessing on books and windmills. *Electron. J. Combin.*, 29:P1.12, 2022.

[22] G. Hegedűs. Sunflowers and l-intersecting families. *AKCE International Journal of Graphs and Combinatorics*, 17:402–406, 2019.

[23] G. Jacobson. Space-efficient static trees and graphs. In *30th Annual Symposium on Foundations of Computer Science*, pages 549–554, 1989.

[24] A. Latyshev K. Kokhas. The power of constructors. *J. Math. Sci.*, 255:124–131, 2021.

[25] V. Retinskiy K. Kokhas, A. Latyshev. Cliques and constructors in "hats" game. ii. *J. Math. Sci.*, 255:58–70, 2021.

[26] C. Knierim, A. Martinsson, and R. Steiner. Hat guessing numbers of strongly degenerate graphs. *SIAM J. Discrete Math.*, 37:1331–1347, 2023.

[27] K. Kokhas and A. Latyshev. For which graphs the sages can guess correctly the color of at least one hat. *J. of Math. Sci.*, 236:503–520, 2018.

[28] K. Kokhas and A. Latyshev. Cliques and constructors in "hats" game, i. *J. Math. Sci.*, 255:39–57, 2021.

[29] A. Latyshev and K. Kokhas. Hat guessing number of planar graphs is at least 22. *Discrete Math.*, 347:113820, 2024.

[30] I.M.J. McInnis. In preparation.

[31] S. Melczer. *An Invitation to Analytic Combinatorics: From One to Several Variables*. Texts and Monographs in Symbolic Computation. Springer International Publishing, 2021.

[32] J. Ian Munro. Tables. In Vijay Chandru and V. Vinay, editors, *Foundations of Software Technology and Theoretical Computer Science, 16th Conference, Hyderabad, India, December 18-20, 1996, Proceedings*, volume 1180 of *Lecture Notes in Computer Science*, pages 37–42. Springer, 1996.

[33] J. Ian Munro, P. K. Nicholson, L. S. Benkner, and S. Wild. Hypersuccinct trees – new universal tree source codes for optimal compressed tree data structures and range minima. In *29th Annual European Symposium on Algorithm*, page 70:1–70:18, 2021.

[34] J. Ian Munro and S. Srinivasa Rao. Succinct representation of data structures. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004.

[35] G. Navarro. *Compact Data Structures: A Practical Approach*. Cambridge University Press, 2016.

[36] A. M. Odlyzko. Some new methods and results in tree enumeration. In *Proceedings of the thirteenth Manitoba conference on numerical mathematics and computing (Winnipeg, Man., 1983)*, volume 42, pages 27–52, 1984.

[37] A. Rao. Coding for sunflowers. *Discrete Analysis*, 2020.

[38] W. Szczechla. The three colour hat guessing game on cycle graphs. *Electron. J. Combin.*, 24:1.37, 2017.