COMBINATORIAL CHANNELS FROM PARTIALLY ORDERED SETS

BY

DANIEL FRANCIS CULLINA

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Associate Professor Negar Kiyavash, Chair
Professor Alexander Barg, University of Maryland
Professor Bruce Hajek
Professor Olgica Milenkovic
Professor R. Srikant

ABSTRACT

A central task of coding theory is the design of schemes to reliably transmit data though space, via communication systems, or through time, via storage systems. Our goal is to identify and exploit structural properties common to a wide variety of coding problems, classical and modern, using the framework of partially ordered sets. We represent adversarial error models as combinatorial channels, form combinatorial channels from posets, identify a structural property of posets that leads to families of channels with the same codes, and bound the size of codes by optimizing over a family of equivalent channels. A large number of previously studied coding problems that fit into this framework. This leads to a new upper bound on the size of s-deletion correcting codes. We use a linear programming framework to obtain sphere-packing upper bounds when there is little underlying symmetry in the coding problem. Finally, we introduce and investigate a strong notion of poset homomorphism: locally bijective cover preserving maps. We look for maps of this type to and from the subsequence partial order on q-ary strings.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A central task of coding theory is the design of schemes to reliably transmit data though space, via communication systems, or through time, via storage systems. Our goal is to identify and exploit structural properties common to a wide variety of coding problems, classical and modern, using the framework of partially ordered sets.

In Chapter 2, we introduce the main ideas. We use the notion of a combinatorial channel to represent an adversarial or worst-case error model. We define partially ordered sets (posets), poset homomorphisms, and the other fundamental concepts. We describe a simple method for forming combinatorial channels from posets and identify a structural property of posets that leads to families of channels with the same codes. This leads to ones of our core techniques: bounding the size of codes by optimizing over a family of equivalent channels. In the second half of the chapter, we give examples of previously studied coding problems that fit into this framework. This chapter is based primarily on [1] and also contains some material from [2].

In Chapter 3, we apply the methods described in Chapter 2 to the subsequence partial order on q-ary strings. This leads to a new upper bound on the size of s-deletion correcting codes, for fixed s and input length going to infinity. The new bounds improves on the previous best bound whenever the number of deletions is greater than the alphabet size. At the end of this chapter, we look at the partial order on integer compositions. This has many features in common with the subsequence partial order, but many enumerative problems are more tractable. The channels related to the composition partial order also have emerging applications, particularly in DNA storage systems. This chapter is based primarily on [3] and Section 3.8 is based on [1].

Chapter 4 covers the linear programming framework for sphere-packing upper bounds. The main contributions are methods for bounding the values

of these linear programs when they involve too many variables and constraints to evaluate or even analyze exactly. These methods are useful when there is little underlying symmetry in the coding problem. This chapter is based on [2].

In Chapter 5, we introduce and investigate a stronger notion of poset homomorphism: locally bijective cover preserving maps. In particular, we look for maps of this type to and from the subsequence partial order on q-ary strings. We have multiple motivations. First, this definition unites several known functions. These connect single deletion correcting binary codes, permutation statistics, and binary trees. Our main result is the construction of a q-ary analogue of one of these functions. The binary specialization of this new function turns out to be distinct from the previously known motivating function, so as a side effect we obtain a new permutation statistic. There are a number of unsolved fundamental enumerative problems related to the subsequence partial order. We hope to attack some of these problems by identifying new structural properties of the subsequence partial order and by finding connections between the subsequence partial order and better-behaved partial orders. Finally, more examples of string statistics are potentially helpful for the construction of deletion correcting codes.

1.1 Notation

Let $\mathbb{N}$ denote the set of non-negative integers and let $[n]$ denote the set of non-negative integers less than n: $\{0, 1, \ldots, n-1\}$.

Let $X \sqcup Y$ be the disjoint union of X and Y, let $X \times Y$ be the Cartesian product of X and Y, let $X \to Y = Y^X$ be the set of functions from X to Y or equivalently strings, sequences, or vectors of elements of Y indexed by X. Let 2^X denote the power set of X. Let $\binom{X}{k} = \{S \in 2^X : |S| = k\}$, the set of k-subsets of X. Note that $|X \sqcup Y| = |X| + |Y|$, $|X \times Y| = |X| \times |Y|$, $|Y^X| = |Y|^{|X|}$, $|2^X| = 2^{|X|}$, and $\left|\binom{X}{k}\right| = \binom{|X|}{k}$.

In particular, let $[q]^n$ be the set of q-ary strings of length n. Sometimes we will wish to emphasize the index set $[n]$ and write $[q]^{[n]}$ or $[n] \to [q]$. A string of length n is indexed by the set $[n]$. Thus x_0 is the first (leftmost) symbol of x. Indices outside of $[n]$ will be interpreted modulo n. This allows us to use x_{-1} to refer to the last (rightmost) symbol of x. A string can be

written as a list of symbols: $x = \langle x_0, x_1, \ldots, x_{-1} \rangle$. This notation makes the distinction between a symbol and a string of length one clear: i vs $\langle i \rangle$. The empty string is represented by $\langle\rangle$. Finally, let $[q]^*$ be the set of q-ary strings of all lengths:

$$[q]^* = \bigsqcup_{n \in \mathbb{N}} [q]^{[n]}.$$

We represent the closed interval of indices starting at i and ending at j as $[i, j]$. Thus $x_{[1,-1]}$ is the string containing all but the first symbol of x, and $x_{[0,-2]}$ contains all but the last symbol of x.

We will define a number of higher order functions, i.e. functions whose domain or codomain is a set of functions. For example if we have $f : X \to (Y \to Z)$, then for $x \in X$ we have $f(x) : Y \to Z$ and for $y \in Y$ we have $f(x)(y) \in Z$.

We will also define several functions that map an element of some set X to a subset of X. For such a function f and a subset $S \subseteq X$, we will use a standard abuse of notation and write $f(S)$ to mean $\bigcup_{x \in S} f(x)$.

When R is a semiring and X and Y are finite, for the purposes of matrix multiplication, we treat elements of R^X as $|X|$-dimensional column vectors of elements of R indexed by X. Similarly, we treat elements of $R^{X \times Y}$ as $|X|$ by $|Y|$ matrices of elements of R with the rows indexed by X and the columns indexed by Y. Let $\mathbf{1}$ be the column vector of all ones and let $\mathbf{0}$ be the column vector of all zeros. For a set $S \subseteq X$, let $\mathbf{1}_S \in \{0, 1\}^X$ be the indicator column vector for the set S.

We will need the following asymptotic notation: let $a(n) \sim b(n)$ denote that $\lim_{n \to \infty} \frac{a(n)}{b(n)} = 1$ and $a(n) \lesssim b(n)$ denote that $\lim_{n \to \infty} \frac{a(n)}{b(n)} \leq 1$. We will use the following asymptotic equality frequently: for fixed c, $\binom{n}{c} \sim \frac{n^c}{c!}$.

CHAPTER 2

COMBINATORIAL CHANNELS FROM POSETS

2.1 Sphere-packing bounds and linear programs

In this section, we review the well-known connections between combinatorial channels, hypergraphs, sphere-packing bounds, and linear programming.

2.1.1 Combinatorial channels

We use the concept of a combinatorial channel to formalize a set of possible errors.

Definition 2.1.1. *A combinatorial channel is a matrix $A \in \{0,1\}^{X \times Y}$, where X is the set of channel inputs and Y is the set of channel outputs. An output y can be produced from an input x by the channel if $A_{x,y} = 1$. Each row or column of A must contain at least a single one, so each input can produce some output and each output can be produced from some input.*

We will often think of a channel as a bipartite graph. In this case, the left vertex set is X, the right vertex set is Y, and A is the bipartite adjacency matrix. We will refer to this bipartite graph as the *channel graph*.[1] For $x \in X$, let $N_A(x) \subseteq Y$ be the neighborhood of x in the channel graph (the

[1] An equivalent approach, taken by Kulkarni and Kiyavash [4], is to represent an error model by a hypergraph. A hypergraph consists of a vertex set and a family of hyperedges. Each hyperedge is a nonempty subset of the vertices. A hypergraph $\mathcal{H} = (V, E)$ can be described by a vertex-hyperedge incidence matrix $H \in \{0,1\}^{V \times E}$. There are two ways to encode an error model as a hypergraph. Let $A \in \{0,1\}^{X \times Y}$ be the combinatorial channel for that error model. The first option is to take $H = A$ to be the incidence matrix of the hypergraph. The hypergraph vertices are the channel inputs and there is an edge for each output. Alternatively, we can let $H = A^T$ and obtain the dual of the previous hypergraph. Now the hypergraph vertices are the channel outputs. This is the option taken by Kulkarni and Kiyavash.

Throughout this dissertation, we use the language of channels and bipartite channel graphs rather than that of hypergraphs. This allows us to refer to channel inputs and

set of outputs that can be produced from x). The degree of x is $|N_A(x)|$. For $y \in Y$, let $N_A(y) \subseteq X$ be the neighborhood of y in the channel graph (the set of inputs that can produce y). In most cases, the channel involved will be evident and we will drop the subscript on N.

Note that $A\mathbf{1}_{\{y\}} = \mathbf{1}_{N(y)}$ and $\mathbf{1}_{\{x\}}^T A = \mathbf{1}_{N(x)}^T$. Thus $A\mathbf{1}$ is the vector of input degrees of the channel graph, $A^T\mathbf{1}$ is the vector of output degrees, and $\mathbf{1}^T A\mathbf{1}$ is the number of edges.

We are interested in the problem of transmitting information through a combinatorial channel with no possibility of error. To do this, the transmitter only uses a subset of the possible channel inputs in such a way that the receiver can always determine which input was transmitted.

Definition 2.1.2. *A code for a combinatorial channel $A \in \{0,1\}^{X \times Y}$ is a set $C \subseteq X$ such that for all $y \in Y$, $|N(y) \cap C| \leq 1$.*

This condition ensures that decoding is always possible: if y is received, the transmitted symbol must have been the unique element of $N(y) \cap C$.

2.1.2 Sphere-packing

A code is a packing of the neighborhoods of the inputs into the output space. The neighborhoods of the codewords must be disjoint and each neighborhood contains at least $\min_{x \in X} |N(x)|$ outputs. Thus the simplest sphere-packing upper bound on the size of a code C is

$$|C| \leq \frac{|Y|}{\min_{x \in X} |N_A(x)|}.$$

This is the *minimum degree upper bound*, because $|N_A(x)|$ is the degree of x in the channel graph of A. The sphere-packing upper bounds discussed in this chapter are generalizations of and improvements on this bound.

Maximum input packing and its dual, minimum output covering, are naturally expressed as integer linear programs.

Definition 2.1.3. *For a channel $A \in \{0,1\}^{X \times Y}$, the size of the largest input*

outputs using symmetric language and avoids any confusion between a hypergraph and its dual.

packing, or code, is

$$p(A) = \max_{w \in \mathbb{N}^X} \mathbf{1}^T w \quad \text{s. t. } A^T w \leq \mathbf{1}.$$

The size of the smallest output covering is

$$\kappa(A) = \min_{z \in \mathbb{N}^Y} \mathbf{1}^T z \quad \text{s. t. } Az \geq \mathbf{1}.$$

An output covering can be thought of as a strategy for the adversary operating the channel. Whenever the transmitter selects an input x, there is at least one $y \in N(x)$ included in the covering and the adversary can select this as the channel output. This restricts the number of distinguishable messages available to the transmitter to the size of the output covering. Note that the adversary's choices do not require knowledge of the transmitter's choice of code.

2.1.3 Fractional relaxations

The maximum independent set and minimum dominating set problems over general graphs are NP-hard [5]. The approximate versions of these problems are also hard. The maximum independent set of an n-vertex graph cannot be approximated within a factor of $n^{1-\epsilon}$ for any ϵ in polynomial time unless any problem in NP can be solved in probabilistic polynomial time [6]. We seek efficiently computable bounds. These bounds cannot be good for all graphs, but they will perform reasonably well for many of the graphs that we are interested in.

The relaxed problem, maximum fractional set packing, provides an upper bound on the original packing problem.

Definition 2.1.4. *Let $A \in \{0,1\}^{X \times Y}$ be a channel. The size of the maximum*

fractional input packing in A is

$$\begin{aligned} p^*(A) = \max_{w \in \mathbb{R}^X} & \mathbf{1}^T w \\ \text{s. t. } & w \geq \mathbf{0} \\ & A^T w \leq \mathbf{1}. \end{aligned}$$

The size of the minimum fractional output covering is

$$\begin{aligned} \kappa^*(A) = \min_{z \in \mathbb{R}^Y} & \mathbf{1}^T z \\ \text{s. t. } & z \geq \mathbf{0} \\ & Az \geq \mathbf{1}. \end{aligned}$$

The fractional programs have larger feasible spaces, so $p(A) \leq p^*(A)$ and $\kappa^*(A) \leq \kappa(A)$. By strong linear programming duality, $p^*(A) = \kappa^*(A)$.

Unlike the integer programs, the values of the fractional linear programs can be computed in polynomial time. However, we are usually interested in sequences of channels with exponentially large input and output spaces. In these cases, finding exact solutions to the linear programs is intractable but we would still like to know as much as possible about the behavior of the solutions. There is a vast literature devoted to iterative algorithms for solving linear programs. Because our programs are exponentially large,[2] we cannot estimate their values by simulating these algorithms. Instead, we analyze the behavior of a few initial iterations of an algorithm. This leads us to value an unusual set of properties and to propose some very simple iterative algorithms that meet our needs.

2.2 Confusability graphs and families of channels

Sphere-packing upper bounds for coding problems are obtained from combinatorial channels. However, it is well known that for any channel there is a simpler object that also characterizes the set of codes: the confusability graph. Furthermore, any particular confusability graph arises from many

[2]Although the matrices in the programs of interest to us are exponentially large, any entry can be computed in polynomial time. Thus the specifications of these programs are highly compressible.

combinatorial channels. To obtain upper bounds on the size of codes for one channel it can be useful to consider the sphere-packing bounds that arise from some other equivalent channel. At the end of this section, we show how the Hamming and Singleton bounds are an example of this phenomenon.

2.2.1 Confusability graphs and independent sets

First we give a few standard definitions, generally following [7], among others.

Definition 2.2.1. *For a channel $A \in \{0,1\}^{X \times Y}$, the confusability graph of A has vertex set X. Distinct vertices u and v are adjacent in the confusability graph of A if and only if $N(u)$ and $N(v)$ intersect.*

Definition 2.2.2. *Let G be an undirected simple graph with vertex set X. A set $S \subseteq X$ is independent in G if and only if for all $u, v \in S$, u and v are not adjacent. The maximum size of an independent set in G is denoted by $\alpha(G)$.*

Now we have a second important characterization of codes.

Lemma 2.2.1. *Let G be the confusability graph for a channel $A \in \{0,1\}^{X \times Y}$. Then a set $C \subseteq X$ is code for a A if and only if it is an independent set in G. Thus $\alpha(G) = p(A)$.*

Proof: A set C is not a code if and only if there is some y such that $N(y)$ contains distinct codewords u and v. Equivalently $N(u)$ intersects $N(v)$ and u and v are adjacent in G. ■

The confusability graph does not contain enough information to recover the original channel graph, but it contains enough information to determine whether a set is a code for the original channel.

2.2.2 Families of channels with the same codes

There are many different channels that have G as a confusability graph. A *clique* in a graph G is a set of vertices S such that for all distinct $u, v \in S$, $\{u, v\} \in E(G)$. If G is the confusability graph for a channel $A \in \{0,1\}^{X \times Y}$, then for each $y \in Y$, $N(y)$ is a clique in G. Let $\Omega \subseteq 2^X$ be a family of cliques that covers every edge in G. This means that for all $\{u, v\} \in E(G)$,

there is some $S \in \Omega$ such that $u, v \in S$. Let $H \in \{0,1\}^{X \times \Omega}$ be the vertex-clique incidence matrix: $H_{x,S} = 1$ is $x \in S$ and $H_{x,S} = 0$ otherwise. Then $\alpha(G) = p(H)$.

Thus each family of cliques that covers every edge gives us an integer linear program that expresses the maximum independent set problem for G. These programs all contain the same integer points, the indicators of the independent sets of G. However, their polytopes are significantly different so the fractional relaxations of these programs give widely varying upper bounds on $\alpha(G)$.

Each edge in G is a clique, so $E(G)$ is one natural choice for Ω. Then $\alpha(G) = p(H_E)$, where $H_E \in \{0,1\}^{X \times E(G)}$ is the vertex edge incidence matrix for G. However, relaxing the integrality constraint for this program gives a useless upper bound. The vector $w = \frac{1}{2}\mathbf{1}$ is feasible, so $p^*(H_E) \geq \frac{|X|}{2}$ regardless of the structure of G.

Definition 2.2.3. *Let Ω be the set of* maximal *cliques in G and let $H_\Omega \in \{0,1\}^{X \times \Omega}$ be the vertex-clique incidence matrix. Define the minimum clique cover of G, $\theta(G) \triangleq \kappa(H_\Omega)$ and the minimum fraction clique cover $\theta^*(G) \triangleq \kappa^*(H_\Omega)$.*

Every edge is contained in at least one maximal clique, so $\alpha(G) = p(H_\Omega)$. Unlike the program derived from the edge set, $\theta^*(G)$ gives a nontrivial upper bound on $\alpha(G)$. In fact, $\theta^*(G)$ is the best sphere-packing bound for any channel that has G as its confusability graph.

Lemma 2.2.2. *Let G be a graph with vertex set X and let $\Omega_1, \Omega_2 \subseteq 2^X$ be families of cliques that cover every edge in G. Let H_1, H_2 be the vertex-clique incidence matrices for Ω_1 and Ω_2 respectively. If for each $R \in \Omega_1$ there is some $S \in \Omega_2$ such that $R \subseteq S$, then $p^*(H_2) \leq p^*(H_1)$.*

Proof: A clique S gives the constraint $\sum_{x \in S} w_x \leq 1$ in p. If $R \in \Omega_1$, $S \in \Omega_2$, and $R \subseteq S$, then the constraint from R is implied by the constraint for S. Any additional cliques in Ω_2 can only reduce the feasible space for $p(H_2)$. Thus the feasible space for $p(H_2)$ is contained in the feasible space for $p(H_1)$. ■

Corollary 2.2.1. *Let $A \in \{0,1\}^{X \times Y}$ be a channel and let G be the confusability graph for A. Then $\theta^*(G) \leq \kappa^*(A)$.*

Proof: For each output $y \in Y$, $N(y)$ is a clique in G and these cliques cover every edge of G. Each clique in G is contained in a maximal clique, so the claim follows immediately from Lemma 2.2.2. ■

Corollary 2.2.1 suggests that we should ignore the structure of our original channel A and try to compute $\theta^*(G)$ instead of $\kappa^*(A)$. However, there is no guarantee that we can efficiently construct the linear program for $\theta^*(G)$ by starting with G and searching for all of the maximal cliques. We are often interested in graphs with an exponential number of vertices. Even worse, the number of maximal cliques in G can grow exponentially in the number of vertices. To demonstrate this, consider a complete k-partite graph with two vertices in each part. If we select one vertex from each part, we obtain a maximal (and also maximum) clique. The graph has $2k$ vertices, but there are 2^k maximal cliques.

The fractional clique cover number has been considered in the coding theory literature in connection with the Shannon capacity of a graph, $\Theta(G)$. The capacity of a combinatorial channel A is $\lim_{n\to\infty} p(A^n)^{\frac{1}{n}}$, the number of possible messages per channel use when the channel can be used many times. Like $p(A)$, the capacity of the channel depends only on its confusability graph. Thus the Shannon capacity of a graph G can be defined as the capacity of a channel with confusability graph G. The Shannon capacity of a graph is at least as large as the maximum independent set and is not known to even be computable. Shannon used something equivalent to a clique cover as an upper bound for Shannon capacity [8]. Rosenfeld showed the connection between Shannon's bound and linear programming [9]. Shannon also showed that the feedback capacity of a combinatorial channel A is $p^*(A)$. Lovasz introduced the Lovasz theta function of a graph, $\vartheta(G)$, and showed that it was always between the Shannon capacity and the fractional clique cover number [7]. All together, we have

$$\alpha(G) \leq \Theta(G) \leq \vartheta(G) \leq \theta^*(G).$$

The Lovasz theta function is derived via semidefinite programming and consequently is not a sphere-packing bound.

There are also several connections between these concepts and communication over probabilistic channels. For a combinatorial channel A, the minimum capacity over the probabilistic channels with support A is $p^*(A)$.

Recently Dalai has proven upper bounds on the reliability function of a probabilistic channel that are finite for all rates above at the (logarithmic) Shannon capacity of the underlying confusion graph, in contrast to previous bounds that were finite for rates above $\log p^*(A)$ [10]. The idea of multiple channels with the same confusion graph plays an important role here.

2.3 Partial orders

Definition 2.3.1. *A* partially ordered set *or* poset *consists of a set and an order relation. A relation $\geq$ is an order relation on a set X if it is*

- *Reflexive: For all x, $x \geq x$,*
- *Antisymmetric: $x \geq y$, $y \geq x \implies x = y$,*
- *Transitive: $x \geq y$, $y \geq z \implies x \geq z$.*

Definition 2.3.2. *Let $(X, \geq)$ be a poset.*

- *A point x is* comparable *to a point y if either $x \geq y$ or $x \leq y$.*
- *A point x is the* bottom element *if for all $y \in X$, $x \leq y$. Denote the bottom element by $\perp$.*
- *A point x* covers *a point y, written $x \succ y$, if $x > y$ and there is no $z \in X$ such that $x > z > y$.*
- *$U(x) = \{y \in P : y \geq x\}$ is the* up-set *of x.*
- *$D(x) = \{y \in P : y \leq x\}$ the* down-set *of x.*

One important way to construct partial orders is the product operation.

Definition 2.3.3. *For $i \in [k]$, let $(X_i, \leq_i)$ be a poset. Then the* product poset *is $(\prod_{i\in[k]} X_i, \bigwedge_{i\in[k]} \leq_i)$. That is, the product poset is defined on the Cartesian product of the component posets and for $y, z \in \prod_{i\in[k]} X_i$, $y \leq z$ if and only if $y_i \leq_i z_i$ for all $i \in [k]$.*

Example 2.3.1. *The natural numbers form a partial order under the usual order relation: for $x, y \in \mathbb{N}$, $x \leq y$ if there is some $z \in \mathbb{N}$ such that $x + z = y$. Any two natural numbers are comparable. The bottom element is 0 and $x \in \mathbb{N}$ is covered by $x + 1$.*

2.3.1 Poset homomorphisms

Definition 2.3.4. *Let (X, P) and (Z, Q) be posets and let $f : X \to Z$.*

- *If for all $x, y \in X$, $x \geq y \implies f(x) \geq f(y)$, then f is* order preserving.
- *If $x > y \implies f(x) > f(y)$, then f is* strict-order preserving.
- *If $x \succ y \implies f(x) \succ f(y)$, then f is* cover preserving.

Each property is strictly stronger than the prior properties.

Definition 2.3.5. *Let (X, P) be a poset. A* rank function *on (X, P) is a cover preserving map $r : X \to \mathbb{N}$.*

All of the posets that we will consider have a rank function and a bottom element. We will always use the canonical rank function that assigns a rank of zero to the bottom element and we will write $|x|$ for the rank of $x \in X$.

Definition 2.3.6. *Let r be a rank function on (X, P). For $x \in X$, define*

- *the rank n elements: $P_n = \{y \in P : \mathrm{r}(y) = n\}$,*
- $U_a(x) = \{y \in U(x) : \mathrm{r}(y) = \mathrm{r}(x) + a\}$,
- $D_a(x) = \{y \in D(x) : \mathrm{r}(y) = \mathrm{r}(x) - a\}$.

Remark 2.3.1. *We will define several posets by specifying their cover relation. The existence of a rank function makes it easy to verify that a particular relation is in fact the cover relation of a partial order: it is sufficient to check that the rank function maps all pairs in the cover relation to successive integers.*

Definition 2.3.7. *Let r be a rank function on $(X, \leq)$. For $x \in X$, define*

$$U_s(x) = \{y \in X : y \geq x,\ \mathrm{r}(y) = \mathrm{r}(x) + s\},$$
$$D_s(x) = \{y \in X : y \leq x,\ \mathrm{r}(y) = \mathrm{r}(x) - s\}.$$

Many of the posets that we encounter will be semilattices or lattices.

Definition 2.3.8. *A poset $(X, \leq)$ is a* meet semilattice *if for all $x, y \in X$, the set $D(x) \cap D(y)$ has a unique maximal element. Call this element the* meet *of x and y, written $x \wedge y$. $(X, \leq)$ is a* join semilattice *if for all $x, y \in X$, the set $U(x) \cap U(y)$ has a unique minimal element. Call this element the* join *of x and y, written $x \vee y$. $(X, \leq)$ is a* lattice *if it has both properties.*

The properties are preserved under products.

2.4 Combinatorial channels from posets

In this section, we show how to define combinatorial channels from a poset and determine the conditions under which these channels have the same confusion graph. We can think of a step up or down in poset as a error. This interpretation allows us to define a family of combinatorial channels poset. Throughout this section, let $(X, \leq)$ be a ranked poset with a bottom element and let $X_n = U_n(\perp)$, the set of elements of rank n. The rank-n, a-down-error, b-up-error channel has input space X_n and output space X_{n-a+b}. The neighborhood of an input x is $U_b(D_a(x))$. In Sections 2.5, 2.6, 2.7, 2.8, and 2.9, we will see that many well-studied combinatorial channels fit into this framework.

We will be particularly interesting ranked posets that have a useful structural property. This property can be expressed in several equivalent ways.

Lemma 2.4.1. *Let P be a ranked poset. The following conditions are equivalent:*

- *For all $x, y \in P$ such that $\mathrm{r}(x) \geq 1$ and $\mathrm{r}(y) \geq 1$,*
  $\exists z \in D_1(x) \cap D_1(y) \iff \exists w \in U_1(x) \cap U_1(y)$
- *For all $x, y \in P$ such that $\mathrm{r}(x) \geq a$ and $\mathrm{r}(y) \geq b$,*
  $\exists z \in D_a(x) \cap D_b(y) \iff \exists w \in U_b(x) \cap U_a(y)$
- *For all x such that $\mathrm{r}(x) \geq 1$, $U_1(D_1(x)) = D_1(U_1(x))$*
- *For all x such that $\mathrm{r}(x) \geq a$, $U_b(D_a(x)) = D_a(U_b(x))$.*

It is easy to show that if two posets each satisfy all of the conditions of Lemma 2.4.1, then their product does as well.

Definition 2.4.1. *Let X be a ranked poset with a bottom element and let* r

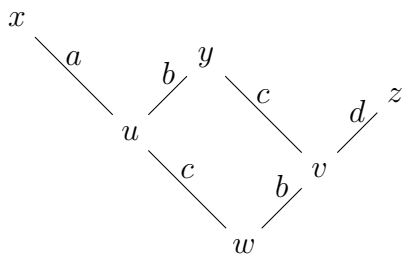

Figure 2.1: The poset elements used in the proof of Lemma 2.4.2. The lines indicate the order relationships and are labeled with the rank differences.

be the rank function such that $\mathrm{r}(\bot) = 0$. *Then for* $x, y \in X$, *let*

$$\begin{aligned} f &: X \times X \to \mathbb{N} \\ f(x,y) &= \max\{\mathrm{r}(z) : z \in D(x) \cap D(y)\} \\ g &: X \times X \to \mathbb{N} \times \mathbb{N} \\ g(x,y) &= (\mathrm{r}(x) - f(x,y), \mathrm{r}(y) - f(x,y)). \end{aligned}$$

Lemma 2.4.2. *The function* $g(x,y)$ *satisfies the following properties:*

- $g(x,y) \geq (0,0)$
- $g(x,y) = (0,0) \iff x = y$
- $g(x,y) = (a,b) \iff g(y,x) = (b,a)$
- *Triangle inequality:* $g(x,y) + g(y,z) \leq g(x,z)$
- *Converse of triangle inequality: If* $g(x,z) = (a+c, b+d)$ *and* $\mathrm{r}(x) - a + b \leq m$, *then there is some* y *such that* $g(x,y) = (a,b)$ *and* $g(y,z) = (c,d)$.

Proof: The first three properties follow immediately from the definition of g.

If $U_b(D_a(x)) \cap U_c(D_d(z))$ is nonempty, then equivalently

$$\begin{aligned}
& (g(x,y) \leq (a,b)) \wedge (g(y,z) \leq (c,d)) \\
\iff & \exists y \in U_b(D_a(x)) \cap U_d(D_c(z)) \\
\iff & \exists u \in D_a(x), v \in D_d(z) : \exists y \in U_b(u) \cap U_c(v) \\
\iff & \exists u \in D_a(x), v \in D_d(z) : \exists w \in D_c(u) \cap D_b(v) \\
\iff & \exists w \in D_c(D_a(x)) \cap D_b(D_d(z)) \\
\iff & g(x,z) \leq (a+c, b+d),
\end{aligned}$$

so equivalently $D_{a+b}(x) \cap D_{a+b}(y)$ is nonempty.

The middle equivalence uses Lemma 2.4.1. Figure 2.1 illustrates the relationships between the strings in this proof. ■

There are many metrics based on g relating poset elements of different ranks. Within each rank, there is just one natural metric derived from g.

Lemma 2.4.3. *The function $d : X_n \times X_n \to \mathbb{N}$, $d(x,y) = n - f(x,y)$ is a metric on X_n.*

Proof: For $x, y \in X_n$, $g(x,y) = (d(x,y), d(x,y))$. By the properties of g from Lemma 2.4.2, d is a metric. ■

Now we can state the main result: poset channels performing different mixtures of up and down errors have the same codes.

Theorem 2.4.1. *Let X be a ranked poset with a bottom element and let X satisfy the conditions of Lemma 2.4.1. Then $C \subseteq X_n$ is a code for the rank-n a-down-error b-up-error channel if and only if for all $x, y \in X_n$, $d(x,y) \geq a + b$.*

Proof: If C is not a code, there are $x, y \in C$ such that $U_b(D_a(x)) \cap U_b(D_a(y))$ is nonempty. Then

$$\begin{aligned}
& \exists w \in U_b(D_a(x)) \cap U_b(D_a(y)) \\
\iff & \exists u \in D_a(x), v \in D_a(y) : \exists w \in U_b(u) \cap U_b(v) \\
\iff & \exists u \in D_a(x), v \in D_a(y) : \exists z \in D_b(u) \cap D_b(v) \\
\iff & \exists z \in D_b(D_a(x)) \cap D_b(D_a(y)),
\end{aligned}$$

so equivalently $D_{a+b}(x) \cap D_{a+b}(y)$ is nonempty. The middle equivalence uses

Lemma 2.4.1. The relationships between the poset elements in this proof are illustrated in Figure 2.2. ■

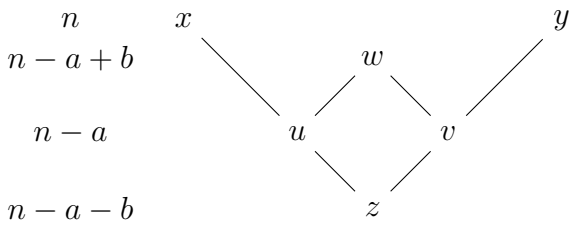

Figure 2.2: The strings used in the proof of Lemma 2.4.1. Their lengths are given on the left and the lines indicate their order relationships.

2.5 Constant weight binary vectors: Substitution errors

In the next few sections we discuss several well-known error models and distance metrics that are connected to posets.

Let **2** be the poset with two comparable elements. Call the elements 0 and 1 and let $1 > 0$. The poset $P = \mathbf{2}^d$ is connected to substitution errors for constant weight binary vectors. The rank function is the Hamming weight. A down error is the replacement of a one with a zero and an up error is the replacement of a zero with a one. This poset is a lattice, so for any x and y, the maximum rank element in $D(x) \cap D(y)$ is unique: it is the meet of x and y. It is not hard to show that as this poset satisfies the parallelogram property. Sphere packing bounds are trivial to derive because all of the channels are input and output regular.

Alternatively, this poset can also be thought of as the poset of subsets of of a set ordered by containment. This perspective connects sphere-packing bounds for the channels derived from this poset to an well-known problem of extremal set theory. A family of n-element subsets of d is t-intersecting if each pair of subsets in the family have at least t elements in common. This means that the distance in the poset metric between any two subsets in the family is at most $2n-2t$. Thus the family is a clique in the rank-n distance-$2s$ confusability graph. Translated from the language of intersecting families to that of combinatorial channels, the Erdős-Ko-Rado theorem [11] states that

if d is much larger than n, the largest clique in $G_{d,n}$ is the neighborhood of an output the s-down error channel.

The complete intersection theorem of Ahlswede and Khachatrian [12] answers the analogous question for all ranks of $\mathbf{2}^d$. This theorem says that for all ranks n and distances $2s$, the largest clique in the confusability graph is the neighborhood of an output of some a-down error b-up error channel. Because the confusability graph is vertex transitive, this implies that covering bound from that channel is equal to $\theta^*(G)$.

2.6 q-ary vectors: Substitution errors and erasures

Consider the poset on $\{\epsilon\} \sqcup [q]$ such that ϵ has rank zero and each element of $[q]$ has rank one. This is depicted in Figure 2.3.

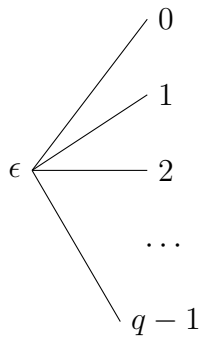

Figure 2.3: The cover relations of the poset $(\{\epsilon\} \sqcup [q], \leq)$.

Then the product poset $(\{\epsilon\} \sqcup [q])^d$ is connect to substitution errors and erasure on q-ary vectors of length d. The simplest nontrivial example, $q = 2$ and $d = 2$, is depicted in Figure 2.4.

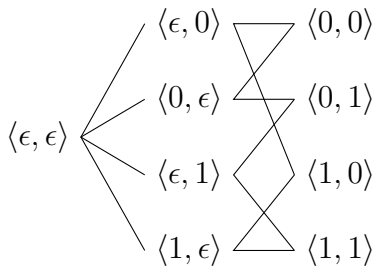

Figure 2.4: $(\{\epsilon\} \sqcup [2])^2$.

The rank function of an element $x \in (\{\epsilon\} \sqcup [q])^d$ is the number of symbols in x that are from $[q]$. Thus the top rank of $(\{\epsilon\} \sqcup [q])^d$ consists of the q-ary vectors and lower ranks contain partially erased vectors. A down-error is an erasure, an up-error in an unerasure, and the combination of a down-error with an up-error is a substitution.

It is easy to show that $(\{\epsilon\} \sqcup [q])^d$ satisfies a weaker variant of the parallelogram property. This condition is sufficient to guarantee that all of the s-error channels with the top rank as their input space have the same codes.

Now we can find the best possible sphere-packing bound for the Hamming distance by optimizing over the following family of channels. Consider the channel that takes a q-ary vector of length n as its input, erases a symbols, and substitutes up to b symbols. Thus there are q^n channel inputs, $\binom{n}{a}q^{n-a}$ outputs, and each input can produce $\binom{n}{a}\sum_{i=0}^{b}\binom{n-a}{i}(q-1)^i$ possible outputs. Two inputs share a common output if and only if their Hamming distance is at most $s = a + 2b$. For each choice of n and s, we have a family of channels with identical confusability graphs. Call the q-ary n-symbol a-erasure b-substitution channel $A_{q,n,a,b}$. These channels are all input and output regular, so

$$\begin{aligned}\kappa^*(A_{q,n,a,b}) &= \frac{\binom{n}{a}q^{n-a}}{\binom{n}{a}\sum_{i=0}^{b}\binom{n-a}{i}(q-1)^i} \\ &= \frac{q^{n-a}}{\sum_{i=0}^{b}\binom{n-a}{i}(q-1)^i}.\end{aligned}$$

Two special cases give familiar bounds. For even s, setting $a = 0$ and $b = s/2$ produces the Hamming bound:

$$\kappa^*(A_{q,n,0,s/2}) = \frac{q^n}{\sum_{i=0}^{s/2}\binom{n}{i}(q-1)^i}.$$

Setting $a = s$ and $b = 0$ produces the Singleton bound:

$$\kappa^*(A_{q,n,s,0}) = q^{n-s}.$$

For $q = 2$, the Hamming bound is always the best bound in this family. When q is at least 3, each bound in the family is the best for some region of the parameter space.

Lemma 2.6.1. *$\kappa^*(A_{q,n,a,b}) \leq \kappa^*(A_{q,n,a+2,b-1})$ when $a + qb \leq n - 1$.*

Proof: We can rewrite the initial inequality as

$$\begin{aligned}
\kappa^*(A_{q,n,a+2,b-1}) &\geq \kappa^*(A_{q,n,a,b}) \\
\frac{q^{n-a-2}}{\sum_{i=0}^{b-1}\binom{n-a-2}{i}(q-1)^i} &\geq \frac{q^{n-a}}{\sum_{i=0}^{b}\binom{n-a}{i}(q-1)^i} \\
\sum_{i=0}^{b}\binom{n-a}{i}(q-1)^i &\geq q^2\sum_{i=0}^{b-1}\binom{n-a-2}{i}(q-1)^i. \qquad (2.1)
\end{aligned}$$

To simplify (2.1), we use the following identity:

$$\begin{aligned}
&\sum_{i=0}^{b}\binom{n-c+2}{i}(q-1)^i \\
&= \sum_{i=0}^{b}\left(\binom{n-c}{i-2} + 2\binom{n-c}{i-1} + \binom{n-c}{i}\right)(q-1)^i \\
&= \sum_{i=0}^{b-2}\binom{n-c}{i}(q-1)^{i+2} + 2\sum_{i=0}^{b-1}\binom{n-c}{i}(q-1)^{i+1} + \\
&\quad \sum_{i=0}^{b}\binom{n-c}{i}(q-1)^i \\
&= \binom{n-c}{b}(q-1)^b - \binom{n-c}{b-1}(q-1)^{b+1} + \\
&\quad ((q-1)^2 + 2(q-1) + 1)\sum_{i=0}^{b-1}\binom{n-c}{i}(q-1)^i \\
&= \binom{n-c}{b-1}(q-1)^b\left(\frac{n-c-b+1}{b} - q + 1\right) + \\
&\quad q^2\sum_{i=0}^{b-1}\binom{n-c}{i}(q-1)^i.
\end{aligned}$$

By setting $c = a + 2$, we can use this to rewrite the left side of (2.1). Eliminating the common term from both sides of the inequality gives

$$\begin{aligned}
\binom{n-a-2}{b-1}(q-1)^b\left(\frac{n-a-b-1}{b} - q + 1\right) &\geq 0 \\
\frac{n-a-b-1}{b} - q + 1 &\geq 0 \\
n - a - 1 - qb &\geq 0,
\end{aligned}$$

which proves the claim. ∎

Theorem 2.6.1. *Let $q, n, s \in \mathbb{N}$ such that $q \geq 3$, $0 \leq s \leq n-1$, and s even. Then*

$$\operatorname*{argmin}_{0 \leq b \leq s/2} \kappa^*(A_{q,n,s-2b,b}) = \begin{cases} s/2 & s \leq \frac{2}{q}(n-1) \\ \left\lfloor \frac{n-1-s}{q-2} \right\rfloor & s \geq \frac{2}{q}(n-1). \end{cases}$$

For fixed $\delta, \frac{2}{q} \leq \delta \leq 1$, and $s = \delta n$

$$\lim_{n\to\infty} \frac{1}{n} \log \min_{0 \leq b \leq s/2} \kappa^*(A_{q,n,s-2b,b}) = (1-\delta)\log(q-1).$$

Proof: Let $a+2b = s$, so $a+qb = s+(q-2)b$. Lemma 2.6.1 allows us to determine the value of b minimizing $\kappa^*(A_{q,n,s-2b,b})$. When $s+(q-2)\frac{s}{2} \leq n-1$, or equivalently $s \leq \frac{2}{q}(n-1)$, $\kappa^*(A_{q,n,0,s/2})$ is the smallest in the family. For $b \geq 1$

$$\kappa^*(A_{q,n,a+2,b-1}) \geq \kappa^*(A_{q,n,a,b}) \leq \kappa^*(A_{q,n,a-2,b+1})$$

if and only if

$$b \leq \frac{n-1-s}{q-2} \leq b+1.$$

Let b^* be the optimal choice of b. Finally,

$$\begin{aligned}
&\lim_{n\to\infty} \frac{1}{n} \log \frac{q^{n-s+2b^*}}{\sum_{i=0}^{b^*} \binom{n-s+2b^*}{i} (q-1)^i} \\
&= \lim_{n\to\infty} \frac{n-s+2b^*}{n} \log q - \frac{n-s+2b^*}{n} H_2\left(\frac{b^*}{n-s+2b^*}\right) - \frac{b^*}{n}\log(q-1) \\
&= \frac{q(1-\delta)}{q-2}\log q - \frac{q(1-\delta)}{q-2} H_2(1/q) - \frac{1-\delta}{q-2}\log(q-1) \\
&= \frac{1-\delta}{q-2}\left(q\log q - \log q - (q-1)\log\frac{q}{q-1} - \log(q-1)\right) \\
&= \frac{1-\delta}{q-2}((q-1)\log(q-1) - \log(q-1)) \\
&= (1-\delta)\log(q-1),
\end{aligned}$$

where we used

$$\begin{aligned}
\lim_{n\to\infty} \quad \frac{b^*}{n} \quad &= \frac{1-\delta}{q-2}, \\
\lim_{n\to\infty} \frac{n-s+2b^*}{n} &= 1-\delta+2\frac{1-\delta}{q-2} = \frac{q(1-\delta)}{q-2}, \\
\lim_{n\to\infty} \frac{b^*}{n-s+2b^*} &= \frac{1}{q}.
\end{aligned}$$

■

Figure 2.5 plots this optimized bound, the Hamming bound, and Singleton bound for $q = 4$. The intermediate region of the optimized bound is linear because each clique used in the intermediate cover is the product of a Hamming-type clique with a Singleton-type clique.

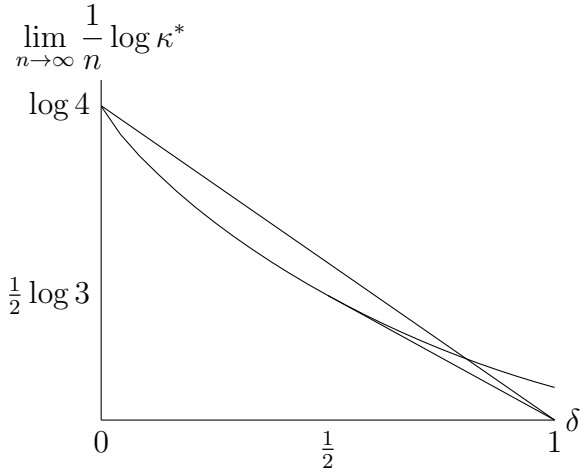

Figure 2.5: Sphere-packing bounds for channel performing substitution errors and erasures. The curved line is the Hamming bound, which is $\lim_{n\to\infty} \frac{1}{n} \log \kappa^*(A_{4,n,0,s/2})$. The upper straight line is the Singleton bound, which is $\lim_{n\to\infty} \frac{1}{n} \log \kappa^*(A_{4,n,s,0})$. The straight line running from $(\frac{1}{2}, \frac{1}{2} \log 3)$ to $(1, 0)$ is the optimized sphere-packing bound, $\lim_{n\to\infty} \frac{1}{n} \log \min_{0\le b\le s/2} \kappa^*(A_{4,n,s-2b,b})$.

For $q = 2$, the Hamming bound is always the best bound in this family. When $q \ge 3$, the Hamming bound is the best for $s \le \frac{2}{q}(n-1)$. For larger number of errors, bounds from other error mixtures become better. Each bound in the family is the best for some region of the parameter space [13]. This error model has been very thoroughly studied and the sphere-packing

bounds are superseded by much more sophisticated results, but the connection with the poset framework is still interesting.

2.7 q-ary strings: Insertions and deletions

Consider the set of q-ary strings of any length ordered by the subsequence relation. This is depicted in Figure 2.6. The rank function for this poset is string length. A down error in this poset is the deletion of a symbol from a string. An up error is the insertion of a symbol.

The order relation can be formally defined as follows.

Definition 2.7.1. *Let $x \in ([m] \to [q])$ and $y \in ([n] \to [q])$. Then $x \leq y$ if there is an order preserving injection $f : [m] \to [n]$ such that $x = y \circ f$.*

It is easy to verify that this defines a partial order. The relation is reflexive because the identity map $[m] \to [m]$ always exists. The relation is antisymmetric because there are no other order preserving injections $[m] \to [m]$. Transitivity follows from composition of order preserving injections.

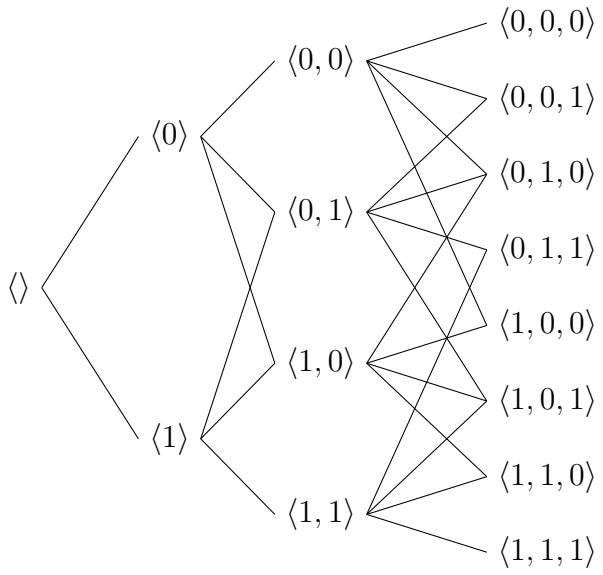

Figure 2.6: The cover relations for ranks 0 through 3 of the $([2]^*, \mathbf{subseq})$ poset.

It is not hard to verify that this channel satisfies the parallelogram property. In Chapters 3 and 5, we will thoroughly investigate this poset. In

particular, we will analyze channels performing a mixture of insertions and deletions on q-ary strings and find that the best sphere-packing bound comes from a channel that performs approximately $\frac{qs}{q+1}$ deletions and $\frac{s}{q+1}$ insertions.

2.8 Edit distances

In Section 2.7, we defined the subsequence partial order on $[q]^*$. This definition can be generalized as follows.

Definition 2.8.1. *Let $(X, \leq)$ be a poset. Define $(X^*, \textbf{\textit{subseq}})$ as follows. Let $x \in ([m] \to X)$ and $y \in ([n] \to X)$. Then $x \leq y$ if there is an order preserving injection $f : [m] \to [n]$ such that $x \leq y \circ f$ in the product poset on $X^{[m]}$. That is, $x_i \leq y_{f(i)}$ for all $i \in [m]$.*

The subsequence order on $[q]$,$([q]^*, \textbf{subseq})$, comes from applying this construction to the partial order on $[q]$ in which all distinct elements are incomparable and the deletion distance arises from $([q]^*, \textbf{subseq})$. If instead we let $X = \{\epsilon\} \sqcup [q]$ with the partial order described in Section 2.6, we obtain the poset depicted in Figure 2.7, which corresponds to another interesting distance.

If $x \prec y$, then y can be produced from x either by inserting an ϵ symbol or by replacing an existing ϵ with a symbol from $[q]$. Thus the rank of x is equal to the length of x plus the number of symbols in x that are from $[q]$. The metric derived from this rank function is an edit distance in which substitutions, deletions, and insertions all cost 2. To produce an edit distance with arbitrary weights on the two types of operations, we can define a generalized "rank" function: α times the length plus β times the number of symbols from $[q]$. In the metric derived from this function, substitutions cost 2β and insertions and deletions cost $\alpha + \beta$. The deletion distance comes from $\alpha = 0$, $\beta = 1$ and the Hamming distance comes from $\alpha = \frac{1}{2}$, $\beta \to \infty$.

2.9 Permutations

There are distance metric and several error models for permutations that are connect to partial order. Hamming distance on permutations has a very

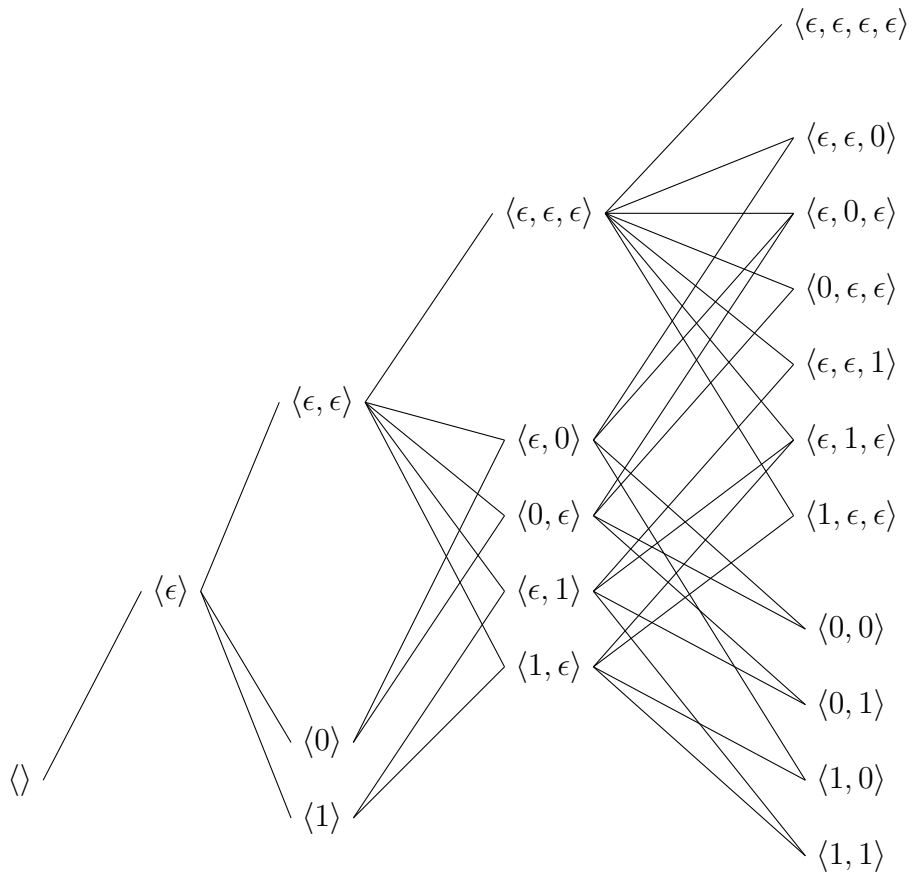

Figure 2.7: The cover relations for ranks 0 through 3 of the $([2]^*, \mathbf{subseq})$ poset.

simple connection: the permutations are a subset of the n-ary vectors of length n.

Kendall-tau distance is associated with adjacent transposition errors. It has a more interesting connection to the erasure poset. A partial order on a set of n distinguishable elements can be represented by $\binom{n}{2}$ comparisons. The comparison of incomparable elements is recorded as an erasure. Any binary vector with erasures of length $\binom{n}{2}$ that satisfies the transitivity condition corresponds to some poset. Thus partial orderings of a n labeled elements form a subposet of the poset of length-$\binom{n}{2}$ binary vectors with erasures. Permutations have no incomparable elements and form the top layer of this poset The number of comparable pairs is the rank function for this poset and the Kendall-tau distance is the natural metric.

CHAPTER 3

THE SUBSEQUENCE PARTIAL ORDER

3.1 Introduction

Deletion channels output only a subsequence of their input while preserving the order of the transmitted symbols. Deletion channels are related to synchronization problems, a wide variety of problems in bioinformatics, and the communication of information over packet networks. This chapter concerns channels that take a fixed-length input string of symbols drawn from a q-ary alphabet and delete a fixed number of symbols. In particular, we are interested in upper bounds on the cardinality of the largest possible s-deletion correcting codebook.

Levenshtein derived asymptotic upper and lower bounds on the sizes of binary codes for any number of deletions [14]. These bounds easily generalize to the q-ary case [15]. He showed that the Varshamov-Tenengolts (VT) codes, which had been designed to correct a single asymmetric error [16, 17], could be used to correct a single deletion. The VT codes establish the asymptotic tightness of the upper bound in the case of a binary alphabet and a single deletion.

Since then, a wide variety of code constructions, which provide lower bounds, have been proposed for the deletion channel and other closely related channels. One recent construction uses constant Hamming weight deletion correcting codes [18]. In contrast, progress on upper bounds has been rare. Levenshtein eventually refined his original asymptotic bound (and the parallel nonbinary bound of Tenengolts) into a nonasymptotic version [19]. Kulkarni and Kiyavash recently proved a better upper bound for an arbitrary number of deletions and any alphabet size [4].

Another line of work has attacked some related combinatorial problems. These include characterization of the sets of supersequences and subsequences

of any string. Levenshtein showed that the number of supersequences does not depend on the starting string [20]. He also gave upper and lower bounds on the number of subsequences using the number of runs in the starting string [14]. Calabi and Hartnett gave a tight bound on the number of subsequences of each length [21]. Hirschberg extended the bound to larger alphabets [22]. Swart and Ferreira gave a formula for the number of distinct subsequences produced by two deletions for any starting string [23]. Mercier et al. showed how to generate corresponding formulas for more deletions and gave an efficient algorithm to count the distinct subsequences of any length of a string [24]. Liron and Langberg improved and unified existing bounds and constructed tightness examples [25]. Some of our intermediate results contribute to this area.

3.1.1 Upper bound technique

To derive our upper bounds, we use a packing argument that can be applied to any combinatorial channel. Any combinatorial channel can be represented by a bipartite graph. Channel inputs correspond to left vertices, channel outputs correspond to right vertices, and each edge connects an input to an output that can be produced from it. If two channel inputs share a common output, they cannot both appear in the same code. The degree of an input vertex in the graph is the number of possible channel outputs for that input. If the degree of each input is at least r and there are N possible outputs, any code contains at most N/r codewords.

Any code capable of correcting s deletions is also capable of correcting any combination of s total insertions and deletions. We discuss this equivalence in Section 3.2.1. Despite this, the packing argument produces different upper bounds for channels that perform different mixtures of insertions and deletions. Let $C_{q,s,n}$ be the size of the largest q-ary n-symbol s-deletion correcting code. Prior to this work, the bounds on $C_{q,s,n}$ coming from the s-insertion channel and the s-deletion channel were known.

For the s-insertion channel, each q-ary n-symbol input has the same degree. For the exact value, see Lemma 3.3.1. For fixed q and s, the degree is

asymptotic to $\binom{n}{s}(q-1)^s$. There are q^{n+s} possible outputs, so

$$C_{q,s,n} \leq \frac{q^{n+s}}{\binom{n}{s}(q-1)^s}(1+o(1)). \tag{3.1}$$

The s-deletion case is slightly more complicated because different inputs have different degrees. For instance, the input strings consisting of a single symbol repeated n times have only a single possible output: the string with that symbol repeated $n-s$ time. Consequently, using the minimum degree over all of the inputs yields a worthless bound. Using the following argument [14], Levenshtein showed that

$$C_{q,s,n} \leq \frac{q^{n}}{\binom{n}{s}(q-1)^s}(1+o(1)). \tag{3.2}$$

The average degree of an input is asymptotic to $\left(\frac{q-1}{q}\right)^s \binom{n}{s}$ and most inputs have a degree close to that. The inputs can be divided into two classes: those with degree at least $1-\epsilon$ times the average degree and those with smaller degree. For an appropriately chosen ϵ that goes to zero as n goes to infinity, the vast majority of inputs fall into the former class. Call members of the former class the typical inputs. The minimum degree argument can be applied to bound the number of typical inputs that can appear in a code. There are q^{n-s} possible outputs, so an asymptotic upper bound on the number of typical inputs in a code is given by (3.2). We have no information about what the fraction of the atypical inputs can appear in a code, but the total number of atypical inputs is small enough to not affect the asymptotics of the upper bound.

The bounds (3.1) and (3.2) have the same growth rates, but the bound on deletion correcting codes is a factor of q^s better than the bound on insertion correcting codes, despite the fact that any s-deletion correcting code is an s-insertion correcting code and vice versa. Note that there is no possible improvement to the insertion channel bound from dividing the inputs into typical and atypical classes.

We extend this bounding strategy to channels that perform both deletions and insertions. We obtain a generalized upper bound that includes Levenshtein's bound as a special case. Recall that Levenshtein's bound is known to be tight for one deletion and alphabet size equal to two. The new

bound improves upon Levenshtein's bound whenever the number of deletions is greater than the alphabet size. The new bound is the best that can be obtained via the technique of discarding atypical vertices.

The rest of the chapter is organized as follows. In Section 3.2, we present some notation and basic results on deletion and insertion channels. In Section 3.4, we construct a class of well-behaved edges in the channel graph. Together with an upper bound on the number of edges in the channel graph, the size of this class establishes the asymptotics of the average input degree. In Section 3.5, we prove a lower bound on the degree of each input vertex and use it to establish our main result: an upper bound on the size of a q-ary s-deletion correcting code.

3.2 Preliminaries

3.2.1 Deletion and insertion channels

The subsequence relation is a partial ordering of $[q]^*$. Consequently for strings x and y, we write $x \preceq y$ when x is a subsequence of y.

We formalize the problem of correcting deletions and insertions by defining the following sets.

Definition 3.2.1. *For $x \in [q]^n$, define $D_a(x) = \{z \in [q]^{n-a} : z \preceq x\}$, the set of subsequences of x that can be produced by a deletions. Define $U_b(x) = \{w \in [q]^{n+b} : w \succeq x\}$, the set of supersequences of x that can be produced by b insertions. Define $S_{a,b}(x) = U_b(D_a(x))$.*

For each input x to an n-symbol a-deletion b-insertion channel $S_{a,b}(x)$ is the set of possible outputs. The following well-known fact about insertions and deletions shows that the sequencing of insertion and deletion errors does not matter.

Lemma 3.2.1. *For all $l, a, b \in \mathbb{N}$ and $x \in [q]^{l+a}$, $U_b(D_a(x)) = D_a(U_b(x))$. For all $x \in [q]^{l+a}$ and $y \in [q]^{l+b}$, $D_a(x) \cap D_b(y) \neq \varnothing$ if and only if $U_b(x) \cap U_a(y) \neq \varnothing$.*

Proof: By the transitivity of the subsequence relation, $D_c(D_d(x)) = D_{c+d}(x)$ and $U_c(U_d(x)) = U_{c+d}(x)$. It is easy to see that $D_1(U_1(x)) = U_1(D_1(x))$, The claim then follows from Lemma 2.4.1. ∎

When two inputs share common outputs they can potentially be confused by the receiver.

Definition 3.2.2. *A q-ary n-symbol a-deletion b-insertion correcting code is a set $C \subset [q]^n$ such that for any two distinct strings $x, y \in C$, $S_{a,b}(x) \cap S_{a,b}(y)$ is empty.*

Applying Theorem 2.4.1, we get the following result. Let $a, b, n \in \mathbb{N}$ and $x, y \in [q]^n$. Then $S_{a,b}(x) \cap S_{a,b}(y) \neq \varnothing$ if and only if $D_{a+b}(x) \cap D_{a+b}(y) \neq \varnothing$. Consequently a set $C \subset [q]^n$ is a q-ary n-symbol a-deletion b-insertion correcting code if and only if it is an $(a+b)$-deletion correcting code.

3.3 Enumerating subsequences and supersequences

Each q-ary string of length n has the same number of supersequences of length n. This fact about insertions was originally proved by Levenshtein [20]. Our proof uses the same ideas as the original to construct an explicit bijection.

Algorithm 1 Encoding and decoding supersequences

```
ENCODE : [q]* × [q]* → [q]*
ENCODE(x, y) = case x of
    ⟨⟩ : y
    ⟨x_0⟩ + x' : case y of
        ⟨⟩ : Error — y is not a supersequence of x.
        ⟨y_0⟩ + y' : case (y_0 − x_0) mod q of
            0 : ⟨0⟩ + ENCODE(x', y')
            i : ⟨i⟩ + ENCODE(x, y')


DECODE : [q]* × [q]* → [q]*
DECODE(x, z) = case x of
    ⟨⟩ : z
    ⟨x_0⟩ + x' : case z of
        ⟨⟩ : Error — z contains too few zeros.
        ⟨z_0⟩ + z' : case z_0 of
            0 : ⟨x_0⟩ + DECODE(x', z')
            i : ⟨(x_0 + i) mod q⟩ + DECODE(x, y')
```

Lemma 3.3.1. *For all $x \in [q]^n$,*

$$|U_s(x)| = \sum_{i=0}^{s} \binom{n+s}{i} (q-1)^i.$$

Proof: The functions $\textsc{Encode}(x, \cdot)$ and $\textsc{Decode}(x, \cdot)$ from Algorithm 1 give a bijection between $|U_s(x)|$ and the strings of length $n + s$ with at most s nonzero entries. ■

The following lemma is originally due to Calabi and Hartnett [21]. Again, we use ideas from the original proof to give an explicit bijection.

Algorithm 2 Encoding and decoding subsequences

$\textsc{Encode} : [q]^* \times [q]^* \to [q]^*$
$\textsc{Encode}(y, x) =$ **case** x **of**
  $\langle\rangle : \langle\rangle$
  $\langle x_0\rangle + x' : \langle j\rangle + \textsc{Encode}(y', x')$ **where** $(j, y') = \textsc{Search}(x_0, \varnothing, y)$

$\textsc{Search} : [q] \times 2^{[q]} \times [q]^* \to ([q] \times [q]^*)$
$\textsc{Search}(i, S, y) =$ **case** y **of**
  $\langle\rangle :$ Error — i does not appear in y.
  $\langle y_0\rangle + y' :$ **case** y_0 **of**
    $y_0 = i : (|S|, y')$
    $y_0 \neq i : \textsc{Search}(i, S + y_0, y')$

$\textsc{Decode} : [q]^* \times [q]^* \to [q]^*$
$\textsc{Decode}(y, z) =$ **case** z **of**
  $\langle\rangle : \langle\rangle$
  $\langle z_0\rangle + z' : \langle i\rangle + \textsc{Decode}(y', z')$ **where** $(i, y') = \textsc{Delete}(z_0, \varnothing, y)$

$\textsc{Delete} : [q] \times 2^{[q]} \times [q]^* \to ([q] \times [q]^*)$
$\textsc{Delete}(j, S, y) =$ **case** y **of**
  $\langle\rangle :$ Error — y does not contains $j + 1$ distinct symbols.
  $\langle y_0\rangle + y' :$ **case** S **of**
    $|S| = j \wedge y_0 \notin S : (y_0, y')$
    $|S| < j \vee y_0 \in S : \textsc{Delete}(j, S + y_0, y')$

Lemma 3.3.2. *For all $x \in [q]^n$, $|D_s(x)|$ is at most the number of q-ary strings of length $n - s$ that sum (over the natural numbers) to at most s:*

$$|D_s(x)| \leq \sum_{i=0}^{s} [z^i](1 + z + \ldots + z^{q-1})^{n-s}.$$

Proof: In Algorithm 2, the functions SEARCH and DELETE serve dual purposes. The SEARCH function deletes symbols from the beginning of y until it finds an appearance of the symbol i. It returns the tail of y (starting after the i) along with the number of distinct symbols that it deleted before finding an i. The DELETE function deletes symbols from the beginning of y until it has deleted j distinct symbols and found a $(j+1)$st distinct symbol. It returns the tail of y and the $(j+1)$st distinct symbol that it found.

Thus $\text{SEARCH}(i, \varnothing, y) = (j, y')$ if and only if $\text{SEARCH}(j, \varnothing, y) = (i, y')$. By induction $\text{ENCODE}(y, x) = z$ if and only if $\text{DECODE}(y, z) = x$. ■

3.4 Constructing edges

Now we will execute the strategy described in Section 3.1.1. The following graph completely describes the behavior of the channel that takes a q-ary input string of length $l + a$ and performs a deletions and b insertions.

Definition 3.4.1. *Let $B_{q,l,a,b}$ be a bipartite graph with left vertex set $[q]^{l+a}$ and right vertex set $[q]^{l+b}$. The neighborhood of each left vertex x is $S_{a,b}(x)$.*

To obtain our upper bound on code size, we will need a lower bound on the degree of each left vertex of $B_{q,l,a,b}$. The goal of this section is an intermediate result: an asymptotically tight lower bound on the number of edges in $B_{q,l,a,b}$. To do this, we find a more easily counted parameter set $P_{q,l,a,b}$ and an injective construction function $\text{CONSTRUCT} : P_{q,l,a,b} \to E(B_{q,l,a,b})$. We demonstrate the injectivity of CONSTRUCT by finding a left inverse of CONSTRUCT and conclude that $|P_{q,l,a,b}| \leq |E(B_{q,l,a,b})|$. We then give a simple upper bound on the number of edges that matches the lower bound asymptotically. In Section 3.5, we obtain our lower bound on input degree by working with the edges of $B_{q,l,a,b}$ that are in the image of CONSTRUCT rather than the complete set of edges.

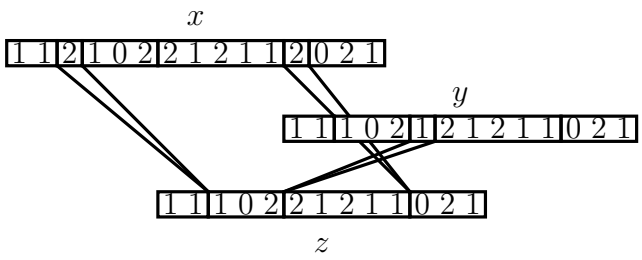

Figure 3.1: An example of an edge $(x, y) \in E(B_{3,13,2,1})$ constructed from a common subsequence $z \in [3]^{13}$.

3.4.1 The construction and deconstruction procedures

Vertices in $B_{q,l,a,b}$ are adjacent if and only if they have a common subsequence of length l. Because of this, to construct an edge $(x, y) \in E(B_{q,l,a,b})$, we start with a string $z \in [q]^l$. Let $s = a + b$. Partition z into $s + 1$ intervals. To produce x, select a of the s boundaries between intervals and insert one new symbol into z at each. To produce y, insert one new symbol into z at each of the other b boundaries. Figure 3.1 gives an example.

This construction procedure is capable of producing every edge in $B_{q,l,a,b}$, but many edges can be produced in multiple ways. We will show that if two restrictions are added to construction procedure, the deconstruction procedure will always be able to recover the construction parameters. This proves that each edge can be produced in at most one way. At the end of the section, we will show that the number of edges that cannot be produced at all under the restrictions is asymptotically negligible.

The first restriction is that each interval must be nonempty and each inserted symbol must differ from the leftmost symbol in the interval to its right. This restriction is needed because inserting a new symbol anywhere within a run of that same symbol has the same effect. Under the restriction, a run in z can only be extended by inserting a matching symbol at the right end.

The second restriction is that each interval of z must be nonalternating.

Definition 3.4.2. *A string is alternating if some $u \in [q]$ appears at all even indices, some $v \in [q]$ appears at all odd indices, and $u \neq v$. A string is nonalternating if it is not alternating. Let $A_{q,*}$ be the set of nonalternating q-ary strings and let $A_{q,n}$ be those of length n.*

The empty string and all strings of length one are trivially alternating, so

the shortest nonalternating strings have length two. For each length $n \geq 2$, each of the q choices for u and $q-1$ choices for v results in a unique alternating string, so $|A_{q,n}| = q^n - q(q-1)$.

Now we decribe the deconstruction procedure. Start with an edge (x, y). Beginning at the left, find the longest matching prefix of x and y and delete it from both. This prefix is the first interval of z. Now the first symbols of x and y differ. One of these symbols is an insertion, but we do not know which one.

To distinguish these two cases, apply the following heuristic. Provisionally delete the first symbol of x and determine the length of the longest common prefix of y and the rest of x. Then do the same with the roles of x and y reversed. Take the longer common prefix to be the next interval of z and the deleted symbol that resulted in this prefix to be the insertion.

After removing this prefix, either the first symbols of x and y again differ or x and y are both the empty string. Apply this heuristic until the latter case is achieved.

3.4.2 Formalization of construction and deconstruction

In this section we will define our construction and deconstruction functions more precisely and prove that the latter inverts the former. Fully formal descriptions of all of the functions described in this section can be found in Section 3.7.

The functions treat strings as lists of symbols. Let $\langle\rangle$ be the empty list. Recall that $x + y$ is the concatenation of x and y and let $(u, v) +\!+ (x, y) = (u + x, v + y)$. Let x_0 be the first symbol of a nonempty string x and let x_{-0} be the rest of the string.

To specify an edge (x, y), we need the following parameters. First, we need $s + 1$ nonalternating strings. When concatenated together, these will form the common subsequence of x and y. Second, for each of the s gaps we pick an element of $\{\mathrm{X}, \mathrm{Y}\}$. This specifies which endpoint of the edge will receive the inserted symbol. Finally, to specify each inserted symbol we pick $\delta \in [q] \setminus \{0\}$. The inserted symbol will be equal to δ plus the first symbol of the next interval modulo q.

An example of the construction procedure can be found in Figure 3.2. The

$$\textsc{Construct}(11, \langle(\mathrm{X}, 1, 102), (\mathrm{Y}, 2, 21211), (\mathrm{X}, 2, 021)\rangle)$$

$$= \begin{matrix}11+\\11+\end{matrix}\textsc{Insert}(\langle(\mathrm{X}, 1, 102), (\mathrm{Y}, 2, 21211), (\mathrm{X}, 2, 021)\rangle)$$

$$= \begin{matrix}11+2102+\\11+\ 102+\end{matrix}\textsc{Insert}(\langle(\mathrm{Y}, 2, 21211), (\mathrm{X}, 2, 021)\rangle)$$

$$= \begin{matrix}11+2102+\ 21211+\\11+\ 102+121211+\end{matrix}\textsc{Insert}(\langle(\mathrm{X}, 2, 021)\rangle)$$

$$= \begin{matrix}11+2102+\ 21211+2021\\11+\ 102+121211+\ 021\end{matrix}$$

$$= \begin{matrix}112102212112021\\11102121211021\end{matrix}$$

Figure 3.2: An example of the construction procedure for a pair of ternary strings. The INSERT function is applied to each triple $(\{\mathrm{X}, \mathrm{Y}\} \times ([3] \setminus \{0\}) \times A_{3,*})$ to produce a pair of string segments. CONSTRUCT concatenates these to produce the final pair.

CONSTRUCT function takes two arguments. The first is w_s, the nonalternating string that will appear at the beginning of both x and y. Because each inserted symbol depends on the following nonalternating string, we group the remaining construction parameters in triples $\{\mathrm{X}, \mathrm{Y}\} \times ([q] \setminus \{0\}) \times A_{q,*}$. The second argument to CONSTRUCT is t, a list of s of these triples. CONSTRUCT uses the INSERT function to turn t into a pair of strings, then prefixes w_s to both members of the pair. The INSERT function starts by translating t_0 into a pair of strings. If w is the string from t_0, one of the output strings is w and the other is $(\delta\ +\ w_0)\ :\ w$. INSERT recursively applies itself to t_{-0} and concatenates the translation of the first triple to the result the recursive call.

The decontruction procedure attempts to recover the parameters to CONSTRUCT from an edge (x, y). An example of the deconstruction procedure can be found in Figure 3.3. The MATCH function takes two strings x and y and finds their longest common prefix. More precisely, $\textsc{Match}(x, y) = (w, (u, v))$ where $x = w{+}u$, $y = w{+}v$, and w is as long as possible. The DECONSTRUCT function uses MATCH to remove the common prefix of the input strings, then calls DELETE on the remaining parts of the input strings. DELETE takes a pair of strings x and y that are either both empty or are both nonempty and differ in their first symbol. In the former case, there are no more construc-

$$
\begin{array}{l}
\textsc{Deconstruct}\begin{pmatrix}1121022121112021\\11102121211021\end{pmatrix}\\
\quad\textsc{Match}\begin{pmatrix}1121022121112021\\11102121211021\end{pmatrix} = 11\begin{matrix}2102212112021\\102121211021\end{matrix}\\
= \left(11, \textsc{Delete}\begin{pmatrix}2102212112021\\102121211021\end{pmatrix}\right)\\
\quad\textsc{Delete}\begin{pmatrix}2102212112021\\102121211021\end{pmatrix}\\
\quad\quad\textsc{Match}\begin{pmatrix}102212112021\\102121211021\end{pmatrix} = 102\begin{matrix}212112021\\121211021\end{matrix}\ \checkmark\\
\quad\quad\textsc{Match}\begin{pmatrix}2102212112021\\02121211021\end{pmatrix} = \langle\rangle\begin{matrix}2102212112021\\02121211021\end{matrix}\\
\quad= \langle(\mathrm{X}, 1, 102)\rangle + \textsc{Delete}\begin{pmatrix}212112021\\121211021\end{pmatrix}\\
\quad\textsc{Delete}\begin{pmatrix}212112021\\121211021\end{pmatrix}\\
\quad\quad\textsc{Match}\begin{pmatrix}12112021\\121211021\end{pmatrix} = 121\begin{matrix}12021\\211021\end{matrix}\\
\quad\quad\textsc{Match}\begin{pmatrix}212112021\\21211021\end{pmatrix} = 21211\begin{matrix}2021\\021\end{matrix}\ \checkmark\\
\quad= \langle(\mathrm{Y}, 2, 21211)\rangle + \textsc{Delete}\begin{pmatrix}2021\\021\end{pmatrix}\\
\quad\textsc{Delete}\begin{pmatrix}2021\\021\end{pmatrix}\\
\quad\quad\textsc{Match}\begin{pmatrix}021\\021\end{pmatrix} = 021\begin{matrix}\langle\rangle\\\langle\rangle\end{matrix}\ \checkmark\\
\quad\quad\textsc{Match}\begin{pmatrix}2021\\21\end{pmatrix} = 2\begin{matrix}021\\1\end{matrix}\\
\quad= \langle(\mathrm{X}, 2, 021)\rangle + \textsc{Delete}\begin{pmatrix}\langle\rangle\\\langle\rangle\end{pmatrix}\\
= (11, \langle(\mathrm{X}, 1, 102), (\mathrm{Y}, 2, 21211), (\mathrm{X}, 2, 021)\rangle)
\end{array}
$$

Figure 3.3: Deconstruction of the edge constructed in Figure 3.2. First, MATCH strips off the common prefix. The DELETE function tests whether it a longer common prefix is achieved by deleting the first symbol of the first string or the second string. The check marks indicate the longer match. It produces a triple specifying that deletion and prefix.

tion parameters to recover so DELETE simply returns $\langle\rangle$. In the latter case, DELETE uses the heuristic described in Section 3.4.1 to determine which first symbol is an insertion. DELETE(x, y) computes MATCH(x_{-0}, y) and MATCH(x, y_{-0}). It assumes that whichever common prefix is longer is the nonalternating string used during construction. The information about the deletion and prefix becomes a triple. Finally, DELETE recursively applies itself to the leftovers from the chosen match.

Now we will show that DECONSTRUCT is a left inverse of CONSTRUCT. The main step is to show that DELETE can recover the parameters given to INSERT.

Lemma 3.4.1. *The function* DELETE *is a left inverse of* INSERT. *That is, for all* $t \in (\{X, Y\} \times ([q] \setminus \{0\}) \times A_{q,*})^*$, $t =$ DELETE(INSERT(t)).

Proof: We show this by induction on the length of t. For the base case, DELETE(INSERT$(\langle\rangle)) =$ DELETE$(\langle\rangle, \langle\rangle) = \langle\rangle$.

If t is nonempty, let $(x, y) =$ INSERT(t). Either the first symbol of x or the first symbol of y is an insertion. Without loss of generality suppose the former case, so $t_0 = (\mathrm{X}, \delta, w)$ where $w = (w_0, \ldots, w_{m-1})$ for some $m \geq 2$. Then $x = \langle \delta + w_0 \rangle + w + x'$ and $y = w + y'$ where $(x', y') =$ INSERT(t_{-0}). Note that the pair of strings produced by INSERT(t_{-0}) are either both empty or both nonempty. If they are nonempty, they have different first symbols.

Recall that DELETE computes both MATCH(x_{-0}, y) and MATCH(x, y_{-0}). MATCH(x_{-0}, y) always equals $(w,$ INSERT$(t_{-0}))$. Let MATCH(x, y_{-0}) be equal to $(z, (u, v))$. Suppose that the length of z is at least $m - 1$. Then we have $z_0 = \delta + w_0 = w_1$ and $z_i = w_{i-1} = w_{i+1}$ for $1 \leq i \leq m - 2$. This implies that w is alternating, which is a contradiction. Thus the length of z is always less than the length of w and DELETE correctly concludes that the first symbol of x was the insertion. The length of t_{-0} is less than the length of t, so by the induction hypothesis DELETE(INSERT$(t_{-0})) = t_{-0}$. Finally, DELETE$(t) = \langle(\mathrm{X}, \delta, w)\rangle + t_{-0} = t_0 + t_{-0} = t$. ∎

In order for the constructed pair of strings to form an edge in $B_{q,l,a,b}$, the lengths of the nonalternating strings must add up to l. Thus each possible vector of string lengths is a composition of l with $s + 1$ parts.

Definition 3.4.3. *A composition of* l *with* t *parts is a list of* t *non-negative integers with sum* l. *Let* $M(t, l, k)$ *be the family of compositions of* l *with* t

parts and each part of size at least k:

$$M(t,l,k) \;=\; \left\{ \lambda \in (\mathbb{N} \setminus [k])^t \;\middle|\; \sum_{i\in[t]} \lambda_i = l \right\}.$$

Each element of $M(t,l,0)$ can be uniquely represented by a string of l item symbols and $t-1$ divider symbols: the dividers partition the items into t groups. Thus $|M(t,l,0)| = \binom{l+t-1}{t-1}$ and $|M(t,l,k)| = |M(t,l-kt,0)| = \binom{l-kt+t-1}{t-1} = \binom{l-t(k-1)-1}{t-1}$.

Definition 3.4.4. *For all $q,l,a,b \in \mathbb{N}$, let $s = a+b$. Let $P_{q,l,s}$ be the set*

$$\bigcup_{\lambda\in M(s+1,l,2)} A_{q,\lambda_s} \times \prod_{i=0}^{s-1} (\{\mathrm{X},\mathrm{Y}\} \times ([q] \setminus \{0\}) \times A_{q,\lambda_i})$$

and let $P_{q,l,a,b}$ contain the elements of $P_{q,l,s}$ in which X *appears exactly a times and* Y *appears exactly b times.*

The argument of this section is summarized in the following lemma.

Lemma 3.4.2. *For all $q,l,a,b \in \mathbb{N}$ and $(w_s,t) \in P_{q,l,a,b}$, let (x,y) be the pair of strings produced by* Construct(w_s,t). *Then (x,y) is an edge of $B_{q,l,a,b}$,* Deconstruct$(x,y) = (w_s,t)$, *and $|E(B_{q,l,a,b})| \geq |P_{q,l,a,b}|$.*

3.4.3 Asymptotically matching lower and upper bounds

Lemma 3.4.3. *For fixed $q,a,b \in \mathbb{N}$ and $s = a+b$, $|P_{q,l,a,b}| \gtrsim q^l \binom{l}{s}\binom{s}{a}(q-1)^s$.*

Proof: In $P_{q,l,a,b}$, there are $\binom{s}{a}$ possibilities for the s elements of $\{\mathrm{X},\mathrm{Y}\}$. There are $(q-1)^s$ possibilities for the s elements$([q] \setminus \{0\})^s$. For $\lambda_i \geq 2$, $|A_{q,\lambda_i}| = q^{\lambda_i} - q(q-1)$, so the number of possibilities for the $s+1$

alternating strings is

$$\begin{aligned}
&\sum_{\lambda \in M(s+1,l,2)} \prod_{i=0}^{s} (q^{\lambda_i} - q(q-1)) \\
&\geq \sum_{\lambda \in M(s+1,l,2)} \prod_{i=0}^{s} (q^{\lambda_i} - q^2) \\
&= q^l \sum_{\lambda \in M(s+1,l,2)} \prod_{i=0}^{s} \left(1 - q^{2-\lambda_i}\right) \\
&\overset{(a)}{\geq} q^l \sum_{\lambda \in M(s+1,l,2+\log_q l)} \prod_{i=0}^{s} \left(1 - q^{2-\lambda_i}\right) \\
&\overset{(b)}{\geq} q^l \binom{l - (2+\log_q l)(s+1) + s}{s} (1 - l^{-1})^{s+1} \\
&\sim q^l \binom{l}{s}.
\end{aligned}$$

In (a), we drop the terms of the sum in which for some i, $\lambda_i < 2 + \log_q l$. This allows us to apply the inequality $q^{2-\lambda_i} \leq q^{-\log_q l}$ in (b). We conclude that for a and b fixed and l large, $|P_{q,l,a,b}| \gtrsim q^l \binom{l}{s}\binom{s}{a}(q-1)^s$. ∎

Recall from Lemma 3.3.1 that each $x \in [q]^{n-s}$ has the same number of supersequences of length n:

$$|U_s(x)| = \sum_{i=0}^{s} \binom{n}{i} (q-1)^i. \tag{3.3}$$

We will call this number $I_{q,s,n}$. For fixed q and s, $I_{q,s,n} \sim \binom{n}{s}(q-1)^s$.

Lemma 3.4.4. *For all $q, l, a, b \in \mathbb{N}$ with $s = a + b$, the number of edges in $B_{q,l,a,b}$ satisfies*

$$\begin{aligned}
|E(B_{q,l,a,b})| &\leq q^l I_{q,a,l+a} I_{q,b,l+b} \\
&\sim q^l \binom{l}{a} (q-1)^a \binom{l}{b} (q-1)^b \\
&\sim q^l \binom{l}{s} \binom{s}{b} (q-1)^s.
\end{aligned}$$

Proof: There are $q^l I_{q,a,l+a} I_{q,b,l+b}$ triples $(z, x, y) \in [q]^l \times [q]^{l+a} \times [q]^{l+b}$ such that $z \preceq x$ and $z \preceq y$. If $x \in [q]^{l+a}$ and $y \in [q]^{l+b}$ are adjacent in $B_{q,l,a,b}$, then they have at least one common subsequence of length l and appear in

at least one triple. ■

Our bounds establish the asymptotic growth of the number of edges.

Theorem 3.4.1. *For fixed $q, a, b \in \mathbb{N}$, the number of edges in $B_{q,l,a,b}$ satisfies $|E(B_{q,l,a,b})| \sim q^l \binom{l}{s}\binom{s}{b}(q-1)^s$. The average of $S_{a,b}(x)$ over all $x \in [q]^n$ is asymptotic to $\binom{n}{s}\binom{s}{b}(q-1)^s q^{-a}$.*

Proof: From Lemma 3.4.2, we have $|E(B_{q,l,a,b})| \geq |P_{q,l,a,b}|$. From Lemma 3.4.3 we have the asymptotic lower bound and from Lemma 3.4.4 we have the asymptotic upper bound.

For $x \in [q]^n$, $S_{a,b}(x)$ is the neighborhood of x in $B_{q,n-a,a,b}$. The average degree of the left vertices is asymptotic to $q^l \binom{l}{s}\binom{s}{b}(q-1)^s/q^n = \binom{n-a}{s}\binom{s}{b}(q-1)^s q^{-a}$. ■

3.5 Bounds on input degree and code size

Lemma 3.5.1. *Let $x \in [q]^n$ be a string with r runs. Let $c-1$ be the length of the longest alternating interval of x. Then $|S_{a,b}(x)|$, the number of unique strings that can be produced from x by a deletions and b insertions, is at least*

$$\binom{r-(a+1)c-1}{a}\binom{n-(2a+b+1)c-2}{b}(q-1)^b.$$

Proof: To lower bound $|S_{a,b}(x)|$, we identify a subset $P_x \subseteq P_{q,n-a,a,b}$ such that for all $p \in P_x$, $\textsc{Construct}(p) = (x, y)$. From Lemma 3.4.2, all y produced this way are in $S_{a,b}(x)$ and $|S_{a,b}(x)| \geq |P_x|$.

To produce an element of P_x, we select a symbols of x for deletion, select b spaces in x for insertion, and specify the b new symbols. The symbols selected for deletion and the spaces selected for insertion partition x into $s+1$ intervals. To ensure that none of these intervals are alternating, we will require that all of the intervals contain at least c symbols.

There are many equivalent ways to extend a run by inserting a matching symbol. Construct extends a run by adding a symbol at the right end, so we only select symbols for deletion from those at the right end of a run. It is easier to ignore the symbols that do not appear at the end of a run for the purpose of spacing as well. We need there to be at least c symbols in each of the $a+1$ intervals produced by the deletions, but we enforce the stronger

condition that in each of these intervals there are at least c symbols that appear at the end of their run. There are $M(a+1,r,c) = \binom{r-(a+1)c-1}{a}$ ways to pick the symbols for deletion that satisfy this condition.

There are $n-1$ potential spaces in which an insertion can be made. Insertions cannot be performed in the c spaces before and after a deleted symbol. In the worst case, all of these forbidden spaces are distinct, leaving $n-1-2ac$ spaces to choose from. There must be c symbols between any two consecutive chosen spaces, before the first chosen space, and after the last chosen space. Thus there must be at least $c-1$ spaces in each of these $b+1$ intervals. Again, it is easier to enforce the stronger condition that there are at least $c+1$ spaces not near a deletion in each interval. Thus there are always at least $M(b+1, n-1-2ac, c+1) = \binom{n-1-2ac-(b+1)c-1}{b} = \binom{n-(2a+b+1)c-2}{b}$ ways to pick the spaces.

Finally, for each of the b insertion points, we must specify the difference between the inserted symbol and its successor. Thus, there are $(q-1)^b$ choices for this step. ■

The following argument, very similar to Lemma 3.4.4, shows that this degree lower bound is asymptotically tight. This is a generalization of a lemma of Levenshtein [14].

Lemma 3.5.2. *For all $q, n, r, a, b \in \mathbb{N}$ with $s = a+b$, if $x \in [q]^n$ has r runs, then*

$$|S_{a,b}(x)| \leq \binom{r+a-1}{a} I_{q,b,n-a+b}.$$

Proof: Any subsequence of x can be specified by the number of symbols deleted from each run. This is a composition of a with r parts, so $|S_{a,0}(x)| \leq |M(a,r,0)| = \binom{r-1+a}{r-1} = \binom{r+a-1}{a}$. Each string in $S_{a,b}(x)$ is a supersequence of one of these subsequences. Each subsequence has exactly $I_{q,b,n-a+b}$ supersequences of length $n-a+b$. ■

If $r = pn$ for fixed p, the bounds of Lemma 3.5.1 and Lemma 3.5.2 are both asymptotic to

$$\binom{r}{a}\binom{n}{b}(q-1)^b.$$

To apply Lemma 3.5.1 to a string, we need two statistics of that string: the number of runs and the length of the longest alternating interval. The next two lemmas concern the distributions of these statistics.

Lemma 3.5.3. *The number of q-ary strings of length n with an alternating interval of length at least c is at most $(n-c+1)q^{n-c+1}(q-1)$.*

Proof: If some interval of length at least c is alternating, at least one of its subintervals of length exactly c is alternating. A string of length n contains $n-c+1$ intervals of length c, so each string of interest fall into at least one of $n-c+1$ classes. In each class, there are $q(q-1)$ choices for the symbols in the alternating interval and q^{n-c} choices for the remaining symbols. ■

Lemma 3.5.4. *The number of q-ary strings of length n with $\left(\frac{q-1}{q}-\epsilon\right)(n-1)+1$ or fewer runs is at most $q^n e^{-2(n-1)\epsilon^2}$.*

Proof: For $x \in [q]^n$, let $x' \in [q]^{n-1}$ be the string of first differences of x. That is, let $x'_i = x_{i+1} - x_i \bmod q$. If x has r runs, then x'_i is nonzero at the $r-1$ boundaries between runs. Thus there are $q\binom{n-1}{r-1}(q-1)^{r-1}$ strings with exactly r runs. The number of strings with few runs is

$$\begin{aligned} & q \sum_{i=0}^{\left(\frac{q-1}{q}-\epsilon\right)(n-1)} \binom{n-1}{i}(q-1)^i \\ &= q^n \sum_{i=0}^{\left(\frac{q-1}{q}-\epsilon\right)(n-1)} \binom{n-1}{i}\left(\frac{q-1}{q}\right)^i \left(\frac{1}{q}\right)^{n-1-i} \\ &\leq q^n e^{-2(n-1)\epsilon^2}. \end{aligned}$$

The upper bound comes from the application of Hoeffding's inequality to the binomial distribution [26]. ■

Now we can show that there are few inputs with degree significantly below the average.

Lemma 3.5.5. *Let $q, a, b \in \mathbb{N}$ be fixed and let $s = a+b$. For all $t \in \mathbb{N}$, there is a sequence of subsets $T_n \subseteq [q]^n$ such that $|T_n|$ is $O(q^n/n^t)$ and*

$$\min_{x \in [q]^n \setminus T_n} |S_{a,b}(x)| \gtrsim \frac{(q-1)^s}{q^a}\binom{n}{s}\binom{s}{b}.$$

Proof: We form two classes of bad strings: strings with a long alternating interval and strings with few runs. Call these classes T'_n and T''_n respectively. Let $T_n = T'_n \cup T''_n$.

A string falls into T'_n if it has an alternating subinterval of length at least c. If we let $c = (t+1)\log_q n$, then by Lemma 3.5.3 we have

$$|T'_n| < nq^{n-c+1}(q-1) = n^{-t}q^{n+1}(q-1),$$

which is $O(q^n/n^t)$.

Over all strings in $[q]^n$, the average number of runs is $\frac{q-1}{q}(n-1)+1$. A string falls into T''_n if it has at most $\left(\frac{q-1}{q} - \epsilon\right)(n-1)+1$ runs. If we let $\epsilon = \sqrt{\frac{t\log n}{2(n-1)}}$, then by Lemma 3.5.4 we have

$$|T''_n| \leq q^n e^{-2(n-1)\epsilon^2} = q^n e^{-t\log n} = q^n/n^t.$$

For fixed t, this ϵ is $o(1)$, so $\left(\frac{q-1}{q} - \epsilon\right)(n-1)+1 \sim \frac{(q-1)n}{q}$.

Now we can apply Lemma 3.5.1 to lower bound the degree of the strings in $[q]^n \setminus T_n$. The first multiplicative term in the lower bound is asymptotic to

$$\begin{aligned}\binom{\frac{q-1}{q}n - (a+1)(t+1)\log_q n - 1}{a} &\sim \binom{\frac{q-1}{q}n}{a} \\ &\sim \left(\frac{q-1}{q}\right)^a \binom{n}{a}.\end{aligned}$$

The second term is asymptotic to

$$\binom{n-(2a+b+1)(t+1)\log_q n - 2}{b} \sim \binom{n}{b}.$$

Thus

$$\begin{aligned}\min_{x\in[q]^n\setminus T_n} |S_{a,b}(x)| &\gtrsim \left(\frac{q-1}{q}\right)^a \binom{n}{a}\binom{n}{b}(q-1)^b \\ &\sim \frac{(q-1)^s}{q^a}\binom{n}{s}\binom{s}{b}.\end{aligned}$$

■

Our main theorem follows easily.

Theorem 3.5.1. *For fixed $q, s \in \mathbb{N}$, the number of codewords in an n-symbol*

q-ary s-deletion correcting code satisfies

$$C_{q,s,n} \lesssim \min_{0\le b\le s} \frac{q^{n+b}}{(q-1)^s \binom{n}{s}\binom{s}{b}}.$$

Proof: Consider an a-deletion b-insertion channel with $a+b=s$. By Lemma 3.2.1 and Theorem 2.4.1, any code for this channel can also correct s deletions. There are q^{n-a+b} possible outputs, so for any $T_n \subseteq [q]^n$,

$$C_{q,s,n} \lesssim \frac{q^{n-a+b}}{\min_{x\in[q]^n\setminus T_n} |S_{a,b}(x)|} + |T_n|.$$

For any choice of t, from Lemma 3.5.5 we obtain a sequence T_n such that $\min_{x\in[q]^n\setminus T_n} |S_{a,b}(x)|$ is $\Omega(n^s)$ and $|T_n|$ is $O(q^n/n^t)$. To make the contribution of the second term negligible, we choose $t=s+1$ and obtain an asymptotic upper bound of

$$C_{q,s,n} \lesssim \frac{q^{n-a+b}}{\frac{(q-1)^s}{q^a}\binom{n}{s}\binom{s}{b}} + O\left(\frac{q^n}{n^{s+1}}\right) \sim \frac{q^{n+b}}{(q-1)^s\binom{n}{s}\binom{s}{b}}.$$

■

This improves (3.2), Levenshtein's upper bound, by a factor of $\binom{s}{b}q^{-b}$. By setting b to zero we recover Levenshtein's bound. Whenever $s>q$, $\binom{s}{1}q^{-1} > \binom{s}{0}q^0 = 1$ so setting b to one in the generalized bound offers an improvement.

Corollary 3.5.1. *If $q+1$ divides s, the size of a q-ary s-deletion correcting code satisfies*

$$C_{q,s,n} \lesssim \frac{3s^{\frac{1}{2}}q^{n+s+\frac{1}{2}}}{(q+1)^{s+1}(q-1)^s\binom{n}{s}}.$$

Proof: We optimize over b in Theorem 3.5.1. The factor $\binom{s}{b}q^{-b}$ is a constant times a binomial distribution:

$$\left(\frac{q+1}{q}\right)^s \binom{s}{b}\left(\frac{1}{q+1}\right)^b\left(\frac{q}{q+1}\right)^{s-b}.$$

The maximum is achieved by $b=\left\lfloor\frac{s+1}{q+1}\right\rfloor$. When $q+1$ divides s, the maximum is at least

$$\left(\frac{q+1}{q}\right)^s \frac{1}{3}\sqrt{\frac{q+1}{qs/(q+1)}} = \frac{(q+1)^{s+1}}{3s^{\frac{1}{2}}q^{s+\frac{1}{2}}}$$

by Lemma 3.5.6. ■

Lemma 3.5.6. *For $a, b, n \in \mathbb{N}$,*

$$\binom{(\alpha+\beta)n}{\alpha n}\left(\frac{\alpha}{\alpha+\beta}\right)^{\alpha n}\left(\frac{\beta}{\alpha+\beta}\right)^{\beta n} \geq \frac{1}{3}\sqrt{\frac{\alpha+\beta}{\alpha\beta n}}.$$

Proof: One form of Stirling's approximation is [27]

$$1 \leq \frac{n!}{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n} \leq e^{\frac{1}{12}}.$$

Then for $\alpha, \beta, n \in \mathbb{N}$, consider the binomial distribution produced by $(\alpha+\beta)n$ trials and success probability $\alpha/(\alpha+\beta)$. The most likely outcome is αn successes and the probability of that outcome is:

$$\begin{aligned}
&\max_i \binom{(\alpha+\beta)n}{i}\left(\frac{\alpha}{\alpha+\beta}\right)^{i}\left(\frac{\beta}{\alpha+\beta}\right)^{(\alpha+\beta)n-i} \\
&= \binom{(\alpha+\beta)n}{\alpha n}\left(\frac{\alpha}{\alpha+\beta}\right)^{\alpha n}\left(\frac{\beta}{\alpha+\beta}\right)^{\beta n} \\
&\geq \frac{\sqrt{2\pi(\alpha+\beta)n}\left(\frac{(\alpha+\beta)n}{e}\right)^{(\alpha+\beta)n}}{e^{\frac{1}{12}}\sqrt{2\pi\alpha n}\left(\frac{\alpha n}{e}\right)^{\alpha n} e^{\frac{1}{12}}\sqrt{2\pi\beta n}\left(\frac{\beta n}{e}\right)^{\beta n}} \frac{\alpha^{\alpha n}\beta^{\beta n}}{(\alpha+\beta)^{(\alpha+\beta)n}} \\
&= \frac{1}{e^{\frac{1}{6}}\sqrt{2\pi}}\sqrt{\frac{\alpha+\beta}{\alpha\beta n}} \\
&\geq \frac{1}{3}\sqrt{\frac{\alpha+\beta}{\alpha\beta n}}.
\end{aligned}$$

■

The degree lower bound in Lemma 3.5.5 cannot be raised any further without excluding an asymptotically non-negligible number of inputs.

Lemma 3.5.7. *For fixed $q, a, b \in \mathbb{N}$ and fixed $\epsilon > 0$, suppose that there is a sequence of subsets $T_n \subseteq [q]^n$ such that*

$$\min_{x \in [q]^n \setminus T_n} |S_{a,b}(x)| \gtrsim (1+\epsilon)\frac{(q-1)^s}{q^a}\binom{n}{s}\binom{s}{b}.$$

Then $|T_n|$ is $\Omega(q^n)$.

Proof: This is essentially an application of Markov's inequality. By

the definition of T_n,

$$\sum_{x\in[q]^n} |S_{a,b}(x)| \geq \sum_{x\in[q]^n\setminus T_n} |S_{a,b}(x)|$$
$$\gtrsim (q^n - |T_n|)(1+\epsilon)\frac{(q-1)^s}{q^a}\binom{n}{s}\binom{s}{b}.$$

From Lemma 3.4.4,

$$\sum_{x\in[q]^n} |S_{a,b}(x)| = |E(B_{q,n-a,a,b})| \lesssim q^{n-a}\binom{n}{s}\binom{s}{b}(q-1)^s.$$

Chaining these inequalities together and dividing both sides of the result by $q^{n-a}\binom{n}{s}\binom{s}{b}(q-1)^s$ yields $\left(1-\frac{|T_n|}{q^n}\right)(1+\epsilon) \lesssim 1$. Thus $|T_n| \gtrsim \frac{\epsilon}{1+\epsilon}q^n$ and $|T_n|$ is $\Omega(q^n)$. ∎

If the number of excluded channel inputs is $\Omega(q^n)$, the excluded inputs are the dominant contribution to the upper bound on code size. Thus our bounds on code size are the best that can be obtained via the technique of excluding atypical inputs.

3.6 Concluding discussion

In this chapter, we extended Levenshtein's strategy for obtaining an upper bound on the size of deletion codes. Levenshtein's bound arises from the deletion channel. We derived the corresponding bounds from channels that perform a mixture of deletions and insertions. This results in an improvement whenever the number of errors, s, is larger than the alphabet size, q. The best version of our bound uses a channel where the ratio of deletions to insertions is q to one.

Our argument relies on the fact that the channel graphs are approximately regular in the asymptotic regime where the number of errors is fixed. A natural question is whether this argument can be extended to the regime where the number of errors is a constant fraction of the input length. However, it is not clear whether the graphs are approximately regular in the latter regime. The argument of this chapter relies on the typical spacing between errors going to infinity. Because this spacing becomes large, any interaction between two errors becomes rare. When the typical spacing does not grow

with input length, interactions will not be rare and it will not be possible to simply discard the cases where they occur. Instead it will be necessary to understand the details of these interactions.

3.7 Algorithms

Our construction function, CONSTRUCT, is specified in Algorithm 3 and our deconstruction function, DECONSTRUCT, is specified in Algorithm 4. The function LENGTH returns the number of symbols in the string.

Algorithm 3 Construct an edge

$\textsc{Construct} : A_{q,*} \times (\{\mathrm{X}, \mathrm{Y}\} \times ([q] \setminus \{0\}) \times A_{q,*})^* \to [q]^* \times [q]^*$
$\textsc{Construct}(w, t) = (w, w) +\!\!+ \textsc{Insert}(t)$

$\textsc{Insert} : (\{\mathrm{X}, \mathrm{Y}\} \times ([q] \setminus \{0\}) \times A_{q,*})^* \to [q]^* \times [q]^*$
$\textsc{Insert}(t) =$ **case** t **of**
$\quad \langle\rangle : (\langle\rangle, \langle\rangle)$
$\quad \langle(z, \delta, w)\rangle + t' :$ **case** z **of**
$\quad\quad \mathrm{X} : (\langle\delta + w_0\rangle, \langle\rangle) +\!\!+ (w, w) +\!\!+ \textsc{Insert}(t')$
$\quad\quad \mathrm{Y} : (\langle\rangle, \langle\delta + w_0\rangle) +\!\!+ (w, w) +\!\!+ \textsc{Insert}(t')$

Algorithm 4 Deconstruct an edge

$\textsc{Deconstruct} : [q]^* \times [q]^* \to A_{q,*} \times (\{\mathrm{X}, \mathrm{Y}\} \times ([q] \setminus \{0\}) \times A_{q,*})^*$
$\textsc{Deconstruct}(x, y) = (w, \textsc{Delete}(x, y))$
  where $(w, (x, y)) = \textsc{Match}(x, y)$

$\textsc{Delete} : [q]^* \times [q]^* \to (\{\mathrm{X}, \mathrm{Y}\} \times ([q] \setminus \{0\}) \times A_{q,*})^*$
$\textsc{Delete}(x, y) =$ **case** (x, y) **of**
  $(\langle\rangle, \langle\rangle) : \langle\rangle$
  $(\langle x_0\rangle + x', \langle y_0\rangle + y') :$ **case** $(\textsc{MLength}(x', y), \textsc{MLength}(x, y'))$ **of**
    $> : \langle(\mathrm{X}, x_0 - y_0, a)\rangle + \textsc{Delete}(b, c)$ **where** $(a, b, c) = \textsc{Match}(x', y)$
    $< : \langle(\mathrm{Y}, y_0 - x_0, a)\rangle + \textsc{Delete}(b, c)$ **where** $(a, b, c) = \textsc{Match}(x, y')$

$\textsc{MLength} : [q]^* \times [q]^* \to \mathbb{N}$
$\textsc{MLength}(x, y) =$
  if $(x = x_0 + x') \wedge (y = y_0 + y') \wedge (x_0 = y_0)$
  then $1 + \textsc{MLength}(x', y')$
  else 0

$\textsc{Match} : [q]^* \times [q]^* \to [q]^* \times [q]^* \times [q]^*$
$\textsc{Match}(x, y) =$
  if $(x = x_0 + x') \wedge (y = y_0 + y') \wedge (x_0 = y_0)$
  then $(\langle x_0\rangle + w, x'', y'')$ **where** $(w, x'', y'') = \textsc{Match}(x', y')$
  else $(\langle\rangle, x, y)$

3.8 The poset of compositions

The natural numbers under the usual order form a ranked totally ordered set, so $\mathbb{N}^d$ is a ranked poset. An composition of width d and size n is a rank-n element of $\mathbb{N}^d$, a vector of d nonnegative integers that sums to n. Let $C(d, n)$ be the set of such vectors. Elementary counting shows that $|C(d, n)| = \binom{n+d-1}{n}$.

In this section we will discuss parallels between composition poset and

the poset of q-ary strings ordered by the subsequence relation. In order to illustrate these ideas, we will compute the asymptotic values of the sphere-packing bounds for the channels that take a composition for various mixtures of up and down errors.

First, the composition poset is very closely connected to the problem of correcting restricted set of insertion and deletion errors: repetition and derepetition error. A repetition error transforms a q-ary string of length n to a string of length $n+1$ by replacing a single symbol with two adjacent copies of that symbol. A derepetition does the reverse, contracting two adjacent copies of a symbol into a single copy. Repetition and a derepetition errors cannot change the number of runs in a string or which symbol appears in the ith run. Thus the space of messages is the union of $\sum_{r=1}^{n} q(q-1)^r$ disconnected components. Within each component, each message can be represented by the vector of run lengths. Each run length is at least one, so a string of length n with r runs can be encoded as an element of $C(r, n-r)$. A repetition increases a single run length by one. Dolecek and Anantharam have exploited this correspondence and constructed repetition error correcting codes [28].

There are further parallels between the posets. In the subsequence poset, the number of supersequences of length $n+b$ is constant across all of the strings in $[q]^n$. Similarly, the number of supercompositions of size $n+b$ is constant across all compositions in $C(d,n)$. In both cases, this property does not extend to down errors. A string of length n with r runs has r subsequences of length $n-1$. The number of subsequences of length $n-a$ depends strongly on r for all a. A composition in $C(d,n)$ with f nonzero entries has f subcompositions of size $n-1$. The number of subcompositions of size $n-a$ depends strongly on f.

For a fixed alphabet size q, the number of length-n strings grows as q^n. There are $\binom{n+d-1}{n}$ compositions in $C(d,n)$. In order to have exponential growth as we increase n, we should fix the ratio $\rho = d/n$. A typical q-ary string has $\frac{q-1}{q}n$ runs. The vector encoding the run lengths of a typical string is in $C\left(\frac{q-1}{q}n, \frac{1}{q}n\right)$. Thus it makes sense to compare the poset of q-ary strings to a sequence of posets of compositions with $\rho = q-1$.

When we make this precise comparison, a few more similarities emerge. In the poset of q-ary strings, the rank $n+1$ contains a factor of q more elements than rank n. For compositions, the ratio between the size of consecutive

ranks

$$\frac{|C(d,n+1)|}{|C(d,n)|} = \frac{\binom{n+d}{n+1}}{\binom{n+d-1}{n}} = \frac{n+d}{n+1} \sim 1+\rho = q.$$

Let $x \in [q]^n$ have $\frac{q-1}{q}n$ runs, which is the typical value. Then $|U_1(x)| = (n+1)(q-1)+1$ and $|D_1(x)| = \frac{q-1}{q}n$. The ratio of these is asymptotically q. Let $x \in C(d,n)$ have $\frac{d-1}{d+n-1}$ nonzero entries, which again is the typical value. Then $|U_1(x)| = d$ and $|D_1(x)| = \frac{d-1}{d+n-1}$. The ratio of these is asymptotically $1+\rho$.

3.8.1 Sphere-packing bounds

Let $A_{d,n,a,b}$ be the channel that takes an element of $C(d,n)$ and performs a down errors and b up errors. To obtain an upper bound on $p^*(A_{d,n,a,b})$, we will apply the local degree upper bound. See Chapter 4 for the full details of this method. This gives

$$p^*(A_{d,n,a,b}) \leq \sum_{y \in C(d,n-a+b)} \frac{1}{\max_{x \in U_a(D_b(y))} |U_b(D_a(x))|}.$$

To evaluate this, we need an lower bound on $|U_b(D_a(x))|$ for each $x \in C(d,n)$. If x has f nonzero entries, then

$$|U_b(D_a(x))| \geq \binom{f}{a}|C(d-a,b)| = \binom{f}{a}\binom{d-a+b-1}{b}.$$

There are f entries where we can perform one of the a down errors. We can distribute the b up error among the other $d-a$ entries. In this way, each element of $U_b(D_a(x)$ is counted at most once.

If y has f nonzero entries, then any $x \in U_a(D_b(y))$ has at least $f-b$ nonzero entries. There are $\binom{d}{f}\binom{n-1}{n-f}$ such vectors with exactly f nonzero entries. Thus f follows a hypergeometric distribution.

Putting this together, we obtain

$$p^*(A_{d,n,a,b}) \leq \sum_f \frac{\binom{d}{f}\binom{n-a+b-1}{n-a+b-f}}{\binom{f-b}{a}\binom{d-a+b-1}{b}}.$$

The average number of nonzero entries is $\frac{dn}{n+d-1} \sim \frac{\rho}{1+\rho}n$. Because the hypergeometric distribution concentrates, it is not too hard to show that for fixed

a and b this sum is asymptotically at most

$$|C(d, n-a+b)| \left(\binom{\frac{\rho}{1+\rho}n}{a}^{-1} \binom{\rho n}{b}^{-1} + O(n^{-a-b-1}) \right).$$

We can make a few further simplifications:

$$\binom{\frac{\rho}{1+\rho}n}{a}\binom{\rho n}{b} \sim (1+\rho)^{-a}\binom{\rho n}{a+b}\binom{a+b}{a}.$$

The size of the output space, $|C(d, n-a+b)|$, is equal to $\binom{n-a+b+d-1}{n-a+b} \sim \binom{n+d-1}{n}(1+\rho)^{-a+b}$. Putting all of this together, we get

$$p^*(A_{d,n,a,b}) \leq \frac{\binom{n+d-1}{n}(1+\rho)^b}{\binom{\rho n}{a+b}\binom{a+b}{a}}.$$

Now we fix the total number of errors $s = a+b$ and optimize across channels. The quantity $\left(\frac{1}{1+\rho}\right)^{s-a}\binom{s}{a}$ follows a rescaled binomial distribution and consequently is maximized by $\frac{a}{s} \approx \frac{1}{1+\frac{1}{1+\rho}} = \frac{1+\rho}{2+\rho}$. The best sphere-packing bound for the deletion insertion channels came from the choice $\frac{a}{s} \approx \frac{q}{q+1}$. Thus we have another example of these two posets exhibiting similar behavior when $\rho = q-1$. It is not hard to show that the largest cliques in the confusability graph over compositions are associated with the channel where $\frac{a}{s} = \frac{\rho}{1+\rho}$, again paralleling the subsequence partial order.

The sizes of the sets $U_b(D_a(x))$ play a central role in the value of the sphere-packing bounds. In the subsequence poset, when a and b grow with n, for most strings we do not have good estimates of the sizes of these sets. Unlike the subsequence partial order, the partial order on compositions in a lattice, which makes it much easier to compute the necessary quantities. We believe that analysis of codes in the composition poset capable of correcting many errors will provide a good first step toward the corresponding analysis of deletion correcting codes.

3.8.2 Applications

Codes on integer compositions have applications in data storage. One example is DNA-based storage. Kiah et al. propose a form of DNA storage in which information is encoded in the relative frequencies of short sequences as

intervals of a long sequence [29]. This allows data recovery using sequencing methods that only make short reads. The computationally expensive step of assembling short reads to recover the whole long sequence is avoided.

Kiah et al. [29] constructed codes on integer compositions by viewing compositions as constant-sum q-ary vectors and modifying existing code constructions for correcting asymmetric errors in that setting. In addition, they add a number of other practical constraints to their codes. Unfortunately, their method of translating results from the q-ary setting to the integer composition setting depends on particular properties of the code construction, so we cannot immediately convert known upper bounds on q-ary asymmetric error correcting codes to bounds on integer composition codes for the purpose of comparison.

CHAPTER 4

SPHERE PACKING AND SPHERE COVERING BOUNDS

4.1 Introduction

The classic problem of coding theory, correcting substitution errors in a vector of q-ary symbols, is highly symmetric. First, if s errors are required to change a vector x into another vector y, then s errors are also required to change y into x. Second, the number of vectors that can be produced from x by making up to s substitutions, the size of the sphere around x, does not depend on x. The sizes of these spheres play a crucial role in both upper and lower bounds on the size of the largest s-substitution-error-correcting codes. The Hamming bound is a sphere-packing upper bound and the Gilbert-Varshamov lower bound is a sphere-covering lower bound. The two symmetries that we have described make the proofs of the Hamming and Gilbert-Varshamov bounds extremely simple.

Many other interesting error models do not have this degree of symmetry. Substitution errors with a restricted set of allowed substitutions are sometimes of interest. The simplest example is the binary asymmetric errors, which can replace a one with a zero but cannot replace a zero with a one. Binary asymmetric errors have neither of the two symmetries we have described. Erasure and deletion errors differ from substitution errors in a more fundamental way: the error operation takes an input from one set and produces an output from another.

In this chapter, we will discuss the generalizations of sphere-packing bounds to arbitrary error models. These generalizations become especially important when the sizes of the error spheres are nonuniform. Sphere-packing bounds are fundamentally related to linear programming and the best possible versions of the bounds are solutions to linear programs. In highly symmetric cases, including many classical error models, it is often possible to

get the best possible sphere-packing bound without directly considering any linear programs. For less symmetric channels, the linear programming perspective becomes essential.

In fact, recently a new bound, explicitly derived via linear programming, was applied by Kulkarni and Kiyavash to find an upper bound on the size of deletion-correcting codes [4]. It was subsequently applied to grain errors [30, 31] and multipermutation errors [32]. We will refer to this as the local degree bound. The local degree bound constructs a dual feasible point for the sphere-packing linear program because computation of the exact solution is intractable. Deletion errors, like most interesting error models, act on an exponentially large input space. Because computation of the best possible packing and covering bounds is often intractable, simplified bounds such as the local degree bound are useful.

Sphere-packing and sphere-covering arguments have been applied in an ad hoc fashion throughout the coding theory literature. We attempt to present a unifying framework that permits such arguments in their most general form applicable to both uniform and nonuniform error sphere sizes. More precisely, we derive a series of bounds from approximations to packing and covering problems. The local degree bound of [4] is one of the bounds in the series. We associate each bound with an iterative procedure such that the original bound is the result of a single step. This characterization makes it easy to study the relationships between the bounds. We apply our generalization of the local degree bound to improve the best known upper bounds on the sizes of single deletion correcting codes and single grain error correcting codes.

We use the concept of a combinatorial channel to represent an error model in a fashion that makes the connection to linear programming natural. These bounds use varying levels of information about structure of the error model and consequently make trade-offs between performance and complexity. For example, one bound uses the distribution of the sizes of spheres in the space while another uses only the size of the smallest sphere.

Our contributions can be summarized as follows. We provide a unified framework for describing upper bounds on code size. This allows us to make very general statements about the relative strengths of the bounds. In particular, our generalization of the local degree bound allows us to improve the best known upper bounds for a few channels.

In Section 2.1, we discussed the linear programs associated with sphere-

packing bounds. In Section 4.2, we present a generalization of the local degree bound that is related to an iterative procedure. We use this to improve the best known upper bounds on the sizes of single deletion correcting code and single grain error correcting codes. In Section 4.3, we discuss sphere-packing bounds related to the degree sequence and average degree of a channel. In Section 2.2.2, we discuss families of channels that have the same codes but give different sphere-packing bounds.

4.2 The local degree iterative algorithm

Let $A \in \{0,1\}^{X \times Y}$ be a channel. We can obtain an upper bound for $p^*(A)$ (and consequently $p(A)$) by finding a feasible point in the program for $\kappa^*(A)$. Given some $t \in \mathbb{R}^Y$ such that $t \geq \mathbf{0}$ and $(At)_x > 0$ for all x, let $z \in \mathbb{R}^Y$ be the smallest scaling of t that is feasible for $\kappa^*(A)$:

$$z_y = \frac{t_y}{\min_{x \in X} (At)_x}. \tag{4.1}$$

We have the upper bound $p^*(A) \leq \mathbf{1}^T z$, which we call $\kappa^*_{\text{MD}}(A, t)$. The special case

$$\kappa^*_{\text{MD}}(A, \mathbf{1}) = \frac{|Y|}{\min_{x \in X} |N_A(x)|},$$

is the minimum degree upper bound, which explains the subscript. Throughout, bounds notated by κ^* with a subscript come from the construction of a particular dual feasible point (i.e. feasible in the program for k^*) and bounds notated by p^* with a subscript come from the value of a relaxation of the primal program.

4.2.1 The local degree bound

For channels that are both input and output regular, computation of the sphere-packing bound p^* is trivial: the minimum degree bound is exact. However, even a single low degree input will ruin the effectiveness of the minimum degree bound. To obtain a better upper bound on $p(A)$ and $p^*(A)$, we will construct a different feasible point in the program for $\kappa^*(A)$ by making a small change to (4.1).

Definition 4.2.1. *Let $A \in \{0,1\}^{X \times Y}$ be a channel. For $t \in \mathbb{R}^Y$ such that $(At)_x > 0$ for all $x \in X$, define $\varphi_A(t) \in \mathbb{R}^Y$ as follows:*

$$\varphi_A(t)_y \triangleq \frac{t_y}{\min_{x \in N(y)} (At)_x}.$$

Define the local degree upper bound $\kappa^*_{\mathrm{LD}}(A, z) = \mathbf{1}^T \varphi_A(z)$.

Lemma 4.2.1. *Let $t \in \mathbb{R}^Y$ such that $t \geq \mathbf{0}$ and $(At)_x > 0$ for all $x \in X$. Then $\varphi_A(t)$ is feasible in the program for $\kappa^*(A)$. If t is feasible for $\kappa^*(A)$, then $\varphi_A(t) \leq t$.*

Proof: To demonstrate feasibility of $z = \varphi_A(t)$, we need $z \geq \mathbf{0}$ and $Az \geq \mathbf{1}$. The first condition is trivially met. For $x \in X$ and $y \in N(x)$, we have

$$\begin{aligned} z_y &= \frac{t_y}{\min_{x' \in N(y)} (At)_{x'}} \geq \frac{t_y}{(At)_x} \\ (Az)_x &= \sum_{y \in N(x)} z_y \geq \frac{1}{(At)_x} \sum_{y \in N(x)} t_y = 1 \end{aligned}$$

and z is feasible.

If t is feasible, then $At \geq \mathbf{1}$. For all $y \in Y$ we have

$$z_y = \frac{t_y}{\min_{x \in N(y)} (At)_x} \leq t_y.$$

■

We can view the application of φ_A as a single iteration of an algorithm with the following intuitive description. Suppose that we have a vector $t \in \mathbb{R}^Y$ that is a feasible vector in the program for $\kappa^*(A)$. For any channel, we can take $t = \mathbf{1}$ as an initial vector. At each input x, the total coverage, $(At)_x$, is at least one. The input x informs each output in $N(x)$ that it can reduce its value by a factor of $(At)_x$. Each output y receives such a message for each input in $N(y)$, then makes the largest reduction consistent with the messages.

An iteration fails to make progress under the following condition. From the definition $\varphi_A(t)_y = t_y$ if and only if $\min_{x \in N(y)} (At)_x = 1$. Thus $\varphi_A(t) = t$ if for all $y \in Y$ there is some $x \in N(y)$ such that $(At)_x = 1$. This algorithm is monotonic in each entry of the feasible vector, so it cannot make progress

if its input is at the frontier of the feasible space.

Scaling the input by a positive constant does not affect the output of φ_A: for $c \in \mathbb{R}$, $c > 0$, $\varphi_A(t) = \varphi_A(ct)$. We could think of $\kappa^*_{\text{MD}}(A, t)$ as involving an iterative procedure as well. It has the same scaling property. In contrast to the local degree iteration, the minimum degree iteration always stops after a single step because the output vector is a constant multiple of the input. The local degree iteration scales different entries in the initial vector by different amounts, so it is possible for it to make progress for multiple iterations.

4.2.2 Application to the single deletion channel

Now we will apply two iterations of the local degree iteration to obtain a new upper bound for the single deletion channel. Because some of the calculations are long, we will state the results in this section and give the proofs in Section 4.2.4.

Let A_n be the n-bit 1-deletion channel. The input to the binary single deletion channel is a string $x \in [2]^n$ and the output is a subsequence of x, $y \in [2]^{n-1}$. Each output vertex in A_n has degree $n + 1$. Thus $\kappa^*(A_n) \geq \kappa^*_{\text{MD}}(A_n) = \frac{2^n}{n+1}$.

Levenshtein [14] showed that

$$\kappa^*(A_n) \leq \frac{2^n}{n+1}(1 + o(1)).$$

Kulkarni and Kiyavash computed the local degree upper bound, or equivalently $\varphi_{A_n}(\mathbf{1})$ [4]. This shows that $\kappa^*(A_n)$ is at most

$$\frac{2^n}{n-1} = \frac{2^n}{n+1}\left(1 + \frac{2}{n-1}\right) = \frac{2^n}{n+1}(1 + O(n^{-1})).$$

Recently, Fazeli et al. found a fractional covering for A_n that provides a better upper bound [33].

For the remainder of this section we abbreviate φ_{A_n} by φ. In this section, we compute $\varphi \circ \varphi(\mathbf{1})$ for these channels and analyze the values of these points. We show that Fazeli's improved covering is related to the covering $\varphi \circ \varphi(\mathbf{1})$, but $\varphi \circ \varphi(\mathbf{1})$ provides a better bound asymptotically.

More precisely, the upper bound from $\varphi \circ \varphi(\mathbf{1})$, given in Theorem 4.2.2,

shows that $\kappa^*(A)$ is at most

$$\frac{2^n}{n-1}\left(1-\frac{2}{n-1}+O(n^{-2})\right)=\frac{2^n}{n+1}(1+O(n^{-2})).$$

The covering in Fazeli et al. gives an upper bound of

$$\frac{2^n}{n+1}\left(1+\frac{1}{n-1}+O(n^{-2})\right).$$

A run in a string is a maximal set of consecutive indices that have the same symbol. Let $r,u,b\in\mathbb{N}^{[2]^*}$ be vectors such that for all $x\in[2]^*$, r_x is the number of runs in x, u_x is the number of length-one runs, or unit runs, in x, and b_x is the number of unit runs at the start or end of x.

Theorem 4.2.1. *Let*

$$f(r,u,b)\triangleq\frac{1}{r}\left(1+\frac{\max(2u-b-2,0)}{(r+2)(r+1)}\right)^{-1}.$$

Then the vector $z_y=f(r_y,u_y,b_y)$ is feasible for $\kappa^*(A_n)$, so $\kappa^*(A_n)\leq\mathbf{1}^Tz$.

Theorem 4.2.2. *For $n\geq 2$,*

$$\kappa^*(A_n)\leq\frac{2^n}{n+1}\left(1+\frac{26}{n(n-1)}\right).$$

Now we will compare this bound to the bound corresponding to the cover of Fazeli et al. Let

$$f'(r,u,b)\triangleq\begin{cases}\frac{1}{r}\left(1-\frac{u-b}{r^2}\right) & u-b\geq 2\\ \frac{1}{r} & u-b\leq 1.\end{cases}$$

Fazeli et al. established that $z_y=f'(r_y,u_y,b_y)$ is feasible for $\kappa^*(A_n)$ [33]. Compare this with the cover given by f' and note that the coefficient on u is 1 in f' and 2 in f.

Lemma 4.2.2. *Let $z_y=f'(r_y,u_y,b_y)$. Then*

$$\mathbf{1}^Tz\geq\frac{2^n-2}{n+1}\left(1+\frac{1}{n-1}-\frac{3}{(n-1)(n-2)}\right).$$

This shows that the bound of Theorem 4.2.2 is asymptotically better than

the bound corresponding to the cover of Fazeli et al. We could continue to iterate φ to produce even better bounds. The fractional covers produced would depend on more statistics of the strings. For example, the value at a particular output of the cover produced by the third iteration of φ would depend on the number of runs of length two in that output string, in addition to the total number of runs and the number of runs of length one.

The largest known single deletion correcting codes are the Varshamov-Tenengolts (VT) codes [14]. The largest length-n VT code, denoted VT_0, contains at least $\frac{2^n}{n+1}$ codewords, so this sequence of codes is asymptotically optimal. VT_0 is known to be a maximum independent set for $n \leq 10$, but this question is open for larger n [34]. Kulkarni and Kiyavash [4] computed the exact value of $\kappa^*(A_n)$ for $n \leq 14$. For $7 \leq n \leq 14$, the gap between $\kappa^*(A_n)$ and the size of the VT codes was at least one, so it is unlikely that sphere-packing bounds will resolve the optimality of the VT codes for larger n. Despite this, it would be interesting to know the asymptotics of the gap between $\kappa(A_n)$ and $\kappa^*(A_n)$. For example, is it true that $\kappa^*(A_n) \leq \frac{2^n}{n+1} + O(2^{cn})$ for some constant $c < 1$?

4.2.3 Application to the single grain error channel

Recently, there has been a great deal of interest in grain error channels, which are related to high-density encoding on magnetic media. A grain in a magnetic medium has a single polarization. If an encoder attempts to write two symbols to a single grain, only one of them will be retained. Because the locations grain boundaries are generally unknown to the encoder, this situation can be modeled by a channel.

Mazumdar et al. applied the degree sequence bound to non-overlapping grain error channels [35]. Sharov and Roth applied the degree sequence bound to both non-overlapping and overlapping grain error channels [36]. We discuss the degree sequence bound and its relationship to the local degree bound in Section 4.3. Kashyap and Zémor [30] applied the local degree bound to improve on Mazumdar et al. for the one, two, or three error cases. They conjectured an extension for larger numbers of errors. Gabrys et al [31] applied the local degree bound to improve on Sharov and Roth.

The input and output of this channel are strings $x, y \in [2]^n$. To produce

n	$\lvert VT_0\rvert$	$\kappa^*(A)$	Thm. 4.2.1	FVY	KK	Thm. 4.2.2
5	6	6	7	7	7	12
6	10	10	12	12	12	17
7	16	17	20	20	21	25
8	30	30	35	35	36	41
9	52	53	61	61	63	69
10	94	96	109	109	113	119
11	172	175	196	197	204	211
12	316	321	357	358	372	377
13	586	593	653	657	682	682
14	1096	1104	1205	1212	1260	1248
15	2048		2237	2251	2340	2301
16	3856		4174	4202	4368	4272
17	7286		7825	7882	8191	7977
18	13798		14727	14845	15420	14969
19	26216		27820	28059	29127	28207
20	49940		52720	53202	55188	53348
21	95326		100194	101163	104857	101226
22	182362		190912	192850	199728	192623
23	349536		364621	368478	381300	367485
24	671092		697865	705511	729444	702697

Figure 4.1: The cardinality of the VT construction and several upper bounds on $p(A_n)$, where A_n is the n-bit single deletion channel. For $n \leq 14$, Kulkarni and Kiyavas [4]h were able computed the exact value of $\kappa^*(A_n)$. This requires solving an exponentially large linear program. Kulkarni and Kiyavash also constructed a dual feasible point with weight $\frac{2^n-2}{n-1}$ (column KK). This is equivalent to the first iteration of the local degree algorithm. Fazeli et al. improved on this construction (column FVY) [33]. Our Theorem 4.2.1 uses two interactions of the local degree algorithm. Computing the value of the FVY and Theorem 4.2.1 columns requires a sum over about n^2 terms. Our Theorem 4.2.2 gives an analytic upper bound on the weight of the feasible point from Theorem 4.2.1, which improves on existing bounds for $n \geq 22$.

an output from an input, select a grain pattern with at most one grain of length two and no larger grains. The grain of length two, if it exists, bridges indices j and $j+1$ for some $0 \le j \le n-2$. Then the channel output is

$$y_i = \begin{cases} x_i & i \neq j \\ x_{i+1} & i = j \end{cases}$$

If $x_j = x_{j+1}$ or if there is no grain of length two, then $y = x$.

The degree of an input string is equal to the number of runs r: each of the $r-1$ run boundaries could be bridged by a grain or there could be no error. A grain error reduces the number of runs by 0,1, or 2. The number of runs is reduced by 1 if $j = 0$ and $x_0 \neq x_1$, by 2 if $j \geq 1$, $x_j \neq x_{j+1}$, and $x_{j-1} = x_{j+1}$, and by 0 otherwise. Equivalently, the number of runs is reduced by 1 if a length-1 run at index 0 is eliminated and by 2 if a length-1 run elsewhere is eliminated. In the previous section, we let u_x be the number of length-1 runs in x and b_x be the number of length-1 runs appearing at the start or end of x. For the grain channel, we need to distinguish between length-1 runs at the start and at the end, so let b_x^L and b_x^R count these.

Theorem 4.2.3. *Let A_n be the n-bit 1-grain-error channel. The vector*

$$z_y = \frac{1}{r_y}\left(1 + \frac{2u_y - 2b_y^R - b_y^L - 2}{(r_y+2)(r_y+1)}\right)^{-1}$$

is feasible for $\kappa^*(A_n)$.

The proof can be found in Section 4.2.4. By applying the techniques used in the proof of Theorem 4.2.2, it can be shown that Theorem 4.2.3 implies that $\kappa^*(A_n) = \frac{2^{n+1}}{n+2}(1 + O(n^{-2}))$.

4.2.4 Proofs

Theorem 4.2.1. *Let*

$$f(r,u,b) \triangleq \frac{1}{r}\left(1 + \frac{\max(2u - b - 2, 0)}{(r+2)(r+1)}\right)^{-1}.$$

Then the vector $z_y = f(r_y, u_y, b_y)$ is feasible for $\kappa^*(A_n)$, so $\kappa^*(A_n) \le \mathbf{1}^T z$.

Proof: By Lemma 4.2.1, $\varphi \circ \varphi(\mathbf{1})$ is feasible for $\kappa^*(A_n)$. From the definition of φ,

$$\frac{z_y}{\varphi(z)_y} = \min_{x \in N(y)} (A_n z)_x.$$

Each $x \in [2]^n$ has r_x total subsequences, so $(A_n z'')_x = r_x$,

$$\frac{1}{\varphi(\mathbf{1})_y} = \min_{x \in N(y)} (A_n \mathbf{1})_x = \min_{x \in N(y)} r_x = r_y,$$

and $\varphi(\mathbf{1})_y = 1/r_y$.

Of the subsequences of x, $u_x - b_x$ have $r_x - 2$ runs, b_x have $r_x - 1$ runs, and $r_x - u_x$ have r_x runs, so

$$\begin{aligned}
&(A_n \varphi(\mathbf{1}))_x \\
&= \sum_{y \in N(x)} \frac{1}{r_y} \\
&= \frac{u_x - b_x}{r_x - 2} + \frac{b_x}{r_x - 1} + \frac{r_x - u_x}{r_x} \\
&= 1 + u_x \left(\frac{1}{r_x - 2} - \frac{1}{r_x} \right) + b_x \left(\frac{1}{r_x - 1} - \frac{1}{r_x - 2} \right) \\
&= 1 + \frac{2u_x(r_x - 1) - b_x r_x}{r_x(r_x - 1)(r_x - 2)} \\
&= 1 + \frac{(2u_x - b_x)(r_x - 2) + 2(u_x - b_x)}{r_x(r_x - 1)(r_x - 2)} \\
&\geq 1 + \frac{2u_x - b_x}{r_x(r_x - 1)}.
\end{aligned}$$

The inequality follows from $u_x - b_x \geq 0$.

Let $y \in [2]^{n-1}$ be a string and let $x \in [2]^n$ be a supersequence of y. It is possible to create a supersequence by extending an existing run, adding a new run at an end of the string, or by splitting an existing run into three new runs, so $r_x \leq r_y + 2$ The only way to destroy a unit run in y is to extend it into a run of length two, so $u_x \geq u_y - 1$. Similarly, $u_x - b_x \geq u_y - b_y - 1$, so $2u_x - b_x \geq 2u_y - b_y - 2$. Applying these inequalities to $(A_n \varphi(\mathbf{1}))_x$, we

conclude that

$$\begin{aligned}\frac{\varphi(\mathbf{1})_y}{(\varphi\circ\varphi(\mathbf{1}))_y} &= \min_{x\in N(y)} (A_n\varphi(\mathbf{1}))_x \\ &\geq 1+\frac{\max(2u-b-2,0)}{(r_y+2)(r_y+1)}, \\ (\varphi\circ\varphi(\mathbf{1}))_y &\leq \frac{1}{r_y}\left(1+\frac{\max(2u-b-2,0)}{(r_y+2)(r_y+1)}\right)^{-1}.\end{aligned}$$

■

Lemma 4.2.3. *There are* $2\binom{n-1}{n-r}$ *strings in* $[2]^n$ *with* r *runs.*

There are $2\binom{n-r-1}{n-2r+u}\binom{r}{r-u}$ *strings in* $[2]^n$ *with* r *runs and* u *unit runs.*

For $r\geq 2$*, there are* $2\binom{n-r-1}{n-2r+u}\binom{r-2}{u-b}\binom{2}{b}$ *strings in* $[2]^n$ *with* r *runs,* u *unit runs and* b *external unit runs.*

Proof: For $k\geq 1$, there are $\binom{n+k-1}{n}$ ways to partition n identical items into k distinguished groups. Thus there are $\binom{n-lk+k-1}{n-lk}=\binom{n-(l-1)k-1}{n-lk}$ ways to partition n items into k groups such that each group contains at least l items.

A binary string is uniquely specified by its first symbol and it run length sequence. We have n symbols to distribute among r runs such that each run contains at least one symbol, so there are $\binom{n-1}{n-r}$ arrangements. This proves the first claim. We can also specify the run sequence of a string by giving the locations of the unit runs and the lengths of the longer runs. The $r-u$ runs of length at least two can appear in r positions so there are $\binom{r}{r-u}$ arrangements, We have $n-u$ symbols to distribute among $r-u$ runs such that each run contains at least 2 symbols, so there are $\binom{n-u-(r-u)-1}{n-u-2(r-u)}=\binom{n-r-1}{n-2r+u}$ arrangements, which proves the second claim. As long as $r\geq 2$, the internal unit runs two can appear in $r-2$ positions and the external unit runs can appear in two positions, so there are $\binom{r-2}{u-b}\binom{2}{b}$ possible arrangements, which proves the third claim. ■

Note that Lemma 4.2.3 uses the polynomial definition of binomial coefficients, which can be nonzero even when the top entry is negative. For example, the number of strings of length n with n runs, n unit runs, and two external unit runs is $2\binom{-1}{0}\binom{n-2}{n-2}\binom{2}{2}=2$.

For compactness, let

$$\mathop{\mathbb{E}}_{r}[f(r)] = \frac{1}{2^{n-1}} \sum_{r \geq 1} \binom{n-1}{n-r} f(r),$$

and let $\mathop{\mathbb{E}}_{u,b}[f(r,u,b)]$ equal

$$\frac{1}{\binom{n-1}{n-r}} \sum_{u=0}^{r} \sum_{b=0}^{2} \binom{n-r-1}{n-2r+u} \binom{r-2}{u-b} \binom{2}{b} f(r,u,b)$$

for $r > 1$ and let $\mathbb{E}_{u,b}[f(1,u,b)] = f(1,0,0)$.

If $z_x = f(r_x, u_x, b_x)$, then

$$\begin{aligned}
\mathbf{1}^T z &= \sum_{x \in [2]^n} f(r_x, u_x, b_x) \\
&= 2f(1,0,0) + \\
&\quad 2 \sum_{r=2}^{n} \sum_{u=0}^{r} \sum_{b=0}^{2} \binom{n-r-1}{n-2r+u} \binom{r-2}{u-b} \binom{2}{b} f(r,u,b) \\
&= 2^n \mathop{\mathbb{E}}_{r} \left[\mathop{\mathbb{E}}_{u,b} [f(r,u,b)] \right]. \qquad (4.2)
\end{aligned}$$

Analysis of the feasible point constructed in Theorem 4.2.1 relies on the following identities. For $k \geq 0$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{r} \left[\frac{1}{\binom{r+k-1}{r-1}} \right] &= \frac{\sum_{r \geq 1} \binom{n+k-1}{n-r}}{2^{n-1} \binom{n+k-1}{n-1}} \\
&\leq \frac{2^k}{\binom{n+k-1}{n-1}} \qquad (4.3) \\
\mathop{\mathbb{E}}_{u,b} \left[\binom{u}{u-k} \right] &= \frac{\binom{r}{r-k} \binom{r-1}{r-k-1}}{\binom{n-1}{n-k-1}} \qquad (4.4) \\
\mathop{\mathbb{E}}_{u,b} [b] &= \frac{2(r-1)}{n-1}. \qquad (4.5)
\end{aligned}$$

Each of these can be easily derived from the binomial theorem and Vandermonde's identity.

Theorem 4.2.2. *For $n \geq 2$,*

$$\kappa^*(A_n) \leq \frac{2^n}{n+1} \left(1 + \frac{26}{n(n-1)} \right).$$

Proof: For $n \leq 13$, this follows from the bound of Kulkarni and Kiyavash [4]. This proof covers $n \geq 10$.

From Theorem 4.2.1 and (4.2), we have $\kappa^*(A_{n+1}) \leq 2^n \, \mathbb{E}_r[\mathbb{E}_{u,b}[f(r,u,b)]]$ where

$$f(r,u,b) = \frac{1}{r}\left(1 + \frac{\max(2u-b-2,0)}{(r+2)(r+1)}\right)^{-1}.$$

For $x > 0$, $(1+x)^{-1} \leq 1 - x + x^2$, so $f(r,u,b)$ is at most

$$\begin{aligned}
&\frac{1}{r}\left(1 - \frac{\max(2u-b-2,0)}{(r+2)(r+1)} + \frac{(\max(2u-b-2,0))^2}{(r+2)^2(r+1)^2}\right) \\
&\leq \frac{1}{r} - \frac{2u-b-2}{(r+2)(r+1)r} + \frac{2u(2u-2)}{(r+2)^2(r+1)^2 r}.
\end{aligned}$$

We will bound this term by term using (4.3), (4.4), and (4.5). First

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r}\left[\mathop{\mathbb{E}}_{u,b}\left[\frac{1}{r} - \frac{2u-b-2}{(r+2)(r+1)r}\right]\right] \\
&= \mathop{\mathbb{E}}_{r}\left[\frac{1}{r} - \frac{1}{(r+2)(r+1)r}\left(2\frac{r(r-1)}{n-1} - \frac{2(r-1)}{n-1} - 2\right)\right] \\
&= \mathop{\mathbb{E}}_{r}\left[\frac{1}{r} - \frac{2}{n-1}\cdot\frac{(r-1)^2 - n + 1}{(r+2)(r+1)r}\right] \\
&= \frac{2}{n-1}\mathop{\mathbb{E}}_{r}\left[\frac{n-1}{2r} - \frac{1}{r} + \frac{5}{(r+1)r} + \frac{n-10}{(r+2)(r+1)r}\right] \\
&= \frac{2}{n-1}\mathop{\mathbb{E}}_{r}\left[\frac{n-3}{2r} + \frac{5}{(r+1)r} + \frac{n-10}{(r+2)(r+1)r}\right] \\
&\leq \frac{2}{n-1}\left(\frac{n-3}{n} + \frac{20}{(n+1)n} + \frac{8n-80}{(n+2)(n+1)n}\right) \\
&= \frac{2}{n-1}\left(\frac{n^2-n-6}{(n+2)n} + \frac{28n-40}{(n+2)(n+1)n}\right) \\
&= \frac{2}{n+2}\left(\frac{n^2-n-6}{n(n-1)} + \frac{28n-40}{(n+1)n(n-1)}\right) \\
&= \frac{2}{n+2}\left(1 + \frac{22n-46}{(n+1)n(n-1)}\right) \\
&< \frac{2}{n+2}\left(1 + \frac{22}{(n+1)n}\right).
\end{aligned}$$

Second,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r}\left[\mathop{\mathbb{E}}_{u,b}\left[\frac{4u(u-1)}{(r+2)^2(r+1)^2r}\right]\right] \\
&= \mathop{\mathbb{E}}_{r}\left[\frac{4}{(r+2)^2(r+1)^2r}\cdot\frac{r(r-1)^2(r-2)}{(n-1)(n-2)}\right] \\
&< \mathop{\mathbb{E}}_{r}\left[\frac{4}{(r+2)(r+1)}\cdot\frac{(r-1)(r-2)}{(n-1)(n-2)}\cdot\frac{1}{r}\right] \\
&\leq \mathop{\mathbb{E}}_{r}\left[\frac{4}{(r+2)(r+1)}\cdot\frac{(r+2)(r+1)}{(n+2)(n+1)}\cdot\frac{1}{r}\right] \\
&= \mathop{\mathbb{E}}_{r}\left[\frac{4}{(n+2)(n+1)}\cdot\frac{1}{r}\right] \\
&\leq \frac{8}{(n+2)(n+1)n}.
\end{aligned}$$

Combining these two terms, we get

$$\kappa^*(A_{n+1}) \leq 2^n\frac{2}{n+2}\left(1+\frac{22}{(n+1)n}+\frac{4}{(n+1)n}\right).$$

■

Lemma 4.2.2. *Let $z_y = f'(r_y, u_y, b_y)$. Then*

$$\mathbf{1}^T z \geq \frac{2^n-2}{n+1}\left(1+\frac{1}{n-1}-\frac{3}{(n-1)(n-2)}\right).$$

Proof:

$$\begin{aligned}
f'(r,u,b) &\geq \frac{1}{r}\left(1-\frac{u-b}{r^2}\right) \\
&\geq \frac{1}{r}\left(1-\frac{u-b}{(r-1)(r-2)}\right)
\end{aligned}$$

$$
\begin{aligned}
&2^n \mathop{\mathbb{E}}_r \left[\mathop{\mathbb{E}}_{u,b} \left[\frac{1}{r} \left(1 - \frac{u-b}{(r-1)(r-2)} \right) \right] \right] \\
&= 2^n \mathop{\mathbb{E}}_r \left[\frac{1}{r} \left(1 - \frac{1}{(r-1)(r-2)} \left(\frac{r(r-1)}{n-1} - \frac{2(r-1)}{n-1} \right) \right) \right] \\
&= 2^n \mathop{\mathbb{E}}_r \left[\frac{1}{r} \left(1 - \frac{1}{n-1} \right) \right] \\
&= 2^n \frac{2^n - 1}{2^{n-1} n} \left(1 - \frac{1}{n-1} \right) \\
&= \frac{2^{n+1} - 2}{n+2} \left(\frac{(n+2)(n-2)}{n(n-1)} \right) \\
&= \frac{2^{n+1} - 2}{n+2} \left(1 + \frac{1}{n} - \frac{3}{n(n-1)} \right).
\end{aligned}
$$

■

Theorem 4.2.3. *Let A_n be the n-bit 1-grain-error channel. The vector*

$$
z_y = \frac{1}{r_y} \left(1 + \frac{2u_y - 2b_y^R - b_y^L - 2}{(r_y + 2)(r_y + 1)} \right)^{-1}
$$

is feasible for $\kappa^*(A_n)$.

Proof: By Lemma 4.2.1, $\varphi \circ \varphi(\mathbf{1})$ is feasible for $\kappa^*(A)$. From the definition of φ,

$$
\frac{z_y}{\varphi(z)_y} = \min_{x \in N(y)} (A_n z)_x.
$$

Each $x \in [2]^n$ has r_x total neighbors, so $(A_n z'')_x = r_x$,

$$
\frac{1}{\varphi(\mathbf{1})_y} = \min_{x \in N(y)} (A\mathbf{1})_x = \min_{x \in N(y)} r_x = r_y,
$$

and $\varphi(\mathbf{1})_y = 1/r_y$.

Of the neighbors of x, $u_x - b_x^L - b_x^R$ have $r_x - 2$ runs, b_x^L have $r_x - 1$ runs,

and $r_x - u_x + b_x^R$ have r_x runs, so $(A_n\varphi(\mathbf{1}))_x$ equals

$$\begin{aligned}
&\sum_{y\in N(x)} \frac{1}{r_y} \\
&= \frac{u_x - b_x^L - b_x^R}{r_x - 2} + \frac{b_x^L}{r_x - 1} + \frac{r_x - u_x + b_x^R}{r_x} \\
&= 1 + (u_x - b_x^R)\left(\frac{1}{r_x - 2} - \frac{1}{r_x}\right) + b_x^L\left(\frac{1}{r_x - 1} - \frac{1}{r_x - 2}\right) \\
&= 1 + \frac{2(u_x - b_x^R)(r_x - 1) - b_x^L r_x}{r_x(r_x - 1)(r_x - 2)} \\
&= 1 + \frac{(2u_x - 2b_x^R - b_x^L)(r_x - 2) + 2(u_x - b_x^R - b_x^L)}{r_x(r_x - 1)(r_x - 2)} \\
&\geq 1 + \frac{2u_x - 2b_x^R - b_x^L}{r_x(r_x - 1)}.
\end{aligned}$$

Let $x \in [2]^n$ be an input and let $y \in N(x)$. A grain error can leave the number of runs unchanged, destroy a unit run at the start of x, or destroy a unit run in the middle of x, merging the adjacent runs. Thus $r_y \geq r_x - 2$. The only way to produce a unit run in y is shorten a run of length two in x, so $u_x \geq u_y - 1$. Similarly, $2u_x - 2b_x^R - b_x^L \geq 2u_y - 2b_y^R - b_y^L - 2$. Applying these inequalities to $(A\varphi(\mathbf{1}))_x$, we conclude that

$$\begin{aligned}
\frac{\varphi(\mathbf{1})_y}{(\varphi \circ \varphi(\mathbf{1}))_y} &= \min_{x\in N(y)} (A\varphi(\mathbf{1}))_x \geq 1 + \frac{2u_y - 2b_y^R - b_y^L - 2}{(r_y + 2)(r_y + 1)}, \\
(\varphi \circ \varphi(\mathbf{1}))_y &\leq \frac{1}{r_y}\left(1 + \frac{2u_y - 2b_y^R - b_y - 2}{(r_y + 2)(r_y + 1)}\right)^{-1}.
\end{aligned}$$

■

4.3 The degree sequence upper bound

If a channel $A \in \{0,1\}^{X\times Y}$ is input regular, then the local degree bound reduces to the minimum degree bound. We have $A\mathbf{1} = \mathbf{1}d$ and $\kappa^*_{\mathrm{LD}}(A, \mathbf{1}) = \kappa^*_{\mathrm{MD}}(A, \mathbf{1}) = |Y|/d$. The advantage of the local degree bound is robustness to variation in the input degree distribution. The degree threshold upper bound is an alternative technique for dealing with nonuniform combinatorial channels that predates the local degree bound by many decades. Levenshtein applied this idea to obtain an upper bound on codes for the deletion channel

[14]. Kulkarni and Kiyavash applied the local degree bound to the deletion channel and showed that the resulting bound improved on Levenshtein's result [4].

The degree threshold bound is a simple generalization of the minimum degree bound and the basic idea behind the bound does not require linear programming. Pick a degree threshold d and let $S = \{x \in X : |N(x)| < d\}$, the set of low degree inputs. Each member of S appears at most once in the maximum packing. Applying the minimum degree bound to $X \setminus S$ gives

$$p(A) \leq \frac{|X| - |S|}{d} + |S|.$$

This approach is effective when the degree distribution concentrates around its mean but still has a few vertices with much lower degree.

This idea can be taken a bit further. For any code $C \subseteq X$, $\sum_{x \in C} |N(x)| \leq |Y|$. The size of the largest input set C satisfying this inequality is an upper bound on the size of the largest code. This set can be found greedily by repeatedly adding the minimum degree remaining input vertex. Call this the degree sequence upper bound.

The degree sequence upper bound is monotonic and decreasing in the degree of each vertex. The degree threshold bound corresponds to a simplified degree sequence containing only degrees 1 and d.

Although its definition does not require a linear program, the degree sequence bound still has a nice linear programming interpretation. Taking this perspective, we compare the performance of the degree sequence bound to the local degree bound for arbitrary channels and show that the local degree bound is always better.

4.3.1 Linear programs for the degree sequence bound

While the local degree upper bound is naturally expressed as a feasible point in the program for κ^*, the easiest linear programming interpretation of the degree sequence bound works differently. The degree sequence upper bound is the value of a further relaxation of the program for p^*. It turns out that the minimum degree upper bound is easily expressed as both a dual feasible point and a primal relaxation. Section 4.2 included the former interpretation and the latter is given here. For a channel $A \in \{0,1\}^{X \times Y}$ and a vector

$t \in \mathbb{R}^Y$, define

$$\begin{aligned} p^*_{\text{MD}}(A,t) = \max_{w \in \mathbb{R}^X} & \mathbf{1}^T w \\ \text{s. t. } & w \geq \mathbf{0} \\ & t^T A^T w \leq t^T \mathbf{1}. \end{aligned}$$

The solution to this program puts all of the weight on the minimum degree input $\text{argmin}_x (At)_x$, so $p^*_{\text{MD}}(A,t) = \kappa^*_{\text{MD}}(A,t)$.

Recall that $A\mathbf{1}$ is the vector of input degrees of the channel graph of A. Thus the main constraint of the program for $\kappa^*_{\text{MD}}(A, \mathbf{1})$ is $\sum_{x \in X} |N(x)| w_x \leq |Y|$. In a code, each vertex can only be included once. We can capture this fact and improve the upper bound by adding the additional constraint $w \leq \mathbf{1}$ to the program.

Definition 4.3.1. *For a channel $A \in \{0,1\}^{X \times Y}$ and a vector $t \in \mathbb{R}^Y$ such that $(At)_x > 0$ for all $x \in X$, define the degree sequence bound*

$$\begin{aligned} p^*_{\text{DS}}(A,t) = \max_{w \in \mathbb{R}^X} & \mathbf{1}^T w \\ \text{s. t. } & \mathbf{0} \leq w \leq \mathbf{1} \\ & t^T A^T w \leq t^T \mathbf{1}. \end{aligned}$$

The degree sequence upper bound is tight: for a given input degree distribution and output space size, there is some channel where the neighborhoods of the small degree inputs are disjoint. For this channel, the degree sequence upper bound is tight. The bound cannot be improved with incorporating more information about the structure of the channel.

The local degree upper bound, which incorporates information about the channel beyond the degree sequence, is always at least as good as the degree sequence bound. To see this, we associate the degree sequence bound with a particular feasible point in the program for $\kappa^*(A)$.

Lemma 4.3.1. *Let $A \in \{0,1\}^{X \times Y}$ be a channel and let $t \in \mathbb{R}^Y$ such that $(At)_x > 0$ for all $x \in X$. Then there is a vector $\psi_A(t) \in \mathbb{R}^Y$ such that*

- $\mathbf{1}^T \psi_A(t) = p^*_{\text{DS}}(A,t)$
- $\varphi_A(t) \leq \psi_A(t)$
- *$\psi_A(t)$ is feasible in the program for $\kappa^*(A)$.*

Proof: The vector At contains the weighted degree of each input under the output weighting t. For $d \in \mathbb{R}$, define the following sets of inputs:

$$S(d) = \{x \in X : (At)_x < d\},$$
$$S'(d) = \{x \in X : (At)_x \leq d\}.$$

Define $f(d) = \mathbf{1}_{S(d)}^T At$, the sum of the weighted degrees of all of the inputs with weighted degree less than d, and $f'(d) = \mathbf{1}_{S'(d)}^T At$. Both $f(d)$ and $f'(d)$ are nondecreasing functions of d and $f(d) \leq f'(d)$. Because $f(0) = 0 \leq \mathbf{1}^T t \leq \mathbf{1}^T At = f'(\mathbf{1}^T t)$, there is some d such that $f(d) \leq \mathbf{1}^T t \leq f'(d)$. Then there is some λ satisfying $\mathbf{1}^T t = \lambda f'(d) + (1-\lambda) f(d)$.

First we establish that $p^*_{\text{DS}}(A, t) = (1-\lambda)|S(d)| + \lambda |S'(d)|$ by constructing a primal feasible point and a dual feasible point with this value.

The point $w = (1-\lambda)\mathbf{1}_{S(d)} + \lambda \mathbf{1}_{S'(d)}$ is feasible for the primal program for $p^*_{\text{DS}}(A, t)$. This puts a weight of one, the maximum possible weight, on each of the inputs with degree below the threshold and fractional weight of λ on inputs with degree equal to the threshold. To see that the nontrivial feasibility condition is satisfied, observe that $\mathbf{1}^T A^T w = (1-\lambda) f(d) + \lambda f'(d) = \mathbf{1}^T t$.

The dual program for $p^*_{\text{DS}}(A, t)$ is

$$\begin{aligned} \min_{c \in \mathbb{R},\, z \in \mathbb{R}^X} \quad & \mathbf{1}^T t c + \mathbf{1}^T z \\ \text{s. t. } & c \geq 0 \\ & z \geq \mathbf{0} \\ & Atc + z \geq \mathbf{1}. \end{aligned}$$

The point $c = \frac{1}{d}$, $z_x = \max(0, 1 - \frac{(At)_x}{d})$ is feasible in the dual program. Note that $z_x > 0$ exactly for those $x \in S(d)$. The value of this point is

$$\begin{aligned} \frac{\mathbf{1}^T t}{d} + \sum_{x \in S(d)} \frac{d - (At)_x}{d} &= |S(d)| + \frac{\mathbf{1}^T t - \mathbf{1}_{S(d)}^T At}{d} \\ &= |S(d)| + \frac{\mathbf{1}^T t - f(d)}{d} \\ &= |S(d)| + \frac{\lambda(f'(d) - f(d))}{d} \\ &= |S(d)| + \lambda(|S'(d)| - |S(d)|). \end{aligned}$$

The final equality follows from the fact that each vertex in $S'(d) \setminus S(d)$ has weighted degree d.

Next, we construct $\psi_A(t)$ from (c, z):

$$\psi_A(t)_y = t_y \left(c + \sum_x \frac{z_x A_{x,y}}{(At)_x} \right).$$

To establish the first claim, we compute

$$\begin{aligned}
\mathbf{1}^T \psi_A(t) &= \mathbf{1}^T tc + \sum_y t_y \sum_x \frac{z_x A_{x,y}}{(At)_x} \\
&= \mathbf{1}^T tc + \sum_x \frac{z_x}{(At)_x} \sum_y A_{x,y} t_y \\
&= \mathbf{1}^T tc + \mathbf{1}^T z.
\end{aligned}$$

Substituting the values $c = \frac{1}{d}$ and $z_x = \max(\frac{d-(At)_x}{d}, 0)$, we obtain

$$\begin{aligned}
\frac{\psi_A(t)_y}{t_y} &= \frac{1}{d} + \sum_{x \in N(y)} \frac{1}{(At)_x} \max\left(\frac{d - (At)_x}{d}, 0 \right) \\
&= \frac{1}{d} + \sum_{x \in N(y)} \max\left(\frac{1}{(At)_x} - \frac{1}{d}, 0 \right) \\
&\geq \frac{1}{d} + \max_{x \in N(y)} \max\left(\frac{1}{(At)_x} - \frac{1}{d}, 0 \right) \\
&= \max_{x \in N(y)} \max\left(\frac{1}{(At)_x}, \frac{1}{d} \right) \\
&\geq \max_{x \in N(y)} \frac{1}{(At)_x} \\
&= \frac{1}{t_y} \varphi_A(t).
\end{aligned}$$

This establishes the second claim.

Because $\varphi_A(t)$ is feasible in $\kappa^*(A)$, $\psi_A(t)$ is as well, which establishes the third claim. ■

Theorem 4.3.1. *Let $A \in \{0,1\}^{X \times Y}$ be a channel and let $t \in \mathbb{R}^Y$ such that $(At)_x > 0$ for all $x \in X$. Then $\kappa^*_{\text{LD}}(A, t) \leq p^*_{\text{DS}}(A, t)$.*

Proof: This follows immediately from the first and second claims of Lemma 4.3.1. ■

This interpretation of the degree sequence bound allows us to iteratively

construct dual feasible points, but these points are dominated by those produced by the local degree algorithm. However, this interpretation does give some intuition about the source of the superior performance of the local degree bound. When multiple low-degree inputs have a common output, the local degree bound takes advantage of this fact. This information is not contained in the degree distribution.

There are several open questions regarding families of channels with the same confusability graphs. Under what conditions can we find these families? What is the relationship between these families and distance metrics? When we have a family of channels that are not input or output regular, what should we do to get the best bounds?

4.4 Conclusion

We have discussed two aspects of the problem of finding upper bounds on the size of codes for combinatorial channels: fractional coverings for a particular channel and families of channels with the same codes. In both cases, there is a well-defined optimal version of the bound: for a particular channel there is the minimum weight fractional covering, and for a family there is the minimum fractional clique cover of the confusion graph. In both cases, finding these optimal bounds can be intractable.

When the channel is input-regular, the minimum degree, degree sequence, and local degree upper bounds here are equivalent, but not necessarily equal to the fractional covering number. The local degree bound is always at least as good as the degree sequence bound but uses more information about the structure of the channel. The local degree bound can be iterated to obtain stronger bounds. The best sphere-packing bound for a given channel can be much weaker than the best sphere-packing bound for some other channel that admits the same codes. Consequently, finding a family of channels equivalent to the channel of interest can be very powerful.

CHAPTER 5

LOCALLY BIJECTIVE COVER PRESERVING HOMOMORPHISMS

5.1 Locally bijective poset homomorphisms

In Section 2.3, we defined three types of homomorphisms between posets. The strongest of these was a cover preserving map. If a map $f : X \to Z$ is cover preserving, then for all $x \in X$ and $\hat{x} \in U_1(x)$, $f(\hat{x}) \in U_1(f(x))$. Another way to say this is that the partial function from $U_1(x)$ to $U_1(f(x))$ defined by f is in fact a function. In this chapter, we will study even stronger notions of homomorphism.

Definition 5.1.1. *Let f be a cover preserving map $X \to Z$. Call f* locally injective *if for $x \in X$, the induced function $U_1(x) \to U_1(f(x))$ is injective. Define* locally surjective *and* locally bijective *analogously.*

5.1.1 Properties

The following definitions will be useful for analyzing such maps. Recall that $x \prec y$ means that x is covered by y, i.e. $x < y$ and there are no points in between them.

Definition 5.1.2. *Let X be a poset.*

- *A set $S \subseteq X$ is a* chain *of X if all of its elements are comparable, i.e., the elements can be indexed by $[|S|]$ such that $x_0 < x_1 < \ldots < x_{|S|-1}$.*
- *A chain S is* rooted *if it contains $\perp$, the bottom element of X.*
- *A chain S is* skipless *if its elements can be indexed by $[|S|]$ such that $x_0 \prec x_1 \prec \ldots \prec x_{|S|-1}$.*

For compactness, we will refer to an n-element rooted skipless chain as an n-chain.

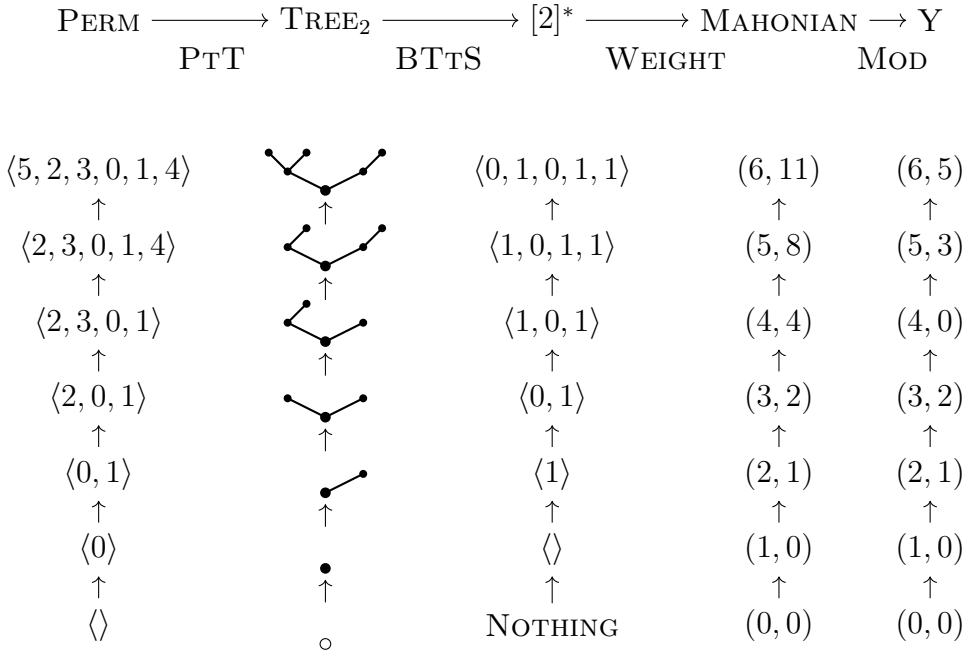

Figure 5.1: Corresponding chains in (PERM, **subseq**), (NOTHING $\sqcup$ TREE$_2$, **subtree**), (NOTHING $\sqcup$ $[2]^*$, **subseq**), (MAHONIAN, $\leq$), (Y, $\leq$). These posets and the locally bijective cover reserving maps connecting them will be defined throughout the chapter.

Lemma 5.1.1. *Let X and Z each be a ranked poset with a bottom element and let $f : X \to Z$ be locally surjective. Let $z \in Z$ and $x \in f^{-1}(z)$. Then for each n, f induces a surjection between the length-n skipless chains of X starting at x and the length-n skipless of Z starting at z. If f is locally bijective, this is a bijection.*

Proof: Let $T = \{t_0, \ldots, t_{n-1}\} \subseteq Z$ be a skipless chain with $t_0 = z$. For each i, if we have s_i such that $f(s_i) = t_i$, from the definition of locally surjective, there is some s_{i+1} such that $s_{i+1} \succ s_i$ and $f(s_{i+1}) = t_{i+1}$. If f is locally bijective, s_{i+1} is unique. Because $f(x) = z$, by induction we can construct a skipless chain $S = \{s_0, \ldots, s_{n-1}\} \subseteq X$ such that $f(s_i) = t_i$. ■

Figure 5.1 illustrates corresponding chains in five posets that are linked by a sequence of locally bijective maps.

Lemma 5.1.2. *Let X and Z each be a ranked poset with a bottom element and let $f : X \to Z$ be a cover preserving map. The following properties are equivalent:*

1. *f is surjective and locally surjective.*

2. *f is locally surjective and bottom preserving ($f(\bot) = \bot$).*

3. *f is surjective and for all $z \in Z$, $y \in U(z)$, and $w \in f^{-1}(z)$, there is some $x \in U(w)$ such that $f(x) = y$.*

Proof:

(1) $\implies$ (2): Because f is cover preserving and $\mathrm{r}(\bot) = 0$ in both X and Z, for all $x \in X$, $\mathrm{r}(x) = \mathrm{r}(x) - \mathrm{r}(\bot) = \mathrm{r}(f(x)) - \mathrm{r}(f(\bot)) \leq \mathrm{r}(f(x))$. Because f is surjective, there is some x such that $f(x) = \bot$. Then $\mathrm{r}(x) \leq \mathrm{r}(f(x)) = \mathrm{r}(\bot) = 0$, so $x = \bot$.

(2) $\implies$ (3): For all $z \in Z$, $y \in U(z)$, there is a skipless chain $T = \{t_{\mathrm{r}(z)}, \ldots, t_{\mathrm{r}(y)}\} \subseteq Z$, with $t_{\mathrm{r}(z)} = z$ and $t_{\mathrm{r}(y)} = y$. From Lemma 5.1.1 there is a skipless chain $S = \{s_{\mathrm{r}(z)}, \ldots, s_{\mathrm{r}(y)}\} \subseteq X$ such that $f(s_i) = t_i$. In particular, $f(s_{\mathrm{r}(y)}) = y$.

We have $f(\bot) = \bot$, so for any $y \in Z$ there is some $x \in f^{-1}(y)$, i.e. f is surjective.

(3) $\implies$ (1): Replacing $y \in U(z)$ with $y \in U_1(z)$ gives the definition of locally surjective. ■

We will be primarily interested in locally bijective maps to and from $([q]^*, \mathbf{subseq})$. Recall from Section 2.7 that $([q]^*, \mathbf{subseq})$ is not a lattice or even a semilattice. A locally surjective map from a meet semilattice to another poset has the following useful property.

Lemma 5.1.3. *Let X be a meet semilattice, let Z be a ranked poset with a bottom element, and let $f : X \to Z$ be surjective and locally surjective. Let $z, z' \in Z$, let y be a maximal element of $D(z) \cap D(z')$, and let $w \in f^{-1}(y)$. Then there are $x, x' \in U(w)$ such that $f(x) = z$, $f(x') = z'$, and $x \wedge x' = w$.*

Proof: Because $z, z' \in U(y)$, from Lemma 5.1.2 there are $x, x' \in U(w)$ such that $f(x) = z$, $f(x') = z'$. We have $x \geq x \wedge x'$ and $x' \geq x \wedge x'$. Because f is order preserving, $z \geq f(x \wedge x')$ and $z' \geq f(x \wedge x')$. Equivalently, $f(x \wedge x') \in D(z) \cap D(z')$. Because y is maximal in $D(z) \cap D(z')$, $f(x \wedge x') \not> y$. On the other hand, $x \wedge x' \geq w$ so $f(x \wedge x') \geq y$, so $f(x \wedge x') = y$. Because f is rank preserving, $\mathrm{r}(x \wedge x') = \mathrm{r}(y) = \mathrm{r}(w)$, and $x \wedge x'$ is comparable to w so they must be equal. ■

In other words, if $D(z) \cap D(z')$ has many maximal elements, z and z' must have many preimages in f.

For any ranked poset with bottom element Z, there is at least one semilattice X and map $f : X \to Z$ such that f is bijective and locally surjective map to Z. Take X to be the rooted skipless chains of Z ordered by the prefix relation. This is a meet semilattice: the meet of two chains is their longest common prefix. However, it is not a lattice. Then let f map each chain to its maximal element. This is clearly surjective and locally bijective.

5.2 Permutations and Mahonian statistics

A string of length n over the alphabet $[n]$ is a permutation if each symbol appears exactly once. Let $[n]!$ be the set of permutations of $[n]$. Let

$$\textsc{Perm} = \bigsqcup_{n \in \mathbb{N}} [n]!.$$

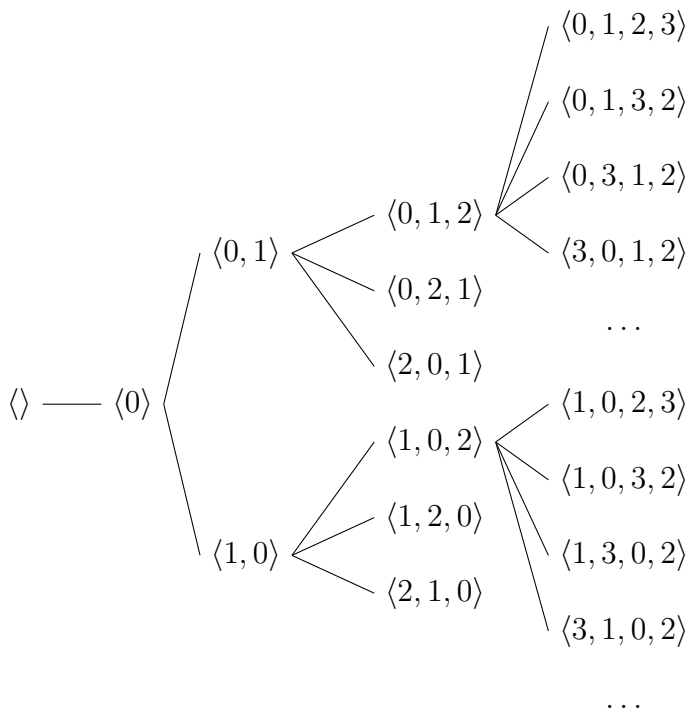

Figure 5.2: The cover relations of (Perm, **subseq**) in ranks 0 through 3, and part of rank 4.

Definition 5.2.1. *Let* (Perm, ***subseq***) *be the subsequence partial order on*

permutations of all lengths. For $x \in [n]!$ and $y \in [n+1]!$, $x \prec y$ if x is produced by deleting the symbol n from y.

(PERM, **subseq**) is depicted in Figure 5.2. The rank of a permutation of $[n]$ in this poset is n.

A permutation statistic is a map PERM $\to \mathbb{N}$. Well-known examples include the parity of permutation, the number of cycles, the number of inversions, and the number of descents. In many cases, multiple permutation statistics have the same distribution of values over all permutations. A permutation statistic f is *Mahonian* if its distribution has the following generating function [37–41]:

$$\sum_{\pi \in [n]!} z^{f(\pi)} = \prod_{i \in [n]} (1 + z + \ldots + z^i) = \prod_{i \in [n]} \frac{1 - z^{i+1}}{1 - z}.$$

There is another simple way to achieve this distribution. Let $S_n = \prod_{i \in [n]} [i+1]$ and let $f : S_n \to \mathbb{N}$, $f(s) = \sum_{i \in [n]} s_i$. Then the distribution of f over S_n is the same as the distribution of a Mahonian statistic over $[n]!$.

The most well-known Mahonian statistic is the number of inversions of a permutation:

$$\mathrm{inv}(\pi) = \left| \left\{ (i,j) \in \binom{[|\pi|]}{2} : \pi_i > \pi_j \right\} \right|.$$

One proof that this is Mahonian involves the subsequence order on permutations. If we have a permutation of $[n-1]$ and insert the symbol n, we preserve all of the existing inversions and create between 0 and $n-1$ new inversions, depending on the number of symbols to the right of the inserted n. In fact, this argument shows that $\pi \to (|\pi|, \mathrm{inv}(\pi))$ is a cover preserving locally bijective map from (PERM, **subseq**) to the following poset, which is depicted in Figure 5.3

Definition 5.2.2. *Let* MAHONIAN $= \{(a,b) \in \mathbb{N}^2 : b \leq \binom{a}{2}\}$. *Define the poset* (MAHONIAN, $\leq$) *by the cover relations*

$$(a,b) \prec (c,d) \iff (c = a+1) \wedge (0 \leq d - b \leq a).$$

Thus $U_1((a,b)) = \{(a+1, b+i) : i \in [a+1]\}$.

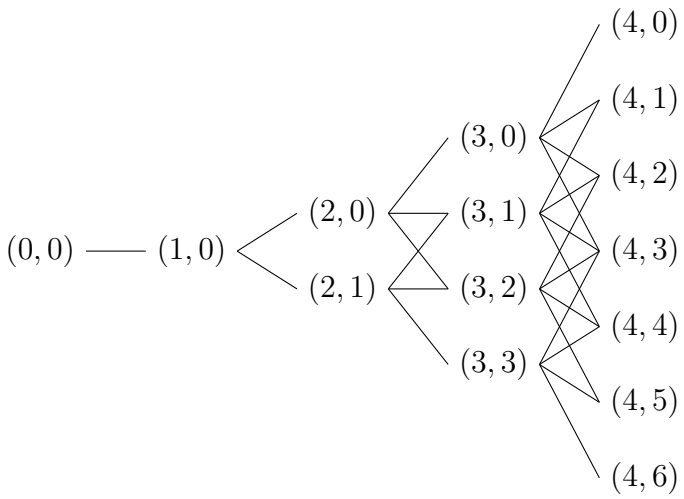

Figure 5.3: Cover relations in ranks 0 through 4 of (MAHONIAN, $\leq$).

A second famous Mahonian statistic is the major index of a permutation. We will define this as maj = vt $\circ$UPDOWN. The proof that this statistic is Mahonian is somewhat more involved The first proof is due to MacMahon [42]. More combinatorial proofs followed much later [43, 44].

Definition 5.2.3. *Define the function taking a permutation to its up-down sequence,* UPDOWN : PERM $\to$ NOTHING $\sqcup\, [2]^*$*, as follows. For a nonempty permutation* π*, let* UPDOWN$(\pi) \in [2]^{|\pi|-1}$ *such that*

$$\text{UPDOWN}(\pi)_i = \begin{cases} 1 & \pi_i < \pi_{i+1} \\ 0 & \pi_i > \pi_{i+1} \end{cases}$$

and let UPDOWN$(\langle\rangle)$ = NOTHING.

Define the Varshamov-Tenengolts weight of a binary string as follows:

$$\begin{aligned} \text{vt} &: \text{NOTHING} \sqcup [2]^* \to \mathbb{N} \\ \text{vt}(\text{NOTHING}) &= 0 \\ \text{vt}(x) &= \sum_{i \in [|x|]} (i+1)x_i \\ \text{WEIGHT} &: \text{NOTHING} \sqcup [2]^* \to \text{MAHONIAN} \\ \text{WEIGHT}(x) &= (\text{r}(x), \text{vt}(x)). \end{aligned}$$

Lemma 5.2.1. UPDOWN : (PERM, ***subseq***) $\to$ ($[2]^*$, ***subseq***) *is a locally bijective cover preserving map.*

Proof: Let π be a permutation of $[n]$ and let $x = \textsc{UpDown}(p)$, and let $\hat{x} \succ x$. Then there is a unique $\hat{\pi} \succ \pi$ such that $\hat{x} = \textsc{UpDown}(\hat{\pi})$. There may be multiple ways to decompose $\hat{x}$ as $\hat{x} = u + \langle j \rangle + v$ where $x = u + v$. However, exactly one of the following decompositions exists:

1. $\hat{x} = \langle 0 \rangle + x$,
2. $\hat{x} = x + \langle 1 \rangle$,
3. $\hat{x} = u + \langle 10 \rangle + v$ and $x = u + \langle 1 \rangle + v$,
4. $\hat{x} = u + \langle 10 \rangle + v$ and $x = u + \langle 0 \rangle + v$.

These decompositions correspond to the following rules about extending runs: only extend a run of zeros by inserting a zero at the left end and only extend a run of ones by inserting a one at the right end.

In the first case, let $\hat{\pi} = \langle n \rangle + \pi$. In the second case, let $\hat{\pi} = \pi + \langle n \rangle$. In the third and fourth cases, let $\hat{\pi}_i = n$, where $i = |u| + 1$. That is, $\hat{\pi} = a + \langle n \rangle + b$, where $\pi = a + b$ and $\textsc{UpDown}(a) = u$. In each case, it is clear that $\hat{x} = \textsc{UpDown}(\hat{\pi})$. ∎

Lemma 5.2.2. $\textsc{Weight} : (\textsc{Nothing} \sqcup [2]^*, \boldsymbol{subseq}) \to (\textsc{Mahonian}, \leq)$ *is a locally bijective cover preserving map.*

Proof: We need to show that for all $x \in \textsc{Nothing} \sqcup [2]^*$, $\textsc{Weight}$ bijectively maps $U_1(x)$ to $U_1(\textsc{Weight}(x))$. If $x = \textsc{Nothing}$, $U_1(x) = \{\langle\rangle\}$ and $U_1(\textsc{Weight}(x)) = U_1((0,0)) = \{(1,0)\}$ so we have a trivial bijection.

Now let $x \in [2]^*$ and let $\#(x)$ be the composition of x. That is, $\# : [2]^* \to \mathbb{N}^{[2]}$, $\#(x)_0$ is the number of 0 symbols in x, and $\#(x)_1$ is the number

of 1 symbols in x. Let $\#(x) = (k_0, k_1)$ and let $n = k_0 + k_1$. Then

$$\begin{aligned}
\mathrm{vt}(x) &= \sum_{j\in[n]} (j+1)x_j \\
&= \sum_{j\in[n]:x_j=1} (j+1) \\
&= \left|\left\{(i,j) \in [n]^2 : i \le j,\, x_j = 1\right\}\right| \\
&= \left|\left\{(i,j) \in [n]^2 : i \le j,\, x_i = x_j = 1\right\}\right| + \\
&\qquad \left|\left\{(i,j) \in [n]^2 : i < j,\, x_i = 0,\, x_j = 1\right\}\right| \\
&= \binom{k_1+1}{2} + \left|\left\{(i,j) \in [n]^2 : i < j,\, x_i = 0,\, x_j = 1\right\}\right| \\
&\triangleq \binom{k_1+1}{2} + g(x).
\end{aligned}$$

For $y \in U_1(x)$, either $\#(y) = (k_0 + 1, k_1)$ or $\#(y) = (k_0, k_1 + 1)$. For y of the former type, $\mathrm{vt}(y) - \mathrm{vt}(x) = g(y) - g(x)$. There are $k_1 + 1$ strings of the former type and for each these $g(y)$ takes a different value. For y of the latter type, $\mathrm{vt}(y) - \mathrm{vt}(x) = k_1 + 1 + g(y) - g(x)$. There are $k_0 + 1$ strings of the latter type and for each these $g(y)$ takes a different value. ■

The property of being locally bijective and cover preserving is preserved under composition, so

$$\begin{aligned}
&\textsc{Weight} \circ \textsc{UpDown} : (\textsc{Perm}, \mathbf{subseq}) \to (\textsc{Mahonian}, \le) \\
&\textsc{Weight} \circ \textsc{UpDown} = (\pi \mapsto |\pi| \times \mathrm{maj}(\pi))
\end{aligned}$$

is a locally bijective cover preserving map.

5.3 Varshamov-Tenengolts codes

The famous Varshamov-Tenengolts (VT) codes are closely connected to the locally bijective maps that we have been describing.

Definition 5.3.1. *Let $Y = \{(a,b) \in \mathbb{N}^2 : b \in [a+1]\}$. Define the poset (Y, Q) by the cover relations*

$$(a,b) \prec (c,d) \iff (c - a = 1).$$

Thus $U_1((a,b)) = \{(a+1,d) : d \in [a+2]\}$.

$(Y, \leq)$ is depicted in Figure 5.4.

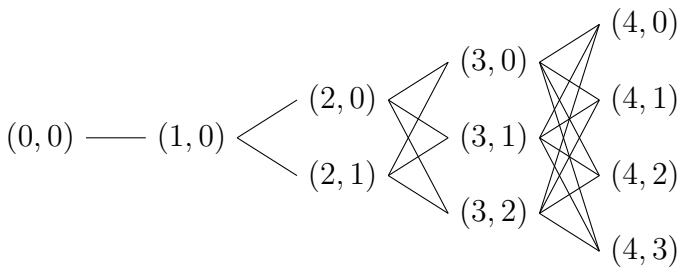

Figure 5.4: The cover relations for ranks 0 through 3 of the $(\mathrm{Y}, \leq)$ poset.

Define $\textsc{Mod} : \textsc{Mahonian} \to Y$ such that $\textsc{Mod}(a,b) = (a, b \bmod (a+1))$.

Lemma 5.3.1. $\textsc{Mod} : (\textsc{Mahonian}, \leq) \to (\mathrm{Y}, \leq)$ *is a locally bijective cover preserving map.*

Proof: The point $(a,b) \in \textsc{Mahonian}$ is covered by the points $(a+1, b+i)$ for $i \in [a+1]$. These are mapped by $\textsc{Mod}$ to distict points in Y. ■

Recall that a set $C \subset [q]^n$ is an single deletion correcting code if for all $y \in [q]^{n-1}$, $|U_1(y) \cap C| \leq 1$. C is *single deletion perfect* if $|U_1(y) \cap C| = 1$ for all y. The Varshamov-Tenengolts construction partitions $[2]^*$ into single deletion perfect codes as follows. The ith VT code of length n is $\{x \in [2]^n : \textsc{Mod}(\textsc{Weight}(x)) = (n,i)\}$. Together, Lemma 5.2.2 and Lemma 5.3.1 give the the Levenshtein decoding algorithm demonstrates for the Varshamov-Tenengolts codes. The channel receiver knows that $\textsc{Mod}(\textsc{Weight}(x)) = (n,i)$ for any transmitted string x, i.e. the transmitted is using the ith VT code. Given the received string y, compute $(n-1, j) = \textsc{Weight}(y)$. From Lemma 5.3.1, there is a unique $(n, \hat{j}) \succ (n-1, j)$ such that $\textsc{Mod}(n, \hat{j}) = (n,i)$. From Lemma 5.2.2, there is a unique $x \succ y$ such that $\textsc{Weight}(x) = (n, \hat{j})$. This is the transmitted string.

5.4 Trees

In the next two sections, we will show that UpDown is the composition of $\textsc{BTtS} : \textsc{Tree}_2 \to [2]^*$ with $\textsc{PtT} : \textsc{Perm} \to \textsc{Nothing} \sqcup \textsc{Tree}_2$. The

intermediate objects in this composition of functions are binary trees. There are several benefits to this perspective.

First, recall from Section 5.1 that we hope to learn something about the structure of the subsequence order on binary strings by finding locally bijective maps into it. The function UpDown does not provide much structural information about $([2]^*, \mathbf{subseq})$ because the set of permutations of $[n]$ is too big. In fact, the subsequence order on permutations is isomorphic to the prefix order on the n-chains of $([2]^*, \mathbf{subseq})$, whose construction we described at the end of Section 5.1. This construction works for arbitrary ranked posets, so it does not tell us anything about $([2]^*, \mathbf{subseq})$ in particular. The set of binary trees is much smaller, so the existence of a locally bijective map from that poset gives us nontrivial structural information about $([2]^*, \mathbf{subseq})$.

Second, working with binary trees will open up a number of generalizations. While the appropriate q-ary generalization of a permutation is not immediately obvious, the generalization of a binary tree to a q-ary tree is straightforward. In Section 5.7, we will define $\mathrm{TtS} : \mathrm{Tree}_q \to [q]^*$. The specialization of TtS will not turn out to be BTtS. This will give us a new Mahonian permutation statistic.

Before we discuss the new results involving q-ary trees, we need to establish some background. In this section, we formally define trees, their vertex sets, their order relation, and the function PtT.

Definition 5.4.1. *A rooted q-ary tree has q slots for children. Each of these slots either is empty or contains a q-tree:*

$$\mathrm{Tree}_q = (\mathrm{Nothing} \sqcup \mathrm{Tree}_q)^{[q]}.$$

A q-ary tree t is characterized by its set of vertices, $V(t)$. The natural partial order on q-ary trees is the inclusion order on their vertex sets: in $(\mathrm{Tree}_q, \mathbf{subtree})$, $t \leq t'$ when $V(t) \subseteq V(t')$. $(\mathrm{Nothing} \sqcup \mathrm{Tree}_2, \mathbf{subtree})$ is depicted in Figure 5.5. Now we will formalize these notions.

The vertices of a tree can be indexed by q-ary strings. In fact, we will go a step further and say that $V(t) \subset [q]^*$. In general, a vertex is synonymous with the path through the tree to that vertex. For any $t \in \mathrm{Tree}_q$, $\langle\rangle \in V(t)$ because the empty string is the root vertex. For a vertex $x \in V(t)$ with $|x| > 0$, i.e. a non-root vertex, the *parent* of x is the prefix of x of length $l - 1$. This indexing scheme is implemented in Algorithm 5.4.

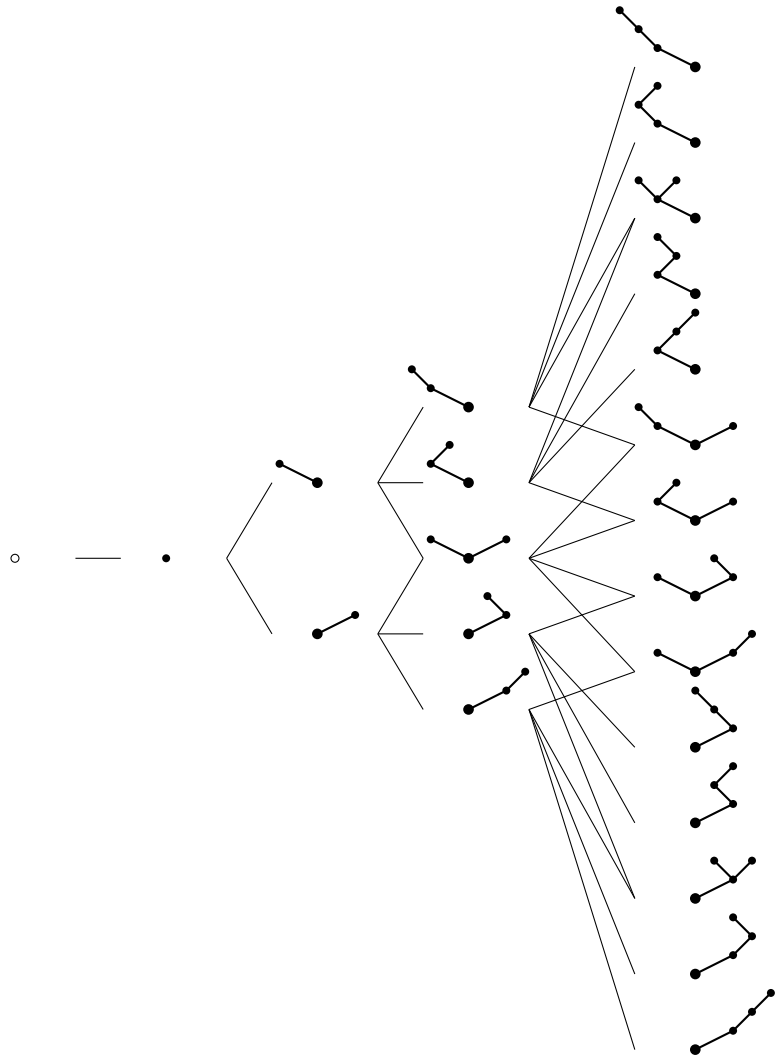

Figure 5.5: The cover relations for ranks 0 through 4 of (NOTHING ⊔ TREE$_2$, **subtree**).

Algorithm 5 Vertex Indexing

$\textsc{VertexAt} : \textsc{Tree}_q \times [q]^* \to (\textsc{Nothing} \sqcup \textsc{Tree}_q)$

$\textsc{VertexAt}(t, x) =$ **case** x **of**

$\langle\rangle : t$

$\langle i \rangle + x' :$ **case** t_i **of**

$\textsc{Nothing} : \textsc{Nothing}$

$\textsc{Tree}_q : \textsc{VertexAt}(t_i, x')$

Now we can define

$$V(t) = \{x \in [q]^* : \textsc{VertexAt}(t, x) \in \textsc{Tree}_q\}.$$

We can extend $(\textsc{Tree}_q, \textbf{subtree})$ to obtain $(\textsc{Nothing} \sqcup \textsc{Tree}_q, \textbf{subtree})$ by defining $V(\textsc{Nothing}) = \varnothing$. The bottom element of $(\textsc{Tree}_q, \textbf{subtree})$ is the tree $(i \mapsto \textsc{Nothing})$ and the bottom element of $(\textsc{Nothing} \sqcup \textsc{Tree}_q, \textbf{subtree})$ is $\textsc{Nothing}$. Thus for $(\textsc{Tree}_q, \textbf{subtree})$, the canonical rank function is $\mathrm{r}(t) = |V(t)| - 1$, the number of non-root vertices of t and for $(\textsc{Nothing} \sqcup \textsc{Tree}_q, \textbf{subtree})$ the canonical rank function is $\mathrm{r}(u) = |V(u)|$.

Observe that for trees t and t', $V(t) \cap V(t')$ is also the vertex set of a tree. Similarly, $V(t) \cup V(t')$ is a valid vertex set. Thus both $(\textsc{Tree}_q, \textbf{subtree})$ and $(\textsc{Nothing} \sqcup \textsc{Tree}_q, \textbf{subtree})$ are lattices.

There is an alternative recursive method of defining **subtree**.

Definition 5.4.2. *Define the* ***subtree*** *order relation on* $\textsc{Nothing} \sqcup \textsc{Tree}_q$ *as follows. For* $u, u' \in \textsc{Nothing} \sqcup \textsc{Tree}_q$*, let* $u \leq u'$ *iff either*

1. $u = \textsc{Nothing}$
2. $u, u' \in \textsc{Tree}_q$ *and for all* $i \in [q]$*,* $u_i \leq u'_i$ *in* ***subtree****.*

This definition is often more convenient for proofs.

5.4.1 Increasing vertex labelings, and chains

A labeling of a q-tree t assigns each vertex a unique label from the set $[|V(t)|]$, i.e. it is a bijection between $V(t)$ and $[|V(t)|]$. A labeling is increasing if each

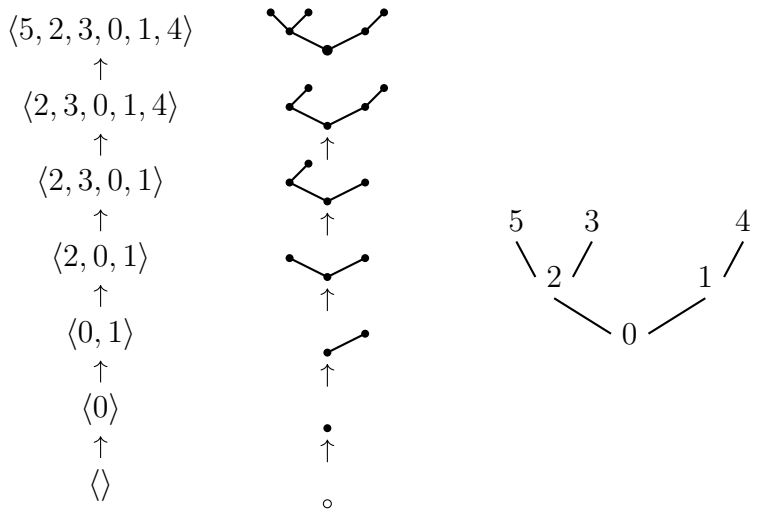

Figure 5.6: The unique skipless chain of permutations from $\langle 2, 3, 0, 1, 4\rangle$ to $\langle\rangle$, the skipless chain of trees that results from PTT, and the increasing labeling of the top tree that summarizes the chain.

nonroot vertex has a larger label than its parent. Clearly, the root vertex must be assigned 0. A q-tree t with an increasing labeling corresponds to a rooted skipless chain of q-trees that ends with t. To generate the chain, start with the empty tree and add vertices in the order of the labeling.

Lemma 5.4.1. *A $(n+1)$-vertex tree t has*

$$\frac{(n+1)!}{\prod_{v\in V(t)} |\textsc{VertexAt}(t, v)|}$$

increasing labelings, where $V(t) \subset [q]^*$ is the set of vertices of t.

Proof: We prove this by induction on the size of the tree. An empty tree has one increasing labeling, and $0!/\prod_{v\in\varnothing} |U(v)| = 1$. Let t be nonempty and let r be the root vertex of t. Let $\mathbf{k}_i \in \mathbb{N}$ be the size of t_i, the ith child of r. Let $n = \sum_{i\in[q]} \mathbf{k}_i$. The zero label must be assigned to r and there are $\binom{n}{\mathbf{k}}$ ways to distribute the other n labels among the children. Because $V(t) = r \sqcup \bigsqcup_i V(t_i)$ and $|U(r)| = n+1$, there are

$$\binom{n}{\mathbf{k}} \frac{n+1}{|U(r)|} \prod_{i\in[q]} \frac{\mathbf{k}_i!}{\prod_{v\in V(t_i)} |U(v)|} = \frac{(n+1)!}{\prod_{v\in V(t)} |U(v)|}$$

increasing labelings of t. ∎

5.4.2 Permutations to Binary Trees

If we take a binary tree with an increasing labeling and traverse the vertices in infix order, i.e. traverse the left subtree of a vertex, then the vertex itself, then the right subtree, we obtain a permutation. In fact, every permutation comes from a unique labeled binary tree. The reverse operation, producing a binary tree from a permutation, is described in Algorithm 6.

Let $S^{\underline{[n]}}$ be the set of strings of length n with symbols from S in which each symbol appears at most once. We use this notation because the cardinality of this set is the falling factorial $|S|^{\underline{n}} = \prod_{i\in[n]}(|S|-i) = |S|(|S|-1)\dots(|S|-n+1)$.

Algorithm 6 Permutation to Binary Tree

$\textsc{PtT} : \mathbb{N}^{\underline{[n]}} \to (\textsc{Nothing} \sqcup \textsc{Tree}_2)$
$\textsc{PtT}(\pi) = \textbf{case } \pi \textbf{ of}$
  $\pi = \langle\rangle : \quad \textsc{Nothing}$
  $\pi \neq \langle\rangle : \quad (t_0, t_1)$
  where
    $j \leftarrow \operatorname{argmin}_{i\in[n]} \pi_i$
    $t_0 \leftarrow \textsc{PtT}(\pi_{[0,j-1]})$
    $t_1 \leftarrow \textsc{PtT}(\pi_{[j+1,n-1]})$

The bijection between labeled binary trees and permutations is also a bijection between n-chains of binary trees and n-chains or permutations. Equivalently, PtT is locally bijective. An example of this is given in Figure 5.6.

5.5 Binary trees to binary strings

The following lemma previews our motivation for introducing $\textsc{Tree}_q$.

Lemma 5.5.1. *Let $x \in [q]^n$. Then $|U_1(x)| = (q-1)(n+1)+1$. Let t be an $(n+1)$-vertex q-tree. Then $|U_1(t)| = (q-1)(n+1)+1$.*

We postpone the proof until Section 5.6, where we will also prove a more refined statement. Because $|U_1(x)| = |U_1(t)|$ if x and t have the same rank, a

locally bijective map from q-ary trees to strings could exist. In this section, we show that BTtS, which we mentioned at the start of Section 5.4, is such a map for $q = 2$. BTtS is formally defined in Algorithm 5.5 and Figure 5.1 illustrates its effect on a chain of trees.

Algorithm 7 Binary Tree to Binary String

$\text{BTtS} : \text{Tree}_2 \to [2]^*$
$\text{BTtS}(t) = \text{BTtS}(0, t_0) + \text{BTtS}(1, t_1)$

$\text{BTtS2} : [2] \times (\text{Nothing} \sqcup \text{Tree}_2) \to [2]^*$
$\text{BTtS2}(k, u) = \textbf{case } u \textbf{ of}$
  $\text{Nothing} : \langle\rangle$
  $\text{Tree}_2 : \text{BTtS3}(k, u)$

$\text{BTtS3} : [2] \times \text{Tree}_2 \to [2]^*$
$\text{BTtS3}(k, t) = \textbf{case } k \textbf{ of}$
  $0 : \text{BTtS}(t) + \langle 0\rangle$
  $1 : \langle 1\rangle + \text{BTtS}(t)$

With BTtS defined, we can prove the claim that we made at the start of Section 5.4.

Lemma 5.5.2. *If π is a nonempty permutation, then*

$$\text{UpDown}(\pi) = \text{BTtS}(\text{PtT}(\pi)).$$

Proof: We show this by induction on the length of the input permutation. Let $\pi \in [n+1]!$ be a nonempty permutation. There are four possibilities: n can appear at the start of π, at the end of π, neither, or (if $n = 0$) both.

If $\pi = \langle 0\rangle$, then $\text{UpDown}(\pi) = \langle\rangle$, $\text{PtT}(\pi) = (i \mapsto \text{Nothing})$, and $\text{BTtS}((i \mapsto \text{Nothing})) = \langle\rangle$.

If $\pi = \langle n\rangle + \pi'$ for some nonempty π', then n is larger than every symbol in π'. Thus n is larger than the first symbol of π' and $\text{UpDown}(\pi) = \langle 1\rangle + \text{UpDown}(\pi')$. Also $t = \text{PtT}(\pi) = (\text{Nothing}, t_1)$ where $t_1 = \text{PtT}(\pi')$. Thus $\text{BTtS}(t) = \langle\rangle + (\langle 1\rangle + \text{BTtS}(t_1))$. By the induction hypothesis, we have that $\text{BTtS}(t_1) = \text{UpDown}(\pi')$, $\text{UpDown}(\pi) = \text{BTtS}(t)$.

The case $\pi = \pi' + \langle n \rangle$ is similar.

Finally, suppose $\pi = \pi' + \langle n \rangle + \pi''$ for nonempty π' and π''. Then $\textsc{UpDown}(\pi) = \textsc{UpDown}(\pi') + \langle 01 \rangle + \textsc{UpDown}(\pi'')$. Also $t = \textsc{PtT}(\pi) = (t_0, t_1)$ where $t_0 = \textsc{PtT}(\pi')$ and $t_1 = \textsc{PtT}(\pi'')$. Thus

$$\textsc{BTtS}(t) = (\textsc{BTtS}(t_0) + \langle 0 \rangle) + (\langle 1 \rangle + \textsc{BTtS}(t_1)).$$

By the induction hypothesis, we have that $\textsc{BTtS}(t_0) = \textsc{UpDown}(\pi')$ and $\textsc{BTtS}(t_1) = \textsc{UpDown}(\pi'')$. ■

Now we will show that BTtS is locally bijective. The basic idea is as follows. The mapping from t to x gives us a recursive decomposition of x into shorter strings. By Lemma 5.5.3, the ends of these shorted strings satisfy some constraints. When we add a new symbol to x to produce some $y \succ x$, at each level of the decomposition, we must choose to which string we add the new symbol. There will be only way to make these choices and preserve the constraints on the ends of the strings at each level. This constructs a unique tree $u \succ t$ that maps to y.

The following definition has significantly more flexibility than the amount required to analyze BTtS. However, this generality will be essential when we consider TtS in Section 5.7 and the proofs there will be clearer if the easier variants in this section are parallel.

Definition 5.5.1. *For $A, B \subseteq [q]$, define*

$$\Gamma(A, B) = [q]^0 \cup \bigcup_{\ell \geq 1} \{y \in [q]^\ell : \ y_0 \in A, \ y_{\ell-1} \in B\}.$$

That is, $\Gamma(A, B)$ contains the empty string as well as strings that begin with an element of A and end with an element of B.

Given this definition, the following properties of BTtS and its auxiliary functions are immediate.

Lemma 5.5.3. *Let $t \in \textsc{Tree}_2$, let $u \in \textsc{Nothing} \sqcup \textsc{Tree}_2$ and let $k, l \in [2]$.*

1. *Let $x = \textsc{BTtS}(t)$. Then $x \in \Gamma([0,1],[0,1])$.*
2. *Let $x = \textsc{TtS2}(k, u)$. Then $x \in \Gamma([k,1],[0,k])$.*
3. *Let $x = \textsc{TtS3}(k, t)$. Then $x \in \Gamma([k,1],[0,k])$.*

The following lemma is the fundamental tool behind the main results of this section and of Section 5.7, where we need its full generality.

Lemma 5.5.4. *Let $[q] \supseteq A_0 \supseteq A_1$ and $B_0 \subseteq B_1 \subseteq [q]$ such that A_1 and B_0 partition $[q]$. Let $s_0 \in \Gamma(A_0, B_0)$, let $s_1 \in \Gamma(A_1, B_1)$, and let $x = s_0 + s_1$.*

1. *Then $x \in \Gamma(A_0, B_1)$. If s_0 is nonempty, then $x \in \Gamma(A_0, B_0)$. If s_1 is nonempty, then $x \in \Gamma(A_1, B_1)$.*

2. *Let $\hat{x} \in \Gamma(A_0, B_1)$ such that $\hat{x} \succ x$. Then there are unique $\hat{s}_0 \geq s_0$ and $\hat{s}_1 \geq s_1$ such that $\hat{s}_0 \in \Gamma(A_0, B_0)$, $\hat{s}_1 \in \Gamma(A_1, B_1)$, and $\hat{x} = \hat{s}_0 + \hat{s}_1$.*

Proof:

Proof of 1: Consider the following four cases. If $s_0 = s_1 = \langle\rangle$, the claim holds trivially. If both are nonempty, again it is trivial. If $s_0 = \langle\rangle$ and s_1 is nonempty, then $x \in \Gamma(A_1, B_1)$. Because $A_0 \supseteq A_1$, $x \in \Gamma(A_0, B_1)$ as well. The fourth case is symmetric with the third.

Proof of 2: Let k be the symbol that can be inserted into x to produce $\hat{x}$. First suppose that the inserted k-symbol extends a run of k-symbols in x. Because B_0 and A_1 are disjoint, if s_0 and s_1 are both nonempty, the last symbol of s_0 differs from the the first symbol of s_1. Thus any run of k-symbols in x is fully contained in either s_0 or s_1 and there is a unique way to extend one of these segments to obtain $\hat{s}_0$ and $\hat{s}_1$. Extending a run preserves the initial and final symbols of string, so $\hat{s}_0$ and $\hat{s}_1$ have the required properties.

If the inserted symbol does not extend a run, there are unique $v, w \in [q]^*$ such that $x = v + w$ and $\hat{x} = v + \langle k \rangle + w$. If the inserted symbol is internal to some s_i, then that segment is extended by the inserted symbol to produce $\hat{s}_i$. Because the insertion is internal, $\hat{s}_i$ has the same initial and final symbols as s_i and is still in $\Gamma(A_i, B_i)$.

In the final case, $\hat{x} = s_0 + \langle k \rangle + s_1$. Then k is a member of exactly one of B_0 and A_1. If $k \in B_0$, let $\hat{s}_0 = s_0 + \langle k \rangle$. If s_0 is nonempty, $\hat{s}_0$ is clearly in $\Gamma(A_0, B_0)$. If s_0 is empty, k is the initial symbol of $\hat{x}$, so $k \in A_0$ and again $\hat{s}_0 \in \Gamma(A_0, B_0)$. The $k \in A_1$ case is analogous. ■

In particular, either $\hat{s}_0 \succ s_0$ and $\hat{s}_1 = s_1$ or $\hat{s}_0 = s_0$ and $\hat{s}_1 \succ s_1$.

The first claim of Lemma 5.5.5 establishes that BTTS is locally bijective.

Lemma 5.5.5. *Let $t \in \text{TREE}_2$, let $u \in \text{NOTHING} \sqcup \text{TREE}_2$, and let $k \in [2]$.*

1. *Let $x = \text{BTTS}(t)$ and let $\hat{x} \succ x$. Then there is a unique $\hat{t} \succ t$ such that $\text{BTTS}(\hat{t}) = \hat{x}$.*

2. *Let* $x = \text{BTTS2}(k, u)$ *and let* $\hat{x} \succ x$ *such that* $\hat{x} \in \Gamma([k, 1], [0, k])$. *Then there is a unique* $\hat{u} \succ u$ *such that* $\text{BTTS2}(k, \hat{u}) = \hat{x}$.

3. *Let* $x = \text{BTTS3}(k, t)$ *and let* $\hat{x} \succ x$ *such that* $\hat{x} \in \Gamma([k, 1], [0, k])$. *Then there is a unique* $\hat{t} \succ t$ *such that* $\text{BTTS2}(k, \hat{t}) = \hat{x}$.

Proof:

Proof of 1: We have $x = s_0 + s_1$, where $s_i = \text{BTTS}(i, t_i)$. By Lemma 5.5.3, s_0 and s_1 satisfy the conditions of Lemma 5.5.4: $A_0 = [0, 1]$, $A_1 = [0, 0]$, $B_0 = [1, 1]$, and $B_1 = [0, 1]$. From the second part of that lemma, there are unique $\hat{s}_0 \geq s_0$ and $\hat{s}_1 \geq s_1$ such that $\hat{x} = \hat{s}_0 + \hat{s}_1$ There is exactly one j such that $\hat{s}_j \succ s_j$. Now we construct the desired $\hat{t}$ as follows. From claim 2, there is a unique $\hat{t}_j \succ t_j$ such that $\text{BTTS2}(j, \hat{t}_j) = \hat{s}_j$. For $i \neq j$, let $\hat{t}_i = t_i$. Thus $\hat{t} \succ t$ and $\text{TREETOSTRING}(\hat{t}) = \hat{x}$.

Proof of 2: If u is a tree, then by claim 3, there is a unique $\hat{u} \succ u$ such that $x = \text{TTS3}(k, \hat{u})$. Otherwise, $u = \text{NOTHING}$, $x = \langle\rangle$, and $\hat{x} = \langle k \rangle$ because this is the only member of $\Gamma([k, q-1], [0, k])$ of length one. Let $\hat{u} = (i \mapsto \text{NOTHING})$ and note that $\text{TTS2}(k, \hat{u}) = \text{TTS3}(k, \hat{u}) = \langle k \rangle = x$.

Proof of 3: First consider $k = 0$. Let $y = \text{BTTS}(t)$, so $x = y + \langle 0 \rangle$. Because $\hat{x} \in \Gamma([0, 1], [0, 0])$ and $\hat{x}$ is nonempty, the final symbol of $\hat{x}$ must be 0 and we can let $\hat{x} = \hat{y} + \langle 0 \rangle$. Clearly, $\hat{y} \succ y$. From claim 1, there is a unique $\hat{t} \succ t$ such that $\text{BTTS}(\hat{t}) = \hat{y}$. We have $\text{BTTS2}(0, \hat{t}) = \hat{y} + \langle 0 \rangle$, so we are done.

The argument for $k = 1$ is analogous. ■

5.6 Compositions of q-ary strings and q-ary trees

The composition of a q-ary string is a standard concept. In this section we introduce the composition of a q-ary tree. We observe that BTOS preserves composition (see Figure 5.7). We also introduce an action of permutations of $[q]$ on q-ary trees. This modifies compositions in the expected way. The action will be essential to defining TTS in Section 5.7.

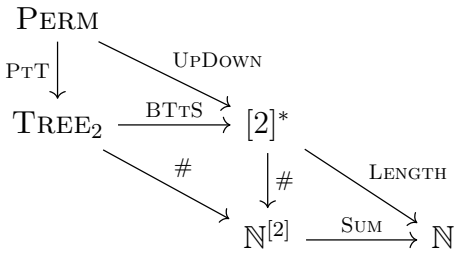

Figure 5.7: A commutative diagram summarizing the results of Sections 5.5 and 5.6.

The composition of a q-ary string is a vector in $\mathbb{N}^q$. We write the composition function as $\# : [q]^* \to \mathbb{N}^q$. The ith entry of the composition of a string is the number of i-symbols that appear in the string. Thus $\#(x)_i = |\{j \in [n] : x_j = i\}|$. Note that $\#$ is a cover preserving function from $([q]^*, \mathbf{subseq})$ to $(\mathbb{N}^q, \leq)$.

Algorithm 8 Compositions

$\textsc{TreeMap} : ([q] \to (\textsc{Nothing} \sqcup \textsc{Tree}_q \to \mathrm{X})) \to (\textsc{Tree}_q \to \mathrm{X}^{[q]})$
$\textsc{TreeMap}(f) = t \mapsto (x_0, \ldots, x_{q-1})$
  where for $i \in [q]$
    $x_i = f(i)(t_i)$

$\# : \textsc{Tree}_q \to \mathbb{N}^{[q]}$
$\#(t) = \sum_{i \in [q]} \mathbf{k}_i$
  where $\mathbf{k} = \textsc{TreeMap}(\#')(t)$

$\#' : [q] \to (\textsc{Nothing} \sqcup \textsc{Tree}_q \to \mathbb{N}^{[q]})$
$\#'(i)(u) =$ **case** u **of**
  $\textsc{Tree}_q : \mathbf{1}_{\{i\}} + \#(u)$
  $\textsc{Nothing} : \mathbf{0}$

We can also define a composition for nonempty q-trees. We will abuse notation and use $\#$ for this composition as well. The ith entry of the composition of a tree is the number of vertices that have a nonempty i-child, or equivalently, the number of vertices that are the ith child of their parent. Thus

$$\#(t)_i = |\{x \in V(t) \setminus \langle\rangle : x_{-1} = i\}|.$$

Like the string composition, the tree composition is a cover preserving function to $(\mathbb{N}^q, \leq)$. Algorithm 8 gives a recursive definition of composition. This formulation, which uses the TreeMap function and the auxilary function $\#'$, will be useful for proving properties of BTtS and TtS.

Now that we have defined the composition, we can state and prove a stronger version of Lemma 5.5.1.

Lemma 5.6.1. *Let* $\mathbf{k} \in \mathbb{N}^q$ *and let* $n = \sum_{i \in [q]} \mathbf{k}_i$. *Let* $x \in [q]^n$ *such that* $\#(x) = \mathbf{k}$. *Then* $U_1(x)$ *contains exactly* $n + 1 - \mathbf{k}_i$ *strings with composition* $\mathbf{1}_{\{i\}} + \mathbf{k}$. *Let* t *be an* $(n+1)$*-vertex tree such that* $\#(t) = \mathbf{k}$. *Then* $U_1(t)$ *contains exactly* $n + 1 - \mathbf{k}_i$ *trees with composition* $\mathbf{1}_{\{i\}} + \mathbf{k}$.

Proof: If $x_j = i$, then the same string is produced by inserting a new i-symbol before or after x_j. There are $n + 1$ possible places to insert a new i-symbol and $\mathbf{k}_i$ equivalances between places, so there are $n + 1 - \mathbf{k}_i$ unique supersequences.

Each of the $n + 1$ vertices has a slot for a child of type i and $\mathbf{k}_i$ of these slots are filled, so there are $n + 1 - \mathbf{k}_i$ ways to add a new vertex of type i. ■

We can also show that $\mathrm{BTtS} : \mathrm{Tree}_2 \to [2]^*$ preserves compositions: $\# \circ \mathrm{BTtS} = \#$.

Lemma 5.6.2. *Let* $t \in \mathrm{Tree}_2$, *let* $u \in \mathrm{Nothing} \sqcup \mathrm{Tree}_2$ *and let* $k \in [q]$.

1. $\#(\mathrm{BTtS}(t)) = \#(t)$.
2. $\#(\mathrm{BTtS2}(k, u)) = \#'(k)(u)$.
3. $\#(\mathrm{BTtS3}(k, t)) = \#(t) + \mathbf{1}_{\{k\}}$.

Proof:

Proof of 1: This follows from claim 2 and the fact that for $x, y \in [2]^*$, $\#(x + y) = \#(x) + \#(y)$.

Proof of 2: If $u \in \mathrm{Tree}_2$, $\mathrm{BTtS2}(k, u) = \mathrm{BTtS3}(k, u)$ and this follows from claim 3. If $u = \mathrm{Nothing}$, we have $\#(\mathrm{BTtS2}(k, \mathrm{Nothing})) = \#(\langle\rangle) = \mathbf{0}$ and $\#'(k)(\mathrm{Nothing}) = \mathbf{0}$.

Proof of 3: This follows from claim 1, $\#(x + \langle 0\rangle) = \#(x) + \mathbf{1}_{\{0\}}$, and $\#(\langle 1\rangle + x) = \#(x) + \mathbf{1}_{\{1\}}$. ■

5.6.1 Permutations actions

In order to define TtS in the next section, we need to introduce permutation actions on trees and strings. Algorithm 9 defines PC (permute composition), PS (permute string), and PT (permute tree). In each case, the permutation acts on the alphabet $[q]$. Note that PC, PS, and PT all treat their argument as a function an compose it with either π or π^{-1}. Recall that $\mathbb{N}^{[q]} = [q] \to \mathbb{N}$, $[q]^* = \bigsqcup_{i \in \mathbb{N}} ([i] \to [q])$, and $\textsc{Tree}_q = [q] \to (\textsc{Nothing} \sqcup \textsc{Tree}_q)$.

Algorithm 9 Permuting the alphabet

$\mathrm{PC} : ([q] \to [q]) \to (\mathbb{N}^{[q]} \to \mathbb{N}^{[q]})$
$\mathrm{PC}(\pi) = \mathbf{k} \mapsto \mathbf{k} \circ \pi^{-1}$

$\mathrm{PS} : ([q] \to [q]) \to ([q]^* \to [q]^*)$
$\mathrm{PS}(\pi) = x \mapsto \pi \circ x$

$\mathrm{PT} : ([q] \to [q]) \to (\textsc{Tree}_q \to \textsc{Tree}_q)$
$\mathrm{PT}(\pi) = t \mapsto \mathrm{PT2}(\pi) \circ t \circ \pi^{-1}$

$\mathrm{PT2} : ([q] \to [q]) \to ((\textsc{Nothing} \sqcup \textsc{Tree}_q) \to (\textsc{Nothing} \sqcup \textsc{Tree}_q))$
$\mathrm{PT2}(\pi) = u \mapsto$
  case u **of**
    $\textsc{Nothing} : \textsc{Nothing}$
    $\textsc{Tree}_q : \mathrm{PT}(\pi)(u)$

These permutation actions interact with the group operation for permutations in the required way:

$$\mathrm{PS}(\pi) \circ \mathrm{PS}(\pi') = \mathrm{PS}(\pi \circ \pi')$$
$$\mathrm{PT}(\pi) \circ \mathrm{PT}(\pi') = \mathrm{PT}(\pi \circ \pi')$$
$$\mathrm{PC}(\pi) \circ \mathrm{PC}(\pi') = \mathrm{PC}(\pi \circ \pi').$$

The compositions of strings and trees are modified by permutations in the

expected way:

$$\# \circ \mathrm{PS}(\pi) = \mathrm{PC}(\pi) \circ \#$$
$$\# \circ \mathrm{PT}(\pi) = \mathrm{PC}(\pi) \circ \#.$$

5.7 q-ary trees to q-ary strings

In this section, we prove the main new result of this chapter.

Theorem 5.7.1. *There is a locally bijective cover preserving map* TtS : $\textsc{Tree}_q \to [q]^*$.

TtS is defined in Algorithm 10 along with several auxiliary functions. It also depends on the string and tree permutation functions, PS and PT, which were defined in Algorithm 9, and TreeMap, which was defined in Algorithm 8.

We can generalize Lemma 5.5.4 to longer concatenations of strings.

Lemma 5.7.1. *Let $s_i \in [q]^*$ for $i \in [m]$ such that $s_i \in \Gamma(A_i, B_i)$. Suppose that for all $i \in [m-1]$, $A_i \supseteq A_{i+1}$, $B_i \subseteq B_{i+1}$, and B_i and A_{i+1} partition $[q]$. Let $x = s_0 + \ldots + s_{m-1}$.*

1. *Then $x \in \Gamma(A_0, B_{m-1})$.*
2. *Let $\hat{x} \in \Gamma(A_0, B_{m-1})$ such that $\hat{x} \succ x$. Then there is a unique tuple of strings $(\hat{s}_0, \ldots, \hat{s}_{m-1}) \in ([q]^*)^m$ such that $\hat{s}_0 + \ldots + \hat{s}_{m-1} = \hat{x}$ and for all i, $\hat{s}_i \geq s_i$ and $\hat{s}_i \in \Gamma(A_i, B_i)$.*

Proof: Let $u_0 = s_0$ and let $u_{j+1} = u_j + s_{j+1}$ (so $u_{q-1} = x$). Inductively applying the first part of Lemma 5.5.4, $u_j \in \Gamma(A_0, B_j)$. Thus u_j and s_{j+1} satisfy the conditions of the second part of Lemma 5.5.4: $A_0 \supseteq A_{j+1}$, $B_j \subseteq B_{j+1}$, and B_j and A_{j+1} partition $[q]$. Suppose we have $\hat{u}_{j+1} \succ u_{j+1}$. From Lemma 5.5.4, we obtain $\hat{u}_j \geq u_j$ and $\hat{s}_{j+1} \geq s_{j+1}$ such that $\hat{u}_j + \hat{s}_{j+1} = \hat{u}_{j+1}$. Let $\hat{u}_{q-1} = \hat{x}$. Then by induction we obtain $\hat{s}_0, \ldots, \hat{s}_{q-1}$ with the desired properties. ∎

Definition 5.7.1. *For $a, b \in [q]$, define the circular interval $[a, b] \subseteq [q]$ as follows:*

$$[a,b] = \begin{cases} \{i \in [q] : a \leq i \wedge i \leq b\} & a \leq b \\ \{i \in [q] : a \leq i \vee i \leq b\} & a > b. \end{cases}$$

Algorithm 10 Tree to String

$\textsc{TtS} : \textsc{Tree}_q \to [q]^*$
$\textsc{TtS} = \ \textsc{Concat} \circ \textsc{TreeMap}(\textsc{TtS2})$

$\textsc{TtS2} : [q] \to (\textsc{Nothing} \sqcup \textsc{Tree}_q \to [q]^*)$
$\textsc{TtS2}(k) = u \mapsto \textbf{case } u \textbf{ of}$
  $\textsc{Tree}_q : \textsc{TtS3}(k)(u)$
  $\textsc{Nothing} : \langle\rangle$

$\textsc{TtS3} : [q] \to (\textsc{Tree}_q \to [q]^*)$
$\textsc{TtS3}(k) = \mathrm{PS}(\pi) \circ \textsc{Concat} \circ \textsc{TreeMap}(\textsc{TtS4}(\pi(k))) \circ \mathrm{PT}(\pi)$
  where $\pi = (i \mapsto q - 1 - i)$

$\textsc{TtS4} : [q] \to ([q] \to (\textsc{Nothing} \sqcup \textsc{Tree}_q \to [q]^*))$
$\textsc{TtS4}(l)(k) = u \mapsto \textbf{case } u \textbf{ of}$
  $\textsc{Tree}_q : \textsc{TtS5}(l,\ k)(u)$
  $\textsc{Nothing} : \textbf{case } (l, k) \textbf{ of}$
    $l = k : \ \langle k \rangle$
    $l \neq k : \ \langle\rangle$

$\textsc{TtS5} : [q] \times [q] \to (\textsc{Tree}_q \to [q]^*)$
$\textsc{TtS5}(l,\ k) = \textbf{case}$
  $l < k : \ \mathrm{PS}(i \mapsto i + l + 1) \circ \textsc{TtS3}(k - l - 1) \circ \mathrm{PT}(i \mapsto i - l - 1)$
  $l > k : \ \mathrm{PS}(i \mapsto i + l) \circ \textsc{TtS3}(k - l) \circ \mathrm{PT}(i \mapsto i - l)$
  $l = k : \ t \mapsto \langle k \rangle + \textsc{TtS}(t) + \langle k \rangle$

$\textsc{Concat} : ([q]^*)^q \to [q]^*$
$\textsc{Concat}(s_0, \ldots, s_{q-1}) = s_0 + \ldots + s_{q-1}$

Note that $(i \mapsto i + k)[a, b] = [a + k, b + k]$ and $(i \mapsto -1 - i)[a, b] = [-1 - b, -1 - a]$.

To prove that TtS is locally bijective, we first show that the outputs of TtS and each of its auxiliary function are members of $\Gamma(A, B)$ where A and B are circular intervals. Lemma 5.7.2 contains the precise version of this statement. Then, in the proof of Lemma 5.7.3, we will use two string decompositions that meet the conditions of Lemma 5.7.1. We will go through these decompositions here to give a high level understanding of the proof.

In the first decomposition, let $A_i = [i, q-1]$ and let $B_i = [0, i]$. Then the

end symbols of x and s_i are from the following sets:

		s_0	$[0, q-1], [0, 0]$
		$\vdots$	$\vdots$
x	$[0, q-1], [0, q-1]$	s_i	$[i, q-1], [0, i]$
		s_{i+1}	$[i+1, q-1], [0, i+1]$
		$\vdots$	$\vdots$
		s_{q-1}	$[q-1, q-1], [0, q-1]$.

Note that B_i and A_{i+1} partition $[q]$ for all i.

In the second decomposition, we start with $x \in \Gamma([0, l], [l, q-1])$ such that at least one symbol of x is an l. We let $x = u + s_l + v$, $u = s_0 \dots s_{l-1}$ and $v = s_{l+1} \dots s_{q-1}$. The restrictions on the end symbols of u, v and each s_i are as follows:

		u	$[0, l]$,$[l+1, l-1]$
x	$[0, l], [l, q-1]$	s_l	$[l, l]$,$[l, l]$
		v	$[l+1, l-1]$,$[l, q-1]$
		s_0	$[0, l]$,$[l+1, 0]$
		$\vdots$	$\vdots$
u	$[0, l], [l+1, l-1]$	s_i	$[i, l]$,$[l+1, i]$
		$\vdots$	$\vdots$
		s_{l-1}	$[l-1, l]$,$[l+1, l-1]$
		s_{l+1}	$[l+1, l-1]$,$[l, l+1]$
		$\vdots$	$\vdots$
v	$[l+1, l-1], [l, q-1]$	s_j	$[j, l-1]$,$[l, j]$
		$\vdots$	$\vdots$
		s_{q-1}	$[q-1, l-1]$,$[l, q-1]$.

The decompositions $u = s_0 \dots s_{l-1}$ and $v = s_{l+1} \dots s_{q-1}$ meet the conditions of Lemma 5.7.1, but the whole sequence $x = s_0 + \dots + s_{q-1}$ does not. In particular $[0, l] \not\supseteq [l+1, l-1]$, so the first symbol of v is less restricted than the first symbol of x, and $[l+1, l-1] \not\subseteq [l, q-1]$, so the last symbol of u is less restricted than the last symbol of x. Thus we also require that s_l be

nonempty, which means it must contains at least one l. With this restriction, the first symbol of x cannot come from v and the last symbol of x cannot come from u, so $x \in \Gamma([0, l], [l, q-1])$ is achieved.

5.7.1 Properties of TtS

Remark 5.7.1. *Lemmas 5.7.2, 5.7.3, and 5.7.4 each contain statements about* TtS*,* TtS2*,* TtS3*,* TtS4*, and* TtS5*. For each lemma, we will prove these five statements by mutual induction on the size of the trees. This parallels the mutually recursive definitions of these functions.*

The claims 2 and 4 for the input Nothing *are the base cases of the induction. The proofs of claims 1 and 3 depend respectively on claims 2 and 4 for strictly smaller trees and* Nothing*. The proofs of claims 2 and 4 depend on claims 3 and 5 for trees of the same size. The proof of claim 5 depends on claims 1 and 3 for tree of the same size. Thus assuming all five claims for trees of size strictly less than n, we can prove claims 1 and 3, then claim 5, then claims 2 and 4 for trees of size n.*

Lemma 5.7.2. *Let $t \in \text{Tree}_q$, let $u \in \text{Nothing} \sqcup \text{Tree}_q$ and let $k, l \in [q]$.*

1. *Let $x = \text{TtS}(t)$. Then $x \in \Gamma([0, q-1], [0, q-1])$.*
2. *Let $x = \text{TtS2}(k)(u)$. Then $x \in \Gamma([k, q-1], [0, k])$.*
3. *Let $x = \text{TtS3}(k)(t)$. Then $x \in \Gamma([k, q-1], [0, k])$.*
4. *Let $x = \text{TtS4}(l)(k)(u)$.*
   (a) *If $l < k$, then $x \in \Gamma([k, l], [l+1, k])$.*
   (b) *If $l = k$, then $x \in \Gamma([k, k], [k, k])$.*
   (c) *If $l > k$, then $x \in \Gamma([k, l-1], [l, k])$.*
5. *Let $x = \text{TtS5}(l, k)(t)$.*
   (a) *If $l < k$, then $x \in \Gamma([k, l], [l+1, k])$.*
   (b) *If $l = k$, then $x \in \Gamma([k, k], [k, k])$.*
   (c) *If $l > k$, then $x \in \Gamma([k, l-1], [l, k])$.*

Proof: Remark 5.7.1 explains the structure of the induction.

Proof of 1: Because $\Gamma([0,q-1],[0,q-1]) = [q]^*$, this is trivially true.

Proof of 2: Either $x = \textsc{TtS3}(k)(t)$ or $x = \langle\rangle$. In former case, $x \in \Gamma([k,q-1],[0,k])$ from claim 3. The empty string is also in $\Gamma([k,q-1],[0,k])$.

Proof of 3: Let $t' = \mathrm{PT}(j \mapsto -1-j)t$ and let $l = -1-k$. Then $x = \mathrm{PS}(j \mapsto -1-j)(s_0 + \ldots + s_{q-1})$, where $s_i = \textsc{TtS4}(l)(i)(t'_i)$ and $t'_i = t_{-1-i}$. From claim 4, $s_i \in \Gamma([i,l],[l,i])$. By the first part of Lemma 5.7.1, $s_0 + \ldots + s_{l-1} \in \Gamma([0,l],[l,l-1])$ and $s_{l+1} + \ldots + s_{q-1} \in \Gamma([l+1,l],[l,q-1])$. Note that $[l,l-1] = [l+1,l] = [q]$. However, from claim 4 s_l is nonempty. By two applications of the first part of Lemma 5.5.4, $(s_0 + \ldots + s_{l-1}) + s_l + (s_{l+1} + \ldots + s_{q-1}) \in \Gamma([0,l],[l,q-1])$ Thus $x \in \Gamma([k,q-1],[0,k])$.

Proof of 4: If $u = \textsc{Nothing}$, then $x = \langle\rangle$ or $x = \langle k\rangle$. We have $\langle\rangle \in \Gamma(A,B)$ for all A and B and $\langle k\rangle \in \Gamma([k,k],[k,k])$. If $u \in \textsc{Tree}_q$, $x = \textsc{TtS5}(l,k)(t)$ and the claim follows from claim 5.

Proof of 5a: Let $t' = \mathrm{PT}(i \mapsto i-l-1)t$. We have $x = \mathrm{PS}(i \mapsto i+l+1)(\textsc{TtS3}(k-l-1,t'))$. From claim 3, we have $\textsc{TtS3}(k-l-1)(t') \in \Gamma([k-l-1,q-1],[0,k-l-1])$ with composition $\#(t') + \mathbf{1}_{\{k-l-1\}}$. Thus $x \in \Gamma([k,l],[l+1,k])$ with composition $\#(t) + \mathbf{1}_{\{k\}}$.

Proof of 5b: In this case, $x = \langle k\rangle + y + \langle k\rangle$, where $y = \textsc{TtS}(t)$, so the first claim is immediate. From claim 1, $\#(y) = \#(t)$.

Proof of 5c: This is completely analogous to claim 5a. ∎

Lemma 5.7.3. *Let $t \in \textsc{Tree}_q$, let $u \in \textsc{Nothing} \sqcup \textsc{Tree}_q$ and let $k,l \in [q]$.*

1. *Let $x = \textsc{TtS}(t)$, and let $\hat{x} \succ x$. Then there is a unique $\hat{t} \succ t$ such that $\textsc{TtS}(\hat{t}) = \hat{x}$.*

2. *Let $x = \textsc{TtS2}(k)(u)$. Let $\hat{x} \succ x$ such that $\hat{x} \in \Gamma([k,q-1],[0,k])$. Then there is a unique $\hat{u} \succ t$ such that $\textsc{TtS2}(k)(\hat{u}) = \hat{x}$.*

3. *Let $x = \textsc{TtS3}(k)(t)$. Let $\hat{x} \succ x$ such that $\hat{x} \in \Gamma([k,q-1],[0,k])$. Then there is a unique $\hat{t} \succ t$ such that $\textsc{TtS3}(k)(\hat{t}) = \hat{x}$.*

4. *Let $x = \textsc{TtS4}(l)(k)(u)$. Let $\hat{x} \succ x$ such that*

   (a) *$\hat{x} \in \Gamma([k,l],[l+1,k])$ if $l < k$,*

   (b) *$\hat{x} \in \Gamma([k,k],[k,k])$ if $l = k$,*

   (c) *and $\hat{x} \in \Gamma([k,l-1],[l,k])$ if $l > k$.*

Then there is a unique $\hat{u} \succ u$ *such that* $\textsc{TtS4}(l)(k)(\hat{u}) = \hat{x}$.

5. *Let* $x = \textsc{TtS5}(l,k)(t)$. *Let* $\hat{x} \succ x$ *such that*

   (a) $\hat{x} \in \Gamma([k,l],[l+1,k])$ *if* $l < k$,

   (b) $\hat{x} \in \Gamma([k,k],[k,k])$ *if* $l = k$,

   (c) *and* $\hat{x} \in \Gamma([k,l-1],[l,k])$ *if* $l > k$.

   Then there is a unique $\hat{t} \succ t$ *such that* $\textsc{TtS5}(l,k)(\hat{t}) = \hat{x}$.

Proof: Remark 5.7.1 explains the structure of the induction.

Proof of 1: We have $x = s_0 + s_1 + \ldots + s_{q-1}$ where $s_i = \textsc{TtS2}(i)(t_i)$. By part 2 of Lemma 5.7.2, for each i, $s_i \in \Gamma([i,q-1],[0,i])$. Thus we satisfy the conditions of Lemma 5.7.1 and there is a unique $(\hat{s}_0,\ldots,\hat{s}_{q-1}) \in ([q]^*)^q$ such that $\hat{s}_0 + \ldots \hat{s}_{q-1} = \hat{x}$ and, for all i, $\hat{s}_i \geq s_i$ and $\hat{s}_i \in \Gamma([i,q-1],[0,i])$. There is exactly one j such that $\hat{s}_j \succ s_j$. From claim 2, there is a unique $\hat{t}_j \succ t_j$ with the desired properties. Let $\hat{t}_i = t_i$ for $i \neq j$, so $\hat{t} \succ t$ and $\textsc{TtS}(\hat{t}) = \hat{x}$.

Proof of 2: If u is a tree, then by claim 3, there is a unique $\hat{u} \succ u$ such that $x = \textsc{TtS3}(k,\hat{u})$. Otherwise, $u = \textsc{Nothing}$, $x = \langle\rangle$, and $\hat{x} = \langle k\rangle$ because this is the only member of $\Gamma([k,q-1],[0,k])$ of length one. Let $\hat{u} = (i \mapsto \textsc{Nothing})$ and note that $\textsc{TtS2}(k)(\hat{u}) = \textsc{TtS3}(k)(\hat{u}) = \langle k\rangle = x$.

Proof of 3: Let $t' = \textsc{PT}(j \mapsto -1-j)(t)$ and let $l = -1-k$. Then $\textsc{PS}(j \mapsto -1-j)(x) = s_0 + \ldots + s_{q-1}$, where $s_i = \textsc{TtS4}(l)(i)(t'_i) = \textsc{TtS4}(l)(i)(t_{-1-i})$. Let $u = s_0 + \ldots s_{l-1}$ and let $v = s_{l+1} + \ldots + s_{q-1}$. Then $u \in \Gamma([0,l],[l+1,l-1])$, $s_l \in \Gamma([l,l],[l,l])$, and $u \in \Gamma([l+1,l-1],[l,q-1])$. By two applications of the second part of Lemma 5.5.4, there are unique $\hat{u} \geq u$, $\hat{s}_l \geq s_l$, and $\hat{v} \geq v$ such that $\hat{u}+\hat{s}_l+\hat{v} = \textsc{PS}(j \mapsto -1-j)(\hat{x})$ and $\hat{u}$, $\hat{s}_l$, and $\hat{v}$ have the same end-symbol restrictions as u, s_l and v, respectively. If $\hat{u} \succ u$, then by Lemma 5.7.1, there is a unique $(\hat{s}_0,\ldots,\hat{s}_{l-1}) \in ([q]^*)^l$ such that $\hat{s}_0 + \ldots \hat{s}_{l-1} = \hat{u}$ and for all i, $\hat{s}_i \geq s_i$ and $\hat{s}_i \in \Gamma([i,l-1],[l,i])$. If $\hat{v} \succ v$, then by Lemma 5.7.1, there is a unique $(\hat{s}_{l+1},\ldots,\hat{s}_{q-1}) \in ([q]^*)^{q-l-1}$ such that $\hat{s}_{l+1} + \ldots \hat{s}_{q-1} = \hat{v}$ and for all i, $\hat{s}_i \geq s_i$ and $\hat{s}_i \in \Gamma([i,l],[l+1,i])$. Thus there is exactly one j such that $\hat{s}_j \succ s_j$. From claim 4, there is a unique $\hat{t}_j \succ t_j$ with the desired properties. Let $\hat{t}_i = t_i$ for $i \neq j$, so $\hat{t} \succ t$ and $\textsc{TtS3}(k)(\hat{t}) = \hat{x}$.

Proof of 4: If u is a tree, then by claim 5, there is a unique $\hat{u} \succ u$ such that $x = \textsc{TtS5}(l,k)(\hat{u})$. If $u = \textsc{Nothing}$ and $l \neq k$, then $x = \langle\rangle$ and $\hat{x} = \langle k\rangle$ because this is the only member of $\Gamma([k,j],[j+1,k])$ of length one.

If $u = \textsc{Nothing}$ and $l = k$, then $x = \langle k \rangle$ and $\hat{x} = \langle kk \rangle$ because this is the only member of $\Gamma([k,k],[k,k])$ of length two. In either case, let $\hat{u} = (i \mapsto \textsc{Nothing})$. Note that for $l \neq k$, $\textsc{TtS4}(l)(k)(\hat{u}) = \textsc{TtS5}(l,k)(\hat{u}) \overset{(a)}{=} \langle k \rangle = x$ and $\textsc{TtS4}(k)(k)(\hat{u}) = \textsc{TtS5}(k,k)(\hat{u}) \overset{(b)}{=} \langle kk \rangle = x$. Here, both (a) and (b) follow from part 5 of Lemma 5.7.2 and $\#(\hat{u}) = \mathbf{0}$.

Proof of 5: If $l = k$, we have $x = \langle k \rangle + x' + \langle k \rangle$ where $x' = \textsc{TtS}(t)$. Because $\hat{x} \in \Gamma([k,k],[k,k])$, $\hat{x} = \langle k \rangle + \hat{x}' + \langle k \rangle$ for some $\hat{x}' \succ x'$. By claim 1, there is a unique $\hat{t} \succ t$ such that $\textsc{TtS}(\hat{t}) = \hat{x}'$.

If $l < k$, x and $\hat{x}$ are in $\Gamma([k,l][l+1,k])$. Let $\pi = (i \mapsto i - l - 1)$, $x' = \mathrm{PS}(\pi)(x)$, $\hat{x}' = \mathrm{PS}(\pi)(\hat{x})$, $t' = \mathrm{PT}(\pi)(t)$. Then x' and $\hat{x}'$ are in $\Gamma([k-l-1,q-1][0,k-l-1])$, $\hat{x}' \succ x'$, and $\textsc{TtS3}(k-l-1)(t') = x'$. From claim 3, there is a unique $\hat{t}' \succ t'$ such that $\textsc{TtS3}(k-l-1)(\hat{t}') = \hat{x}'$ Let $\hat{t} = \mathrm{PT}(\pi^{-1})(\hat{t}')$, so $\hat{x} = \textsc{TtS5}(l,k)(\hat{t})$ as required.

The case $l > k$ is analogous. ■

Theorem 5.7.1 follows immediately from the first claim.

TtS also preserves compositions.

Lemma 5.7.4. *Let $t \in \textsc{Tree}_q$, let $u \in \textsc{Nothing} \sqcup \textsc{Tree}_q$ and let $k, l \in [q]$.*

1. $\# \circ \textsc{TtS} = \#$.
2. $\# \circ \textsc{TtS2}(k) = \#'(k)$.
3. $\#(\textsc{TtS3}(k)(t)) = \#(t) + \mathbf{1}_{\{k\}}$.
4. *Let $x = \textsc{TtS4}(l)(k)(u)$.*
   (a) *If $l = k$, then $\#(x) = \#'(k)(u) + \mathbf{1}_{\{k\}}$.*
   (b) *If $l \neq k$, then $\#(x) = \#'(k)(u)$.*
5. *Let $x = \textsc{TtS5}(l,k)(t)$.*
   (a) *If $l = k$, then $\#(x) = \#(t) + 2 \cdot \mathbf{1}_{\{k\}}$.*
   (b) *If $l \neq k$, then $\#(x) = \#(t) + \mathbf{1}_{\{k\}}$.*

Proof: This proof follows the same inductive structure as the previous two lemmas, so we will only sketch the two main ideas.

If we have $\#(f(i)(u)) = \#'(i)(u)$ for some $f : [q] \to (\textsc{Nothing} \sqcup \textsc{Tree}_q) \to [q]^*$, then

$$\begin{aligned}\#(\textsc{Concat}(\textsc{TreeMap}(f)(t))) &= \sum_{i\in[q]} \#(f(i)(t_i)) \\ &= \sum_{i\in[q]} \#'(i)(t_i) \\ &= \#(t).\end{aligned}$$

If we have $\# \circ g = \#$ for some $g : \textsc{Tree}_q \to [q]^*$, then

$$\begin{aligned}\# \circ \mathrm{PS}(\pi^{-1}) \circ g \circ \mathrm{PT}(\pi) &= \mathrm{PC}(\pi^{-1}) \circ \# \circ g \circ \mathrm{PT}(\pi) \\ &= \mathrm{PC}(\pi^{-1}) \circ \# \circ \mathrm{PT}(\pi) \\ &= \mathrm{PC}(\pi^{-1}) \circ \mathrm{PC}(\pi) \circ \# \\ &= \#.\end{aligned}$$

This generalizes to handle g satisfying the relation $(\# \circ g)(t) = \#(t) + \mathbf{1}_{\{k\}}$. ■

5.8 Conclusion

We have found locally bijective maps from $\textsc{Tree}_q \to [q]^*$ for all q. Because $(\textsc{Tree}_q, \leq)$ is a lattice, by Lemma 5.1.3 these maps quantify the failure of $([q]^*, \mathbf{subseq})$ to be a lattice. Is $(\textsc{Tree}_q, \leq)$ the smallest lattice for which locally bijective maps to $([q]^*, \mathbf{subseq})$ exist? More precisely, these maps give upper bounds on the number of maximal elements of $D(x) \cap D(y)$. Can corresponding lower bounds be found?

CHAPTER 6

REFERENCES

[1] D. Cullina and N. Kiyavash, "Combinatorial channels from partially ordered sets," in *2015 IEEE Information Theory Workshop (ITW)*, Apr. 2015, pp. 1–5.

[2] D. Cullina and N. Kiyavash, "Generalized Sphere-Packing Bounds on the Size of Codes for Combinatorial Channels," *IEEE Transactions on Information Theory*, vol. 62, no. 8, pp. 4454–4465, Aug. 2016.

[3] D. Cullina and N. Kiyavash, "An improvement to Levenshtein's upper bound on the cardinality of deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3862–3870, July 2014.

[4] A. A. Kulkarni and N. Kiyavash, "Nonasymptotic upper bounds for deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 59, no. 8, pp. 5115–5130, 2013.

[5] D. B. West, *Introduction to Graph Theory*. Prentice Hall Upper Saddle River, 2001, vol. 2.

[6] J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," in *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*. IEEE, 1996, pp. 627–636.

[7] L. Lovasz, "On the Shannon capacity of a graph," *IEEE Transactions on Information Theory*, vol. 25, no. 1, pp. 1–7, Jan. 1979.

[8] C. Shannon, "The zero error capacity of a noisy channel," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 8–19, 1956.

[9] M. Rosenfeld, "On a problem of C. E. Shannon in graph theory," *Proceedings of the American Mathematical Society*, vol. 18, no. 2, pp. 315–319, Apr. 1967.

[10] M. Dalai, "Lower bounds on the probability of error for classical and classical-quantum channels," *Information Theory, IEEE Transactions on*, vol. 59, no. 12, pp. 8027–8056, 2013.

[11] P. Erdos, C. Ko, and R. Rado, "Intersection theorems for systems of finite sets," *The Quarterly Journal of Mathematics*, vol. 12, no. 1, pp. 313–320, 1961.

[12] R. Ahlswede and L. H. Khachatrian, "The complete intersection theorem for systems of finite sets," *European Journal of Combinatorics*, vol. 18, no. 2, pp. 125–136, 1997.

[13] D. Cullina and N. Kiyavash, "Generalized sphere-packing upper bounds on the size of codes for combinatorial channels," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*. IEEE, 2014, pp. 1266–1270.

[14] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, 1966, pp. 707–710.

[15] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 766–769, 1984.

[16] R. Varshamov and G. Tenengolts, "Codes which correct single asymmetric errors," *Avtomatika i Telemekhanika*, vol. 26, pp. 288–292, 1965.

[17] R. Varshamov, "On an arithmetic function with an application in the theory of coding," *Doklady Akademii nauk SSSR*, vol. 161, no. 3, pp. 540–543, 1965.

[18] D. Cullina, A. Kulkarni, and N. Kiyavash, "A coloring approach to constructing deletion correcting codes from constant weight subgraphs," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, July 2012, pp. 513–517.

[19] V. I. Levenshtein, "Bounds for deletion/insertion correcting codes," in *IEEE International Symposium on Information Theory Proceedings*, 2002, p. 370.

[20] V. I. Levenshtein, "Elements of coding theory," *Diskretnaya Matematika i Matematicheskie Voprosy Kibernetiki*, pp. 207–305, 1974.

[21] L. Calabi and W. E. Hartnett, "Some general results of coding theory with applications to the study of codes for the correction of synchronization errors," *Information and Control*, vol. 15, no. 3, pp. 235–249, 1969.

[22] D. Hirschberg, "Bounds on the number of string subsequences," in *Combinatorial Pattern Matching*, 1999, pp. 115–122.

[23] T. G. Swart and H. C. Ferreira, "A note on double insertion/deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 269–273, 2003.

[24] H. Mercier, M. Khabbazian, and V. Bhargava, "On the number of subsequences when deleting symbols from a string," *IEEE Transactions on Information Theory*, vol. 54, no. 7, pp. 3279–3285, 2008.

[25] Y. Liron and M. Langberg, "A characterization of the number of subsequences obtained via the deletion channel," in *IEEE International Symposium on Information Theory Proceedings*, 2012, pp. 503–507.

[26] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, Mar. 1963.

[27] W. Feller, "Stirling's formula," in *An Introduction to Probability Theory and Its Applications*, 3rd ed. John Wiley & Sons, 1968, vol. 1, pp. 52–54.

[28] L. Dolecek and V. Anantharam, "Repetition error correcting sets: Explicit constructions and prefixing methods," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2120–2146, 2010.

[29] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," in *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 814–818.

[30] N. Kashyap and G. Zmor, "Upper bounds on the size of grain-correcting codes," *arXiv preprint arXiv:1302.6154*, 2013. [Online]. Available: http://arxiv.org/abs/1302.6154

[31] R. Gabrys, E. Yaakobi, and L. Dolecek, "Correcting grain-errors in magnetic media," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2013, pp. 689–693.

[32] S. Buzaglo, E. Yaakobi, T. Etzion, and J. Bruck, "Error-correcting codes for multipermutations," 2013. [Online]. Available: http://authors.library.caltech.edu/36946/

[33] A. Fazeli, A. Vardy, and E. Yaakobi, "Generalized sphere packing bound," *arXiv:1401.6496 [cs, math]*, Jan. 2014. [Online]. Available: http://arxiv.org/abs/1401.6496

[34] N. J. A. Sloane, *Challenge Problems: Independent Sets in Graphs*. [Online]. Available: http://neilsloane.com/doc/graphs.html

[35] A. Mazumdar, A. Barg, and N. Kashyap, "Coding for high-density recording on a 1-d granular magnetic medium," *Information Theory, IEEE Transactions on*, vol. 57, no. 11, pp. 7403–7417, 2011.

[36] A. Sharov and R. M. Roth, "Bounds and constructions for granular media coding," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2011, pp. 2343–2347.

[37] D. Foata and M.-P. Schtzenberger, "Major index and inversion number of permutations," *Mathematische Nachrichten*, vol. 83, no. 1, pp. 143–159, 1978.

[38] E. Babson and E. Steingrmsson, "Generalized permutation patterns and a classification of the Mahonian statistics," *Sm. Lothar. Combin*, vol. 44, no. B44b, pp. 547–548, 2000.

[39] R. J. Clarke, E. Steingrmsson, and J. Zeng, "New Euler-Mahonian permutation statistics," *Sm. Lothar. Combin. 35 (1995), Art. B35c, approx. 29 pp.(electronic)*, 1996.

[40] M. C. Wilson, "An interesting new Mahonian permutation statistic," *The Electronic Journal of Combinatorics*, vol. 17, no. 1, p. R147, 2010.

[41] M. Skandera, "An Eulerian partner for inversions," *Sm. Lothar. Combin*, vol. 46, p. B46d, 2001.

[42] P. A. MacMahon, *Combinatory Analysis*. Courier Corporation, 1916, vol. 2.

[43] G. Viennot, "Maximal chains of subwords and up-down sequences of permutations," *Journal of Combinatorial Theory, Series A*, vol. 34, no. 1, pp. 1–14, 1983.

[44] N. G. De Bruijn, "Permutations with given ups and downs," *Nieuw Arch. Wisk*, vol. 18, no. 3, pp. 61–65, 1970.