FLEXIBLE SCHEMES AND BEYOND: EXPERIMENTAL ENUMERATION OF PATTERN AVOIDANCE CLASSES

By

YONAH BIERS-ARIEL

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Mathematics

Written under the direction of

Doron Zeilberger

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2020

ABSTRACT OF THE DISSERTATION

Flexible Schemes and Beyond: Experimental Enumeration of Pattern Avoidance Classes

By YONAH BIERS-ARIEL

Dissertation Director: Doron Zeilberger

This thesis demonstrates several applications of experimental methods to the enumeration of pattern-avoiding permutations. First, we introduce flexible schemes – a new extension of enumeration schemes. We show how a computer can find flexible schemes and use them to count permutations. We establish sufficient conditions for the existence of finite flexible schemes, and in particular show that they exist whenever finite enumeration schemes or regular insertion encodings do. Next, we combine enumeration schemes with structural arguments to give a new algorithm for counting 1342-avoiding permutations. We find a recurrence for the number of permutations avoiding four dashed patterns, and finally generalize Claude Lenormand's "raboter" sequence. We have implemented all the algorithms described here in Maple and provide links to the Maple code in each section.

Acknowledgements

This thesis would not have been possible without the support, first and foremost, of my advisor Doron Zeilberger. He is responsible for my introduction to the wide world of experimental math, and most of the problems appearing here were originally his suggestions. His enthusiasm and encouragement have made my PhD a truly rewarding experience.

I would also like to thank Vince Vatter for suggesting the problem of uniting enumeration schemes and the insertion encoding. The first section of this dissertation would not exist without the papers he pointed me to and the techniques I found in them. I am also grateful to him as well as to Bhargav Narayanan and Swastik Kopparty for their service on my dissertation committee.

The experimental mathematician is only as powerful as the computers they use. In writing this thesis, I made frequent use of the Amarel computing cluster, and so I am grateful to Rutgers' Office of Advanced Research Computing for creating and maintaining this important resource.

No one can spend all their time researching, and I'd further like to thank all my fellow graduate students who made my time at Rutgers so much fun. There are too many people in this category to list them all, but I'd especially like to thank Keith Frankston, Justin Semonsen, Abigail Raz, and Matthew Russell.

Finally, I'd like to thank my wife, Michelle Flores, who moved to New Jersey so that I could pursue this opportunity and who feeds me cookies when I get angry trying to debug my code.

Table of Contents

Chapter 1

Introduction

What an exciting time to be a combinatorialist! Nearly a century of advances in computation has made mathematical experimentation easier than ever. It has inspired new questions and given new tools to answer them. It has even caused us to rethink what an answer is [28]. However, while our computers are able assistants, they have not yet outgrown the need for their fleshy, fallible masters. There is still a great deal for us humans to do in collaboration with them.

This collaboration, which we refer to as experimental math, takes many forms. Its most common and fundamental form happens when mathematicians computationally generate many, many objects and then make conjectures about the objects' properties based on this huge supply of data. Once we know what to prove, proving it is often not so difficult. The most obvious instance of this sort of experimental math in this thesis is in Chapter 4, but in a looser sense every result here was conceived of through experimentation.

This collaboration can also be reversed, with a human supplying the conjecture and the start of a proof, and the computer finishing it. Often these proofs involve a human showing that a minimal counterexample must have a specific form, and the computer showing that no objects of that form are counterexamples. This sort of experimental math famously proved the four color theorem [4][5].

A more specialized kind of experimental math occurs when a computer is responsible both for conjecturing and proving its own theorems. A human explains to a computer what sort of object would constitute an answer, and then sets the computer to work. This sort of experimental math can be as simple as interpolating a degree-d polynomial through $d + 1$ points, but can be applied to much more difficult problems as well.

Zeilberger's algorithm for finding hypergeometric sum identities [18] is a classic example of this. Chapters 2 and 5 of this thesis are concerned with this sort of experimental math, and contain additional examples of it.

While experimental math is a powerful methodology with many deep results, this thesis focuses mainly on its applications to pattern avoidance. The modern study of pattern avoidance began in *The Art of Computer Programming* where Knuth asked which permutations can be sorted using only a single stack, and how many of them are there [16]? The field has grown tremendously since then (see [25] for a detailed survey), but our focus is on Knuth's classic question: given some class of permutations, how many are there?

More is known about this question than we could possibly describe in this introduction, but we will cover a few highlights. The answer to Knuth's original question is that the stack-sortable permutations are precisely those that avoid the pattern 231, and they are counted by the Catalan numbers [16]. In fact, Knuth showed that for any pattern σ of length 3, the number of length-n permtuations avoiding σ is the n^{th} Catalan number. Of the seven symmetry classes of length 4 patterns, six have known enumeration sequences [15][26][12], while the 1324-avoiding permutations are so far unenumerated. Wikipedia is an excellent reference for the enumeration sequences and (when known) generating functions of individual classes [27].

Not only do we have enumeration sequences for many permutation classes, we have some asymptotic knowledge of how every class grows. Letting $\mathrm{Av}(\sigma; n)$ be the number of length-n permutations avoiding σ, we know that $\lim_{n\to\infty} |\mathrm{Av}(\sigma; n)|^{1/n} = c_\sigma$ [17] and that c_σ is generally exponentially large in the pattern length [14].

The first part of this thesis is concerned with experimental strategies for enumerating pattern avoiding permutations. We introduce a few existing strategies including enumeration schemes and the insertion encoding, and then develop a new method with the power of both of these. Additionally, we provide the (to our knowledge) first experimental conditions to establish the nonexistence of an enumeration scheme.

The second part of this thesis looks at two individual permutation classes. The first class consists of the permutations avoiding the pattern 1342 which were famously

enumerated by Bóna [12] using a bijection with a certain class of labeled trees. Here, we prove a recurrence which we can justify directly in terms of the permutation class' avoidance properties. This recurrence uses elements both from enumeration schemes and from the structure paradigm for automatic enumeration. Next, we consider the permutations avoiding the dashed patterns $1-32-4, 1-42-3, 2-31-4, 2-41-3$ for which we prove a recurrence conjectured by Callan [22]. Finally, we leave the world of pattern avoidance to generalize Claude Lenormand's "Raboter" sequence.

The material in Section 4 has been previously published in the Journal of Integer Sequences [10]. We also intend to publish the material in Section 2. All other material appears only in this thesis.

Chapter 2

Automatic Avoidance Class Enumeration

2.1 Background

In the last twenty years, an increasing amount of research has focused on the experimental enumeration of permutation classes. Rather than the traditional approach of considering a single avoidance class (or family of avoidance classes), these algorithms seek to input any set of patterns and then (hopefully) return some certificate which gives a polynomial-time algorithm to generate the enumeration sequence of the permutations which avoid these patterns. By polynomial-time, we mean that the time to find the n^{th} term of the sequence is a polynomial in n; what Wilf calls a *p-solution* [28].

The first of these were enumeration schemes, developed by Zeilberger [29] and expanded by Vatter [23]. The idea is to separate permutations into groups by their prefixes and then attempt to reduce these classes to simpler groups by deleting prefix elements. Of course, we can ask about much more than just classical patterns in permutations. The dashed patterns created by Babson and Steingrímsson [6] can be counted by schemes of Baxter and Pudwell [7], and the pattern-avoiding words introduced by Burstein [13] can be counted by schemes of Pudwell [19] and Shar and Zeilberger [20].

This thesis is concerned with both classical and dashed patterns, but only in permutations. Vatter's enumeration schemes are described in greater detail in Section 2.1.2; Baxter and Pudwell's schemes, while not described explicitly, provide some of the key ideas of Section 2.5.

The other enumeration algorithm which we are interested in is the insertion encoding developed by Albert, Linton, and Ruškuc [3]. The strategy here is to consider a finite automaton which moves between states, each state specifying which elements are

already included in the permutation and where new elements will be inserted. It is described in greater detail in section 2.1.3.

Both of these algorithms, along with the new enumeration scheme that we propose in this paper, follow the paradigm of building a permutation element by element and finding rules to reduce partially-completed permutations to simpler ones. Another paradigm began with Albert and Atkinson [1] and has been developed further by, among others, Bean, Gudmundsson, and Ulfarsson [8]. The general idea is to decompose pattern-avoiding permutations into component structures which can more easily be counted. While we don't describe this algorithm in detail, and it isn't needed for our original work in Section 2.2, it is a powerful strategy, and is able to enumerate all classes of pattern-avoiding permutations with a regular insertion encoding, all classes where the number of elements of length n is polynomial in n, and all classes avoiding at least six patterns of length four.

The object of this section is to answer a question posed by Vatter [23]: Is there an automatic enumeration algorithm that applies to all permutation classes with finite enumeration schemes and all classes with regular insertion encodings? Here we provide an algorithm to produce what we will call flexible schemes. Finite flexible schemes exist for every class with a finite enumeration scheme or a regular insertion encoding (and many classes with neither). Like traditional schemes, flexible schemes do not provide generating functions – indeed, they exist for avoidance classes which are believed not to have any D-finite generating function [2] – but they do provide polynomial-time enumeration of their permutation classes.

We begin by presenting two existing strategies for the automatic enumeration of pattern-avoiding permutations: traditional enumeration schemes and the insertion encoding. In Section 2.2, we describe how flexible schemes differ from traditional ones, and present an automatic way to find them. In Section 2.2.4 we show that a finite flexible scheme exists whenever either a finite traditional scheme or regular insertion encoding does. In Section 2.3, we present some of the successes we have had on specific avoidance classes, and in Section 2.5 we adapt flexible schemes to count permutations avoiding vincular patterns.

2.1.1 Definitions

In general, we use standard definitions for permutation patterns and related objects. A *permutation* is some reordering of the numbers in $[n] = \{1, 2, \ldots, n\}$. Permutations are written as a string of ordered values (like 24513), and the length-0 permutation is written as $\emptyset$. A string s of k distinct elements of $[n]$ can be turned into a permutation by taking its *reduction*, the unique permutation of length k whose elements occur in the same order as s.

A permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ *contains* a pattern $p = p_1p_2\ldots p_k$ if there exists a sequence $i_1, i_2, \ldots, i_k$ such that $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$ reduces to p. In this case, we call $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$ an *occurrence* of p in σ; if there is no occurrence of p in σ, then σ *avoids* p. The following example illustrates these definitions.

Example 2.1.1. The permutation 24513 contains the pattern 132 because 253 reduces to 132 (and hence is an occurrence of 132). However, 24513 avoids 321 because it does not contain three elements in decreasing order.

When discussing Zeilberger's enumeration schemes, we need to categorize permutations by their prefixes. A *prefix* of a permutation is the reduction of one of the permutation's initial segments. When we turn to Vatter's enumeration schemes and then the new ones introduced in this thesis, we will instead categorize permutations by their downfixes. A *downfix* of a permutation consists of the permutation's elements which lie below a particular value kept in their proper order.

Example 2.1.2. The permutation 24513 has six prefixes: $\emptyset, 1, 12, 123, 2341$, and 24513. It also has six downfixes: $\emptyset, 1, 21, 213, 2413$, and 24513.

When discussing Vatter's enumerations schemes, we will also need to categorize permutations by their *gap vectors*. A permutation $\sigma_1\sigma_2\ldots\sigma_n$ with downfix $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_l}$ has corresponding gap vector $[i_1 - 1, i_2 - i_1 - 1, \ldots, i_l - i_{l-1} - 1, n - i_l]$. We are particularly interested in gap vectors whose components are larger than those of some other vector, and so we say a gap vector $\mathbf{g} = [g_1, g_2, \ldots, g_{l+1}]$ *satisfies* a gap condition $\mathbf{h} = [h_1, h_2, \ldots, h_{l+1}]$ if $g_i \geq h_i$ for all $1 \leq i \leq l+1$. When $\mathbf{g}$ satisfies $\mathbf{h}$, we write $\mathbf{g} \succeq \mathbf{h}$.

Example 2.1.3. Consider the permutation 24513 and its downfix 21. It has corresponding gap vector $[0, 2, 1]$ which satisfies $[0, 1, 1]$ and $[0, 2, 0]$, but not $[1, 1, 1]$. Notice if a permutation has length n and its downfix has length l, then the corresponding gap vector must have length $l + 1$, and its elements must sum to $n - l$.

We will often want to talk about the set of permutations avoiding some set of patterns B with downfix π and corresponding gap vector $\mathbf{g}$. We call this set $Z(B, \pi, \mathbf{g})$. When this set is nonempty, we say that $\mathbf{g}$ is *viable* for B and π and when it is empty we say that $\mathbf{g}$ is *nonviable* for B and π.

Sometimes, we also want to talk about the set of permutations with downfix π and corresponding gap vector $\mathbf{g}$ without worrying about what patterns they avoid. We call this set $Y(\pi, \mathbf{g})$.

Example 2.1.4. The gap vector $[2, 1, 0]$ is viable for pattern set $\{123\}$ and downfix 12 because

$$Z(\{123\}, 12, [2, 1, 0]) = \{35142, 43152, 45132, 53142, 54132\}.$$

On the other hand, the gap vector $[0, 0, 1]$ is nonviable. When we don't worry about avoiding 123, we find that

$$Y(12, [2, 1, 0]) = \{34152, 35142, 43152, 45132, 53142, 54132\}.$$

The key operation at the heart of enumeration schemes is that of deleting a downfix element to yield a simpler permutation. When a permutation with downfix π and gap vector $\mathbf{g}$ has the i^{th} element of its downfix deleted, it yields a permutation with downfix $d_i(\pi)$ and gap vector $d_i(\mathbf{g})$. Precisely, we define $d_i(\pi_1\pi_2 \dots \pi_k)$ to be the reduction of $\pi_1 \dots \pi_{i-1}\pi_{i+1} \dots \pi_k$, and $d_i([g_1, g_2, \dots, g_{k+1}]) = [g_1, \dots, g_{i-1}, g_i + g_{i+1}, g_{i+2}, \dots, g_{k+1}]$.

We also need to consider the *refinements* of π, that is all the permutations formed by inserting the element $|\pi| + 1$ somewhere in π. When we refine π, we also need to change $[g_1, g_2, \dots g_{k+1}]$ by deleting 1 from g_i (representing the element which is now in the downfix), inserting a new element g' between g_i and g_{i+1}, and splitting the mass of g_i between it and g' (representing the elements that were between the π_i and π_{i+1}, but

are now between π_{i+1} and π_{i+2}). Precisely, we let

$$f_i(\pi_1\pi_2\dots\pi_k) = \pi_1\dots\pi_i(k+1)\pi_{i+1}\dots\pi_k, \text{ and}$$

$$f_{i,j}([g_1, g_2, \dots, g_{k+1}]) = [g_1, \dots, g_{i-1}, j, g_i - j - 1, g_{i+1}, \dots, g_{k+1}].$$

Example 2.1.5. Let $\pi = 24513$ and $\mathbf{g} = [1, 2, 1, 2, 1, 2]$. Then $d_1(\pi) = 3412$ and $d_1(\mathbf{g}) = [3, 1, 2, 1, 2]$. Similarly, $f_2(\pi) = 264513$ and $f_{2,1}(\mathbf{g}) = [1, 1, 0, 1, 2, 1, 2]$.

2.1.2 Enumeration Schemes

The history and basic idea of enumeration schemes have already been discussed; here we illustrate the schemes with an extended example due to Vatter in [23].

Example 2.1.6. Suppose we want an algorithm to enumerate the permutations which avoid the two patterns 1342 and 1432. We begin by considering all permutations with the downfix 1 (i.e. all permutations). Our goal is to find elements of the downfix such that if an occurrence of a forbidden pattern uses the element, then there is a different occurrence of a forbidden pattern which does not use the element. When we find such an element, we can delete it, secure in the knowledge that the resulting permutation contains a forbidden pattern if and only if the original one did. These elements are called *reversely deletable* (in [29]) or *ES-reducible* (in [23]). We follow the more recent source and call them ES-reducible. When a downfix has an ES-reducible element, we will also call the downfix itself ES-reducible. When a downfix is not ES-reducible, we say that it is *ES-irreducible*.

One can easily find permutations with the downfix 1 which contain either the pattern 1342 or 1432 but which contain neither pattern when the 1 is removed (for example, 1342 itself is such a permutation). Therefore, we consider the *refinements* of 1, that is all the length-2 permutations for which 1 is a downfix. These refinements are 12 and 21. Looking at 21, we see that the second element is ES-reducible; if any permutation uses the 1 in an occurrence of 1342 or 1432, the pattern could just as easily be formed using the 2 instead. Turning to 12, though, we find that neither element is ES-reducible; indeed we can form forbidden subpatterns which use both of them.

We are saved, though, by considering gap vectors. If a permutation has at least two elements between the 1 and the 2 (i.e. if its gap vector corresponding to 12 satisfies $[0, 2, 0]$) , then either a 1342 or 1432 pattern must occur. On, the other hand, if it has fewer than two elements between the 1 and the 2 (i.e. if the gap vector fails to satisfy $[0, 2, 0]$), then 2 is ES-reducible. Therefore, we either replace 12 with a shorter downfix, or else we can ignore it entirely.

In Section 2.2.2, we explain precisely how a computer could record these rules, and how it would use them to generate terms of the permutation class' enumeration sequence.

2.1.3 Insertion Encoding

The insertion encoding encodes a process in which a permutation is built up by inserting its elements from smallest to largest, but only in designated slots. These slots, indicated by $\diamond$, are the only places in which a new element can be added, and they must end up containing an element. Each permutation is constructed by a unique sequence of insertions. The following example shows how 24513 is constructed.

Example 2.1.7.

$$\diamond$$

$$\diamond 1 \diamond$$

$$2 \diamond 1 \diamond$$

$$2 \diamond 13$$

$$24 \diamond 13$$

$$24513$$

Albert et al. in [3] describe this sequence by recording at each step which slot an element is inserted into, numbering them beginning with 1 on the left. They also record where in its slot each new element is added. An m represents inserting an element in the middle of a slot (leaving $\diamond$s on both sides), an r represents inserting it on the right of a slot (leaving a $\diamond$ on its left), an l represents inserting it on the left of a slot (leaving

a $\diamond$ on the right), and an f represents filling the slot (leaving no $\diamond$ at all). So, 24513 is recorded as $m_1 l_1 f_2 l_1 f_1$.

When the strings that represent valid permutations in the avoidance class form a regular language, we say the insertion encoding is regular, and the avoidance class can be efficiently enumerated. The authors were able to precisely characterize classes with regular insertion encodings: they are the classes which contain only finitely many *vertical alternations*, i.e. permutations where each odd element is greater than each even element or vice-versa. Later, Vatter showed in [24] that the these classes could also be characterized as those for which any sufficiently long partial permutation has an insertion encoding reducible element, i.e. an element which can be removed without affecting the set of insertion sequences that could finish the permutation.

2.2 Flexible Schemes

2.2.1 Motivation

In this section, we introduce a new idea to extend traditional enumeration schemes and enable them to count many more avoidance classes. It is motivated by the following question: What if there is a downfix and gap condition which do not guarantee that all permutations with that downfix and satisfying that condition contain a forbidden pattern, but which do allow some element of the downfix to be deleted?

If a downfix has such an element for every possible gap vector, then we always are able to reduce it to a simpler downfix. Of course, this is not very useful if it is hard to tell, for a given gap vector, which element is the reducible one. However, if a downfix contains such an element for every possible gap vector, and if we can determine which element that is by comparing the gap vector to finitely many gap conditions, then we can efficiently reduce the downfix. Because we allow the element being deleted to change based on the gap vector, we call such a downfix *Flexible Scheme reducible* (or FS-reducible).

The following example shows how this lets us count the avoidance class $\mathrm{Av}(1423, 2314)$.

Example 2.2.1. Note that a (moderately) quick calculation with Vatter's Maple package `WILFPLUS` reveals that 3214 and 4321 are both irreducible using traditional schemes. In fact, $k \dots 21$ is ES-irreducible for all k (this can be proven using the main theorem of Section 2.4), and no finite scheme exists.

However, 321 is FS-reducible, and thus so are its refinements 3214 and 4321. Suppose that $\mathbf{g}$ is a gap vector satisfying the condition [0,1,0,0], i.e. $\mathbf{g}$ has length 4 and $g_2 \geq 1$. It is not true that any permutation with prefix 321 and gap vector $\mathbf{g}$ must contain a forbidden pattern; for instance, $\mathbf{g} = [0, 1, 0, 0]$ yields the permutation 3421. It is true, however, that as long as this gap condition is satisfied, the third element of the downfix is deleteable, i.e. $|Z(\{1423, 2314\}, 321, [g_1, g_2, g_3, g_4])| = |Z(\{1342, 3124\}, d_3(321), d_3([g_1, g_2, g_3, g_4]))|$.

To see this, suppose that a permutation σ has downfix 321 and gap vector $[g_1, g_2, g_3, g_4]$ (thus, $\sigma_{g_1+1} = 3, \sigma_{g_1+g_2+2} = 2$, and $\sigma_{g_1+g_2+g_3+3} = 3$). By way of contradiction, suppose that $g_2 \geq 1$, but the 1 in σ is not FS-reducible. In other words σ contains either a 1423 or 2314 pattern when the 1 is present, but contains no forbidden pattern when it is removed. Suppose that the 1 participates in a 1423 pattern. When the 1 is removed, either the 2 or 3 can fill in for it in the 1423 pattern, so a forbidden pattern still occurs.

The more difficult case is if 1 participates in a 2314 pattern. Obviously, 1 serves as the 1 in this pattern. We consider three subcases based on which element of σ serves as the 2. First, suppose that 2 serves as the 2. To complete the pattern, we find σ_i, σ_j with $3 < \sigma_i < \sigma_j$ such that $g_1 + g_2 + 2 < i < g_1 + g_2 + g_3 + 3 < j$. Choose k such that $g_1 + 1 < k < g_1 + g_2 + 2$ (we know this is possible since $g_2 \geq 1$). If $\sigma_k < \sigma_j$, then $3\sigma_k 2\sigma_j$ form a 2314 pattern, while if $\sigma_k > \sigma_j$, then $3\sigma_k\sigma_i\sigma_j$ for a 1423 pattern.

In the second case, some σ_l with $l \leq g_1 + 1$ serves as the 2. As before, complete the pattern, this time by finding σ_i, σ_j such that $l < i < g_1 + g_2 + g_3 + 3 < j$. If $i < g_1 + g_2 + 2$ (in other words, if σ_i occurs before 2) then $\sigma_l\sigma_i 2\sigma_j$ is a 2314 pattern. Otherwise, $g_1 + g_2 + 2 < i$, and we are back in case 1.

In the third case, some σ_l with $g_1 + 1 < l < g_1 + g_2 + g_3 + 3$ serves as the 2. Complete the pattern by finding σ_i, σ_j with $l < \sigma_i < g_1 + g_2 + g_3 + 3 < \sigma_j$, and note that $3\sigma_i 1\sigma_j$

is also a 2314 pattern and so we are back in case 2.

We have now dealt with the difficult case when $g_2 \geq 1$, and we are left with the easier case when $g_2 = 0$. Let σ be a permutation with a 321 downfix and gap vector $[g_1, 0, g_3, g_4]$. Since 2 and 3 are consecutive both in terms of their position in σ and their values, but neither 1423 nor 2314 contain decreasing elements which are consecutive in both senses, it follows that no occurrence of a forbidden pattern uses both 2 and 3. Further, 3 can be replaced in any forbidden pattern with 2 (or vice-versa) and so 3 is deletable.

As the previous example shows, verifying one of these reducibility rules by hand is largely a matter of making simple arguments for many tedious special cases. We will see in Section 2.2.3 how a computer can do all this work for us.

2.2.2 Counting with Schemes

In this section, we describe flexible schemes from the perspective of a computer, beginning with the data structure in which they are stored and then showing that a scheme allows for polynomial-time enumeration of a permutation class.

A scheme is a collection of *replacement rules* which allow any sufficiently long permutation downfix to be replaced by a shorter one. As much as possible, we model these rules after Zeilberger's *VZ-triples* defined in [30].

A replacement rule is a pair $[\pi, \mathbf{H}]$, where π is a downfix and $\mathbf{H} = \big[[\mathbf{h_1}, r_1], \ldots, [\mathbf{h_k}, r_k]\big]$ is a list of pairs. In each pair, $\mathbf{h_k}$ is a gap condition and r_k is either 0 (if every gap vector satisfying $\mathbf{h_k}$ is nonviable for π) or a positive integer i (if the i^{th} downfix element is deletable whenever $\mathbf{h_k}$ is satisfied). If $\mathbf{H}$ is an empty list, then π is not FS-reducible, otherwise the final $\mathbf{h_k}$ should be the list of all zeros so that every possible gap vector satisfies some $\mathbf{h_k}$.

In order for a scheme to be complete, it should have replacement rules for downfixes of every length up to d (called the *depth* of the downfix), and none of the rules for length d downfixes should have empty $\mathbf{H}$, i.e. every downfix of length d should be FS-reducible. It's worth noting that it's not strictly necessary to have a replacement

rule for every downfix; if some downfix π has a downfix of its own that already has a replacement rule, we can use that to reduce π, and we don't need a rule specifically for π. On the other hand, in this scenario we definitely can find a replacement rule for π, and so we might as well.

Example 2.2.2. Suppose we are trying to avoid the pattern 123. We obtain the scheme

$$\left\{\Big[[],[]\Big],\Big[[1],[]\Big],\Big[[1,2],\big[[[0,0,1],0],[[0,0,0],2]\big]\Big],\Big[[2,1],\big[[[0,0,0],2]\big]\Big]\right\}$$

of depth is 2.

This scheme is interpreted as follows. If a permutation has the empty downfix, we do not know how to reduce it, and so we consider its refinement 1. But, we also don't know how to reduce 1, so we next consider both refinements of 1: 12 and 21. If a permutation has the downfix 12, we check to see if it satisfies the gap condition [0,0,1], i.e. if it has at least one element following the 2. If so, the permutation cannot possibly avoid 123 (this is indicated by the 0 following the gap condition). If not, we check to see if it satisfies the gap condition [0,0,0]; since every permutation always does, we find that the second element of the downfix is FS-reducible. Finally, if a permutation has the downfix 21, we check to see if it satisfies the gap condition [0,0,0]; again it must, and so the second element of the downfix is FS-reducible.

We call the algorithm which follows these rules `FindTerm`. This algorithm inputs a scheme S, a downfix π, and a gap vector $\mathbf{g}$, and it outputs the number of elements in $Y(\pi, \mathbf{g})$ which are in the permutation class that S describes. Pseudocode for `FindTerm` is given in Algorithm 1.

Given a scheme S for a permutation class, we find the number of permutations of length n which avoid it as $\texttt{FindTerm}(S, [], [n])$. We claim that the runtime is polynomial in n.

Proposition 2.2.3. *Let S be a scheme with depth d. Then, $\texttt{FindTerm}(S, [], [n])$ runs in $O(n^{d+2})$ time.*

Proof. Note that in every recursive call, $\mathbf{g}$ has no more than $d+1$ elements, each of which is in $[0, n-1]$. Similarly, π has no more than d elements. Based on these (extremely

Algorithm 1: `FindTerm`

```
Find r ∈ S such that r[1] = π
H := r[2]
if H = [] then
    if g = [0, 0, ..., 0] then
        return (1)
    output := 0
    for i := 1 to length (g) + 1 do
        for j := 0 to g[i] − 1 do
            output += (FindTerm(S, f_i(π), f_{i,j}(g)))
    return (output)
FindFirst h ∈ H such that h[1] ⪯ g
i := h[2]
if i = 0 then
    return (0)
else
    return (FindTerm(S, d_i(π), d_i(g)))
```

rough) bounds, we conclude that we need to call `FindTerm` recursively at most $d!n^{d+1}$ times. Within each call, if $\mathbf{H} = []$, we loop first over all $i \in [1, \texttt{length}(\mathbf{g}) + 1]$ ($\leq d + 2$ values), and then over all $j \in \mathbf{g}[i]$ ($\leq n$ values). On the other hand, if $\mathbf{H} \neq []$, we need to look through some fixed number of possible $\mathbf{h}$s (making $\leq n$ comparisons each time) to find the first one with $\mathbf{h}[1] \preceq \mathbf{g}$, and then make a single recursive call. □

Notice that this will not run in polynomial time if S is not finite. In particular, suppose that S contains just one downfix of length m for each positive integer m. Then, we can enumerate the permutations of length n in the class described by S by using the partial scheme of consisting of the n downfixes with length $\leq n$. While there are very few downfixes, each one comes with many gap vectors; in particular there are $O\big((\frac{n}{2})^{\frac{n}{2}}\big)$ gap vectors corresponding to the downfix of length $\frac{n}{2}$, and so we must call `FindTerm` recursively exponentially many times.

2.2.3 Automatic Scheme Discovery

Of course, a scheme is only useful if we can find it in the first place. Given some downfix π, the idea is to test every possible gap condition $\mathbf{h}$ to see if it either guarantees

a forbidden pattern or has some element r which is FS-reducible for all gap vectors satisfying $\mathbf{h}$. Once we find such an $\mathbf{h}$, we know how to reduce π whenever its gap vector satisfies $\mathbf{h}$, so for future $\mathbf{h'}$, we only need to find r which is FS-reducible for all gap vectors satisfying $\mathbf{h'}$ and failing to satisfy $\mathbf{h}$ (or have $\mathbf{h'}$ guarantee a forbidden pattern). The following proposition, based on Proposition 6.2 from [23], lets a computer verify that r is FS-reducible for all gap vectors satisfying $\mathbf{h}$ and not satisfying $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$ by checking only a finite number of gap vectors. Note that $||B||_\infty$ is the length of the largest element of B and $||\mathbf{g}||_1$ is the sum of the elements of $\mathbf{g}$.

Proposition 2.2.4. *Let B be a set of forbidden patterns, π be a downfix, and $\mathbf{h}, \mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$ be gap conditions. Suppose that*

$$|Z(B; \pi; \mathbf{g})| = |Z(B; d_r(\pi); d_r(\mathbf{g}))|$$

for all $\mathbf{g}$ with $||\mathbf{g}||_1 \leq ||B||_\infty - 1 + ||\mathbf{h}||_1$ which satisfy $\mathbf{h}$ but fail to satisfy any of $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$. Then the equality holds for all $\mathbf{g}$ which satisfy $\mathbf{h}$ but fail to satisfy any of $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$.

Proof. Choose some $\mathbf{g}$ satisfying $\mathbf{h}$ and no other $\mathbf{h_i}$. It is clear that $|Z(B; \pi; \mathbf{g})| \leq |Z(B; d_r(\pi); d_r(\mathbf{g}))|$ always holds since each permutation in $Z(B; \pi; \mathbf{g})$ can have the downfix element r removed to yield a distinct permutation in $Z(B; d_r(\pi); d_r(\mathbf{g}))$.

To see that $\geq$ also holds, fix $\mathbf{g}$ satsifying $\mathbf{h}$ but not $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$. Our strategy is to show that if there is any $\sigma \notin Z(B; \pi; \mathbf{g})$ such that removing the downfix element r gives a permutation in $Z(B; d_r(\pi); d_r(\mathbf{g}))$, then there is such a σ corresponding to a small gap vector.

Consider a permutation $\sigma \in Y(\pi, \mathbf{g})$ but not in $Z(B; \pi; \mathbf{g})$; that is a permutation with downfix π and corresponding gap vector $\mathbf{g}$ which contains a pattern of B. Suppose further that removing the r in the downfix of σ eliminates the forbidden pattern so that the resulting permutation is in $Z(B; d_r(\pi); d_r(\mathbf{g}))$. Choose an occurrence of a pattern $b \in B$ in σ, and suppose it uses the elements $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ (where $k \leq ||B||_\infty$). Form σ' by removing from σ all the elements which do not participate in this pattern; since at least one element from the pattern (r) came from the downfix, we have at most $||B||_\infty - 1$ elements not in the downfix.

Up until this point, the proof has been identical to the proof of Proposition 6.2 in [23], but now we need to ensure that σ' has a gap vector $\mathbf{g}' \succeq \mathbf{h}$. This can be done by replacing the elements of σ which ensured that $\mathbf{g} \succeq \mathbf{h}$; in the worst case we have removed all the elements of σ which lay in gaps with positive minimum sizes, and so we need to replace $||\mathbf{h}||_1$ elements. Altogether, σ' has $\leq ||B||_\infty - 1 + ||\mathbf{h}||_1$ elements following its prefix.

To complete the proof, we note that by construction σ' has gap vector $\mathbf{g}'$ satisfying $||\mathbf{g}'||_1 \leq ||B||_\infty - 1 + ||\mathbf{h}||_1$. Also by construction, $\mathbf{g}'$ satisfies $\mathbf{h}$, and since $\mathbf{g}' \preceq \mathbf{g}$, it also fails to satisfy any of $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$. Finally, when the r in the downfix is removed, σ' ceases to contain any pattern of B. □

There are two ways to possible good outcomes when considering a potential gap condition: some downfix element is deletable for all gap vectors satisfying the gap condition or all gap vectors satisfying the gap condition are nonviable. Proposition 2.2.4 shows how to check for the first good outcome, and checking for the second is even easier. Since $\mathbf{h}$ is a gap vector satisfying itself, it must be nonviable if all gap vectors satisfying it are nonviable. Conversely, if some other gap vector is viable, then there exists $\sigma \in Z(B, \pi, \mathbf{g})$ for $\mathbf{g} \succeq \mathbf{h}$, and removing elements from σ will give $\sigma' \in Z(B, \pi, \mathbf{h})$. Therefore, it is enough to check if $Z(B, \pi, \mathbf{h})$ is empty.

We can now precisely describe how to determine if a downfix has a reduction rule. Algorithm 2 inputs a set B of forbidden patterns, a downfix π, and a maximum l_1 norm for gap conditions l. Let $\mathrm{Comp}(i, m)$ be the set of nonnegative, length-m vectors whose components sum to i.

There is a subtlety here in that the order in which we choose $\mathbf{h} \in$ *gap_restricts* matters. It may be that there is a way to reduce π whenever its gap vector satisfies $\mathbf{h_1}$ and doesn't satisfy $\mathbf{h_2}$, but not when it satisfies $\mathbf{h_1}$ and $\mathbf{h_2}$. Therefore, we could miss a reduction rule if we consider $\mathbf{h_1}$ before $\mathbf{h_2}$. As a result, after every new addition to the reduction rule Algorithm 2 is building, we have to start over and consider every potential $\mathbf{h}$ again to be sure that we find every possible reduction rule.

Now that we have an algorithm to see if a particular downfix is FS-reducible, we

Algorithm 2: `HasReductionRule`

```
1  k := length (π)
2  gap_restricts := ⋃_{i=0}^{l} Comp (i, k + 1)
3  used_gap_restricts := {}
4  partial_reduction := []
5  for h ∈ gap_restricts\used_gap_restricts do
6      if Z(B, π, h) = ∅ then
7          append [h, 0] to partial_reduction
8          used_gap_restricts := used_gap_restricts ∪ {h}
9          go to(5)
10     else
11         for i := 1 to k do
12             if |Z(B, π, g)| = |Z(B, d_i(π), d_i(g)| for all g ∈ {x : ||x||_1 ≤
               ||B||_∞ + ||h||_1 − 1, x ⪰ h, x ⋡ h' for all h' ∈ used_gap_restricts then
13                 append [h, i] to partial_reduction
14                 used_gap_restricts := used_gap_restricts ∪ {h}
15                 go to(5)
16 if [0, 0, ..., 0] ∈ used_gap_restricts then
17     return (partial_reduction)
18 else
19     return ([π, []])
```

can describe the overall algorithm `HasScheme`. This algorithm inputs a set of forbidden subpatterns B, a maximum depth M and a maximum l_1 norm for gap conditions l. This is a recursive algorithm, and so it also inputs three sets *partial_scheme*, $downfixes\_reduced$ and $downfixes\_needed$ to keep track of the rules found so far, the downfixes which have reduction rules, and the downfixes that need reduction rules respectively. In the initial function call, *partial_scheme* $:= \emptyset$, $downfixes\_reduced := \emptyset$, and $downfixes\_needed := \{[]\}$. Pseudocode is given in Algorithm 3.

We conclude this section with a caveat regarding the performance of flexible schemes. In the next section we will show that flexible schemes provide polynomial-time enumeration for any permutation class with a finite traditional scheme or a regular insertion encoding, and many other classes besides. However, there is no free lunch. Compared with regular insertion encodings, enumeration is much slower because flexible schemes simply do not recognize the underlying C-finite structure of the enumeration sequences they produce. As a result, instead of enumeration in linear time, we have to settle for

Algorithm 3: `HasScheme`

```
k := length (π)
if downfixes_needed = ∅ then
    return (partial_scheme)
for π ∈ downfixes_needed do
    new_rule := HasReductionRule (B, π, l)
    if new_rule[2] ≠ [] then
        partial_scheme := partial_scheme ∪ {new_rule}
        downfixes_reduced := downfixes_reduced ∪ {π}
        if new_rule[2] > 0 then
            downfixes_needed := downfixes_needed ∪ {d_{new_rule[2]}(π)}
    else if k = M then
        return (FAIL)
    else
        partial_scheme := partial_scheme ∪ {new_rule}
        downfixes_reduced := downfixes_reduced ∪ {π}
        downfixes_needed := downfixes_needed ∪ {f_i(π) : 0 ≤ i ≤ k}
downfixes_needed := downfixes_needed\downfixes_reduced
return (HasScheme
(B, M, l, partial_scheme, downfixes_reduced, downfixes_needed))
```

enumeration in $O(n^{d+2})$ time. Compared to traditional schemes, meanwhile, flexible schemes may require much longer gap conditions, and so may be more difficult to build. With traditional schemes, we need only consider gap conditions of size $\leq ||B||_\infty - 1$, and so it is reasonable to simply try all possible gap conditions. Here, we may need much longer gap conditions, and so we must impose an artificial limit on the ones we will consider. In practice, though, this seems to be only a minor disadvantage (see Section 2.3).

2.2.4 Sufficient Conditions for Flexible Schemes

For some automatic enumeration algorithms, we know precisely when they will succeed. As noted earlier, permutation classes have regular insertion encodings if and only if they contain finitely many vertical alternations. Enumeration schemes have proven more difficult to analyze, however. For all but a handful of special cases, we can only conclude that a finite enumeration scheme exists when we find one, and we can only conjecture that one does not exist when we do a lot of work and still fail to find one.

In this section, we prove that finite flexible schemes exist whenever a finite traditional scheme or regular insertion encoding does.

First, we show that every downfix which is ES-reducible is also FS-reducible. As defined in [23], a downfix is ES-reducible if and only if there exists r such that for all $\mathbf{g}$ either $|Z(B, \pi, \mathbf{g})| = 0$ or $|Z(B, \pi, \mathbf{g})| = |Z(B, d_r(\pi), d_r(\mathbf{g}))|$.

A downfix is FS-reducible, meanwhile, if and only if there exists a finite list of gap-conditions $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$ with $\mathbf{h_k} = [0, 0, \ldots, 0]$ and a corresponding list of integers $r_1, r_2, \ldots r_k$ (where each integer is in $\{0, 1, \ldots, |\pi|\}$) such that for all $\mathbf{g}$, if i is chosen minimally so that $\mathbf{g} \succeq \mathbf{h_i}$, then

$$|Z(B, \pi, \mathbf{g})| = \begin{cases} 0 & \text{if } r_i = 0 \\ |Z(B, d_{r_i}(\pi), d_{r_i}(\mathbf{g}))| & \text{otherwise} \end{cases}$$

We claim that if a downfix is ES-reducible, then that downfix is also FS-reducible. As noted in [23], the set $\{\mathbf{g} : |Z(B, \pi, \mathbf{g})| \neq 0\}$ is a lower order ideal in the lattice $\mathbf{N}^{|\pi|+1}$ and has a finite basis $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$ such that $|Z(B, \pi, \mathbf{g})| = 0$ if and only if $\mathbf{g} \succeq \mathbf{h_i}$ for some i. Therefore, we can simply use these $\mathbf{h}$s along with $\mathbf{h_{k+1}} = [0, 0, \ldots, 0]$ and $(r_i)_{i=1}^{k+1}$ where $r_i = 0$ for $i \leq k$ and $r_{k+1} = r$ to fulfill the conditions for FS-reducibility.

Next, we show that a regular insertion encoding also guarantees a finite flexible scheme.

Theorem 2.2.5. *Let B be a set of forbidden patterns, and suppose the class of permutations avoiding B has a regular insertion encoding. Then, that class also has a finite flexible scheme.*

Proof. Recall that the permutation classes with regular insertion encodings are exactly those with finitely many vertical alternations. For such a class, we can find k such that no vertical alternation in the class is longer than $2k$, and so, for every downfix π, any gap vector $\mathbf{g}$ with $k+1$ positive entries is nonviable (because every $\sigma \in Y(\pi, \mathbf{g})$ contains as a subsequence a vertical alternation of length $2k + 1$).

Fix some sufficiently long π; our strategy is as follows. Our first gap conditions are the ones with $k + 1$ 1s and all other entries 0. As noted earlier, any vector satisfying

one of these conditions is nonviable, and so we now only need to consider gap vectors with at most k positive entries. We next show that every gap condition $\mathbf{h}$ consisting of k 1 entries and (length($\mathbf{h}$) $- k$) 0 entries has an element r which is FS-reducible for every $\mathbf{g}$ satisfying $\mathbf{h}$. Then, for every gap condition $\mathbf{h}'$ consisting of $(k-1)$ 1 entries and (length($\mathbf{h}$) $- k + 1$) 0 entries, we find an r which is FS-reducible for every $\mathbf{g}$ satisfying $\mathbf{h}'$ and not satisfying any of the earlier $\mathbf{h}$ with k 1 entries. We continue in this way until we have found an r which is FS-reducible for every $\mathbf{g}$ satisfying $[0, 0, \ldots, 0]$ but not satisfying any $\mathbf{h}$ which contains a 1. Every $\mathbf{g}$ with j positive entries satisfies an $\mathbf{h}$ with j 1 entries but not any $\mathbf{h}$ with more than j 1 entries, and so this will show that π is FS-reducible.

Let π be a downfix and $\mathbf{h}$ be a gap condition with k ones. Consider some element i of π, and suppose it is not FS-reducible. Then, there exists some σ with downfix π and gap vector $\mathbf{g} \succeq \mathbf{h}$ which contains a pattern of B, but which does not contain such a pattern when i is removed. Furthermore, by Proposition 2.2.4 we can assume that $||\mathbf{g}||_1 \leq ||B||_\infty - 1 + k$. Following Vatter in [24], we say that σ witnesses i. Note that i is present in every occurrence in σ of every pattern of B, and so σ can witness at most $||B||_\infty$ elements i.

Now, we just need to show that only finitely many σs can witness elements. Since $\mathbf{g} \succeq \mathbf{h}$ but has no more than k positive entries, all entries of $\mathbf{g}$ are 0 except for the k which $\mathbf{h}$ forced to be positive. Thus, there are finitely many possible $\mathbf{g}$s which can provide witnesses (the precise number is the number of ordered integer partitions of numbers less than or equal to $||B||_\infty - 1 + k$ into k parts). For some fixed $\mathbf{g}$ there are $||\mathbf{g}||_1!$ possible permutations σ with downfix π and gap vector $\mathbf{g}$. Therefore, only a fixed number of downfix elements can be witnessed, and this number is independent of the length of π. So, we can take π large enough, and there will be a downfix element which is not witnessed and hence is FS-reducible.

At this point, we have shown that for all sufficiently long π and gap conditions $\mathbf{h}$ with k ones, there exists an element r which is FS-reducible for every viable $\mathbf{g} \succeq \mathbf{h}$. The proof is essentially the same for any $j < k$, but we will write it out anyway for the sake of completeness.

Let π be a downfix and $\mathbf{h}$ be a gap condition with j ones for some $0 \leq j < k$. Consider some element i of π, and suppose it is not FS-reducible. Then, there exists some σ which contains a pattern of B, but which does not contain such a pattern when i is removed. This σ has downfix π and gap vector $\mathbf{g}$ satisfying $\mathbf{h}$ but not satisfying $\mathbf{h}'$ for any $\mathbf{h}'$ with more than j positive entries. Again Proposition 2.2.4 lets us assume that $||\mathbf{g}||_1 \leq ||B||_\infty - 1 + j$. Also as before, σ can witness at most $||B||_\infty$ elements i.

We know that all entries of $\mathbf{g}$ are 0 except for the j which $\mathbf{h}$ forced to be positive (otherwise, $\mathbf{g}$ would satisfy some other $\mathbf{h}'$ with more 1s). Again, this means that only finitely many $\mathbf{g}$s can provide witnesses, and each $\mathbf{g}$ provides at most $||\mathbf{g}||_1!$ witnesses. Thus, if π is large enough, there is a downfix element which is FS-reducible. As noted at the end of the second paragraph of this proof, that is enough to show that π is FS-reducible. Since this is true for all sufficiently large π, a finite flexible scheme exists. □

While Theorem 2.2.5 shows that a finite scheme does exist, it could, in principle, be extremely deep and thus practically impossible to find. In our experiments recorded in Section 2.3, though, we considered 48 permutation classes with a regular insertion encoding and found finite flexible schemes for all but 3 of them.

2.2.5 Maple Implementation

The algorithms described in this section have been implemented in the Maple package Flexible Scheme, available at the author's website. To use the package, download and save this text file. Then, open a Maple worksheet and run the command `read 'Flexible_Scheme.mpl';`. To display the two main functions (`HasScheme` and `FindTerm`), call the help function by running the command `Help();`, and then to learn more about a function, say `HasScheme`, call help again with `HasScheme` as its argument.

The function `HasScheme` takes four arguments. The first is the set of forbidden subpatterns that define the permutation class it is trying to find a scheme for. The second is the maximum depth it will search, if it cannot reduce the downfixes at this depth, then it will give up. The third argument is the maximum gap size, that is the largest l_1 norm we allow putative gap restrictions to have. The final argument is

a boolean called quick_fail which, when set to true, will cause `HasScheme` to look for irreducible downfix in a depth-first manner. If no scheme of sufficiently small depth exists, this strategy provides a substantial speed-up in proving that no such scheme exists. If a scheme is found, `HasScheme` returns it, and if no scheme is found, `HasScheme` returns the reduction rules that it was able to find along with the downfixes it was unable to reduce.

The second function `SeqS` takes two arguments. The first is a scheme describing a permutation class, and the second is an integer n. The function returns the n^{th} term of the enumeration sequence of the permutation class.

2.3 Empirical Results

We tried to find regular insertion encodings, traditional schemes, and flexible schemes for several different permutation classes. In particular we looked at the avoidance classes of pattern sets B where B consisted of either a single pattern of length 3, 4, or 5, a pair of patterns of length 3, a pair of patterns of length 4, or a pattern of length 4 and another of length 5.

For each of these possible pattern lengths, Table 2.1 shows how many classes have regular insertion encodings, how many have finite traditional schemes, how many have finite flexible schemes, and how many of those with finite flexible schemes did not have either a regular insertion encoding or a finite traditional scheme. For finding regular insertion encodings, we used Vatter's package `InsEnc` from [24], for finding traditional schemes we used both Zeilberger's package `VATTER` from [30] and Vatter's package `WILFPLUS` from [23], and for finding flexible schemes we used our own `Flexible_Scheme`.

While we can safely conclude that we found regular insertion encodings whenever they exist, the same is not true of schemes. When finding traditional schemes using `WILFPLUS`, we only considered downfixes of length ≤ 8; it is conceivable that finite schemes exist, but simply require longer downfixes, and so we did not find them. When finding traditional schemes using `VATTER` and flexible schemes using `Flexible_Scheme`, we only considered downfixes of length ≤ 8 and gap conditions with l_1 norm ≤ 2. In

addition, if we could not determine whether an avoidance class had a finite scheme after 36 hours of computation, we recorded it as not having a scheme.

Because of these constraints, it is conceivable that either `InsEnc` or `WILFPLUS` could have found a regular insertion encoding or a finite scheme for some class that `Flexible_Scheme` failed to find one for. In practice, this happened for three of the avoidance classes we considered (all of which were avoiding a length 4 and length 5 pattern).

Table 2.1: Empirical Results

Pat length[1]	Sym Classes[2]	Ins. Enc.	ES	FS	New with FS
[3]	2	0	2	2	0
[4]	7	0	2	2	0
[5]	23	0	2	2	0
[3], [3]	5	5	5	5	0
[4], [4]	56	13	33	44	9
[4], [5]	434	30	112	173	59

The permutation classes avoiding two length four permutations have been particularly well studied by previous authors. Generating functions are known for all but three of them, and these are conjectured not to have any D-finite generating functions in [2]. Two of these (avoiding {4321, 4231} and {4312, 4123}) have finite traditional schemes, and the remaining one (avoiding {4231, 4123}) has a finite flexible scheme. As indicated in Table 2.1, 12 of the 56 symmetry classes lack finite flexible schemes as far as we can tell. These are the classes avoiding $\{1234, 3412\}$, $\{1324, 2143\}$, $\{1324, 3412\}$, $\{1324, 2341\}$, $\{1324, 4231\}$, $\{1324, 2413\}$, $\{1324, 2431\}$, $\{1342, 1423\}$, $\{1342, 2413\}$, $\{1432, 2413\}$, $\{2143, 2413\}$, and $\{2413, 3142\}$. The interested reader can find links to the enumeration sequences and generating functions of all of these avoidance classes at [27].

[1] $[n], [m]$ refers to classes avoiding one pattern of length n and one of length m, $[n]$ refers to patterns avoiding one pattern of length n

[2] Every permutation class has up to 8 symmetries given by inverting, reversing, and complementing its patterns. Every class has the same enumeration sequence as its symmetries, so we really only care about the number of symmetry classes which can be enumerated, not the total number of permutation classes

2.4 Necessary Conditions for Schemes

We next consider when we can safely give up on finding a traditional enumeration scheme (unfortunately the results do not yet extend to flexible schemes). While a few pattern avoidance classes have been proven to lack finite enumeration schemes (for example $\{1234, 4231\}$ and $\{2413, 3142\}$ in [23]), there is in general no way to tell if a class has a finite scheme without simply trying to find one and eventually giving up. Here, we show that if $12\ldots k$ is ES-irreducible for sufficiently large k ($k = 4$ is sufficient in some cases of interest), then no finite traditional scheme exits.

Before proving our result, we need to introduce a new (but standard) definition: an *interval* of a permutation π is a set of indices I such that both I and $\{\pi_i : i \in I\}$ are contiguous. An *increasing interval*, then, is simply an interval where $\pi_i < \pi_j \iff i < j$ for all $i, j \in I$. Sometimes, we will abuse this definition by referring to the subpermutation of π whose elements have indices in I as an interval as well.

Example 2.4.1. Consider the permutation 51243. Its intervals are $\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{2, 3\}, \{2, 3, 4, 5\}$ and $\{1, 2, 3, 4, 5\}$. Its increasing intervals are $\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}$, and $\{2, 3\}$.

Note that a downfix π is ES-irreducible for a set of patterns B if and only if for each $i \in \pi$ there exists $\mathbf{g_i}$ satisfying the following conditions:

Conditions 1.

1. $Z(B, \pi, \mathbf{g_i}) \neq \emptyset$.
2. There exists $\sigma \in Y(\pi, \mathbf{g_i}) \backslash Z(B, \pi, \mathbf{g_i})$ such that every occurrence in σ of a pattern in B involves the element i.

By Proposition 6.2 in [23], we can further assume that $||\mathbf{g_i}||_1 \leq ||B||_\infty - 1$.

The idea behind the following two proofs is simple: if the longest increasing interval in a pattern has length l, then any occurrence of that pattern can use at most l elements from any increasing interval in a permutation. Therefore, if that permutation has an

increasing interval with more than l elements, one of them can be ignored. Conditions 1 essentially require us to find two permutations for a given downfix element and gap vector; one to witness that the gap vector is viable, and one to witness that the element is not deletable. If we have these two permutations for each element of the downfix $12\ldots k-1$, and they have sufficiently long increasing intervals, we can turn them into witnesses for each element of the downfix $12\ldots k$ by extending their increasing intervals by a single element.

Before giving an actual theorem, we motivate it with the following example:

Example 2.4.2. Consider the pattern set $B = \{3241, 4132\}$), and note that 1234 is ES-irreducible for the set of patterns (this can be checked using WILFPLUS, for instance). To show that 12345 is also ES-irreducible, we need to show that Conditions 1 hold for each of the five elements; in this example we will show that they hold for 1. Notice that, because 1234 is ES-irreducible, there exists a $\mathbf{g_1'}$ satisfying Conditions 1 for the 1 in 1234, for instance $[1,1,0,0,0]$ will do the job, since $516234 \in Z(B, 1234, [1,1,0,0,0])$, and $615234 \in Y(1234, [1,1,0,0,0])\backslash Z(B, 1234, [1,1,0,0,0])$ has three 4132 patterns all using the 1 and no 3241 patterns.

Now, we'll show that $\mathbf{g_1} = [1,1,0,0,0,0]$ satsifies Conditions 1 for the 1 in 12345. Take 516234, increase all elements larger than 1 by 1, and insert a 2 between the new 7 and 3 to get 6172345, and note that this permutation is in $Z(B, 12345, [1,1,0,0,0,0])$. Similarly, take 615234, increase all elements larger than 1 by 1 and insert a 2 between the new 6 and 3 to get 7162345, and note that this permutation is in $Y(12345, [1,1,0,0,0,0])\backslash Z(B, 12345, [1,1,0,0,0,0])$, and the four 4132 patterns all use the 1 (and there are still no 3241 patterns).

Lemma 2.4.3. *Let B be a set of patterns. Suppose that the longest increasing interval of a pattern in B has length l. Then, two related statements hold:*

1. *Suppose that $\mathbf{g'}$ satisfies Conditions 1 for an element $i-1$ of the downfix $\pi' = 12\ldots k-1$. Suppose further that $g'_j, g'_{j+1}, \ldots g'_{j+l-2}$ are all 0 for some $j \geq 2$ where $i > j+l-1$. Form $\mathbf{g}$ by inserting a 0 into $\mathbf{g'}$ immediately before entry g'_j. Then $\mathbf{g}$ satisfies Conditions 1 for the element i of $\pi = 12\ldots k$.*

2. *Suppose that* $\mathbf{g}'$ *satisfies Conditions 1 for an element* i *of the downfix* $\pi' = 12\ldots k-1$. *Suppose further that* $g'_j, g'_{j+1}, \ldots g'_{j+l-2}$ *are all 0 for some* $j > i+1$ *where* $j+l-1 \leq k$. *Form* $\mathbf{g}$ *by inserting a 0 into* $\mathbf{g}'$ *immediately before entry* g'_j. *Then* $\mathbf{g}$ *satisfies Conditions 1 for the element* i *of* $\pi = 12\ldots k$.

Proof. We begin with the first statement. There is a bijection F between permutations $\sigma' \in Y(\pi', \mathbf{g}')$ and permutations $\sigma \in Y(\pi, \mathbf{g})$ given by inserting a copy of the element $j-1$ immediately before the existing $j-1 \in \sigma'$ and then increasing all elements of σ' greater than or equal to $j-1$ (except for the new copy) by one. Choose $\sigma' \in Z(B, \pi', \mathbf{g}')$, let $a = \sum_{p=1}^{j-1} g'_p$, and note that $\{a+j-1, a+j, \ldots, a+j+l-2\}$ is an increasing interval of σ'. Let $\sigma = F(\sigma')$ (now $I = \{a+j-1, a+j, \ldots, a+j+l-1\}$ is an increasing interval of σ), and suppose by way of contradiction that $\sigma \notin Z(B, \pi, \mathbf{g})$. There must be an occurrence of a pattern of B in σ which uses j, but j is in I which has length $l+1$, and this pattern can use at most l elements in I. Since it doesn't matter which l elements are used, the pattern can just as well be formed with the l elements of the increasing interval that aren't j. Thus, this pattern already existed in σ' and $\sigma' \notin Z(B, \pi', \mathbf{g}')$. This contradicts our choice of σ', so we conclude that $\sigma \in Z(B, \pi, \mathbf{g})$.

Next, we verify that $\mathbf{g}$ satisfies the second of Conditions 1 for i. Let $\sigma' \in Y(\pi', \mathbf{g}') \backslash Z(B, \pi', \mathbf{g}')$ be the permutation promised by the second condition, and set $\sigma = F(\sigma')$. We immediately have that $\sigma \in Y(\pi, \mathbf{g}) \backslash Z(B, \pi, \mathbf{g})$. Consider some occurrence p_1 in σ of a pattern in B. Again, let I be the increasing interval of length $l+1$ beginning with $j-1$. As noted in the preceding paragraph, this occurrence p_1 contains $\leq l$ elements of I, and, so we can find an occurrence p_2 of the same pattern using all the same elements of σ outside of I, but not using $j-1$. This occurrence p_2 corresponds to an occurrence p_3 of the same pattern in σ' which must have used $i-1$ (because σ' satisfies Conditions 1 for $i-1$). When we turn σ' into σ, we increase all elements at least as large as $j-1$ by 1; since $i-1 \geq j-1$ it is increased to i. Therefore, p_2 used i, and so did p_1 since i lies outside I and p_1 and p_2 use all the same elements except for the ones with indices in I. Thus, σ satisfies Conditions 1 for i.

The proof of the second statement is essentially the same, except now I falls after

i. □

Proposition 2.4.4. *Let B be a set of patterns whose longest increasing interval has length l. Suppose that $\lfloor k/2 \rfloor - 1 \geq (l-1)(||B||_\infty - 1) + l$, and $\pi' = 12\ldots(k-1)$ is ES-irreducible for B. Then $\pi = 12\ldots k$ is also ES-irreducible for B.*

Proof. We claim that π is ES-irreducible for B, so we will find $\mathbf{g}$ satisfying Conditions 1 for every $i \in \pi$. Choose such an $i \in \pi$, and suppose $i \geq \lfloor k/2 \rfloor + 1$. Let $\pi' = 12\ldots k-1$; by our hypothesis, π' is ES-irreducible for B, and so there exists $\mathbf{g}'$ which satisfies Conditions 1 for $i-1$. Now, π' has at least $\lfloor k/2 \rfloor - 1$ elements less than $i-1$, which by assumption is at least $(l-1)(||B||_\infty - 1) + l$. Since $||\mathbf{g}'||_1 \leq ||B||_\infty - 1$, even if all the positive entries in $\mathbf{g}'$ are placed before $i-1$, we can find at least l entries of π' occurring before $i-1$ such that the $l-1$ g'_j corresponding to the gaps between them are all 0. Thus the conditions of Lemma 2.4.3 part 1 are satisfied, and if we form σ as per the lemma, we find that σ satisfies Conditions 1 for i.

Now suppose that $i \leq \lfloor k/2 \rfloor$. Then, π' has at least $\lfloor k/2 \rfloor - 1$ elements greater than i, so we can find at least l entries of π occurring after i such that the $l-1$ g'_j corresponding to the gaps between them are all 0. Thus the conditions of Lemma 2.4.3 part 2 are satisfied, and if we form σ as per the lemma, we find that σ satsifies Conditions 1 for i. □

This result shows that as long as $12\ldots k$ is irreducible for k large enough (when $l = 1, k = 4$ will do), then B has no finite traditional scheme. A similar argument shows that the same is true if $k\ldots 21$ is irreducible (and thus that $\{1423, 2314\}$ from Example 2.2.1 has no such scheme). While we have not been able to prove the corresponding result for flexible schemes, we suspect that it is also true.

2.5 Flexible Schemes For Covincular Patterns

The purpose of this section is to extend the work of Baxter and Pudwell [7] on enumeration schemes for vincular patterns to flexible schemes. A vincular, or dashed, pattern is much like a classical pattern, except in an occurrence of a vincular pattern some of

the permutation elements participating in the pattern may be required to occur consecutively in the permutation. In Section 4 we talk more about vincular patterns, but in this section we are actually interested in a related object: what Bean et al. [9] call *covincular patterns*.

The reason for this is that the previous authors count with prefix schemes and we count with downfix schemes, and so we are essentially counting the inverses of the permutations that they count. The inverse of a vincular pattern is not another vincular pattern, but rather a covincular patterns. This is a pair $\tau = (\sigma, X)$ where σ is the underlying permutation which must be avoided and X indicates which elements in an occurrence of σ must be consecutive values (rather than having consecutive indices like a vincular pattern). Similarly, a *spaced permutation* is a pair $\rho = (\pi, Y)$ where π is the underlying permutation and Y indicates which elements of π should not be treated as occurring consecutively. If $X = \emptyset$, then τ is just an ordinary pattern, and if $Y = \emptyset$, then ρ is an ordinary permutation.

Containment of covincular patterns is slightly more complicated than classical containment. Let $\pi = \pi_1\pi_2\dots\pi_n$ and $\sigma = \sigma_1\sigma_2\dots\sigma_k$ be permutations. For π to contain the covincular pattern (σ, X), we first must find a classical occurrence of σ in π, i.e. we need $i_1 < i_2 < \cdots < i_k$ such that $\pi_{i_1}\pi_{i_2}\dots\pi_{i_k}$ has the same relative order as σ (ignoring dots). This occurrence must have the further property that for all σ_l, σ_m with $\sigma_l + 1 = \sigma_m$, if $\sigma_l \in X$, then π_{i_m} is one larger than π_{i_l} and $\pi_{i_l} \notin Y$.

Finding a downfix of a spaced permutation is done more or less how one would expect. A downfix of (π, Y) is a pair (π', Y') where π' is the subpermutation of π consisting of elements $\leq l$ for some integer l, and Y' is the subset of Y whose elements are no greater than (the same integer) l.

We define $S(B, \pi, \mathbf{g})$ similarly to how we defined $Z(B, \pi, \mathbf{g})$. Just like $Z(B, \pi, \mathbf{g})$, it is the set of permutations with downfix π and corresponding gap vector $\mathbf{g}$, the only difference is that both π and the elements of $S(B, \pi, \mathbf{g})$ are spaced permutations.

Example 2.5.1.

$$S(\{(123,\{2\})\},(21,\{1\}),[0,1,1]) =$$
$$\big\{(2413,\{1\}),(2413,\{1,3\}),(2413,\{1,4\}),$$
$$(2413,\{1,3,4\}),(2314,\{1,3\}),(2314,\{1,3,4\})\big\}$$

Notice that $(2314,\{1,3\})$ avoids $(123,\{2\})$ because, although 234 is an occurrence of 123 and are all consecutive values, we pretend that there is some placeholder value between 3 and 4 (since $3 \in \{1,3\}$). Since the 2 and 3 in any occurrence of $(123,\{2\})$ can't have any values between them, this is not a valid occurrence.

As with classical pattern avoidance, the key operation here will be deleting a downfix element to get a smaller permutation. Unlike with classical pattern avoidance, though, when we delete a downfix element, we need to add a placeholder element to Y to mark where the element used to be, while retaining all previous placeholders as well. Let $d_r((\pi,Y)) = \tau' = (\pi',Y')$. We define $\pi' = d_r(\pi)$ (we abuse notation by treating d_r as a function both on classical and spaced permutations), and $Y' = \{y : y \in Y, y < r\} \cup \{y-1 : y \in Y, y \geq r\} \cup \{r-1\}$. (Notice that if $r = 1$, then we add the element 0 to Y' which has no effect on whether a permutation contains τ'; this makes sense since 1 can't prevent a pattern by separating two pattern elements, and so we don't actually need to add a placeholder when deleting it.) Since gap vectors are the same for spaced permutations as for classical ones, we do not need to redefine $d_r(\mathbf{g})$. Lemma 2.5.2 shows that we cannot create a new pattern when we delete a permutation element in this way.

Lemma 2.5.2. *Suppose (π,Y) is a permutation which does not contain the pattern (σ,X). Then, $d_r((\pi,Y))$ does not contain (σ,X) either.*

Proof. Let $(\pi',Y') = d_r((\pi,Y))$. Suppose (π',Y') contains (σ,X), and suppose this occurrence uses the elements $\pi'_{i_1},\pi'_{i_2},\ldots,\pi'_{i_k}$. These elements correspond to the elements $\pi_{j_1},\pi_{j_2},\ldots,\pi_{j_k}$ of π (note that $\pi_{j_x} \neq r$ for all x), and these elements form a (classical) occurrence of σ. Since (π,Y) doesn't contain (σ,X), we can find π_{j_l},π_{j_m} corresponding to σ_l,σ_m such that $\sigma_l+1=\sigma_m$, $\sigma_l \in X$ and either $\pi_{j_l}+1 < \pi_{j_m}$ or $\pi_{j_l} \in Y$.

First, suppose $\pi_{j_l} + 1 < \pi_{j_m}$. Either $\pi'_{j_l} + 1 < \pi'_{j_m}$, in which case we didn't actually have an occurrence of (σ, X) in (π', Y), or $\pi_{j_l} = r - 1$ and $\pi_{j_m} = r + 1$. In this case, Y' contains $\pi_{j_l} = r - 1$, and so again we didn't actually have an occurrence of (σ, X). Next, suppose that $\pi_{j_l} \in Y$. If $r < \pi_{j_l}$, then $\pi_{j_l} - 1 = \pi'_{j_l} \in Y'$, and if $r > \pi_{j_l}$, then $\pi'_{j_l} = \pi_{j_l} \in Y$ and so again we didn't actually have an occurrence of (σ, X). □

We can now proceed much as we did in Section 2.2.3. As before, we need to know two things: when we can delete downfix elements, and when every permutation with a particular downfix and gap vector contains a forbidden pattern.

2.5.1 Deleting Downfix Elements

There is a simple bijection F_r between $S(\emptyset, \rho, \mathbf{g})$ and $S(\emptyset, d_r(\rho), d_r(\mathbf{g}))$ given by deleting the r in the permutation's downfix, and adding an appropriate place holder. Let f_r be the restriction of F_r to $S(B, \rho, \mathbf{g})$; in the classical case, it was obvious that f_r was an injection into $S(B, d_r(\rho), d_r(\mathbf{g}))$; in the covincular case, this is a consequence of Lemma 2.5.2. Sometimes, f_r is not just an injection, but is a bijection; we would like some analogue of Proposition 2.2.4 which tells us when that is the case. Luckily, this proposition is easily adapted to our new situation.

Proposition 2.5.3. *Let B be a set of forbidden patterns, $\rho = (\pi, Y)$ be a downfix, and $\mathbf{h}, \mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$ be gap conditions. Suppose that*

$$|S(B, \rho, \mathbf{g})| = |S(B, d_r(\rho), d_r(\mathbf{g}))|$$

for all $\mathbf{g}$ with $||\mathbf{g}||_1 \leq ||B||_\infty - 1 + ||\mathbf{h}||_1$ which satisfy $\mathbf{h}$ but fail to satisfy any of $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$. Then the equality holds for all $\mathbf{g}$ which satisfy $\mathbf{h}$ but fail to satisfy any of $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$.

Proof. Choose some $\mathbf{g}$ satisfying $\mathbf{h}$ and no other $\mathbf{h_i}$. The observation in the previous paragraph is enough to show that $|S(B, \rho, \mathbf{g})| \leq |S(B, d_r(\rho), d_r(\mathbf{g})|$, so we just need to show the reverse inequality.

Suppose that there is some spaced permutation $(\sigma, X) \in S(\emptyset, \rho, \mathbf{g})$, but which isn't in $S(B, \rho, \mathbf{g})$ because it contains some pattern of B. Suppose further that when the

r in the downfix is deleted (and $r-1$ added to X), it ceases to contain any pattern of B. Choose some occurrence of a pattern $b \in B$ in (σ, X), and suppose that it uses the elements $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ (where $k \leq ||B||_\infty$). Form σ' by iteratively finding non-downfix elements s which do not participate in this occurrence and which are not necessary to ensure that σ' has a gap vector satisfying $\mathbf{h}$, and then deleting these elements by replacing (σ, X) with $d_s((\sigma, X))$. Since there are $\leq ||B||_\infty - 1$ elements other than r participating in the forbidden occurrence and $\leq ||h||_1$ elements needed to have a gap vector satisfy $\mathbf{h}$, we can delete all by $||B||_\infty - 1 + ||h||_1$ non-downfix elements in this way. Reduce the resulting permutation.

At this point, we have a permutation $(\sigma', X') \in S(\emptyset, \rho, \mathbf{g}) \backslash S(B, \rho, \mathbf{g})$ with $||g||_1 \leq ||B||_\infty - 1 + ||h||_1$ where $\mathbf{g}$ satisfies $\mathbf{h}$ and no other $\mathbf{h_i}$. All that remains is to show that $d_r((\sigma', X')) \in S(B, d_r(\rho), d_r(\mathbf{g}))$. To do this, notice that we can form $d_r((\sigma', X'))$ by first finding $d_r((\sigma, X))$, which we assumed has no forbidden pattern, and deleting all the s's that we deleted from (σ, X) in the previous paragraph. By Lemma 2.5.2 these deletions cannot create any forbidden pattern, and so $d_r((\sigma', X')) \in S(B, d_r(\rho), d_r(\mathbf{g}))$ as required. □

Strictly speaking, $|S(B, \rho, \mathbf{g})| = |S(B, d_r(\rho), d_r(\mathbf{g}))|$ is not exactly the equality we want. This is an equality between sets of spaced permutations where placeholders can occur anywhere, but we actually want an equality between sets of spaced permutations where placeholders can only occur in the downfix. Precisely, the equality we want is $|R(B, \rho, \mathbf{g})| = |R(B, d_r(\rho), d_r(\mathbf{g}))|$ where $R(B, \rho, \mathbf{g})$ is the set of spaced patterns (σ, X) with downfix $\rho = (\pi, Y)$ and corresponding gap vector $\mathbf{g}$ such that $X = Y$. Fortunately, the equality we have is stronger than the one we actually want.

Corollary 2.5.4. Under the same hypotheses as Proposition 2.5.3, we also have that

$$|R(B, \rho, \mathbf{g})| = |R(B, d_r(\rho), d_r(\mathbf{g}))|$$

for all $\mathbf{g}$ which satisfy $\mathbf{h}$ but fail to satisfy any of $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$.

Proof. Let $\rho = (\pi, Y)$, and by way of contradiction, suppose there is some $(\sigma', X') \in R(B, d_r(\rho), d_r(\mathbf{g}))$ whose preimage (σ, Y) under F_r is not in $R(B, \rho, \mathbf{g})$. It follows that

either $(\sigma, X) \notin R(\emptyset, \rho, \mathbf{g})$ or $(\sigma, X) \notin S(B, \rho, \mathbf{g})$. That $(\sigma, X) \in R(\emptyset, \rho, \mathbf{g})$ is clear simply from the definition of R, and that $(\sigma, X) \in S(B, \rho, \mathbf{g})$ follows from Proposition 2.5.3. Therefore, $(\sigma, X) \in R(\emptyset, \rho, \mathbf{g}) \cap S(B, \rho, \mathbf{g}) = R(B, \rho, \mathbf{g})$ and we have obtained a contradiction. □

Notice that while the conclusion of Proposition 2.5.3 implies that of Corollary 2.5.4, the converse is not true. This opens up the possibility that we could verify Proposition 2.5.3 for just some subset of permutations in $S(B, \rho, \mathbf{g})$ and find a condition which is true more often and which still implies Corollary 2.5.4.

Specifically, we would like to find a subset

$$S'_r\big(B, \rho, \mathbf{h}, (\mathbf{h_i})_{i=1}^k\big) \subseteq \bigcup_{||\mathbf{g}||_1 \leq ||B||_\infty + ||\mathbf{h}||_1 - 1} S(B, \rho, \mathbf{g})$$

which contains exactly those spaced permutations resulting from taking a permutation in $R(B, \rho, \mathbf{g})$ (for any $\mathbf{g}$) and performing the deletions described in the proof of Proposition 2.5.3 to check if r is deletable for permutations with gap vectors satisfying $\mathbf{h}$ and not any $\mathbf{h_i}$. Our next step is to build such an S'_r, based on the containment scenarios of Baxter and Pudwell [7]. While we do not prove that this S'_r is minimal, it is much smaller than S.

We define $S'_r\big(B, (\pi, Y), \mathbf{h}, (\mathbf{h_i})_{i=1}^m\big)$ to be the set of spaced permutations (σ, X) satisfying the following conditions:

1. (π, Y) is a downfix of (σ, X).
2. (σ, X) contains an occurrence of a pattern $(\mu, Z) \in B$. Let this occurrence be $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$, it must be that $r = \sigma_{i_l}$ for some l.
3. The gap vector corresponding to (σ, X) (i.e. $\mathbf{g}$ such that $(\sigma, X) \in S(\emptyset, (\pi, Y), \mathbf{g})$) satisfies $\mathbf{h}$ but not $\mathbf{h_i}$ for any i.
4. X contains all elements except possibly those in $\{1, 2, \ldots, |\pi| - 1\}$ and $\{\sigma_{i_l} : \mu_l \in Z\}$. In other words, X contains all elements except the ones that would violate conditions 1 or 2.
5. (σ, X) is minimal.

Note that S'_r is actually missing some of the spaced permutations resulting from taking a permutation in $R(B, \rho, \mathbf{g})$ and performing the deletions described in the proof of Proposition 2.5.3. In particular, condition 4 demands that X contain every element that it can hold without violating conditions 1 or 2, but some resulting permutations may have a smaller (or even empty) X. It is enough, though, to check the spaced permutations with maximal X; we want to know if there is a spaced permutation such that deleting r erases all occurrences of a forbidden pattern, and if that's true of a spaced permutation with more adjacent elements, it is certainly true of a spaced permutation with fewer adjacent elements.

The following is an example of S'_r:

Example 2.5.5.

$$\begin{aligned} S'_2(\{(123, \{1\})\}, 21, [1, 0, 0]) = \big\{&(52341, \{3, 4\}), (52314, \{3, 4\}), \\ &(42351, \{3, 4\}), (42315, \{3, 4\}), \\ &(52134, \{3, 4\}), (42135, \{3, 4\})\big\}. \end{aligned}$$

This set is built as follows. The downfix element 2 must participate in a forbidden subpattern, and the only role it can fill in such a pattern is the smallest element. Therefore, it must be followed by a 3 and then some larger element, and these elements can occur either before or after the 1 in the downfix. This gives us three options: $2341, 2314$, and 2134. However, these have gap vectors $[0, 2, 0], [0, 1, 1]$, and $[0, 0, 2]$, none of which satisfy $[1, 0, 0]$, so we need to add an extra element before the 2. This element can't be between the 2 and 3, but can either be larger than the 3 and smaller than the 4 or larger than the 4. This gives us the six permutations $42351, 52341, 42315, 52314, 42135$, and 52134. To find Y for each permutation, we note that $1 \notin Y$ because it and 2 are both in the downfix and $2 \notin y$ because it serves as the 1 in a $(123, \{1\})$ pattern, but both 3 and 4 can be in Y.

2.5.2 Guaranteeing Forbidden Patterns

We now can tell when some downfix element r can be deleted from every permutation in $R(B, \rho, \mathbf{g})$ with $\mathbf{g}$ satisfying $\mathbf{h}$ but not $\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_k}$. Our next task is to determine

when $R(B, \rho, \mathbf{g}) = \emptyset$ for every $\mathbf{g}$ satisfying $\mathbf{h}$. Just like how we defined $R(B, \rho, \mathbf{g})$ to be all $(\sigma, X) \in S(B, (\pi, Y), \mathbf{g})$ where $X = Y$ (i.e. contains as few elements as possible), we can define $T(B, \rho, \mathbf{g})$ to be all $(\sigma, X) \in S(B, (\pi, Y), \mathbf{g})$ where $X = Y \cup \{|\pi|, |\pi| + 1, \ldots, |\pi| + ||\mathbf{g}||_1 - 1\}$ (i.e. contains as many elements as possible).

Proposition 2.5.6. *Suppose that $T(B, \rho, \mathbf{h}) = \emptyset$. Then, $R(B, \rho, \mathbf{g}) = \emptyset$ for all $\mathbf{g} \succeq \mathbf{h}$.*

Proof. We prove the contrapositive. Choose some $(\sigma, X) \in R(B, (\pi, Y), \mathbf{g})$ where $\mathbf{g} \succeq \mathbf{h}$. Form (σ', X') by deleting non-downfix elements from σ until the only non-downfix elements left are those forced to be present by $\mathbf{h}$. As in the proof of Proposition 2.5.3, put a placeholder element in X' for each deletion. The resulting permutation avoids all the patterns in B by Lemma 2.5.2. Add more elements to X' until $X' = Y \cup \{|\pi|, |\pi| + 1, \ldots, |\pi| + ||\mathbf{g}||_1 - 1\}$; adding more elements to X' can't cause to contain a pattern that it didn't already, and so $(\sigma', X') \in T(B, \rho, \mathbf{h})$. □

Much like Theorem 4 in [7], Proposition 2.5.6 gives only a sufficient condition for $R(B, \rho, \mathbf{g}) = \emptyset$. Example 2.5.7 illustrates the non-necessity of the condition.

Example 2.5.7. We will show that $R(\{(312, \{2\})\}, (12, \emptyset), \mathbf{g}) = \emptyset$ for all $\mathbf{g} \succeq [1, 0, 0]$ even though $(312, \{2\}) \in T(B, \rho, [1, 0, 0])$. It is clear that $(312, \{2\}) \in T(B, \rho, [1, 0, 0])$ since the 2 in the permutation can't serve as the 2 in the pattern since there is a placeholder element between it and the 3.

However, it still holds that $R(\{(312, \{2\})\}, (12, \emptyset), \mathbf{g}) = \emptyset$ for all $\mathbf{g} \succeq [1, 0, 0]$. Choose some $(\sigma, X) \in R(\emptyset, (12, \emptyset), \mathbf{g})$ (note that $X = \emptyset$). Because $\mathbf{g} \succeq [1, 0, 0]$, there is at least one element of σ greater than 2 preceding the 1. Let σ_{i_1} be the smallest such element. This choice forces $\sigma_{i_1} - 1$ to occur after the 1, and so $\sigma_{i_1} 1 (\sigma_{i_1} - 1)$ forms a (312, {2}) pattern.

2.5.3 Finding a Scheme

Now that we can determine when a downfix element is deletable and when all permutations with a certain downfix and gap vector are non-viable, we are ready to define the covincular versions of `FindTerm`, `HasReductionRule`, and `HasScheme`. We do so

in much the same way that we did for classical patterns. In fact, both `FindTerm` and `HasScheme` do not even need to be rewritten. The pseudocode provided in Algorithms 1 and 3 works just as well in the covincular case; all that is necessary is to remember that π is now a spaced permutation and $d_r(\pi)$ and $f_i(\pi)$ are slightly different when π is a spaced permutation. Further, `HasReductionRule` needs only slight modification; specifically $Z(B, \pi, \mathbf{h})$ in Algorithm 2 needs to be replaced with $T(B, \pi, \mathbf{h})$ and $Z(B, \pi, \mathbf{g})$ and $Z(B, d_r(\pi), d_r(\mathbf{g}))$ need to be replaced with $S(B, \pi, \mathbf{g})$ and $S(B, d_r(\pi), d_r(\mathbf{g}))$ respectively.

All these functions are implemented in the Maple package `Flexible_Covincular_Scheme`, available on the author's website. As with `Flexible_Scheme`, the main two functions are `HasScheme` which attempts to find a enumeration scheme for a set of forbidden patterns and `SeqS` which uses a scheme to generate as many terms of the enumeration sequence as the user requests.

Our method of building schemes is a bit different than that of Baxter and Pudwell [7]. There, the authors don't add placeholders to downfixes when deleting elements, but instead check whether deleting an element can introduce a new pattern (we just check whether deleting an element can eliminate a pattern). Our primary reason for this difference is that we wanted our framework to be as analogous as possible to our framework for finding schemes for classical pattern avoidance. Both frameworks have advantages in terms of which sets of forbidden patterns they can find schemes for; in fact we found some pattern sets which could be enumerated using existing vincular schemes but not using flexible ones.

2.5.4 Empirical Results

We tried to find schemes for enumerating the permutations avoiding covincular permutations of length 4 as well as pairs of covincular permutations of length 4. In both cases, we compared the successes of our new flexible schemes (FS) with existing enumeration schemes (ES). We allowed both types of schemes to have depth up to 5, and use gap restrictions with l_1 norm up to 2. As Table 2.2 shows, we found that both types of schemes managed to enumerate the same permutation classes avoiding a single pattern

of length 4. For pairs of length-4 patterns, flexible schemes were able to enumerate significantly more classes than existing schemes, but there were a handful of classes that flexible schemes couldn't enumerate even though previous ones could. As in Section 2.3, we record a permutation class as not having a scheme if we couldn't find on after 36 hours of computation.

Table 2.2: Empirical Results

Pat length	Sym Classes	ES Only	FS Only	Both ES and FS	Neither ES nor FS
[4]	56	0	0	35	21
[4],[4]	4776	9	269	1649	2849

2.6 Future Work

There are a number of potential avenues for future research. As noted in the introduction, traditional schemes have been extended beyond pattern avoidance in permutations to pattern avoidance in words [19]. We believe that flexible schemes can also be extended to cover this case just as Section 2.5 extends them to covincular pattern avoidance.

Another tempting possibility is to unite flexible schemes (or traditional schemes for that matter) with the structure paradigm exemplified by the `TileScope` algorithm [8]. In Section 3, we do this in a special case by attempting to find a scheme for the permutations that avoid 1423 and then, once we find an ES-irreducible downfix, finding its structural properties to complete the enumeration. Unfortunately, Section 3 does this in an ad hoc way which only works for the specific class under consideration. It would be very interesting to build a general algorithm for many different permutation classes.

Finally, this section only partially answers Vatter's original question [23] which was to build a framework capable of enumerating any permutation class with a finite enumeration scheme, a regular insertion encoding, or finitely many simple permutations. While we have made partial progress to an answer the whole question remains open.

Chapter 3

Permutations avoiding 1423 (and Equivalently 1342)

The permutations avoiding 1342 have the generating function $\frac{32x}{1+20x-8x^2-(1-8z)^{3/2}}$ and are also given by the formula

$$|\mathrm{Av}(\{1342\};n)| = \frac{(7n^2-3n-2)}{2}\cdot(-1)^{n-1}+3\sum_{i=2}^{n} 2^{i+1}\cdot\frac{(2i-4)!}{i!(i-2)!}\cdot\binom{n-i+2}{2}\cdot(-1)^{n-i}.$$

These results were originally found by Bóna [12] through a bijection between $\mathrm{Av}(\{1342\})$ and a certain class of labelled trees. Later, the generating function was rederived by Bloom and Elizalde through a bijection from the permutations in $\mathrm{Av}(\{3124\})$ to specific board minimal rook placements, and from there to certain Dyck paths [11]. In this section, we derive a fast recurrence to count the 1342-avoiding permutations in the style of enumeration schemes.

Like Bloom and Elizalde, we find it more convenient to actually count a different class of permutations. In our case, we will enumerate permutations avoiding 1423, but counting one class is equivalent to counting the other since 1423 and 1342 are inverses.

We will find it convenient to talk about permutations with particular downfixes and *index vectors* (rather than gap vectors like in Section 2). An index vector gives the index of each downfix element in the permutation. We define $A(B,\pi,\mathbf{i},n)$ to be the set of permutations of length n avoiding the patterns in B with downfix π and corresponding index vector $\mathbf{i}$. For example, $A(\{123\},21,[1,3],4)=\{2413\}$.

Note that we could equivalently express this set as $Z(\{123\},21,[0,1,1])$, and, in fact, it is simple to translate any set of permutations defined using an index vector to one defined using a gap vector. The arguments which we will be making, however, are much more natural when expressed using index vectors, and so we choose that convention.

3.1 Structure of 1423 Avoiders

We are trying to find $|A(\{1423\}, \emptyset, [], n)|$, and we begin with the standard enumeration scheme trick of separating the permutations into classes by where their 1 element occurs. Algebraically,

$$|A(\{1423\}, \emptyset, [], n)| = \sum_{i=1}^{n} |A(\{1423\}, 1, [i], n)|. \tag{3.1}$$

Next, we separate them further by where their 2 element occurs, and find:

$$|A(\{1423, 1, [i], n)| = \sum_{j=i+1}^{n} |A(\{1423\}, 12, [i, j], n)| + \sum_{j=0}^{i-1} |A(\{1423\}, 21, [j, i], n)|.$$

The second element of the downfix 21 is ES-reducible, and so we can actually write

$$|A(\{1423, 1, [i], n)| = \sum_{j=i+1}^{n} |A(\{1423\}, 12, [i, j], n)| + \sum_{j=0}^{i-1} |A(\{1423\}, 1, [j], n-1)| \tag{3.2}$$

Now it gets tricky, though. Not only is 12 not ES-reducible, there appears to be no finite traditional (or flexible) scheme for 1423. Luckily, we still know a great deal about the structure of a permutation with a 12 prefix, as shown in the following observation.

Observation 3.1.1. *If π avoids 1423 and has a 12 downfix, then all the elements in π occurring between the 1 and 2 are smaller than all the elements occurring after the 2.*

In fact, much more is true.

Lemma 3.1.2. *For all $\pi \in A(\{1423\}, 12, [i, j], n)$ there exists $k \leq i$ such that $\{\pi_k, \pi_{k+1}, \ldots, \pi_j\} = \{1, 2, \ldots, j - k + 1\}$; that is the $j - k + 1$ elements between π_k and π_j are smaller than all other elements of π.*

Proof. We proceed by induction on i. If $i = j - 1$, then the lemma is trivial. Otherwise the open interval (i, j) is nonempty. Let π_{k_1} be the first (left-most) element in π which is less than an element whose index is in (i, j) and let $\pi_{j_1} = \max\{\pi_l : l \in (i, j)\}$. If $k_1 = i$, then the lemma follows from Observation 3.1.1, so assume $k_1 < i$. Form π' by taking the subpermutation of π consisting of π_{k_1}, π_{j_1}, and all $\pi_l > \pi_{j_1}$, and then reducing this subpermutation. Note that π' is a permutation avoiding 1423 which has a 12 downfix (provided by k_1 and j_1); let i' and j' be the indices of the 1 and 2 elements

respectively. Note also that $i' < i$, and so the induction hypothesis promises an index $k' \leq i' < i$ such that the elements of π' between $\pi_{k'}$ and $\pi_{j'}$ are smaller than all other elements of π'.

Let k be the index of the element from π which was reduced to $\pi_{k'}$. We claim that, in π, every element between π_k and π_j is smaller than all other elements of π. Suppose this is not the case, then there exist $x \in [k, j], y \notin [k, j]$ with $\pi_x > \pi_y$. If $y > j$, then $\pi_y > \pi_{j_1}$ by Observation 3.1.1, and if $y < k$ then $\pi_y > \pi_{j_1}$ by the choice of k_1 (since $k \leq k_1$). Once we have $\pi_y > \pi_{j_1}$, we get that $\pi_x > \pi_{j_1}$, and so π' contains elements $\pi'_{y'}$ and $\pi'_{x'}$ corresponding to both π_y and π_x respectively with $\pi'_{x'} > \pi'_{y'}$. But, $x' \in [k', j']$ and $y \notin [k', j']$ which contradicts the last sentence of the previous paragraph. Thus, the lemma holds. □

Lemma 3.1.2 suggests dividing a permutation π with a 12 downfix into two pieces: π_{down} consisting of the elements with indices in $[k, j]$ and π_{up} consisting of all the others (reduced so that it is also a permutation). The following lemma establishes conditions on π_{down} and π_{up} that guarantee that π will avoid 1423.

Lemma 3.1.3. *Let π_{down} and π_{up} be defined as in the previous paragraph, and let $\pi_{up'}$ be the reduced subpermutation of π consisting of all the elements with indices not in $[k+1, j]$ (so $\pi_{up'}$ has one more element than π_{up}). Then, π avoids 1423 if and only if π_{down} and $\pi_{up'}$ both avoid 1423.*

Proof. The only if direction is easy; π_{down} and π_{up} are both subpermutations of π, and so if either contains 1423, π does as well.

To show the if direction, suppose that π contains 1423 and write this occurrence as $\pi_{i_1}\pi_{i_2}\pi_{i_3}\pi_{i_4}$. We claim that either $\pi_{i_1}\pi_{i_2}\pi_{i_3}\pi_{i_4} \in \pi_{\text{down}}, \pi_{i_1}\pi_{i_2}\pi_{i_3}\pi_{i_4} \in \pi_{\text{up}}$, or $\pi_{i_2}\pi_{i_3}\pi_{i_4} \in \pi_{\text{up}}$ and $i_2 > k$. (Note that the element of $\pi_{\text{up}'}$ corresponding to the k in π is the smallest element of $\pi_{\text{up}'}$ and so can replace π_{i_1} as the 1 in the 1423 occurrence.)

Suppose that $\pi_{i_1}\pi_{i_2}\pi_{i_3}\pi_{i_4} \notin \pi_{\text{down}}$ and $\pi_{i_1}\pi_{i_2}\pi_{i_3}\pi_{i_4} \notin \pi_{\text{up}}$, it follows that $\pi_{i_1} \in \pi_{\text{down}}$ and $\pi_{i_2} \in \pi_{\text{up}}$. Since $\pi_{i_1}\pi_{i_2}\pi_{i_3}\pi_{i_4}$ is an occurrence of 1423, we can conclude that $i_2, i_3, i_4 > j$, so $pi_{i_2}\pi_{i_3}\pi_{i_4} \in \pi_{\text{up}}$, and, since $j > k$, $i_2 > k$. Thus, $\pi_{\text{up}'}$ contains 1423. □

3.2 Recurrences for 1423 Avoiders

Combining Lemmas 3.1.2 and 3.1.3 suggests we compute the number of permutations with a 12 downfix by finding the number of possible π_{down} and multiplying by the number of possible π_{up}, which would give the (incorrect) recurrence

$$|A(\{1423\}, 12, [i,j], n)| = \sum_{k=1}^{i} |A(\{1423\}, 1, [i-k+1], j-k)| \cdot |A(\{1423\}, 1, [k], n-j+k)| \tag{3.3}$$

A few elements of this recurrence require explanation. The first term represents all possible π_{down}; these are 1423-avoiding permutations with 12 downfixes where the 2 is the final element of the permutation. These permutations have $j-k+1$ elements, but since we know that the last element must be a 2, we can just ignore it and treat it as a length $j-k$ permutation. For the second term, we need to count all π_{up} which avoid 1423 even when they are turned into $\pi_{\text{up}'}$, i.e. even when a 1 is inserted in the k^{th} position. Even though the π_{up} only have length $n-j+k-1$, and no restrictions on where their first element can be, we actually count the permutations of length $n-j+k$ with a 1 in their k^{th} position.

Unfortunately, Equation 3.3 is incorrect because the right-hand side counts permutations once for every k such that the permutation can be separated into π_{down} and π_{up} with π_{down} containing all the elements at most $j-k+1$ and π_{up} containing all other elements. This leads to a lot of double-counting!

To fix this, we will count permutations only once, corresponding to the maximal k that allows them to be appropriately broken up. If we separate π into π_{down} and π_{up} and find that π_{down} can be broken into two parts where the second part has at least $j-i+1$ elements and all elements in the first part are larger than all elements in the last part, that would imply that we could have taken just that second part as π_{down} and included the first part in π_{up}. In this decomposition π_{down} starts later in the permutation, so k is larger.

When a permutation can be broken up in this way we say it is decomposable. Precisely, we say that a suffix whose elements are all smaller than all the other elements of the permutation is a *small suffix*, and a permutation with a proper small suffix of

length $\geq p$ is *p-decomposable*. Similarly, a permutation which is not p-decomposable is *p-indecomposable*. This is definition is reminiscent of but not exactly the same as Bóna's definition of decomposable in [12].

Now, we can define $A(\{1423\}, 12, [i,j], n, p)$ to be the subset of $A(\{1423\}, 12, [i,j], n)$ consisting of p-indecomposable permutations. Similarly, $A(\{1423\}, 1, [i], n, p)$ is the subset of $A(\{1423\}, 1, [i], n)$ consisting of p-indecomposable permutations.

This leads us to the final and correct system of recurrences.

Theorem 3.2.1. *The following system of recurrences holds:*

$$|A(\{1423\}, \emptyset, [], n)| = \sum_{i=1}^{n} |A(\{1423\}, 1, [i], n, n)|$$

$$\begin{aligned}
|A(\{1423\}, 1, [i], n, p)| = &\begin{cases} \sum_{j=n-p}^{i-1} |A(\{1423\}, 1, [j], n-1, p)| & \text{if } p \geq 2 \text{ or } (p = 1, n \neq i) \\ 0 & \text{otherwise} \end{cases} \\
&+ \begin{cases} \sum_{j=1}^{n-p-1} |A(\{1423\}, 1, [j], n-1, n-j)| & \text{if } p \geq 2 \text{ or } (p = 1, n \neq i) \\ 0 & \text{otherwise} \end{cases} \\
&+ \sum_{j=i+1}^{n} |A(\{1423\}, 12, [i,j], n, p-1)|
\end{aligned}$$

$$\begin{aligned}
|A(\{1423\}, 12, [i,j], n, p)| = &\sum_{k=1}^{n-p-1} |A(\{1423\}, 1, [i-k+1], j-k, j-i-1)| \\
&\cdot |A(\{1423\}, 1, [k], n-j+k, n-j+1)| \\
&+ \sum_{k=n-p}^{i} |A(\{1423\}, 1, [i-k+1], j-k, j-i-1)| \\
&\cdot |A(\{1423\}, 1, [k], n-j+k, p-j+k)|.
\end{aligned}$$

Proof. The first equation is almost identical to the Equation 3.1. Note that $A(\{1423\}, 1, [i], n) = A(\{1423\}, 1, [i], n, n)$ because a length-n permutation has no proper suffix of length n.

The second equation is closely related to Equation 3.2. The first sum in Equation 3.2 is exactly the same as the final term in this new equation, except we have to

ensure that the p-indecomposibility which the left-hand side requires is also respected by the left-hand side. The second sum in Equation 3.2 gives rise to the first two sums in this new equation. These two are indexed by the second downfix element being added to a permutation in $A(\{1423\}, 1, [i], n, p)$. If $p = 0$ or $p = 1$ and $n = i$, then $|A(\{1423\}, 1, [i], n, p)| = 0$ since there is a (possibly empty) proper small suffix of every permutation in $A(\{1423\}, 1, [i], n)$. Otherwise, any such proper small suffix would have to include the new downfix element. Note that (just like in Equation 3.2) we drop the old downfix element since it is ES-reducible.

Now, there are two cases based on how close to the end of the permutation the new downfix element occurs. If it occurs far away from the end of the permutation (specifically by element $n-p-1$) then any proper small suffix which is long enough to include this new element (i.e. has length $\geq n-j$) already has length p. On the other hand, any proper suffix which doesn't include this new element is not small. Therefore, permutations obey the left-hand side's p-indecomposibility condition if and only if they lack a proper small suffix of length $\geq n-j$.

The other case is when j is close to the end of the permutation (i.e. $\geq n-p$). Any small suffix which is long enough to cause a permutation counted by the left-hand side to be p-decomposable is already long enough to include j, and so we just need to check for proper suffixes of length $\geq p-1$ (p is not necessary because we're ignoring the 1 in the downfix which, when added back in, would add one to the small suffix's length.)

We're now ready to consider the third equation. This equation is based on Equation 3.3. Recall that this equation describes breaking a permutation π into π_{down} and π_{up} where π_{down} includes all the elements with indices between k (the index of the right-hand side sum) and j, while π_{up} includes all other elements. To count all possible π_{down}, we see that they are length $j-k$ permutations (we know that the last element of π_{down} is 2, and so we ignore it) avoiding 1423 where the 1 occurs in position $i-k+1$. As noted earlier, to ensure that k is chosen maximally, we also insist that no π_{down} have a small suffix of length $\geq j-i-1$.

To count all possible π_{up}, though, we need two cases based on the value of k. If k is small (i.e. $\leq n-p-1$), then any small suffix long enough to reach the point where

π_{down} starts (as any small suffix of length ≥ 1 must) will give a small suffix of length $\geq p$. Therefore, we just need to ensure that π_{up} has no small suffix of length $n-j+1$. On the other hand, if k is large, then merely reaching π_{down} is not enough to give a small suffix of length-p. To do so, π_{up} must have a small suffix long enough that it's length, when added to the length of π_{down}, gives a small suffix of length p. So, we need to ensure that π_{up} has no small suffix of length $p-j+k$ to prevent this from happening. Once we have the number of possible π_{down} and π_{up} for each k, we multiply them and sum over all k to find the number of possible π. □

With these recurrences in hand, we only need the initial conditions

$$|A(\{1423\},1,[i],1,p)| = \begin{cases} 1 & \text{if } p>0 \\ 0 & \text{if } p=0 \end{cases}$$

to have a fast algorithm to compute the number of length-n permutations which avoid 1423 (and equivalently 1342).

3.3 Maple Implementaion

An implementation of this recurrence is available on the author's website here. This package has three functions: `a(n)`, `b(i,n,p)`, and `c(i,j,n,p)` which calcuate $|A(\{1423\},\emptyset,n)|, |A(\{1423\},1,[i],n,p)|$, and $|A(\{1423\},12,[i,j],n,p)|$ respectively. Users can see all these functions by calling the `Help()` function or can call this function with either `a`, `b`, or `c` as an argument to see descriptions of these functions.

Chapter 4

A New Quantity Counted by OEIS Sequence A006012

In this section, we prove a conjecture of Callan in [22] that OEIS sequence A006012 counts a certain kind of permutation. Call this sequence $(a_n)_{n=1}^{\infty}$; then a_n is defined by $a_1 = 1$, $a_2 = 2$, and $a_n = 4a_{n-1} - 2a_{n-2}$ (the actual sequence in the OEIS is offset by one, so $a_0 = 1$, $a_1 = 2$, and the recursion is the same). The conjecture states that a_n is equal to the number of permutations of length n for which no subsequence *abcd* has the following two properties: b and c occur consecutively and $\max\{a, c\} < \min\{b, d\}$. As noted in the introduction, this work was originally published in the *Journal of Integer Sequences* [10].

We can rewrite this conjecture in the language of pattern avoidance, in particular, using the dashed notation for vincular pattern avoidance introduced by Babson and Steingrímsson [6]. They define a dashed pattern to be a permutation $\pi_1...\pi_k$, some of whose elements may be separated by dashes. A subsequence of a permutation is an occurrence of a pattern if (i) all the elements have the same relative order as the elements of the pattern, and (ii) if there is no dash between the i^{th} and $i+1^{\text{th}}$ elements of the pattern, then the i^{th} and $i+1^{\text{th}}$ element of the subsequence occur consecutively in the permutation. Just like in ordinary pattern avoidance, a permutation avoids a pattern if it does not contain any occurrence of the pattern, and a permutation avoids a set of patterns if it does not contain any occurrence of any of them. As in the previous section, $\text{Av}(B)$ is the set of permutations avoiding all the patterns in a set B, and $\text{Av}(B; n)$ is the set of length n permutations in $\text{Av}(B)$. The following two examples should help clarify these definitions.

Example 4.0.1. The permutation 251346 contains the subsequence 5146 which is an occurrence of the pattern 3-1-24 because the elements of the subsequence occur in the

same relative order as 3124, and the 4 and 6 are consecutive in the original permutation (the 5 and 1 are also consecutive - that is allowed but not necessary).

Example 4.0.2. The permutation 251346 avoids 32-1-4 (i.e. $251346 \in \mathrm{Av}(\{\text{32-1-4}\}; 6) \subseteq \mathrm{Av}(\{\text{32-1-4}\})$).

We are now ready to rewrite the conjecture using this notation. To express it and the theorems in the next section concisely, we let $A = \{\text{1-32-4}, \text{1-42-3}, \text{2-31-4}, \text{2-41-3}\}$ and $B = \{\text{1-3-2-4}, \text{1-4-2-3}, \text{2-3-1-4}, \text{2-4-1-3}\}$.

Proposition 4.0.3. *A subsequence of a permutation abcd has the properties that b and c occur consecutively in the permutation and* $\max\{a, c\} < \min\{b, d\}$ *if and only if that subsequence is an occurrence of a pattern in A.*

Proof. Let *abcd* be a subsequence with the two indicated properties. Suppose that $a < c$ and $b < d$. Because $\max\{a, c\} < \min\{b, d\}$, it follows that $a < c < b < d$, and so *abcd* is an occurrence of 1-3-2-4. Also, because b and c occur consecutively, we can remove the middle dash and say that *abcd* is an occurrence of 1-32-4. If $a > c$ or $b > d$, we can apply the same argument as long as we switch 1-32-4 with another appropriately chosen member of A.

Conversely, suppose that *abcd* is an occurrence of a pattern in A. In every pattern in A, the first and third elements are 1 and 2, while the second and fourth are 3 and 4. Therefore, we have $\max\{a, c\} < \min\{b, d\}$. Further, since there is no dash between the second and third elements of any pattern in A, it must be that b and c occur consecutively. □

Using Proposition 4.0.3, we rewrite the conjecture as

$$a_n = |\mathrm{Av}(A; n)|. \tag{4.1}$$

We will prove two theorems. The first is that $\mathrm{Av}(A)$ and $\mathrm{Av}(B)$ are the same set. In particular, this theorem establishes that $|\mathrm{Av}(A; n)| = |\mathrm{Av}(B; n)|$ for $n \geq 1$. The second theorem is that $a_n = |\mathrm{Av}(B; n)|$ for $n \geq 1$, so together these two theorems prove that Equation 4.1 holds.

Theorem 4.0.4. *The equality $Av(A) = Av(B)$ holds.*

Proof. It is immediately clear that any permutation containing an occurrence of an element of A must contain an occurrence of an element of B, so we only need to show that the converse is also true. Let π be a permutation. As established by Proposition 4.0.3, subpermutation $\pi_a\pi_b\pi_c\pi_d$ of π is an occurrence of a pattern in A if and only if $c = b + 1$ and $\max\{\pi_a, \pi_c\} < \min\{\pi_b, \pi_d\}$. Similarly, a subpermutation $\pi_a\pi_b\pi_c\pi_d$ of π is an occurrence of a pattern in B if and only if $\max\{\pi_a, \pi_c\} < \min\{\pi_b, \pi_d\}$.

Choose any element of B, and suppose that π contains an occurrence of this element. Find $a < b < c < d$ such that $\max\{\pi_a, \pi_c\} < \min\{\pi_b, \pi_d\}$. Let e be the largest index less than c such that $\pi_e > \max\{\pi_a, \pi_c\}$, i.e., $e = \max\{i : i < c, \pi_i > \max\{\pi_a, \pi_c\}\}$. Because b is an element of $\{i : i < c, \pi_i > \max\{\pi_a, \pi_c\}\}$, it follows that e exists and $a < b \leq e < e + 1 \leq c < d$. Now, we claim that $\pi_a\pi_e\pi_{e+1}\pi_d$ is an occurrence of a pattern in A. Obviously, $e+1 = e+1$, and so it remains to check that $\max\{\pi_a, \pi_{e+1}\} < \min\{\pi_e, \pi_d\}$. Because $\max\{\pi_a, \pi_c\} < \min\{\pi_b, \pi_d\}$, we conclude that $\pi_a < \pi_d$, and by the choice of e, we also have $\pi_a < \pi_e$. Now, either $e+1 = c$, in which case $\pi_{e+1} = \pi_c$, or else $\pi_{e+1} < \max\{\pi_a, \pi_c\}$ because otherwise we would have chosen $e + 1$ as the $\max\{i : i < c, \pi_i > \max\{\pi_a, \pi_c\}\}$ instead of e. It follows that $\pi_{e+1} \leq \max\{\pi_a, \pi_c\} < \pi_d, \pi_e$ for the same reasons as π_a. Therefore, $\max\{\pi_a, \pi_{e+1}\} < \min\{\pi_e, \pi_d\}$ and $\pi_a\pi_e\pi_{e+1}\pi_d$ is an occurrence of a pattern in A. We conclude that the permutations avoiding the patterns of A are the same as the permutations avoiding the patterns of B. □

Armed with this theorem, we now need only show that $\big(|\mathrm{Av}(B; n)|\big)_{n\geq 1}$ satisfies the same recurrence as $(a_n)_{n\geq 1}$. Our strategy will be as follows: given $\mathrm{Av}(B; n-1)$, define four maps which, when all of them are applied to all the permutations of $\mathrm{Av}(B; n-1)$, will generate all of the permutations of $\mathrm{Av}(B; n)$. Then we will count how many permutations of $\mathrm{Av}(B; n)$ are double-counted in this way, and find that there are two for every element of $\mathrm{Av}(B; n-2)$, thereby establishing the recurrence.

Note that, for a permutation to avoid all the patterns of A, it must be the case that either 1 and 2 occur consecutively (not necessarily in that order) or either 1 or 2 is the last element of the permutation. This observation motivates the following definitions

of the four maps $f_{\text{before}}, f_{\text{after}}, f_{\text{end}}$, and f_{bump}. Let f_{before} be the function that inputs a permutation and outputs that permutation with all elements increased by 1 and a 1 inserted immediately before the new 2. Let f_{after} be the function that also inputs a permutation and outputs that permutation with all the elements increased by 1 and a 1 inserted immediately after the new 2. Similarly, let f_{end} be the function that inputs a permutation, increases all its elements by 1 and puts a 1 at the end of it, and let f_{bump} be the function that inputs a permutation, increases all its elements by 1, replaces the new 2 with a 1 and puts a 2 at the end. The following example gives a concrete illustration of the four functions.

Example 4.0.5. Let $\pi = 31542$. Then $f_{\text{before}}(\pi) = 412653, f_{\text{after}}(\pi) = 421653$, $f_{\text{end}}(\pi) = 426531$, and $f_{\text{bump}}(\pi) = 416532$. Note that $\pi \in \text{Av}(B)$, and so are all its images.

Suppose $n \geq 2$. The following two lemmas will together establish that

$$\begin{aligned} &f_{\text{before}}(\text{Av}(B; n-1)) \cup f_{\text{after}}(\text{Av}(B; n-1)) \cup f_{\text{end}}(\text{Av}(B; n-1)) \cup f_{\text{bump}}(\text{Av}(B; n-1)) \\ &= \text{Av}(B; n). \end{aligned} \tag{4.2}$$

Lemma 4.0.6. *The functions $f_{before}, f_{after}, f_{end}$, and f_{bump} all map elements of $Av(B; n-1)$ to elements of $Av(B; n)$.*

Proof. Choose some $\sigma \in \text{Av}(B; n-1)$, and consider each function in turn. If $f_{\text{before}}(\sigma)$ or $f_{\text{after}}(\sigma)$ contains an occurrence τ of a pattern in B, then this occurrence must use the element 1 or else τ would already be an offending pattern in σ. But then replacing 1 by 2 would again give an offending pattern in σ. Thus, no such occurrence is possible in either $f_{\text{before}}(\sigma)$ or $f_{\text{after}}(\sigma)$. In addition, if $f_{\text{end}}(\sigma)$ or $f_{\text{bump}}(\sigma)$ contains an occurrence of a pattern in B, then this occurrence cannot use the last element because that element is either a 1 or a 2, and patterns in B only end with 3 or 4. So, this occurrence would already be an occurrence of the pattern in σ, and therefore cannot exist. □

Lemma 4.0.7. *Every permutation in $Av(B; n)$ is the image of a permutation in $Av(B; n-1)$ under at least one of the functions $f_{before}, f_{after}, f_{end}$, or f_{bump}.*

Proof. Choose some $\pi \in \mathrm{Av}(B;n)$. As previously noted, either 1 and 2 occur consecutively in π, or else either 1 or 2 is the final element of π. Let π' be π with the 1 removed and each element decreased by 1. We have introduced no new patterns, and so $\pi' \in \mathrm{Av}(B;n-1)$. Suppose that 1 occurs immediately before 2 in π, then $f_{\mathrm{before}}(\pi') = \pi$. If the 1 occurs immediately after 2 in π, then $f_{\mathrm{after}}(\pi') = \pi$. If the 1 occurs at the end of π, then $f_{\mathrm{end}}(\pi') = \pi$. If the 2 occurs at the end of π, then we need to define π'', which is π with the 1 removed, the 2 moved the position where the 1 used to be, and each element decreased by 1. Again, we have introduced no new patterns, and so $\pi'' \in \mathrm{Av}(B;n-1)$, and $f_{\mathrm{bump}}(\pi'') = \pi$. □

Now that we have established Equation 4.2, we can prove the main result.

Theorem 4.0.8. *The sequence* $\left(|Av(B;n)|\right)_{n\geq 1}$ *satisfies the same recurrence as* $(a_n)_{n\geq 1}$.

Proof. Since $\mathrm{Av}_1(B) = \{1\}$ and $\mathrm{Av}_2(B) = \{12, 21\}$, the initial conditions hold. If the four functions $f_{\mathrm{before}}, f_{\mathrm{after}}, f_{\mathrm{end}}$, and f_{bump} all had disjoint ranges, we could conclude from Equation 4.2 that $|\mathrm{Av}(B;n)| = 4 \cdot |\mathrm{Av}(B;n-1)|$. Unfortunately, some permutations are counted in the range of multiple functions. Each f outputs a certain kind of permutation: f_{before} outputs permutations where 1 immediately precedes 2, f_{after} outputs permutations where 2 immediately precedes 1, f_{end} outputs permutations where 1 occurs at the end, and f_{bump} outputs permutations where 2 occurs at the end. A permutation in $\mathrm{Av}(B;n)$ that fulfills two of these criteria will be hit twice, hence double-counted. Such permutations must be counted once by either f_{before} or f_{after} and again by either f_{end} or f_{bump} because no permutation can be hit by both f_{before} and f_{after} or both f_{end} and f_{bump}. Thus, the final two elements of such permutations are 1 and 2 (not necessarily in that order). Let $g : \mathrm{Av}(B;n) \to \mathrm{Av}(B;n-2)$ be defined as the function that takes a permutation, removes from it the elements 1 and 2, and reduces all other elements by 2. If we restrict g to those permutations that end in either 12 or 21, g becomes a 2-to-1 map from the double-counted permutations of $\mathrm{Av}(B;n)$ to the permutations of $\mathrm{Av}(B;n-2)$, and so the number of double-counted permutations is twice $|\mathrm{Av}(B;n-2)|$. It follows that $|\mathrm{Av}(B;n)| = 4 \cdot |\mathrm{Av}(B;n-1)| - 2 \cdot |\mathrm{Av}(B;n-2)|$

for $n \geq 3$. $\square$

Chapter 5

A Generalization of the "Raboter" Operation

In a 2018 talk at Rutgers' Experimental Math Seminar, Neil Sloane described Claude Lenormand's "raboter" operation for the base two representation of a number [21]. From this representation, one reduces by one the length of each run of consecutive 1s and 0s. Denote this operation by $r(n)$; so, for example, $r(12) = 2$ because 12 is represented in binary as 1100, and reducing the length of each run by one yields 10.

Sloane also defined $L(k) = \sum_{n=2^k}^{2^{k+1}-1} r(n)$ and conjectured that $L(k) = 2 \cdot 3^{k-1} - 2^{k-1}$, a fact which was quickly proven by Doron Zeilberger [31] and Chai Wah Wu [22].

In Section 5.1, we generalize this theorem to bases other than 2. Let $r(b, n)$ be the the number whose base-b representation is generated by taking the base-b representation of n and shortening each run of consecutive identical elements by one. Further, let $L(b, n) = \sum_{n=b^k}^{b^{k+1}-1} r(b, n)$. We will prove that

$$L(b, k) = \frac{b(b-1)}{2b-1}(2b-1)^k - \frac{b-1}{2}b^k.$$

In Section 5.2, we raise $r(b, n)$ to various powers. Define $L(p, b, k) = \sum_{n=b^k}^{b^{k+1}-1} r(b, n)^p$; we develop an algorithm in Maple to rigorously compute $L(p, b, k)$ as an expression in terms of k for any fixed p, b. In addition, for any fixed p, we can conjecture an expression for $L(p, b, k)$ in terms of b and k.

5.1 More General Bases

Following the example of Zeilberger, we find a recurrence satisfied by $L(b, k)$ and then find a closed form expression satisfying the same recurrence.

Theorem 5.1.1. *$L(b, k) = (2b-1) \cdot L(b, k-1) + b^{k-1}\dfrac{(b-1)^2}{2}$ for $k \geq 2$.*

Proof. The $b^{k+1} - b^k = b^k(b-1)$ numbers which contribute to $L(b,k)$ are exactly those numbers whose base-b representations use $k+1$ digits, so each can be written as Ab_1b_2 where $A \in \{1,\ldots,b-1\} \times \{0,\ldots,b-1\}^{k-2}$ and $b_1, b_2 \in \{0,\ldots,b-1\}$. If $b_1 \neq b_2$, then b_2 is a run of just one element, so the raboter operation eliminates it and $r(b, Ab_1b_2) = r(b, Ab_1)$. Numbers with representations Ab_1 are exactly those which were counted in the calculation of $L(b,k-1)$, and each is counted $b-1$ times here, once for each $b_2 \neq b_1$.

If $b_2 = b_1$, then the base-b representation of $r(b, Ab_1b_2)$ is the representation of $r(b, Ab_1)$ with b_2 appended to the end, and so $r(b, Ab_1b_2) = b \cdot r(b, Ab_1) + b_2$. Thus,

$$\begin{aligned}
L(b,k) &= \sum_A \Big(\sum_{b_1} r(b, Ab_1b_1) + \sum_{b_2 \neq b_1} r(b, Ab_1b_2) \Big) \\
&= \sum_A \Big(\sum_{b_1} \big(b \cdot r(b, Ab_1) + b_1\big) + \sum_{b_2 \neq b_1} r(b, Ab_1) \Big) \\
&= \sum_A \sum_{b_1} (2b-1) r(b, Ab_1) + \sum_A \sum_{b_1} b_1 \\
&= (2b-1)L(b,k-1) + (b-1)b^{k-2}\frac{b(b-1)}{2} \\
&= (2b-1)L(b,k-1) + b^{k-1}\frac{(b-1)^2}{2}.
\end{aligned}$$

□

Together with initial condition $L(b,1) = \frac{b(b-1)}{2}$, this determines the sequence $(L(b,k))_{k=1}^{\infty}$. Finding an explicit formula for $L(b,k)$ is now just a matter of finding a formula which obeys this same recurrence.

Corollary 5.1.2. $L(b,k) = \frac{b(b-1)}{2b-1}(2b-1)^k - \frac{b-1}{2}b^k$.

Proof. With some help from Doron Zeilberger's Maple package Cfinite, we conjecture that the formula for $L(b,k)$ has the form $\alpha_1(2b-1)^k + \alpha_2 b^k$, so we solve the system of equations

$$\begin{aligned}
\alpha_1(2b-1) + \alpha_2 b &= \frac{b(b-1)}{2} \\
\alpha_1(2b-1)^2 + \alpha_2 b^2 &= (2b-1)\frac{b(b-1)}{2} + b\frac{(b-1)^2}{2}
\end{aligned}$$

for α_1, α_2 and find $\alpha_1 = \frac{b(b-1)}{2b-1}$ and $\alpha_2 = -\frac{b}{2} + \frac{1}{2}$. Let $L'(b,k) = \dfrac{b(b-1)}{2b-1}(2b-1)^k - \dfrac{b-1}{2}b^k$. Proving that $L(b,k) = L'(b,k)$ is simply a matter of verifying that $L'(b,1) = \frac{b(b-1)}{2}$ and $L'(b,k) = (2b-1)\cdot L'(b,k-1) + b^{k-1}\dfrac{(b-1)^2}{2}$ for $k \geq 2$, which can easily be done using Maple or any other computer algebra system. □

5.2 Higher Moments

With a formula for $L(b,k)$ found, we consider the following additional generalization:

$$L(p,b,k) = \sum_{n=2^k}^{2^{k+1}-1} r(b,n)^p;$$

that is the sum of $r(b,n)^p$ taken over all numbers n whose base-b representation has $k+1$-digits. The trick in this case is to work inductively beginning with the (solved) $p = 1$ case, and along the way compute $L(l,p,b,k)$ which we define to be the sum of $r(b,n)^p$ taken over all numbers n whose base-b representation has $k+1$-digits, the last of which is l.

In order to compute $L(l,p,b,k)$, we use the following recurrence:

Theorem 5.2.1. $L(l,p,b,k) = (b^p - 1)\cdot L(l,p,b,k-1) + L(p,b,k-1) + \sum_{i=1}^{p} l^i b^{p-i}\binom{p}{i} L(l,p-i,b,k-1)$.

Proof. The numbers with length-$(k+1)$ base-b representations ending in l are exactly those which can be written as Ab_1b_2 with $A \in \{1,\ldots,b-1\}\times\{0,\ldots,b-1\}^{k-2}, b_1 \in$

$\{0, \ldots, b-1\}$, and $b_2 = l$. Therefore,

$$\begin{aligned}
L(l,p,b,k) &= \sum_A \Big(\sum_{b_1 \neq l} r(b, Ab_1 l)^p + r(b, All)^p \Big) \\
&= \sum_A \Big(\sum_{b_1 \neq l} r(b, Ab_1)^p + (b \cdot r(b, Al) + l)^p \Big) \\
&= L(p,b,k-1) - L(l,p,b,k-1) + \sum_A \sum_{i=0}^{p} \binom{p}{i} b^{p-i} r(b, Al)^{p-i} l^i \\
&= L(p,b,k-1) - L(l,p,b,k-1) + b^p L(l,p,b,k-1) \\
&\qquad + \sum_{i=1}^{p} \binom{p}{i} b^{p-i} L(l, p-i, b, k-1)^{p-i} l^i \\
&= (b^p - 1) \cdot L(l,p,b,k-1) + L(p,b,k-1) + \sum_{i=1}^{p} l^i b^{p-i} \binom{p}{i} L(l, p-i, b, k-1).
\end{aligned}$$

□

We find a similar recurrence for $L(p,b,k)$.

Theorem 5.2.2. $L(p,b,k) = (b^p + b - 1)L(p,b,k-1) + \sum_{l=0}^{b-1} \sum_{i=1}^{p} b^{p-i} l^i \binom{p}{i} L(l, p-i, b, k-1)$.

Proof. Again, note that the numbers counted by $L(p,b,k)$ are those which can be written as Ab_1b_2 with $A \in \{1, \ldots, b-1\} \times \{0, \ldots, b-1\}^{k-2}$, and $b_1, b_2 \in \{0, \ldots, b-1\}$. Therefore, the following equations hold:

$$\begin{aligned}
L(p,b,k) &= \sum_A \Big(\sum_{b_1 \neq b_2} r(b, Ab_1 b_2)^p + \sum_{b_1} r(b, Ab_1 b_1)^p \Big) \\
&= \sum_A (b-1) \sum_{b_1} r(b, Ab_1)^p + \sum_A \sum_{b_1} (br(Ab_1) + b_1)^p \\
&= (b-1)L(p,b,k-1) + \sum_A \sum_{b_1} \sum_{i=0}^{p} \binom{p}{i} b^{p-i} r(Ab_1)^{p-i} b_1^i \\
&= (b-1)L(p,b,k-1) + \sum_A \sum_{b_1} b^p r(Ab_1)^p + \sum_{b_1} \sum_{i=1}^{p} \sum_A \binom{p}{i} b^{p-i} r(Ab_1)^{p-i} b_1^i \\
&= (b^p + b - 1)L(p,b,k-1) + \sum_{b_1} \sum_{i=1}^{p} b_1^i b^{p-i} \binom{p}{i} L(b_1, p-i, b, k-1).
\end{aligned}$$

Change the name of b_1 to l to maintain consistent notation, and we have derived the claimed equation. □

5.3 Maple Implementation

The Maple package `raboter.txt` available here contains functions to implement this recurrence. The most important are `SumPowers(b,k,p)` which finds an expression in terms of k for $L(p,b,k)$ (for fixed b and p) and `GuessGeneralForm(b,n,p)` which conjectures an expression in terms of k and b for $L(p,b,k)$ (for fixed p).

For example, this package proves that

$$L(2,2,k) = \frac{2}{3}5^k - \frac{1}{6}2^k - \frac{2}{3}3^k$$

and conjectures that

$$\begin{aligned} L(2,b,k) = &\Big(\frac{1}{6}b^2 - \frac{1}{6}b - \frac{1}{3}\Big)(b-1)^k + \Big(-\frac{1}{6}b^2 + \frac{1}{3}b - \frac{1}{6}\Big)b^k \\ &- \frac{b(b-1)}{2b-1}(2b-1)^k + \frac{2b^3+3b^2-3b-2}{6(b^2+b-1)}(b^2+b-1)^k. \end{aligned}$$

References

[1] M. Albert and M. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300:1–15, 2005.

[2] M. Albert, C. Homberger, J. Pantone, N. Shar, and V. Vatter. Generating permutations with restricted containers. *Journal of Combinatorial Theory*, 157:205–232, 2018.

[3] M. Albert, S. Linton, and N. Ruškuc. The insertion encoding of permutations. *The Electronic Journal of Combinatorics*, 12, 2005.

[4] K. Appel and W. Haken. Every planar map is four colorable. part i: Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.

[5] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. part ii: Reducibility. *Illinois Journal of Mathematics*, 21:491–567, 1977.

[6] E. Babson and E. Steingrímsson. Generalized permutation patterns and a classification of the mahonian statistics. *Séminaire Lotharingien de Combinatoire*, 2000.

[7] A. Baxter and L. Pudwell. Enumeration schemes for vincular patterns. *Discrete Mathematics*, 312:1699–1712, 2012.

[8] C. Bean, B. A. Gudmundsson, and H. Ulfarsson. Automatic discovery of structural rules of permutation classes. *Mathematics of Computation*, 88:1967–1990, 2019.

[9] C. Bean and H. Ulfarsson. Enumerations of permutations simultaneously avoiding a vincular and a covincular pattern of length 3. *Journal of Integer Sequences*, 20, 2017.

[10] Y. Biers-Ariel. The number of permutations avoiding a set of generalized permutation patterns. *Journal of Integer Sequences*, 20, 2017.

[11] J. Bloom and S. Elizalde. Pattern avoidance in matchings and partitions. *The Electronic Journal of Combinatorics*, 2013.

[12] M. Bóna. Exact enumeration of 1342-avoiding permutations: A close link with labeled trees and planar maps. *Journal of Combinatorial Theory*, 80:257–272, 1997.

[13] A. Burstein. *Enumeration of Words with Forbidden Patterns*. PhD thesis, University of Pennsylvania, 1998.

[14] J. Fox. Stanley-Wilf limits are typically exponential. *arXiv*, 2013.

[15] I. M. Gessel. Symmetric functions and p-recursiveness. *Journal of Combinatorial Theory*, 53:257–285, 1990.

[16] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.

[17] A. Marcus and G. Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *Journal of Combinatorial Theory*, 107:153–160, 2004.

[18] M. Petkovsek, H. Wilf, and D. Zeilberger. *A=B*. A K Peters, 1996.

[19] L. Pudwell. *Enumeration Schemes*. PhD thesis, Rutgers University, 2008.

[20] N. Shar and D. Zeilberger. The (ordinary) generating functions enumerating 123-avoiding words with r occurrences of each of $1, 2, \ldots, n$ are always algebraic. *Ann. Comb.*, 20:387–396, 2016.

[21] N. J. A. Sloane. Coordination sequences, planing numbers, and other recent sequences, 2018. Rutgers Experimental Math Seminar.

[22] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences, 2020.

[23] V. Vatter. Enumeration schemes for restricted permutations. *Combinatorics, Probability and Computing*, 17:137–159, 2008.

[24] V. Vatter. Finding regular insertion encodings for permutation classes. *Journal of Symbolic Computation*, 47:259–265, 2012.

[25] V. Vatter. Permutation classes. In M. Bóna, editor, *Handbook of Enumerative Combinatorics*, pages 754–833. CRC Press, Boca Raton, Florida, 2015.

[26] J. West. *Permutations with forbidden subsequences, and, stack-sortable permutations*. PhD thesis, Massachusetts Institute of Technology, 1990.

[27] Wikipedia. Enumerations of specific permutation classes, 2019.

[28] H. S. Wilf. What is an answer? *American Mathematical Monthly*, 89:289–292, 1982.

[29] D. Zeilberger. Enumeration schemes and, more importantly, their automatic generation. *Annals of Combinatorics*, 2:185–195, 1998.

[30] D. Zeilberger. On Vince Vatter's brilliant extension of Doron Zeilberger's enumeration schemes for counting Herb Wilf's classes. *Personal J. of Shalosh B. Ekhad and Doron Zeilberger*, 2007.

[31] D. Zeilberger. Proof of a conjecture of Neil Sloane concerning Claude Lenormand's "Raboter" operation (OEIS sequence A318921). *The Personal Journal of Shalosh B. Ekhad and Doron Zeilberger*, 2018.